

**TRƯỜNG ĐẠI HỌC KINH TẾ - LUẬT**

**MÔN HỌC: BIG DATA VÀ ỨNG DỤNG**

**Thử nghiệm mô hình SVD và mô hình TF-IDF  
xây dựng hệ thống đề xuất phim trên bộ dữ liệu  
phim thu thập từ nền tảng IMDb**

**(NHÓM 2)**

**Thành viên:**

K204110558 Vũ Thị Phương Anh (NT)

K204111767 Nguyễn Thị Thanh Bình

K204111780 Lê Thị Thuỳ Linh

K204111762 Đinh Thị Thúy An

K204110587 Mai Lê Ngọc Trâm

**Giảng viên hướng dẫn:**

Thạc sĩ Nguyễn Thôn Dã

*Thành phố Hồ Chí Minh , 2022*

## MỤC LỤC

<b>PHẦN I: NỘI DUNG CHÍNH.....</b>	<b>3</b>
Tóm tắt.....	3
1. Giới thiệu .....	3
1.1 Lý do chọn đề tài .....	3
1.2 Đề tài nghiên cứu.....	4
1.3 Đối tượng, phạm vi nghiên cứu; sơ lược lịch sử nghiên cứu đề tài .....	5
2. Nghiên cứu liên quan .....	6
2.1 Mô hình lý thuyết trên thế giới.....	6
2.2 Các mô hình đã được áp dụng ở Việt Nam .....	11
3. Nghiên cứu sơ bộ .....	13
3.1 Các mô hình khả thi.....	13
3.2 Dữ liệu yêu cầu.....	18
3.3 Phương pháp thu thập dữ liệu.....	18
4. Khám phá và trực quan hóa dữ liệu .....	20
4.1 Quá trình khai phá .....	20
4.2. Thực hiện cào dữ liệu review movie .....	36
4.3. Kết quả thu thập dữ liệu .....	40
4.4 Mô tả dữ liệu khai phá.....	43
4.5 Tiền xử lý dữ liệu .....	45
4.6 Khai phá và trực quan hóa dữ liệu.....	54
5. Đề xuất mô hình.....	64
5.1 Mô hình áp dụng TF-IDF .....	64
5.2 Mô hình áp dụng SVD.....	64
6. Kết quả thực nghiệm.....	64
6.1    Kết quả thực nghiệm mô hình TF-IDF .....	64
6.2    Kết quả thực nghiệm mô hình SVD .....	67
6.3 Đánh giá kết quả .....	69
7. Kết luận.....	81
7.1 Đóng góp của dự án.....	81
7.2 Hạn chế của dự án và đề xuất.....	81

<b>PHẦN II: MÃ NGUỒN .....</b>	<b>84</b>
1. Tiền xử lý dữ liệu.....	84
2. Trực quan hóa và khám phá dữ liệu .....	89
3. Mô hình TF-IDF .....	96
4. Mô hình SVD.....	99
5. Đánh giá mô hình SVD.....	103
6. Đánh giá mô hình TF-IDF.....	105
<b>ĐÁNH GIÁ LÀM VIỆC NHÓM .....</b>	<b>107</b>
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>108</b>
<b>DỮ LIỆU TỰ THU THẬP .....</b>	<b>110</b>

# **PHẦN I: NỘI DUNG CHÍNH**

## **Tóm tắt**

Hệ thống đề xuất đang là một đề tài hấp dẫn trong lĩnh vực học máy, đặc biệt là học máy không giám sát, áp dụng cho nhiều lĩnh vực như kinh tế, y tế, giáo dục, v.v. Trong khuôn khổ môn học “Big data và ứng dụng”, nhóm tác giả quyết định sẽ tìm hiểu sâu hơn về một hệ thống đề xuất, cụ thể ở đây là hệ thống đề xuất phim. Nhóm tác giả xây dựng hệ thống đề xuất phim dựa trên hai phương pháp là hệ thống đề xuất dựa trên nội dung (Content-based Recommenders) và hệ thống đề xuất dựa trên lọc cộng tác (Collaborative Filtering Recommenders). Kết quả đưa ra là danh sách các phim phù hợp với một bộ phim được yêu cầu đề xuất.

## **1. Giới thiệu**

### **1.1 Lý do chọn đề tài**

Ngày nay cùng với sự phát triển bùng nổ của internet, số lượng người dùng trực tuyến và các kênh thông tin, giải trí, thương mại điện tử, các nhà cung cấp cùng lượng hàng hóa & dịch vụ,... cũng phát triển nhanh chóng và mạnh mẽ, bằng chứng là theo báo cáo VIỆT NAM DIGITAL 2021 cho thấy có đến 68,72 triệu người sử dụng Internet tại Việt Nam. Trong đó, với mỗi phút có khoảng từ 500-600 giờ phát video clip trên các nền tảng Tiktok, Facebook, Youtube, mỗi ngày số lượng tweet, bài post trên các nền tảng Instagram, Facebook, Twitter vào khoảng 500 triệu,... Với lượng thông tin đồ sộ đến như vậy, thật không ngoa khi nói rằng mạng Internet đã làm cho người sử dụng không còn đủ thời gian để xem xét, cân nhắc các lựa chọn trên tổng các thông tin: tin tức, mặt hàng, phim, tạp chí hay bài hát,... Người dùng thường xuyên gặp phải các vấn đề như không biết mình nên xem phim gì, đọc cuốn sách nào phù hợp với sở thích, nhu cầu của bản thân, làm thế nào để tìm kiếm sản phẩm hay dịch vụ thích hợp nhưng nhanh chóng do lượng thông tin quá lớn, chưa được phân loại trong khi thời gian lại có hạn. Đó là lý do trong thời đại kỹ thuật số như hiện nay, bất kỳ trang web nào cũng đều nên sử dụng một số loại hệ thống gợi ý tìm kiếm và đề xuất cho khách hàng

những thông tin liên quan một cách nhanh chóng và tiện lợi. Nó không chỉ làm giảm sự phức tạp, khó khăn trong quá trình sử dụng của người dùng mà còn tối ưu hóa mức độ ưa thích và trải nghiệm của người dùng từ đó giúp thu hút nhiều người dùng hơn và làm tăng tỷ lệ chuyển đổi khách hàng cho trang web.

Với những kiến thức chuyên môn và kinh nghiệm bổ ích mà thầy đã truyền đạt đến chúng em cùng niềm mong muốn, đam mê muốn nghiên cứu, thực hiện một đồ án có thể áp dụng được những kiến thức đã được tiếp thu, Nhóm chúng em quyết định chọn đề tài xây dựng một hệ thống gợi ý có thể được áp dụng trên các nền tảng trực tuyến mà ở đây nhóm chúng em sẽ tập trung nghiên cứu về Hệ thống đề xuất phim với bộ dữ liệu về phim ảnh được thu thập trên nền tảng IMDB. Đây có thể không phải là một đề tài có tính mới nhưng luôn là một đề tài có tính cấp thiết, có tính hữu dụng và có thể áp dụng vào thực tế, mà hơn hết đây là một đề tài mà nhóm chúng em có khả năng thực hiện và phát huy những kiến thức đã học được.

## 1.2 Đề tài nghiên cứu

Hệ thống gợi ý (Recommendation System - RS) là một kỹ thuật lọc thông tin được dùng để dự đoán thị hiếu, sở thích của người dùng, đưa ra đề nghị hoặc gợi ý các đối tượng như nhạc, phim, ảnh, tin tức, sách,... hoặc hành động nào đó thích hợp cho người dùng. Những gợi ý cá nhân hóa đưa ra danh sách các items đã được xếp hạng theo sở thích hoặc những ràng buộc khác để quyết định và đưa ra dự đoán các sản phẩm hoặc dịch vụ nào phù hợp nhất. Những quyết định và dự đoán sẽ khác nhau dựa trên những quyết định, sở thích, thị hiếu khác nhau của từng người dùng cụ thể về việc mua những mặt hàng nào, xem những bộ phim nào, nghe những bản nhạc nào, hay đọc những tin tức nào.

Hiện nay, có nhiều phương pháp và giải thuật khác nhau có thể được áp dụng để xây dựng các hệ thống gợi ý, cụ thể như sau:

- Hệ thống gợi ý dựa vào giải thuật lọc theo nội dung (content-based filtering) -  
Hệ thống gợi ý người dùng những items tương tự những mục tin người dùng đã từng thích trước đó trong quá khứ hoặc hiện tại.

- Hệ thống gợi ý dựa vào nhóm giải thuật lọc cộng tác (collaborative filtering) - Hệ thống thu thập và phân tích những thông tin liên quan như hành vi, hoạt động hoặc sở thích của người dùng và dự đoán những gì người dùng sẽ thích dựa trên sự tương đồng của họ với người dùng khác.
- Hệ thống gợi ý dựa trên nhóm giải thuật lai ghép (hybrid filtering) kết hợp phương pháp cộng tác và dựa trên nội dung - Hệ thống này đề xuất bằng cách so sánh thói quen xem và tìm kiếm của những người dùng tương tự (phương pháp lọc cộng tác) kết hợp cung cấp những items có chung đặc điểm với những items mà người dùng đã đánh giá cao (phương pháp lọc dựa trên nội dung).
- Hệ thống gợi ý dựa trên nhóm giải thuật không cá nhân hóa (non-personalization) - Hệ thống gợi ý các mục tin được tương tác nhiều nhất, có tính chuyển đổi cao nhất,...

Trong đồ án này, chúng em đề xuất một hệ thống gợi ý kết hợp 2 nhóm giải thuật lọc dựa trên nội dung (content-based filtering) và lọc cộng tác (collaborative filtering) mang tên: “**Thử nghiệm mô hình SVD và mô hình TF-IDF xây dựng hệ thống đề xuất phim trên bộ dữ liệu phim thu thập từ nền tảng IMDb**”. Mục tiêu chính của đề tài chúng em là tạo ra một hệ thống gợi ý được tối ưu hóa để cung cấp các đề xuất, gợi ý chính xác và nhanh chóng đến cho người dùng.

### **1.3 Đối tượng, phạm vi nghiên cứu; sơ lược lịch sử nghiên cứu đề tài**

#### **1.3.1. Đối tượng**

Chúng tôi sẽ xây dựng hệ thống đề xuất phim dựa trên dữ liệu thu thập từ nền tảng Internet Movie Database và bộ dataset MovieLens. IMDb là trang dữ liệu điện ảnh trực tuyến, đây là nơi lưu trữ và đem đến những thông tin về những chủ đề điện ảnh, truyền hình, trong đó bao gồm cả bộ phim, diễn viên và đạo diễn. Bên cạnh đó, IMDb còn là nơi tổng hợp những bình luận, đánh giá của người dùng khi trải nghiệm những bộ phim phát sóng. Có ba loại tập dữ liệu thu được từ quá trình thu thập thông tin bao gồm:

- User dataset: thông tin của người dùng (user\_id);
- Movie dataset: thông tin về phim (movie\_id, moviename, genres, timestamp);
- Rating dataset: Chứa thông tin xếp hạng đã được người dùng cung cấp cho một bộ phim cụ thể (rating).

### **1.3.2. Phạm vi nghiên cứu**

Về phạm vi nghiên cứu, nhóm thực hiện thu thập dữ liệu bằng 2 phương pháp là: Thứ nhất, lọc và càو dữ liệu, vì lượng bộ phim trên nền tảng IMDb rất nhiều, nên chúng tôi quyết định chọn lọc và cào dữ liệu của tầm 180.000 bộ phim, nhóm đã cào và xử lý được bộ dataset hơn 800,000 lượt đánh giá (1-10) trên 180.000 bộ phim. Thứ hai, sử dụng bộ dữ liệu có sẵn trên MovieLens gồm 1.048.575 ratings trên 62.423 bộ phim.

### **1.3.3. Sơ lược lịch sử nghiên cứu đề tài**

Nhận thấy rằng hiện nay người dùng trong các trang mạng thường khó khăn trong việc tìm kiếm và lựa chọn những thông tin cần thiết, phù hợp để giải quyết một vấn đề, nhu cầu. Nhóm quyết định nghiên cứu về hệ thống đề xuất để hỗ trợ cho người dùng về mảng này. Và cụ thể là hệ thống đề xuất phim. Sau quá trình nghiên cứu và tìm hiểu, nhóm quyết định lựa chọn 2 mô hình áp dụng cho hệ thống này bao gồm: Phân tích giá trị suy biến (SVD) và TF-IDF (Term Frequency – Inverse Document Frequency).

## **2. Nghiên cứu liên quan**

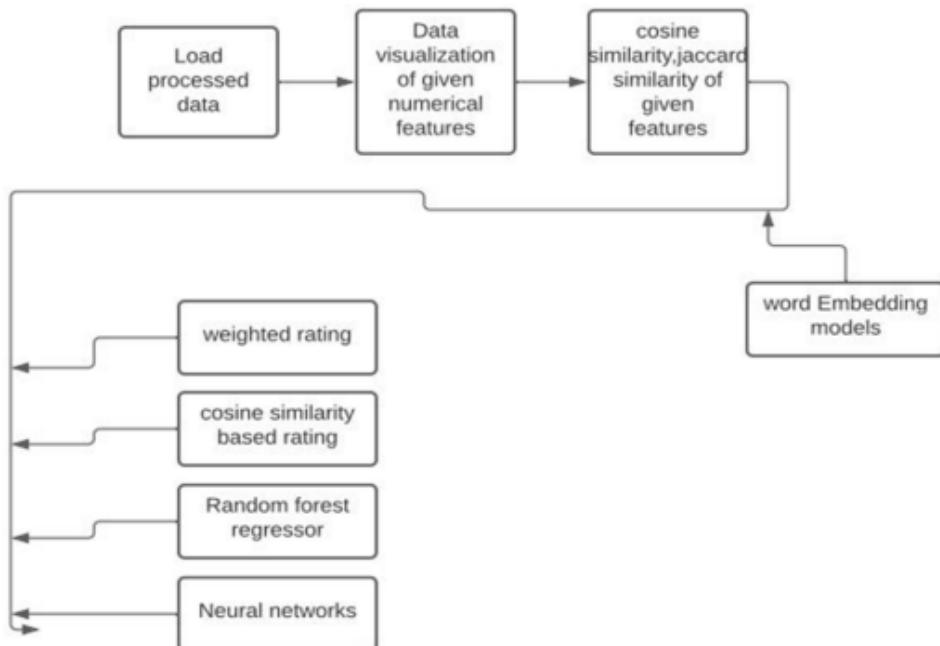
### **2.1 Mô hình lý thuyết trên thế giới**

#### **2.1.1 Xây dựng hệ thống đề xuất phim bằng cách sử dụng Lọc cộng tác với TF-IDF (Nixon, 2020)**

Bài nghiên cứu đưa ra đề xuất liên quan đến việc dự đoán một tập hợp các items để tối đa hóa tiện ích cho người dùng. Như một giải pháp đối với vấn đề này, hệ thống giới

thiệu là một hệ thống lọc thông tin nhằm tìm cách dự đoán xếp hạng do người dùng đưa ra cho một mục. Có các loại hệ thống đề xuất cụ thể là hệ thống đề xuất dựa trên nội dung, dựa trên cộng tác và dựa trên kết hợp các hệ thống khuyến nghị. Lọc cộng tác được phân loại thêm thành lọc cộng tác dựa trên người dùng và mục lọc cộng tác dựa trên mục. Hệ thống khuyến nghị dựa trên lọc cộng tác (CF) có khả năng nắm bắt tương tác hoặc tư quan của người dùng và các items đang được xem xét.

Nhóm tác giả đã khám phá hầu hết các nghiên cứu dựa trên bộ lọc cộng tác hiện có trên tập dữ liệu phim TMDB phổ biến và phát hiện ra rằng, một số tính năng chính đã bị hầu hết các nghiên cứu trước đây bỏ qua. Nhóm tác giả đã thử nghiệm với các phương pháp thống kê điển hình như TF-IDF, Bằng cách sử dụng TF-IDF kích thước của các đường viền của họ (tổng quan và các tính năng văn bản khác) bùng nổ, tạo ra các vấn đề, họ đã giải quyết các vấn đề đó bằng cách sử dụng kỹ thuật giảm kích thước có tên là Singular Value Decomposition (SVD). Sau khi xử lý trước này, dữ liệu Tiền xử lý sẽ được sử dụng để xây dựng các mô hình. Chúng ta có đã đánh giá hiệu suất của các machine learning khác nhau như Random Forest và mạng thần kinh sâu dựa trên Bi-LSTM.



*Mô hình của hệ thống đề xuất*

Kết quả thử nghiệm cung cấp một mô hình đáng tin cậy về MAE (sai số tuyệt đối trung bình), RMSE (Bình phương trung bình gốc) và Bi-LSTM hóa ra là một mô hình tốt hơn với MAE là 0,65 và RMSE là 1,04, nó tạo ra nhiều cá nhân hóa hơn giới thiệu phim so với các mô hình khác.

### **2.1.2 Hệ thống đề xuất phim toàn diện (Hrisav Bhowmick, Ananda Chatterjee & Jaydip Sen, 2021)**

Hệ thống gợi ý, còn được gọi là hệ thống gợi ý, là một loại hệ thống lọc thông tin có gắng dự báo xếp hạng hoặc sở thích của người dùng đối với một mặt hàng. Bài báo này thiết kế và triển khai nguyên mẫu hệ thống đề xuất phim hoàn chỉnh dựa trên Thể loại, Hệ số tương quan Pearson, Độ tương tự Cosine, Dựa trên KNN, Lọc dựa trên nội dung bằng TFIDF và SVD, Lọc cộng tác bằng TFIDF và SVD, công nghệ hệ thống đề xuất dựa trên Thư viện bất ngờ. Ngoài ra, trong bài báo này, nhóm tác giả đã trình bày một ý tưởng mới áp dụng các kỹ thuật học máy để xây dựng một cụm cho phim dựa trên thể loại và sau đó quan sát giá trị quán tính của số cụm đã được xác định. Các hạn chế của các phương pháp được thảo luận trong bài báo này đã được mô tả, cũng như cách một phương pháp khắc phục nhược điểm của một phương pháp khác.

Trong bài báo này, năm phương pháp đã được kết hợp để đưa ra đề xuất, đó là Lọc cộng tác, Lọc dựa trên nội dung, Phân tách giá trị đơn, Dựa trên thể loại, Dựa trên hệ số tương quan Pearson, Tương tự cosine, KNN (với số liệu khoảng cách cosine).

Các phương pháp được sử dụng trong bài báo là:

#### **a, Dựa trên thể loại (Genre Based)**

Đề xuất dựa trên thể loại nằm trong quá trình lọc dựa trên nội dung. Dạng hệ thống đề xuất này hiển thị các mục có liên quan dựa trên nội dung của các mục đã tìm kiếm trước đó của người dùng. Thuộc tính/thể của sản phẩm mà người dùng thích được gọi là nội dung trong trường hợp này. Các mục được gắn nhãn bằng từ khóa trong loại hệ thống này, sau đó

hệ thống sẽ có găng hiếu những gì người dùng muốn bằng cách tìm kiếm cơ sở dữ liệu của nó và cuối cùng cống găng để xuất các sản phẩm khác nhau mà người dùng muốn

Để xuất dựa trên thể loại dựa trên thể loại mà người dùng thích xem. Giả sử nếu thể loại 'hành động' là người dùng muốn xem thì người dùng đó sẽ được đề xuất những bộ phim hàng đầu thuộc thể loại hành động dựa trên điểm số có trọng số. Công thức của điểm trọng số như dưới đây:

$$\text{Score} = \frac{v}{v+m} * R + \frac{m}{m+v} * C, \text{ where}$$

*v = number of ratings for the movie*

*m = minimum number of ratings required to be eligible*

*R = average rating of the movie*

*C = mean rating across whole data*

### b. Hệ số tương quan Pearson (Pearson Correlation Coefficient)

Hệ số tương quan Pearson là một phương pháp đơn giản nhưng mạnh mẽ để xác định cách một biến thay đổi tuyến tính với một biến khác. Ưu điểm của phương pháp này đã được áp dụng trong hệ thống tư vấn này. Sau đây là công thức tính hệ số tương quan.

$$r_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

*Trong đó:*

- $x_i, y_i$  là các điểm mẫu riêng lẻ
- $n$  là cỡ mẫu
- $x$  ngang,  $y$  ngang là giá trị trung bình
- $R_{xy}$  là ký hiệu để xác định hệ số tương quan.

### c. Độ tương tự Cosine

Cách tiếp cận này xác định mức độ giống nhau của hai người dùng bằng cách tính cosin của góc giữa hai vecto, ở đây vecto là phim. Công thức sau đây tính toán độ tương tự cosine giữa hai phim:

$$\text{Sim}(x,y)=\cos(\vec{X}, \vec{Y}) = \frac{\vec{X} \cdot \vec{Y}}{|X|.|Y|}$$

Ở đây X và Y là hai phim mà giữa đó góc cosin được tìm ra để xác định xem chúng giống nhau như thế nào. Chỉ số Cosine càng dần về 0 thì hai bộ phim càng giống nhau.

### d. Thuật toán KNN với số liệu cosin (KNN algorithm with cosine metric)

Đề xuất dựa trên thuật toán KNN nằm trong quá trình lọc cộng tác. Trong quá trình nghiên cứu của nhóm tác giả, lọc cộng tác dựa trên mục đã được sử dụng khi độ tương tự giữa một mục cụ thể và K mục cụ thể khác đã được tính toán bằng cách sử dụng khoảng cách Euclide và độ tương tự cosine.

Ở đây, giá trị của góc cosin giữa mục tiêu và các mục khác nhỏ hơn, chúng sẽ giống nhau hơn. Giá trị của K được lấy là 5 trong trường hợp của nhóm tác giả khi khoảng cách nhỏ nhất giữa 5 phim hàng đầu với phim mục tiêu đã được đánh giá và sau đó được đề xuất cho người dùng đã xem phim mục tiêu.

### e. Phân cụm phim (Clustering of Movies)

Phân cụm là một cách tiếp cận khác để đề xuất phim trong đó bằng cách xem các phim mà người dùng xem phân cụm được sử dụng để tìm các phim tương tự và cuối cùng để xuất những phim giống nhất.

Thuật toán phân cụm K-means đã được sử dụng trong nghiên cứu của nhóm tác giả để đưa ra khuyến nghị. Mục tiêu chính của thuật toán K-mean là giảm tổng khoảng cách giữa các điểm và trọng tâm cụm tương ứng của chúng. Ở đây, khái niệm quan tính bắt đầu

hoạt động để tính toán tổng khoảng cách của tất cả các phim trong một cụm từ trọng tâm cụm của chúng. Mục tiêu là giữ quán tính thấp hơn trong một cụm và càng xa càng tốt trong hai cụm khác nhau.

Ngoài ra còn 3 phương pháp khác như:

- Ma trận tiềm ẩn nội dung sử dụng TFIDF & SVD (Content latent matrix using TFIDF & SVD)
- Ma trận tiềm ẩn hợp tác với TFIDF & SVD (Collaborative latent matrix with TFIDF & SVD)
- Surprise Library với KNN Basic (Surprise Library with KNN Basic)

Kết quả của bài báo nghiên cứu cho thấy trong số tất cả các cách tiếp cận được ghi lại, cách tiếp cận dựa trên lọc cộng tác liên quan đến phương pháp TFIDF và SVD tỏ ra là tốt nhất.

## 2.2 Các mô hình đã được áp dụng ở Việt Nam

### 2.2.1 Xây dựng một hệ thống gợi ý phim đơn giản với Python (Quang, 2019)

Về cơ bản, có 2 loại hệ thống đưa ra gợi ý phổ biến nhất là:

- Content-based: đề xuất dựa theo nội dung
- Collaborative Filtering: đề xuất dựa theo thị hiếu của người dùng khác. Cách tiếp cận này tiếp tục được chia ra thành 2 mảng nhỏ là:
  - User-based collaborative filtering
  - Item-based collaborative filtering

Có hai loại mô hình collaborative đó là Content-based và Model-based:

- Content based sử dụng tập dữ liệu chứa nhiều thông tin như thể loại (genre), nhà sản xuất (producer), diễn viên (actor), nhạc sĩ (musician) để đề xuất các mục nói phim hoặc nhạc.

- Model-based dựa trên yếu tố ma trận (matrix factorizatio) và tốt hơn trong việc xử lý sự thưa thớt. Ví dụ về các phương Model-based như vậy bao gồm cây quyết định (decision trees), mô hình dựa trên quy tắc (rule based models), phương pháp Bayes và mô hình nhân tố tiềm ẩn (latent factor models).

Trong phạm vi của bài hướng dẫn, tác giả đã hướng dẫn cách để xây dựng hệ thống đề xuất đơn giản dựa trên hệ thống Item-based collaborative filtering.

Bộ dữ liệu đầu vào gồm có:

- user\_id - Cột ID của người dùng đã đánh giá bộ phim.
- item\_id - Cột ID of bộ phim.
- rating - Xếp hạng người dùng đã cho bộ phim, từ 1 đến 5.
- timestamp - Thời gian bộ phim được đánh giá.
- title - Tiêu đề của bộ phim.

### **2.2.2 Xây dựng Hệ thống Gợi ý phim dựa trên mô hình nhân tố lảng giềng (Triệu Vĩnh Viêm, 2013)**

Trong mô hình này, nhóm tác giả giới thiệu một tiếp cận tích hợp các ưu điểm của cả hai cách tiếp cận của lọc cộng tác (collaborative filtering) là mô hình nhân tố tiềm ẩn (latent factor models) và mô hình lảng giềng (neighborhood approach). Cách tiếp cận này đã được đề xuất bởi Koren (2010). Ở đây, bên cạnh việc xây dựng một hệ thống trên nền tảng website để gợi ý những bộ phim phù hợp cho người dùng, nhóm tác giả cũng đã điều chỉnh mô hình đã có bằng cách đưa vào các hệ số regularization trên từng tham số khác nhau của mô hình nhằm cải tiến kết quả dự đoán.

Mô hình lảng giềng (Neighborhood models): gồm User-based CF và Item-based CF. Các tác giả xây dựng mô hình tính thêm các trọng số lảng giềng để tăng độ chính xác của mô hình.

$$\hat{r}_{ui} = b_{ui} + \sum_{j \in S^k(i,u)} \theta_{ij}^u (r_{uj} - b_{uj}) \quad (2)$$

Mô hình nhân tố tiềm ẩn (Latent factor models): gồm phân giải giá trị đơn (Singular Value Decomposition - SVD) và phân rã ma trận (Matrix Factorization - MF).

- SVD chỉ xem xét mối quan hệ giữa hai thực thể chính, vì vậy sẽ bị thiếu độ chính xác.
- Để hỗ trợ cho SVD, nhóm tác giả đã sử dụng Maximum Margin Matrix Factorization - MMMF để phối hợp thêm nhiều thông tin khác nhằm tăng độ chính xác của mô hình.

Từ đó, nhóm tác giả xây dựng mô hình nhân tố tiềm ẩn mới gồm phân rã ma trận đa quan hệ (Multi-Relational Matrix Factorization - MRMF) cho lĩnh vực thực nghiệm là phim ảnh (movies) và gene function prediction.

Để khắc phục những hạn chế và hỗ trợ những điểm mạnh của hai cách tiếp cận, nhóm tác giả quyết định chọn mô hình Asymmetric-SVD làm chức năng gợi ý của hệ thống.

Bộ dữ liệu đầu vào gồm: người dùng (users), đánh giá (ratings), bình luận (comments), thể loại (genre).

### **3. Nghiên cứu sơ bộ**

#### **3.1 Các mô hình khả thi**

##### **3.1.1 Mô hình áp dụng TF-IDF**

Trong các thuật toán máy học, người ta thường dùng dữ liệu số để tính toán. Vì vậy khi cần xử lý một văn bản ngôn ngữ, người ta thường gặp khó khăn. Từ đó có một phương pháp ra đời để khắc phục những hạn chế đó là vector hóa dữ liệu văn bản. Một trong những cách để vector hóa dữ liệu văn bản là sử dụng TF-IDF. Quá trình vector hóa TF-IDF liên

quan đến việc tính điểm TF-IDF cho mỗi từ trong kho văn bản và sau đó đưa thông tin đó vào một vector. Sau khi có các vector TF-IDF, người xử lý có thể kết hợp với nhiều kỹ thuật khác để đưa ra kết quả mình mong muốn.

#### **a, Khái niệm**

TF-IDF là một trong những kỹ thuật quen thuộc để xử lý ngôn ngữ tự nhiên. TF-IDF là viết tắt của “term frequency – inverse document frequency”. Đây là trọng số thể hiện mức độ quan trọng của một từ trong một văn bản, và văn bản ở đây cũng được xét trong một tập hợp nhiều văn bản. (Dương, 2016)

#### **b, Các chỉ số quan trọng của TF-IDF (Hiếu, 2019)**

TF-IDF được tính từ 2 chỉ số quan trọng là TF và IDF:

- TF (term frequency) là tần suất một từ xuất hiện trên một văn bản, được tính bởi công thức:

$$tf(t, d) = \frac{f(t, d)}{\max\{f(w, d) : w \in d\}}$$

Trong đó:

-  $tf(t, d)$ : tần suất xuất hiện của từ  $t$  trong văn bản  $d$

-  $f(t, d)$ : Số lần xuất hiện của từ  $t$  trong văn bản  $d$

-  $\max(\{f(w, d) : w \in d\})$ : Số lần xuất hiện của từ có số lần xuất hiện nhiều nhất trong văn bản  $d$

- IDF (inverse document frequency) là nghịch đảo tần suất của văn bản được tính bởi công thức:

$$\text{idf}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

Trong đó:

- $\text{idf}(t, D)$ : giá trị idf của từ  $t$  trong tập văn bản
- $|D|$ : Tổng số văn bản trong tập  $D$
- $|\{d \in D : t \in d\}|$ : thể hiện số văn bản trong tập  $D$  có chứa từ  $t$

Theo tờ Capital One thì lý do chúng ta cần IDF là để giảm tác động của các từ thường xuyên xuất hiện trong văn bản nhưng lại không có giá trị nhiều về ý nghĩa, ví dụ “là”, “thì”, “ở”. Do đó, bằng cách lấy tần suất tài liệu nghịch đảo, chúng ta có thể giảm thiểu trọng số của các thuật ngữ thường xuyên trong khi làm cho các thuật ngữ không thường xuyên có tác động cao hơn. (Simha, 2021)

- TF-IDF là chỉ thể hiện sự liên quan của từ đối với tập văn bản cụ thể. Chỉ số TF-IDF càng cao, từ càng có liên quan đối với tập văn bản cụ thể. Chỉ số này được tính bằng công thức (Nam, 2022):

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D)$$

#### *c, TF-IDF trong các thư viện thường dùng*

Trong bài nghiên cứu này, nhóm có tìm hiểu hai thư viện nổi tiếng hỗ trợ tính toán TF-IDF là Scikit learn và MLlib. Tuy công thức chuẩn của TF-IDF đã được trình bày như trên, nhưng vẫn có sự khác biệt trong cách tính TF-IDF giữa 2 thư viện lớn này.

Đối với Scikit learn, trọng số IDF được tính bằng công thức:

$$\text{idf}(t) = \log \frac{n}{\text{df}(t)} + 1$$

Trong khi đó, đối với thư viện Mlib, trọng số này được tính bằng công thức:

$$IDF(t, D) = \log \frac{|D| + 1}{DF(t, D) + 1},$$

Điều này gây ra một số khác biệt trong kết quả tính toán TF-IDF giữa Scikit learn và Mlib.

### 3.1.2 Mô hình áp dụng SVD

#### a, Giới thiệu chung

Phương pháp phân tích suy biến (Singular Value Decomposition), là một kỹ thuật đại số tuyến tính khá phổ biến trong lĩnh vực khoa học dữ liệu và học máy để phân tách một ma trận thành tích của 3 ma trận nhỏ hơn. Ban đầu, bản chất của phương pháp này sẽ tìm ra một phép xoay không gian sao cho tích vô hướng của các vector không thay đổi, để tìm ra những phép xoay đặc biệt, người ta đã sử dụng khái niệm về ma trận trực giao. Từ đó, phương pháp SVD đã được hình thành dựa trên tính chất của ma trận trực giao và đường chéo.

Phương pháp SVD là một kỹ thuật phân tích thừa số ma trận giúp giảm số lượng tính năng của tập dữ liệu bằng cách giảm kích thước không gian từ kích thước N thành K ( $K < N$ ). Dựa trên khoảng cách norm Frobenius, chúng ta sẽ tìm ra một lớp các ma trận xấp xỉ với ma trận đầu vào, lúc này ma trận đầu vào sẽ được phân tách thành 2 *ma trận trực giao* và 1 *ma trận đường chéo*. Thực chất, quá trình nhân ma trận sẽ biến đổi các điểm dữ liệu của ma trận đầu vào qua những phép thay đổi độ lớn và phép xoay quanh trục để tạo ra những điểm dữ liệu mới trong không gian mới. Sau đó, thông qua phép truncated SVD, những điểm dữ liệu này sẽ có thể giữ được hoàn toàn thông tin ban đầu hoặc giữ một phần lớn thông tin.

Để tăng độ chính xác cao cho việc dự đoán cũng như tiết kiệm thời gian và chi phí, chúng ta sẽ sử dụng phương pháp sắp xếp giá trị riêng giảm dần trên đường chéo chính, lúc này thuật toán SVD có thể tìm được ma trận xấp xỉ tốt và vẫn đảm bảo giảm được hạng và kích thước ma trận.

### b. Công thức

Trong thuật toán SVD, ma trận gốc được biểu diễn dưới dạng tích của 3 ma trận:

$$\mathbf{A}_{m \times n} = \mathbf{U}_{m \times m} \Sigma_{m \times n} (\mathbf{V}_{n \times n})^T$$

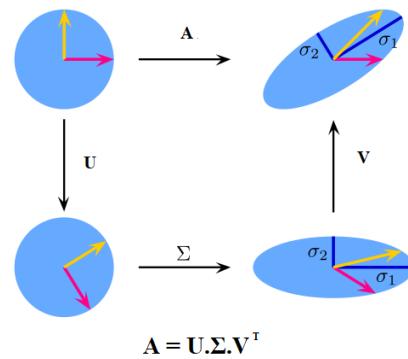
Trong đó:

- $\mathbf{U}, \mathbf{V}$  là ma trận trực giao
- $\Sigma$  là ma trận đường chéo không vuông

$$\begin{aligned} \mathbf{A}_{m \times n} &= \mathbf{U}_{m \times m} \times \begin{array}{c} \text{red squares} \\ \vdots \\ \text{white squares} \end{array} \Sigma_{m \times n} \times \mathbf{V}_{n \times n}^T \\ &\quad (m < n) \end{aligned}$$
  

$$\begin{aligned} \mathbf{A}_{m \times n} &= \mathbf{U}_{m \times m} \times \begin{array}{c} \text{red squares} \\ \vdots \\ \text{white squares} \end{array} \Sigma_{m \times n} \times \mathbf{V}_{n \times n}^T \\ &\quad (m > n) \end{aligned}$$

Mô tả SVD trong 2 trường hợp



Quá trình biến đổi hình học của phương pháp phân tích suy biến

### **3.2 Dữ liệu yêu cầu**

Dữ liệu dùng để khám phá và trực quan hóa bao gồm:

- Dữ liệu về người dùng: mã người dùng, tên người dùng
- Dữ liệu về Phim: mã số phim, tên phim, năm xuất bản, thể loại phim, mô tả nội dung phim, đạo diễn, diễn viên, rating phim, số lượt vote, điểm đánh giá Metacritic
- Dữ liệu về những bình luận đánh giá phim của người dùng: mã số phim, mã review, mã người dùng, tên người dùng, tiêu đề đánh giá, nội dung đánh giá, ngày review

Để thực hiện xây dựng mô hình đề xuất phim, cần có những dữ liệu sau:

- Dữ liệu yêu cầu của mô hình TF-IDF bao gồm mã phim, tên phim, thể loại (MovieID, MovieName, Genre)
- Dữ liệu yêu cầu của mô hình SVD bao gồm mã người dùng, mã phim, xếp hạng (UserId, MovieId, Rating)

### **3.3 Phương pháp thu thập dữ liệu**

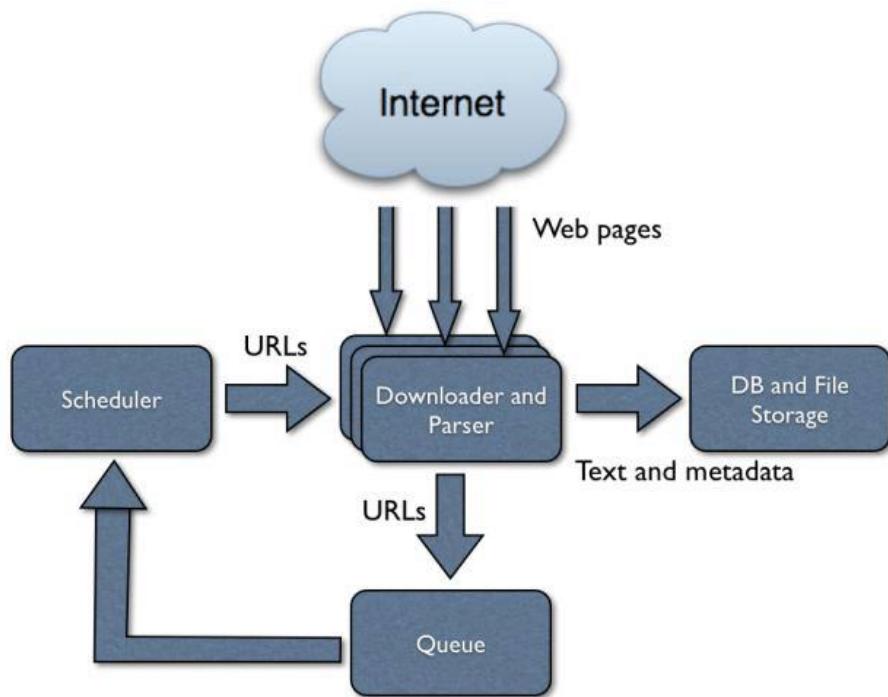
Trong đồ án này, Nhóm sẽ tiến hành thu thập dữ liệu web bằng cách sử dụng chức năng Web scraping/web crawler để lấy dữ liệu về từ các trang web.

Một web scraping/crawler là một chương trình máy tính có khả năng bóc tách và trích xuất thông tin chứa trên các web page một cách tự động, hỗ trợ cho quá trình thu thập dữ liệu trên các website. Trong đó, Việc thu thập dữ liệu web là hoạt động gửi một request HTTP/HTTPS trực tiếp hoặc dùng một browser driver truy cập đến trang đích để lấy nội dung trang về rồi tiến hành parse nội dung để lấy dữ liệu.

**Quy trình thu thập dữ liệu web:**

1. Cung cấp một hoặc nhiều URL có chứa các dữ liệu cần thu thập.

2. Chương trình sẽ load toàn bộ dữ liệu hoặc cấu trúc HTML trong trang được cung cấp.
3. Tiến hành bóc tách và trích xuất tất cả dữ liệu trên trang hoặc những dữ liệu cụ thể được người dùng chỉ định từ trước.
4. Thực hiện xuất ra và lưu xuống tất cả dữ liệu đã thu thập được thành các file có định dạng như sql, csv, json để lưu trữ hoặc để sử dụng sau.



*Quy trình thu thập dữ liệu web*

Trang web mà nhóm lựa chọn để thu thập dữ liệu là IMDB - Internet Movie Database (Cơ sở dữ liệu điện ảnh trên Internet). Đây là thư viện điện ảnh, nơi không chỉ cung cấp các thông tin về các bộ phim với đầy đủ các thể loại, chủ đề điện ảnh, truyền hình, chương trình TV mà còn tổng hợp những nhận xét, đánh giá, phê bình hay xếp hạng phim.

Sau khi nghiên cứu cấu trúc và logic hoạt động của trang web IMDB này, nhóm quyết định tiến hành thu thập dữ liệu bằng cách sử dụng thư viện BeautifulSoup và Framework Selenium của Python:

- BeautifulSoup: Là một thư viện Python dùng để lấy dữ liệu ra khỏi các file HTML và XML bằng cách duyệt qua DOM (mô hình đối tượng tài liệu)(Trung, 2020). Nó hoạt động cùng với các parser (trình phân tích cú pháp) cung cấp các cách để điều hướng, tìm kiếm và chỉnh sửa trong parse tree (cây cú pháp được tạo từ parser). Nhờ vậy mà BeautifulSoup dễ dàng biến một tài liệu HTML phức tạp thành một cây cú pháp mà ở đó ta thao tác với các object một cách dễ dàng và nhanh chóng.
- Selenium: Là một framework được thiết kế để tự động kiểm tra các ứng dụng web. Nó cung cấp một cách để nhà phát triển viết các bài kiểm tra bằng một số ngôn ngữ lập trình phổ biến như C#, Java, Python, Ruby, v.v. Các bài kiểm tra do nhà phát triển viết có thể sử dụng lại hầu hết các trình duyệt web như Chrome, IE và Firefox (Trung, 2020) .Quá trình thu thập dữ liệu bằng Selenium cũng tương tự như quá trình kiểm thử ứng dụng tự động bởi chúng đều thực hiện một chuỗi thao tác tương tác với các trang web một cách tự động và liên tục.

Trong đó, dữ liệu được hiển thị bởi các liên kết JavaScript có thể được cung cấp tự động hóa các lần nhấp vào nút, điều hướng trang tự động bằng cách sử dụng Selenium và sau đó có thể được trích xuất dữ liệu bằng BeautifulSoup.

## **4. Khám phá và trực quan hóa dữ liệu**

### **4.1 Quá trình khai phá**

#### **4.1.1. Xác định dữ liệu và các trang cần thu thập**

Đầu tiên, chúng tôi cần xác định những dữ liệu cần thu thập và các trang chứa dữ liệu cần thu thập có mối liên hệ với nhau hoặc có cùng một logic hoạt động ở trang web IMDB: <https://www.imdb.com>.

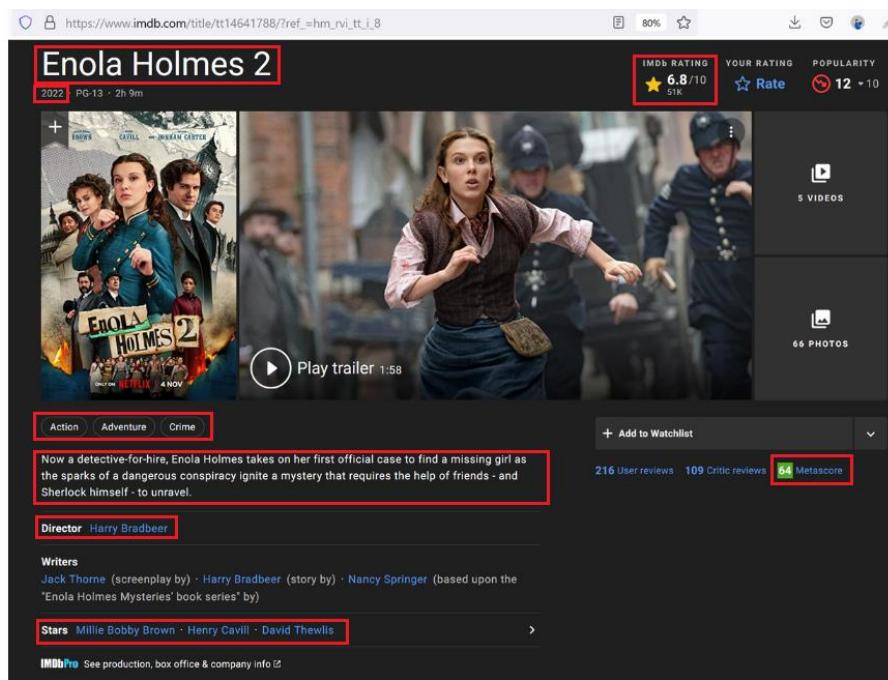
Trong đồ án này, chúng tôi cần các dữ liệu tổng quan về những bộ phim như mã số phim, tên phim, năm xuất bản, thể loại phim, mô tả nội dung phim, đạo diễn, diễn viên, rating phim, số lượt vote, điểm đánh giá Metacritic và các dữ liệu về những đánh giá, bình

luận, review của các người dùng như mã số phim, mã review, mã người dùng, tên người dùng, tiêu đề đánh giá, nội dung đánh giá, ngày review.

Với các dữ liệu tổng quan về những bộ phim, nhóm tìm thấy có 2 trang chứa những dữ liệu mà nhóm cần:

Một, trang chi tiết của từng phim, mỗi trang chứa thông tin dữ liệu tổng quan của một bộ phim, để truy cập cần một tập hợp mã số phim, mỗi lần cào chỉ được dữ liệu của một bộ phim. Như vậy thì hạn chế rất nhiều vì phải load nhiều trang, tốn nhiều thời gian, tài nguyên máy.

Hai, trang tìm kiếm phân loại phim, mỗi trang chứa thông tin dữ liệu tổng quan của 50 bộ phim, mỗi lần cào sẽ lấy được dữ liệu của 50 bộ phim. Như vậy, tuy vẫn có sự hạn chế nhất định nhưng đã load ít trang hơn, tiết kiệm được thời gian và tài nguyên máy hơn nên nhóm lựa chọn trang này để cào dữ liệu tổng quan về những bộ phim.



*Trang chi tiết từng phim*

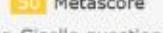
https://www.imdb.com/search/title/?genres=comedy

## Top 50 Comedy Movies and TV Shows

1-50 of 2,018,015 titles. [Next »](#)

View Mode: [Compact](#) | [Detailed](#)

Sort by: [Popularity](#) ▲ | [A-Z](#) | [User Rating](#) | [Number of Votes](#) | [US Box Office](#) | [Runtime](#) | [Year](#)  
[Release Date](#) | [Date of Your Rating](#) | [Your Rating](#)

	<b>1. <a href="#">The White Lotus (2021–2022)</a></b>	<a href="#">+</a>
TV-MA   60 min	Comedy, Drama	
 <b>7.7</b>	 Rate this	
Set in a tropical resort, it follows the exploits of various guests and employees over the span of a week.		
Stars: <a href="#">Jennifer Coolidge</a> , <a href="#">Jon Gries</a> , <a href="#">Eleonora Romandini</a> , <a href="#">Federico Ferrante</a>		
Votes: <b>80,787</b>		
	<b>2. <a href="#">The Menu (2022)</a></b>	<a href="#">+</a>
R   107 min	Comedy, Horror, Thriller	
 <b>7.5</b>	 Rate this	 <b>71</b> Metascore
A young couple travels to a remote island to eat at an exclusive restaurant where the chef has prepared a lavish menu, with some shocking surprises.		
Director: <a href="#">Mark Mylod</a>   Stars: <a href="#">Ralph Fiennes</a> , <a href="#">Anya Taylor-Joy</a> , <a href="#">Nicholas Hoult</a> , <a href="#">Hong Chau</a>		
Votes: <b>19,029</b>		
	<b>3. <a href="#">Disenchanted (2022)</a></b>	<a href="#">+</a>
PG   119 min	Animation, Adventure, Comedy	
 <b>5.8</b>	 Rate this	 <b>50</b> Metascore
Fifteen years after her happily ever after, Giselle questions her happiness, inadvertently turning the lives of those in the real world and Andalasia upside down in the process.		
Director: <a href="#">Adam Shankman</a>   Stars: <a href="#">Amy Adams</a> , <a href="#">Patrick Dempsey</a> , <a href="#">Maya Rudolph</a> , <a href="#">Gabriella Baldacchino</a>		
Votes: <b>10,616</b>		

Trang tìm kiếm phân loại phim

Với các dữ liệu về những đánh giá, bình luận, review của các người dùng, chỉ có một trang chứa các dữ liệu mà nhóm cần là trang review của từng phim, để truy cập cần có mã số phim của bộ phim đó..

**User Reviews**

**A movie about journalism at its best**

This is a very exciting and shocking movie. It is reminiscent of "Spotlight" (about the priests and children) and "All the President's Men" (about the break-in of a national campaign headquarters). I was on the edge of my seat right through the end (2 hours 9 min).

But it's not about the ending which everyone knows because of news media reports. This movie concerns the nuts and bolts of how investigative journalists were able to persuade terrified people to come forward, and the way the journalists acquired real evidence that could stand up in a court of law. The evidence they uncover is genuinely astounding (and mostly heartbreaking).

The two main actresses, Carey Mulligan and Zoe Kazan, were fantastic! This is a story that really needed to be told. We need to appreciate our free press and continue to try to keep it.

9 out of 12 found this helpful. Was this review helpful? Sign in to vote.

[Permalink](#)

**A thrilling, if not innovative, journalism film**

seamcoconnor92 | 17 October 2022

**User Lists**

- 2022 Coronado Island Film Festival
- 2022 Watchlist
- Kommende film
- Watching
- 2022 Most Anticipated

*Trang review của từng phim*

#### 4.1.2. Xác định cấu trúc logic URL

Thách thức của nhóm bây giờ là phải hiểu logic của URL để đảm bảo rằng tập hợp các trang nhóm muốn thu thập có cùng một logic và cách hoạt động.

**ON THE SCENE.**

What It Takes to Portray Wartime Pilots

Watch the 'Devotion' Cast Interview

**Up next**

- WHAT TO WATCH
- HOW WELL DO YOU KNOW JASON MOMOA
- Avatar: The Way of Water

[Browse trailers >](#)

*Trang chủ IMDB*

Từ trang chủ IMDB, nhóm có thể truy cập đến trang phân loại thể loại phim bằng cách search hoặc nhập vào từ khóa Genres. Từ đó, nhóm bắt đầu test thử trang thể loại phim Comedy bằng cách click vào thẻ loại Comedy có URL như sau: <https://www.imdb.com/search/title/?genres=comedy>

The screenshot shows the IMDB search results for comedy movies and TV shows. The title is "Top 50 Comedy Movies and TV Shows". It displays three results:

- 1. The White Lotus (2021–2022)**  
TV-MA | 60 min | Comedy, Drama  
7.7 Rate this Metascore 71  
Set in a tropical resort, it follows the exploits of various guests and employees over the span of a week.  
Stars: Jennifer Coolidge, Jon Gries, Eleonora Romandini, Federico Ferrante  
Votes: 80,772
- 2. The Menu (2022)**  
R | 107 min | Comedy, Horror, Thriller  
7.5 Rate this Metascore 71  
A young couple travels to a remote island to eat at an exclusive restaurant where the chef has prepared a lavish menu, with some shocking surprises.  
Director: Mark Mylod | Stars: Ralph Fiennes, Anya Taylor-Joy, Nicholas Hoult, Hong Chau  
Votes: 18,936
- 3. Disenchanted (2022)**  
PG | 119 min | Animation, Adventure, Comedy  
5.8 Rate this Metascore 71  
Fifteen years after her happily ever after, Giselle questions her happiness, inadvertently turning the lives of those in the real world and Andalasia upside down in the process.  
Director: Adam Shankman | Stars: Amy Adams, Patrick Dempsey, Maya Rudolph

### *Trang phân loại phim thể loại Comedy*

Trong URL, có tham số genres để chỉ hiển thị những bộ phim thuộc thể loại đó, kiểm tra logic của URL này bằng cách thử thay giá trị của tham số genres bằng thể loại khác như Action và thử thay tham số genres bằng release\_date với giá trị mặc định là 2022 để xem kết quả. Nhóm thu được kết quả là liên kết chuyển đến trang tìm kiếm phân loại phim theo năm xuất bản, có URL như sau: [https://www.imdb.com/search/title/?release\\_date=2022](https://www.imdb.com/search/title/?release_date=2022)

**Released between 2022-01-01 and 2022-12-31  
(Sorted by Popularity Ascending)**

View Mode: Compact | **Detailed**

Sort by: Popularity ▲ A-Z | User Rating | Number of Votes | US Box Office | Runtime | Year | Release Date | Date of Your Rating | Your Rating

- 1. Black Panther: Wakanda Forever** (2022)  
PG-13 | 161 min | Action, Adventure, Drama  
★ 7.3 Rate this 67 Metascore  
The people of Wakanda fight to protect their home from intervening world powers as they mourn the death of King T'Challa.  
Director: Ryan Coogler | Stars: Letitia Wright, Lupita Nyong'o, Danai Gurira, Winston Duke  
Votes: 95,832
- 2. 1899** (2022-)  
TV-MA | 60 min | Drama, History, Horror  
★ 7.8 Rate this  
Multinational immigrants traveling from the old continent to the new encounter a nightmarish riddle aboard a second ship adrift on the open sea.  
Stars: Emily Beecham, Aneurin Barnard, Andreas Pietschmann, Miguel Bernardeau  
Votes: 34,614
- 3. Andor** (2022-)  
TV-14 | 40 min | Action, Adventure, Drama  
★ 8.4 Rate this  
Prequel series to Star Wars' 'Rogue One'. In an era filled with danger, deception and intrigue, Cassian will embark on the path that is destined to turn him into a Rebel hero.  
Stars: Diego Luna, Kyle Soller, Stellan Skarsgård, Genevieve O'Reilly

### *Trang tìm kiếm phân loại phim theo năm xuất bản*

Tiếp tục kiểm tra logic URL điều hướng, chuyển tiếp trang bằng cách nhấp vào nút chuyển sang trang kế tiếp 2-3 lần, nhận thấy có sự xuất hiện của một tham số mới là start chỉ định số thứ tự của phim, vì mỗi trang có 50 phim => mỗi lần chuyển trang thì cộng giá trị tham số start thêm 50 đơn vị, có URL như sau:

[https://www.imdb.com/search/title/?release\\_date=2022&start=51](https://www.imdb.com/search/title/?release_date=2022&start=51)

**Released between 2022-01-01 and 2022-12-31  
(Sorted by Popularity Ascending)**

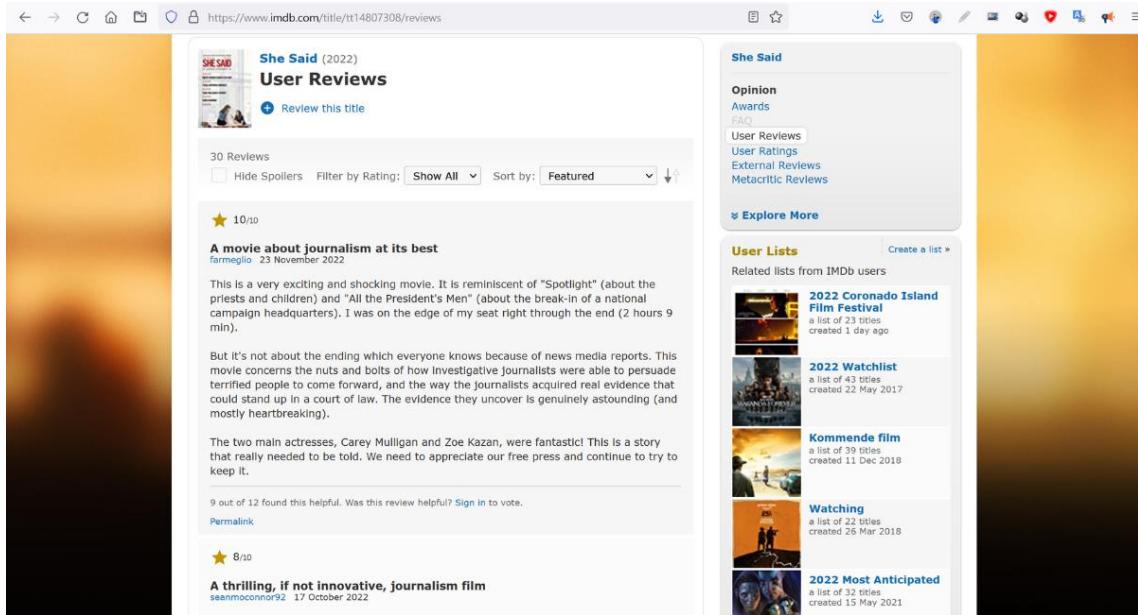
View Mode: Compact | **Detailed**

Sort by: Popularity ▲ A-Z | User Rating | Number of Votes | US Box Office | Runtime | Year | Release Date | Date of Your Rating | Your Rating

- 51. She Said** (2022)  
R | 129 min | Drama, History  
★ 7.4 Rate this 74 Metascore  
New York Times reporters Megan Twohey and Jodi Kantor break one of the most important stories in a generation - a story that helped ignite a movement and shattered decades of silence around the subject of sexual assault in Hollywood.  
Director: Maria Schrader | Stars: Carey Mulligan, Zoe Kazan, Patricia Clarkson, Andre Braugher  
Votes: 2,946
- 52. Wednesday** (2022-)  
TV-14 | 45 min | Comedy, Crime, Family  
★ 8.5 Rate this  
Follows Wednesday Addams' years as a student, when she attempts to master her emerging psychic ability, thwart and solve the mystery that embroiled her parents.  
Stars: Jenna Ortega, Gwendoline Christie, Riki Lindhome, Jamie McShane  
Votes: 23,944
- 53. From Scratch** (2022)  
TV-MA | 422 min | Drama  
★ 7.9 Rate this  
An American woman falls in love with a Sicilian man while studying abroad in Italy.  
Stars: Zoe Saldana, Eugenio Mastrandrea, Danielle Deadwyler, Judith Scott  
Votes: 9,107

### *Trang kế tiếp của trang tìm kiếm phân loại phim theo năm xuất bản*

Đến trang review của từng phim, có URL: <https://www.imdb.com/title/tt14807308/reviews> có logic của URL là mã số phim + từ khóa reviews. Đặc biệt ở trang review này logic chuyển tiếp trang không theo tham số start hay địa chỉ url, href mà phải nhấp vào nút load more để load thêm dữ liệu review thăng trên trang đó.



### Trang review của từng phim

```

<div class="load-more-data" data-target="reviews-container">
  <div data-ajaxurl="/title/tt14807308/reviews/_ajax">
    <div class="ipl-load-more ipl-load-more--loaded">
      <div class="ipl-load-more__load-indicator" data-target="load-more-indicator"></div>
      <div class="ipl-message-box ipl-message-box--alert ipl-load-more__message-box ipl-load-more__message-box--hidden" data-target="load-more-message">
        <button id="load-more-trigger" class="ipl-load-more__button" data-target="load-more-trigger">
          <span>Load More</span>
        </button>
      </div>
    </div>
  </div>
</div>

```

### Load thêm dữ liệu trang review

### 4.1.3. Xác định cấu trúc HTML của trang

Vì tất cả các trang mà nhóm muốn cào có cùng một logic và cấu trúc tổng thể giống nhau nên nhóm chỉ cần xác định được cấu trúc HTML của một trang là có thể suy ra các trang còn lại.

The screenshot shows the IMDb movie details page for "Kong: Skull Island (2017)". A context menu is open over the movie's title, listing options like "Open link in new tab", "Save link as...", and "Inspect". An arrow points from the "Inspect" option to the developer tools window. The developer tools show the HTML structure for the movie item, including the header, content, and footer sections. The "Elements" tab is selected, displaying the DOM tree with various CSS classes and IDs.

The screenshot shows the IMDb "Most Voted Titles Released 2017-01-01 to 2017-12-31" page. A context menu is open over the title "1. Logan (2017)", listing options like "View Mode: Compact | Details". An arrow points from the "Details" option to the developer tools window. The developer tools show the HTML structure for the movie item, including the header, content, and footer sections. The "Elements" tab is selected, displaying the DOM tree with various CSS classes and IDs.

IMDb

Find Movies, TV shows, Celebrities and more...

Movies, TV & Showtimes | Celebs, Events & Photos | News & Community | Watchlist

**Most Voted Titles Released 2017-01-01 to 2017-12-31**

1 to 50 of 117,071 titles | Next »

View Mode: Compact | Detail

Sort by: Popularity | Alphabetical | IMDb Rating | **Number of Votes ▾** | US Box Office | Runtime | Year | Release Date

**1. Logan (2017)**

R | 15.56 x 12 mina, Sci-Fi

Rate this  77 Metascore

In the near future, a weary Logan cares for an ailing Professor X somewhere on the Mexican border. However, Logan's attempts to hide from the world and his legacy are upended when a young mutant arrives, pursued by dark forces.

Director: James Mangold | Stars: Hugh Jackman, Patrick Stewart, Dafne Keen, Boyd Holbrook

Votes: 309,245 | Gross: \$226.23M

Elements Console Sources Network Performance » 0 / 1 : x

... div div.lister-item mode-advanced div.lister-item-content h3.lister-item-header a

Styles Event Listeners DOM Breakpoints Properties

Filter element.style { }

.lister-item-mode-advanced:nth-child(odd), .lister-item-mode-advanced:nth-child(odd), .lister-item-mode-detail:nth-child(odd), .lister-item-mode-simple:nth-child(odd), .lister-item-mode-simple:nth-child(odd) { background-color: #f6f6f6; }

position: 0 margin: - padding: 15 border: 2px solid #f6f6f6; width: 580px; height: 174px; top: 0; left: 0; z-index: 10; opacity: 0.8; border-radius: 10px; box-sizing: border-box; border: none; background-color: white; box-shadow: 0 0 10px 2px #f6f6f6; }

Show all

```

▼<div class="lister-list">
  ▷<div class="lister-item mode-advanced"> ← 1st container
    ▷<div class="lister-top-right">...</div> ← 1st div
    ▷<div class="lister-item-image float-left">...</div> ← 2nd div
    ▷<div class="lister-item-content"> ← 3rd div
      ▷<h3 class="lister-item-header"> ← <h3>
        <span class="lister-item-index unbold text-primary">1.</span>
        <a href="/title/tt3315342/?ref=adv_li_tt"> ← <a>
          Logan</a> == $0
        <span class="lister-item-year text-muted unbold">(2017)</span>
      </h3>
      ▷<p class="text-muted ">...</p>
      ▷<div class="ratings-bar">...</div>
      ▷<p class="text-muted">...</p>
      ▷<p class="sort-num_votes-visible">...</p>
    </div>
    ▷<div class="lister-item mode-advanced">...</div>
    ▷<div class="lister-item mode-advanced">...</div>
    ▷<div class="lister-item mode-advanced">...</div>
    ▷<div class="lister-item mode-advanced">...</div>
  </div>
  | The other movie containers
  |

```

The <h3> from before

```

  ▷<div class="lister-item-content">
    ▷<h3 class="lister-item-header">...</h3>
    ▷<p class="text-muted ">...</p>
    ▷<div class="ratings-bar">
      ▷<div class="inline-block ratings-imdb-rating" name="ir" data-value="8.3">
        <span class="globalSprite rating-star imdb-rating"></span>
      <strong>8.3</strong> == $0
    </div>
    ▷<div class="inline-block ratings-user-rating">...</div>
    ▷<div class="inline-block ratings-metacore">...</div>
    ▷<p class="text-muted">...</p>
  </div>

```

Cấu trúc HTML trang movie

#### 4.1.4. Thực thi thu thập dữ liệu

##### a. Cài đặt thư viện

```
#pip install selenium
#pip install webdriver-manager
#pip install findspark
from selenium import webdriver
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By
from time import sleep
import requests
import re
from bs4 import BeautifulSoup
import urllib.request
import pandas as pd
import csv
from selenium.webdriver.support.ui import WebDriverWait

import findspark
findspark.init()

import pyspark
from pyspark.sql import SparkSession, SQLContext, functions
from pyspark.sql.functions import *
from pyspark.sql.types import *

spark = SparkSession.builder.appName("MovieRecommender").getOrCreate()
```

*Code nhập thư viện*

##### b. Thực hiện cào dữ liệu list movie

- . Bản thử nghiệm

```

movies=[]
years_url = [str(i) for i in range(2002,2005)]
pages = [str(i) for i in range(1, 100, 50)]
# For every year in the interval 2002-2004
for year_url in years_url:
    # For every page in the interval 1-2 page, 50 movies per each page
    for page in pages:
        try:
            # Make a get request
            url = "https://www.imdb.com/search/title/?release_date=" +
year_url + "&start=" + page
            page = urllib.request.urlopen(url)
            page_source = BeautifulSoup(page, "html.parser")
            driver = webdriver.Chrome(ChromeDriverManager().install())
            driver.get(url)
            # Wait 1s to load
            driver.implicitly_wait(1)
            # For every movie of these 50
            for item in page_source.find_all(class_='lister-item
mode-advanced'):
                # extract Data
                try:
                    movieid = item.find(class_='ribbonize')
                    movieid = movieid['data-tconst']
                except:
                    movieid = "NaN"
                try:
                    name = item.find('h3', {'class':'lister-item-header'})
                    name = name.find('a').get_text(strip=True)
                except:
                    name = "NaN"
                try:
                    year = item.find(class_='lister-item-year text-muted
unbold').get_text(strip=True)
                except:
                    year = "NaN"

```

```

try:
    genre = item.find(class_='genre').get_text(strip=True)
except:
    genre = "NaN"
try:
    content = item.select('p')[1].get_text(strip=True)
except:
    content = "NaN"
try:
    rating = item.find(class_='inline-block
ratings-imdb-rating').get_text(strip=True)
except:
    rating = "NaN"
try:
    score = item.find('span',
{'class':'metascore'}).get_text(strip=True)
except:
    score = "NaN"
try:
    star = item.find('p',
{'class':''}).get_text(strip=True)
except:
    star = "NaN"
try:
    votes = item.find('span', attrs = {'name':'nv'})
    votes = votes['data-value']
except:
    vote = "NaN"
movies.append({'MovieID' : movieid,
               'MovieName' : name,
               'Year' : year,
               'Genre': genre,
               'Content': content,
               'Star' : star,
               'Rating' : rating,
               'Votes' : votes,
               'Metascore' : score})
except:
    pass

```

```

#print & Create file csv using Pandas
dataframe = pd.DataFrame(data=movies)
print(dataframe)
dataframe.to_csv('Movie/'f'PandasMovieTest.csv')
#print & Create file csv using Spark
schema = StructType([ \
    StructField('MovieID',StringType(),True), \
    StructField('MovieName',StringType(),True), \
    StructField('Year',StringType(),True), \
    StructField('Genre',StringType(),True), \
    StructField('Content',StringType(),True), \
    StructField('Star',StringType(),True), \
    StructField('Rating',StringType(),True), \
    StructField('Votes',StringType(),True), \
    StructField('Metascore', StringType(), True)])
moviedata=spark.createDataFrame(data=movies,schema=schema)
moviedata.show()
moviedata.write.option("header",True).option("delimiter","|").csv('Movie/'f
'SparkMovieTest.csv')

```

*Code thu thập dữ liệu test movies*

## Bản đầy đủ

```
movies=[]
years_url = [str(i) for i in range(2002,2023)]
pages = [str(i) for i in range(1, 10000, 50)]
# For every year in the interval 2002-2022
for year_url in years_url:
    # For every page in the interval 1-200 page, 50 movies per each page
    for page in pages:
        try:
            # Make a get request
            url = "https://www.imdb.com/search/title/?release_date=" +
year_url + "&start=" + page
            page = urllib.request.urlopen(url)
            page_source = BeautifulSoup(page, "html.parser")
            driver = webdriver.Chrome(ChromeDriverManager().install())
            driver.get(url)

            # Wait 1s to load
            driver.implicitly_wait(1)
            # For every movie of these 50
            for item in page_source.find_all(class_='lister-item mode-advanced'):
                # extract Data
                try:
                    movieid = item.find(class_='ribbonize')
                    movieid = movieid['data-tconst']
                except:
                    movieid = "NaN"
                try:
                    name = item.find('h3', {'class':'lister-item-header'})
                    name = name.find('a').get_text(strip=True)
                except:
                    name = "NaN"
                try:
                    year = item.find(class_='lister-item-year text-muted unbold').get_text(strip=True)
                except:
                    year = "NaN"
                try:
                    genre = item.find(class_='genre').get_text(strip=True)
                except:
                    genre = "NaN"
```

```

try:
    content = item.select('p')[1].get_text(strip=True)
except:
    content = "NaN"
try:
    rating = item.find(class_='inline-block
ratings-imdb-rating').get_text(strip=True)
except:
    rating = "NaN"
try:
    score = item.find('span',
{'class':'metascore'}).get_text(strip=True)
except:
    score = "NaN"
try:
    star = item.find('p',
{'class':''}).get_text(strip=True)
except:
    star = "NaN"
try:
    votes = item.find('span', attrs = {'name':'nv'})
    votes = votes['data-value']
except:
    vote = "NaN"
movies.append({'MovieID' : movieid,
               'MovieName' : name,
               'Year' : year,
               'Genre': genre,
               'Content': content,
               'Star' : star,
               'Rating' : rating,
               'Votes' : votes,
               'Metascore' : score})
except:
    pass

```

```

#print & Create file csv using Pandas
dataframe = pd.DataFrame(data=movies)
print(dataframe)
dataframe.to_csv('Movie/'f'PandasMovie.csv')
#print & Create file csv using Spark
schema = StructType([ \
    StructField('MovieID',StringType(),True), \
    StructField('MovieName',StringType(),True), \
    StructField('Year',StringType(),True), \
    StructField('Genre',StringType(),True), \
    StructField('Content',StringType(),True), \
    StructField('Star',StringType(),True), \
    StructField('Rating',StringType(),True), \
    StructField('Votes',StringType(),True), \
    StructField('Metascore', StringType(), True)])
moviedata=spark.createDataFrame(data=movies,schema=schema)
moviedata.show()
moviedata.write.option("header",True).option("delimiter","|").csv('Movie/'f
'SparkMovie.csv')

```

*Code thu thập dữ liệu full movies*

## 4.2. Thực hiện cào dữ liệu review movie

### a, Bản thử nghiệm

```
def get_review():
    data = ['tt13406094', 'tt6226232', 'tt2935622']
    # Create the pandas DataFrame
    moviedata = pd.DataFrame(data, columns=['movieid'])
    #After the webpage opened, we can extract data of each movies
    #Set initial empty list for each element:
    for i in range(len(moviedata['movieid'])):
        driver = webdriver.Chrome(ChromeDriverManager().install())
        url = "https://www.imdb.com/title/" + moviedata['movieid'][i] +
        "/reviews?"
        driver.get(url)
        driver.implicitly_wait(3) # tell the webdriver to wait for 3
        seconds for the page to load to prevent blocked by anti spam software
        # Set up action to click on 'load more' button
        # note that each page on imdb has 25 reviews
        page = 1 #Set initial variable for while loop
        #Test at least 5 pages
        while page<5:
            try:
                #find the load more button on the webpage
                load_more = driver.find_element(By.ID, 'load-more-trigger')
                #click on that button
                load_more.click()
                page+=1 #move on to next loadmore button
            except:
                #If couldnt find any button to click, stop
                break
        # After fully expand the page, we will grab data from whole website
        review = driver.find_elements(By.CLASS_NAME, 'lister-item')
        #Set list for each element:
        title = []
        review_id = []
        content = []
        rate = []
        date = []
        user_id = []
        user_name = []
        #run for loop to get
        for n in range(0,50):
            try:
                #Grab each element before append them to the list
                reviewtitle = review[n].find_element(By.CLASS_NAME,
                'title').text
```

```

        reviewcontent = review[n].find_element(By.CLASS_NAME,
'content').get_attribute("textContent")
        reviewrate = review[n].find_element(By.CLASS_NAME,
'rating-other-user-rating').text
        reviewdate = review[n].find_element(By.CLASS_NAME,
'review-date').text
        userid = review[n].find_element(By.CLASS_NAME,
'display-name-link').get_attribute("innerHTML")
        username = review[n].find_element(By.CLASS_NAME,
'display-name-link').text
        reviewid = review[n].get_attribute('data-review-id')

        #Then add them to list
        review_id.append(reviewid)
        user_id.append(userid)
        title.append(reviewtitle)
        content.append(reviewcontent)
        rate.append(reviewrate)
        date.append(reviewdate)
        user_name.append(username)

    except:
        continue

#Build data dictionary for dataframe
reviewdata = {'Review_ID' : review_id,
              'User_ID' : user_id,
              'User_name' : user_name,
              'Review_title' : title,
              'Review_rate' : rate,
              'Review_date' : date,
              'Review_body' : content}

#Build dataframe for each review movie to export
review = pd.DataFrame(data = reviewdata)
movie = moviedata['movieid'][i]
#create new column with the movie id value
review['movieid'] = movie
#print & Create file csv
print(review)
review.to_csv('Review/Test/' + f'review{movie}.csv')
driver.quit()

if __name__ == "__main__": get_review()

```

*Code*

*thu thập dữ liệu test reviews*

## b. Bán đầy đủ

```
def get_review():
    data =
    spark.read.option("header",True).option("sep","|").csv("moviedata.csv")
    moviedata = data.toPandas()
    #After the webpage opened, we can extract data of each movies
    #Set initial empty list for each element:
    for i in range(len(moviedata['MovieID'])):
        driver = webdriver.Chrome(ChromeDriverManager().install())
        url = "https://www.imdb.com/title/" + moviedata['MovieID'][i] +
        "/reviews?"
        driver.get(url)
        # let webdriver wait for 3 seconds for the page to load
        driver.implicitly_wait(3)
        # Set up action to click on 'load more' button
        page = 1
        # 0-1000 review => 40 pages
        while page<=40:
            try:
                #find the load more button on the webpage
                load_more = driver.find_element(By.ID, 'load-more-trigger')
                #click on that button
                load_more.click()
                #move on to next loadmore button
                page+=1
            except:
                #If couldnt find any button to click, stop
                break
        # After fully expand the page, grab data from whole website
        review = driver.find_elements(By.CLASS_NAME, 'lister-item')
        #Set list for each element:
        title = []
        review_id = []
        content = []
        rate = []
        date = []
        user_id = []
        user_name = []
        #run for loop to get
        for n in range(0,1000):
            try:
                #Grab each element before append them to the list
                reviewtitle = review[n].find_element(By.CLASS_NAME,
                'title').text
```

```

reviewcontent = review[n].find_element(By.CLASS_NAME,
'content').get_attribute("textContent")
reviewrate = review[n].find_element(By.CLASS_NAME,
'rating-other-user-rating').text
reviewdate = review[n].find_element(By.CLASS_NAME,
'review-date').text
userid = review[n].find_element(By.CLASS_NAME,
'display-name-link').get_attribute("innerHTML")
username = review[n].find_element(By.CLASS_NAME,
'display-name-link').text
reviewid = review[n].get_attribute('data-review-id')

#Then add them to the list
review_id.append(reviewid)
user_id.append(userid)
title.append(reviewtitle)
content.append(reviewcontent)
rate.append(reviewrate)
date.append(reviewdate)
user_name.append(username)

except:
    continue

reviewdata = {'Review_ID' : review_id,
              'User_ID' : user_id,
              'User_name' : user_name,
              'Review_title' : title,
              'Review_rate' : rate,
              'Review_date' : date,
              'Review_body' : content}
#Create dataframe for each review movie to export
review = pd.DataFrame(data = reviewdata)
movie = moviedata['MovieID'][i]
#Create new column with the movie id value
review['MovieID'] = movie
#print & Create file csv
print(review)
review.to_csv('Review/f' + review[move].csv')
driver.quit()

if __name__ == "__main__":
    get_review()

```

*Code thu thập dữ liệu full reviews*

### 4.3. Kết quả thu thập dữ liệu

Với thông tin tổng quan về những bộ phim, nhóm đã thu thập dữ liệu được hơn 200.000 bộ phim trong khoảng thời gian từ năm 2002 đến 2022, cụ thể là 209.995 bộ phim với 209.995 dòng dữ liệu, 9 cột thuộc tính bao gồm MovieID, MovieName, Year, Genre, Content, Star, Rating, Votes, Metascore.

```
/tmp/ipykernel_19988/1794441476.py:10: DeprecationWarning: executable_path has been deprecated,  
please pass in a Service object  
    driver = webdriver.Chrome(ChromeDriverManager().install())  
[WDM] - Downloading: 100%|██████████| 6.35M/6.35M [00:00<00:00, 10.9MB/s]  
  
Output exceeds the size limit. Open the full output data in a text editor  
      MovieID          MovieName      Year  \  
0     tt0306414       Đường Dây Tội Phạm (2002-2008)  
1     tt0145487       Người Nhện   (2002)  
2     tt0304669       Ông Già Tuyết 2 (2002)  
3     tt0295297       Harry Potter và Phòng Chứa Bí Mật (2002)  
4     tt0312172       Monk      (2002-2009)  
...     ...           ...        ...  
209990 tt13099434      NGFL: Nice Guys Finish Last (2022)  
209991 tt8811670       The Library Boys (2022)  
209992 tt15226984      Once upon a time in Serbia (2022)  
209993 tt20159188      Elektra, My Love (2022)  
209994 tt13323300      Army Baby   (2022)  
  
      Genre  \  
0     Crime, Drama, Thriller  
1     Action, Adventure, Sci-Fi  
2     Comedy, Family, Fantasy  
3     Adventure, Family, Fantasy  
4     Comedy, Crime, Drama  
...     ...  
209990       Comedy  
209991       Comedy  
209992       Comedy  
209993       Drama  
209994       Comedy  
...  
209994     NaN  
  
[209995 rows x 9 columns]  
22/11/26 02:06:15 WARN TaskSetManager: Stage 0 contains a task of very large size (29989 KiB). The  
maximum recommended task size is 1000 KiB.
```

MovieID	MovieName	Year	Genre	Content	Star	Rating	Votes	Metascore
tt0306414	Đường Dây Tội Phạm	(2002-2008)	Crime, Drama, Thr...	The Baltimore dru... Stars:Dominic Wes...	9.3	339003	NaN	
tt0145487	Người Nhện	(2002)	Action, Adventure...	After being bitte... Director:Sam Raim...	7.4	815798	73	
tt0304669	Ông Già Tuyệt 2	(2002)	Comedy, Family, F...	Scott Calvin has ... Director:Michael ...	5.7	57807	48	
tt0295297	Harry Potter và P...	(2002)	Adventure, Family...	An ancient prophe... Director:Chris Co...	7.4	633363	63	
tt0312172	Monk	(2002-2009)	Comedy, Crime, Drama	The series follow... Stars:Tony Shalho...	8.0	75953	NaN	
tt0317248	Thành Phố Của Chúa	(2002)	Crime, Drama	In the slums of R... Directors:Fernand...	8.6	755016	79	
tt0274812	Secretary	(2002)	Comedy, Drama, Ro...	A young woman, re... Director:Steven S...	6.9	92211	63	
tt0303461	Tàu Đom Đóm	(2002-2003)	Adventure, Drama...	Five hundred year... Stars:Nathan Fill...	9.0	265736	NaN	
tt0264464	Hay Bất Tối Nêu C...	(2002)	Biography, Crime,...	Barely 21 yet, Fr... Director:Steven S...	8.1	976546	75	
tt0167261	Chúa Tể Của Những...	(2002)	Action, Adventure...	While Frodo and S... Director:Peter Ja...	8.8	1658555	87	
tt0310455	Foyle's War	(2002-2015)	Crime, Drama, Mys...	As WWII rages, DC... Stars:Michael Kit...	8.6	15770	NaN	
tt0267913	Scooby-Doo: Chú C...	(2002)	Adventure, Comedy...	After an acrimoni... Director:Raja Gos...	5.2	114822	35	
tt0253474	Nghệ Sĩ Dương Cầm	(2002)	Biography, Drama,...	A Polish Jewish m... Director:Roman Po...	8.5	828468	85	
tt0409591	Naruto	(2002-2007)	Animation, Action...	Naruto Uzumaki, a... Stars:Junko Takeu...	8.4	102177	NaN	
tt0290673	Irréversible	(2002)	Crime, Drama, Mys...	Events over the c... Director:Gaspar N...	7.3	136034	51	
tt0289043	28 Ngày Sau	(2002)	Drama, Horror, Sc...	Four weeks after ... Director:Danny Bo...	7.5	415834	73	
tt0298203	8 Mile	(2002)	Drama, Music	A young rapper, s... Director:Curtis H...	7.2	291060	77	
tt0313043	CSI: Miami	(2002-2012)	Action, Crime, Drama	Follows the cases... Stars:David Carus...	6.4	57920	NaN	
tt0256415	Quê Nhà Alabama	(2002)	Comedy, Romance	A young woman who... Director:Andy Ten...	6.2	114872	45	
tt0250797	Ngoại Tình	(2002)	Drama, Romance, T...	A New York suburb... Director:Adrian L...	6.7	90609	63	

only showing top 20 rows

### Kết quả thu thập dữ liệu movie

Với dữ liệu review của các bộ phim, nhóm thu thập được review của 1698 bộ phim, mỗi bộ phim sẽ có từ 0-1000 reviews tương đương với số dòng dữ liệu. Tổng kết, thu được 1698 file review phim tương ứng với tổng cộng 808761 dòng dữ liệu reviews.

Review_ID	User_ID
0 rw8673424	< a href="/user/ur44965663/?ref_=tt_urv">Drawmo...
1 rw8668903	< a href="/user/ur62070774/?ref_=tt_urv">danny...
2 rw8668496	< a href="/user/ur26226712/?ref_=tt_urv">Jeremy...
3 rw8668856	< a href="/user/ur23240045/?ref_=tt_urv">kjproul...
4 rw8666142	< a href="/user/ur18374284/?ref_=tt_urv">MrDHWo...
.. ..	...
698 rw8673845	< a href="/user/ur118451636/?ref_=tt_urv">iwish...
699 rw8684679	< a href="/user/ur37607845/?ref_=tt_urv">nonimus...
700 rw8679260	< a href="/user/ur40051567/?ref_=tt_urv">childsey...
701 rw8685895	< a href="/user/ur159053181/?ref_=tt_urv">yahya...
702 rw8687575	< a href="/user/ur0688704/?ref_=tt_urv">drz</a>
User_name	Review_title
0 Drawmort	Not what I expected...
1 dannylee-78082	I Yield
2 Jeremy_Urquhart	I liked it a lot
3 kjproulx	A Fantastic, Emotional, and Mature Marvel Film
4 MrDHWong	Pays an appropriately sombre tribute to its re...
.. ..	...
698 iwichihadabird	A shell of the former movie
699 nonimus81-524-891052	Mediocre
700 childsey01	Mr. Bean's first 3D camera

```

                    Review_ID          User_ID \
0    rw6661743  <a href="/user/ur6387867/?ref_=tt_urv">Superma...
1    rw3855396  <a href="/user/ur20552756/?ref_=tt_urv">TheLit...
2    rw4163022  <a href="/user/ur25192034/?ref_=tt_urv">skybri...
3    rw2412620  <a href="/user/ur15311310/?ref_=tt_urv">Sleepi...
4    rw7026611  <a href="/user/ur115572916/?ref_=tt_urv">balde...
..      ...
956   rw4498324  <a href="/user/ur91508196/?ref_=tt_urv">Challe...
957   rw5114831  <a href="/user/ur56504966/?ref_=tt_urv">athan...
958   rw5526055  <a href="/user/ur38118921/?ref_=tt_urv">pcgnes...
959   rw8330368  <a href="/user/ur24398121/?ref_=tt_urv">baka_l...
960   rw4374901  <a href="/user/ur2127796/?ref_=tt_urv">pejoar</a>

                    User_name          Review_title \
0      Supermanfan-13  Going to miss it but it's prob time for it to go!
1      TheLittleSongbird  Very much alive for Seasons 1-5, Season 7 dead
2        skybrick736  The Walking Dead: Season 1 (10/10)
3      Sleepin_Dragon  Utterly surprised
4       balder777  Amazing if you binge-watch.
..      ...
956       Challedon  Bored to death (II)
957  athanatou1996  My favorite show forever
958       pcgness  Reader of the comics. Disappointed with their ...
959       baka_land  Watch first 5 seasons and then stop, not kidding
960       pejoar  ten seasons would be enough
..      ...
959 \n      The first 5 seasons captured... tt1520211
960 \n      I like this show, but the la... tt1520211

```

[961 rows x 8 columns]

```

                    Review_date          Review_body \
0      27 January 2019  \n  (Review updated after Season...
1      15 October 2022  \n  Peaky Blinders is a tale tha...
2      11 August 2021  \n  Peaky Blinders is not only o...
3      29 September 2022  \n  Peaky Blinders is not only o...
4      24 May 2020  \n  One of the very well made br...
..      ...
808771  15 February 2012  \n  In my opinion, the creators ...
808772  26 February 2012  \n  I think this was quite a dec...
808773  27 April 2012  \n  The movie starts really slow...
808774  10 May 2013  \n  entertaining nice but it cou...
808775  27 August 2021  \n  I almost never write reviews...
                    MovieID
0      tt2442560
1      tt2442560
2      tt2442560
3      tt2442560
4      tt2442560
..      ...
808771  tt1706593
808772  tt1706593
808773  tt1706593
808774  tt1706593
808775  tt1706593

[808776 rows x 9 columns]
808776

```

*Kết quả thu thập dữ liệu review*

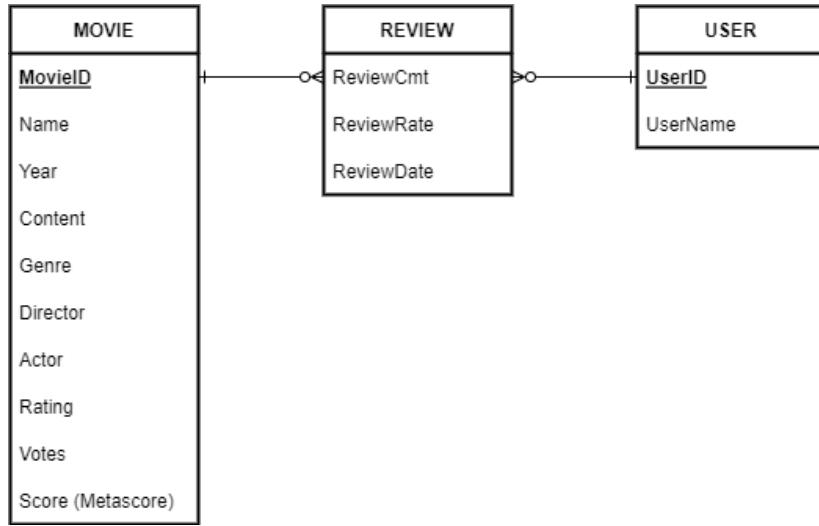
#### 4.4 Mô tả dữ liệu khai phá

Dữ liệu của bài nghiên cứu này gồm 3 trường dữ liệu chính gồm: Movie, Review, User. Được thu thập bằng cách cào dữ liệu trên trang web IMDB - Internet Movie Database (Cơ sở dữ liệu điện ảnh trên Internet). Sau đây là mô tả thuộc tính của các trường dữ liệu:

MOVIE			
Attribute	Data type	Key	Null?
MovieID		PK	
Name			
Year			Yes
Content			
Genre			
Director			Yes
Actor			Yes
Rating			
Votes			
Score (Metascore)			Yes

REVIEW			
Attribute	Data type	Key	Null?
MovieID		FK	
UserID		FK	
ReviewCmt			
ReviewRate			
ReviewDate			

USER			
Attribute	Data type	Key	Null?
UserID		PK	
UserName			



*Sơ đồ ERD mô tả mối quan hệ giữa các trường dữ liệu*

Dữ liệu khai phá bao gồm:

- 209.995 bộ phim trong khoảng thời gian từ 2002 tới 2022 với 209.995 dòng dữ liệu, 9 cột thuộc tính bao gồm MovieID, MovieName, Year, Genre, Content, Star, Rating, Votes, Metascore.
- review của 1698 bộ phim, mỗi bộ phim sẽ có từ 0-1000 reviews tương đương với số dòng dữ liệu.

#### 4.5 Tiết xu lý dữ liệu

- *Loại bỏ giá trị null ở cột Rating, Votes và MovieID*

```

# Drop null values in Rating, Votes, MovieID column
movie1 = moviedata.na.drop(subset=["Rating","Votes","MovieID"])
movie1.show()
print(movie1.count())
    
```

*Code loại bỏ giá trị null ở cột Rating, Votes và MovieID*

## Kết quả:

MovieID	MovieName	Year	Genre	Content	Star	Rating	Votes
NaN	Black Tree Forest...	(2012)	Horror	Add a Plot Director:Dustin F...	2.5	23	
NaN	A Night at the Si...	(2012)	Comedy	A raucous comedy ... Director:Tim Russ...	7.6	21	
NaN	The Daniel Project	(2012 TV Movie)	Documentary	The Daniel Projec... Director:Stewart ...	5.0	166	
NaN	No te enamores de mí	(2012)	Comedy, Drama, Ro...	A collection of s... Director:Federico...	6.1	126	
NaN	Andamio	(2012)	Short, Comedy, Drama	Eduardo, a snob p... Director:Juanna C...	6.7	91	
68	They Call It Myan...	(2012)	Documentary, Hist...	shot clandestinel... Director:Robert H...	7.0	311	
NaN	Secret Eaters	(2012- )	Reality-TV	Secret Eaters put... Stars:Anna Richar...	6.8	129	
NaN	Passenger: Let He...	(2012 Music Video)	Music	Music video for "... Directors:Dave Je...	7.0	29	
NaN	Bloopers	(2012- )	Comedy	Bloopers is a loo... Stars:Dean Cain,J...	5.3	69	
NaN	Combat Countdown	(2012- )	Documentary	Series showcases ... Stars:Mike New,Mi...	6.4	36	
NaN	Dr. Fubalous	(2012- )	Comedy, Musical	Dr. Fubalous and ... Stars:Donnivin Jo...	5.8	52	
NaN	The Luckiest Man ...	(2012)	Short, Comedy	Everything that c... Director:Matthew ...	6.5	6	
NaN	Airport 24/7: Miami	(2012- )	Reality-TV	Features a behind... Stars:Lauren Stov...	7.2	102	
NaN	Loon Lake	(2012)	Drama, Family, Th...	Madeleine and Ros... Director:Mary Sel...	7.4	9	
NaN	A Thousand Cuts	(2012)	Thriller	A stranger with a... Director:Charles ...	4.5	814	

209995

## Kết quả sau khi loại bỏ

Sau khi loại bỏ những dòng có giá trị null ở 3 cột Rating, MovieID và Votes thì dữ liệu còn lại 209995 dòng.

- Lọc ra những dữ liệu không có giá trị NaN trong cột Rating, Votes, MovieID

```
# Filter out data without NaN value in Rating, Votes, MovieID column
movie2 = movie1.filter(movie1.MovieID != 'NaN')
movie3 = movie2.filter(movie2.Rating != 'NaN')
movie4 = movie3.filter(movie3.Votes != 'NaN')
movie5 = movie4.distinct()
print(movie5.count())
movie5.show()
```

181469

Code lọc ra dữ liệu không có giá trị NaN trong cột Rating, Votes, MovieID

## Kết quả:

MovieID	MovieName	Year	Genre	Content	Star	Rating	Votes
Metascore							
tt6796652	Christina Aguilera... (2012 Music Video)		Music music video for "... Director:Melina M...	7.1	67		
NaN							
tt2479848	Goodnight Mr. Foot	(2012) Animation, Short,... Bigfoot checks in...	Director:Genndy T...	5.3	470		
NaN							
tt1525580	Magic Camp	(2012)  Documentary, Family Welcome to the re...	Director:Judd Ehr...	6.2	241		
NaN							
tt2343495	FOX 25th Annivers... (2012 TV Special)		Nan Add a Plot Director:Louis J....	6.3	54		
NaN							
tt2366218	Sound of Heimat -...	(2012)  Documentary	Add a Plot Directors:Arne Bi...	7.3	61		
NaN							
tt1961334	In the Shadows of...	(2012) Documentary, Biog... A documentary tha...	Director:J.C. Bur...	7.9	12		
NaN							
tt2415946	Oglum Bak Git	(2012)  Comedy During the life o...	Director:Kamil Ce...	2.0	1995		
NaN							
tt1989486	Dead Money	(2012)  Action Four drug kingpin...	Directors:Frank E...	8.1	10		
NaN							
tt2188867	Ray Bradbury's Ka...	(2012) Short, Drama, Sci-Fi We were hurdling ...	Director:Eric Toz...	8.1	87		
NaN							
tt2223820	David Bowie & the...	(2012 TV Movie)  Documentary, Music Both a visual fla...	Director:James Ha...	7.4	229		
NaN							
tt2137351	The Bitter Buddha	(2012)  Documentary, Comedy FEATURING ZACH GA...	Director:Steven F...	7.0	294		
72							
tt5879400	The Inside Line	(2012- )  Sport THE POWER OF FORM...	null	6.9	9		
NaN							
tt3265530	Jingang Jing	(I) (2012)  Short Kang-sheng Lee's ... Director:Ming-lia...	6.5	12			
NaN							
tt2249634	Rousseaus Children	(2012 TV Movie) Documentary, History A reality-check i...	Director:Monika S...	8.8	9		
NaN							
tt2342088	Crimes of Mike Re...	(2012)  Drama A failed real est...	Director:Bruce Sw...	5.6	16		
NaN							

181478

### Kết quả khi lọc dữ liệu NaN

Sau khi chọn ra những dữ liệu không có giá trị NaN ở 3 cột Rating, MovieID và Votes thì dữ liệu còn lại 181478 dòng.

- Xử lý số liệu: Tách cột Star thành cột Director, sau đó xử lý số liệu ở cột Star & Director.

```
# Processing Data: Split Star Column into Director Column
movie6 = movie5.withColumn("Director", split(col("Star"), 'Stars:').getItem(0))
# Processing Data: Star & Director Column
movie7 = movie6.withColumn("Director", regexp_replace(col("Director"), 'Director:', ''))
movie8 = movie7.withColumn("Star", split(col("Star"), 'Stars:').getItem(1))
movie8.show()
```

Code xử lý số liệu ở cột Star và Director

## Kết quả:

MovieID	MovieName	Year	Genre	Content	Star	Rating	Votes
Metascore	Director						
tt6796652	Christina Aguilera... (2012 Music Video)		Music	music video for "... Christina Aguilera..."	7.1	67	
NaN	Melina Matsoukas						
tt2479848	Goodnight Mr. Foot	(2012)	Animation, Short,...	Bigfoot checks in... Rose Abdoo, Corey ...	5.3	470	
NaN	Gennady Tartakovsky						
tt1525580	Magic Camp	(2012)	Documentary, Family	Welcome to the re... Jonah Conlin, Zach...	6.2	241	
NaN	Judd Ehrlich						
tt2343495	FOX 25th Annivers...	(2012 TV Special)	Nan	Add a Plot Ryan Seacrest, Chr...	6.3	54	
NaN	Louis J. Horvitz						
tt2366218	Sound of Heimat - ...	(2012)	Documentary	Add a Plot	null	7.3	61
NaN	Directors: Arne Bi...						
tt1961334	In the Shadows of...	(2012)	Documentary, Biog...	A documentary tha... J.C. Burdine, Lois...	7.9	12	
NaN	J.C. Burdine						
tt2415946	Oglum Bak Git	(2012)	Comedy	During the life o... Yavuz Seçkin, Orha...	2.0	1995	
NaN	Kamil Cetin						
tt1989486	Dead Money	(2012)	Action	Four drug kingpin... Chaka Balamani, Ba...	8.1	10	
NaN	Directors: Frank E...						
tt2188867	Ray Bradbury's Ka...	(2012)	Short, Drama, Sci-Fi	We were hurdling ... Brett Stimely, Nat...	8.1	87	
NaN	Eric Tozzi						
tt22223820	David Bowie & the...	(2012 TV Movie)	Documentary, Music	Both a visual fla... Jarvis Cocker, Dav...	7.4	229	
NaN	James Hale						
tt2137351	The Bitter Buddha	(2012)	Documentary, Comedy	FEATURING ZACH GA... Eddie Pepitone, Za...	7.0	294	
72	Steven Feinartz						
tt5879400	The Inside Line	(2012- )	Sport	THE POWER OF FORM...	null	6.9	9
NaN	null						
tt3265530	Jingang Jing	(I) (2012)	Short	Kang-sheng Lee's ...	null	6.5	12
NaN	Ming-liang Tsai						
tt2249634	Rousseau's Children	(2012 TV Movie)	Documentary, History	A reality-check i... Wurtila Kilcher-H...	8.8	9	
NaN	Monika Schärer						
tt2342088	Crimes of Mike Re...	(2012)	Drama	A failed real est... Nicholas Lea, Gabr...	5.6	16	
NaN	Bruce Sweeney						

### Kết quả xử lý số liệu ở cột Star và Director

Ban đầu, cả Star và Director đều được để chung trong cột Star. Sau khi tác dữ liệu, đã tạo ra 2 cột riêng biệt là cột Star và Director. Cột Director sẽ lấy vị trí chuỗi đầu tiên, và Star là phần còn lại.

- Xử lý dữ liệu cột Year

```
# Processing Data: Year Column
movie9 = movie8.withColumn("Year", regexp_replace(col("Year"), ' - ', ''))
movie10 = movie9.withColumn("Year", when(length(col("Year"))>=13, substring("Year",1,5)).when(col("Year").contains(" ( "),
                                         |split(col("Year"), " ").getItem(1).otherwise(col("Year"))))
movie11 = movie10.withColumn("Year", when(length(col("Year"))<=6, substring("Year",2,4)).otherwise(substring("Year",2,9)))
movie11.show()
movie11.printSchema()
```

### Code xử lý số liệu ở cột Year

## Kết quả:

MovieID	MovieName Year Director	Genre	Content	Star Rating	Votes Metascore
tt6796652	Christina Aguilera... 2012 Melina Matsoukas	Music music video for "... Christina Aguilera...	Bigfoot checks in... Rose Abdoo, Corey ...	7.1  67	NaN
tt2479848	Goodnight Mr. Foot 2012 Animation, Short,... Bigfoot checks in... Rose Abdoo, Corey ...	Animation, Short,... Bigfoot checks in...	Rose Abdoo, Corey ...	5.3  470	NaN
tt1525580	Gennady Tartakovsky   Magic Camp 2012  Documentary, Family Welcome to the re... Jonah Conlin, Zach...	Magic Camp 2012  Documentary, Family Welcome to the re...	Jonah Conlin, Zach...	6.2  241	NaN
tt2343495	Judd Ehrlich   FOX 25th Annivers... 2012  NaN  Add a Plot Ryan Seacrest, Chr...	NaN  Add a Plot Ryan Seacrest, Chr...	Ryan Seacrest, Chr...	6.3  54	NaN
tt2366218	Louis J. Horvitz   Sound of Heimat -... 2012  Documentary  Add a Plot  null	Documentary  Add a Plot  null		7.3  61	NaN
tt1961334	irectors:Arne Bi...  In the Shadows of... 2012  Documentary, Biog... A documentary tha... J.C. Burdine, Lois...	Documentary, Biog... A documentary tha...	J.C. Burdine, Lois...	7.9  12	NaN

*Kết quả xử lý số liệu ở cột Year*

Dữ liệu ở cột Year được chuyển đổi từ ‘-’ sang ‘’ (làm biến mất ‘-’).

Những dữ liệu  $\geq 13$  ký tự là xử lý kiểu dữ liệu dạng (2021 TV Show), sẽ loại bỏ và lấy 4 ký tự đầu tiên trừ ngoặc đơn ‘’ (‘ để lấy giá trị năm. Đối với giá trị có ‘’) (‘ sẽ được xử lý tiếp bằng cách cắt dấu cách ra lấy giá trị kiểu ‘(2021)’, sau đó loại bỏ ngoặc đơn.

Những dữ liệu  $\leq 6$  có dạng (2021), sẽ được tách 4 ký tự đầu còn, còn lớn hơn thì (2021-2022) thì tách lấy 9 số (2021-2022). Sau khi tách xong sẽ loại bỏ giá ngoặc đơn.

- **Chuyển đổi kiểu dữ liệu**

Chuyển đổi dữ liệu cột Rating thành kiểu dữ liệu Double

Chuyển đổi dữ liệu cột Votes và Metascore thành kiểu dữ liệu Integer

```
# Convert DataType
movie12 = movie11.withColumn("Rating", col("Rating").cast(DoubleType()))
movie13 = movie12.withColumn("Votes", col("Votes").cast(IntegerType()))
movie14 = movie13.withColumn("Metascore", col("Metascore").cast(IntegerType()))
movie14.show()
movie14.printSchema()
```

*Code chuyển đổi kiểu dữ liệu*

## Kết quả:

MovieID	MovieName	Year	Genre	Content	Star	Rating	Votes	Metascore
tt6796652	Christina Aguilera... Melina Matsoukas	2012	Music music video for "... Christina Aguilera... Judd Ehrlich	music video for "... Christina Aguilera... Judd Ehrlich	7.1	67	null	
tt2479848	Goodnight Mr. Foot Goodnight Mr. Foot Sonya Tayeh	2012	Animation, Short,... Bigfoot checks in... Rose Abdoo,Corynne Tartakovsky	Bigfoot checks in... Rose Abdoo,Corynne Tartakovsky	5.3	470	null	Genre
tt1525580	Magic Camp Judd Ehrlich	2012	Documentary, Family Welcome to the re... Jonah Conlin,Zach...	Welcome to the re... Jonah Conlin,Zach...	6.2	241	null	
tt2343495	FOX 25th Anniversary Special Louis J. Horvitz	2012	Nan	Add a Plot Ryan Seacrest,Chris...	6.3	54	null	
tt2366218	Sound of Heimat - Actors:Arne Bi...	2012	Documentary	Add a Plot	null	7.3	61	null Dir
tt1961334	In the Shadows of... J.C. Burdine	2012	Documentary, Biog... A documentary tha... J.C. Burdine,Lois...	A documentary tha... J.C. Burdine,Lois...	7.9	12	null	
tt2415946	Oglum Bak Git Kamil Cetin	2012	Comedy During the life o... Yavuz Seçkin,Orhan...	During the life o... Yavuz Seçkin,Orhan...	2.0	1995	null	
tt1989486	Dead Money Actors:Frank E...	2012	Action Four drug kingpin... Chaka Balamani,Ba...	Four drug kingpin... Chaka Balamani,Ba...	8.1	10	null Dir	
tt2188867	Ray Bradbury's Ka... Eric Tozzi	2012	Short, Drama, Sci-Fi We were hurdling ... Brett Stimely,Nat...	We were hurdling ... Brett Stimely,Nat...	8.1	87	null	
tt2223820	David Bowie & the... James Hale	2012	Documentary, Music Both a visual fla... Jarvis Cocker,Dav...	Both a visual fla... Jarvis Cocker,Dav...	7.4	229	null	
tt2137351	The Bitter Buddha Steven Feinartz	2012	Documentary, Comedy FEATURING ZACH GA... Eddie Pepitone,Za...	FEATURING ZACH GA... Eddie Pepitone,Za...	7.0	294	72	
tt5879400	The Inside Line null	2012	Sport THE POWER OF FORM...	THE POWER OF FORM...	null	6.9	9	null
tt3265530	Jingang Jing g-liang Tsai S...	2012	Short Kang-sheng Lee's ...	Kang-sheng Lee's ...	null	6.5	12	null Min
tt2249634	Rousseaus Children Monika Schärer	2012	Documentary, History A reality-check i... Wurtilla Kilcher-H...	A reality-check i... Wurtilla Kilcher-H...	8.8	9	null	
tt2342088	Crimes of Mike Re... Bruce Sweeney	2012	Drama A failed real est... Nicholas Lea,Gabri...	A failed real est... Nicholas Lea,Gabri...	5.6	16	null	

*Kết quả sau khi chuyển đổi kiểu dữ liệu*

Sau khi chuyển đổi dữ liệu thì kết quả hiện ra vẫn như trước, chỉ có kiểu dữ liệu được thay đổi.

- *In và tạo file csv*

Tạo ra 2 loại file, 1 file phù hợp với việc xử lý dữ liệu trên pyspark, 1 file phù hợp với việc xử lý trên pandas

```
# Print & Create file csv
movie14.write.option("header",True).option("delimiter","|")
                  .csv('MovieFixed/f'MovieFullSpark.csv')
movie15 = movie14.toPandas()
movie15.to_csv('MovieFixed/f'MovieFullPandas.csv')
```

*Code In và tạo file csv*

- *Xử lý và lưu trữ các file dữ liệu Reviews riêng lẻ*

Vì dữ liệu reviews thu thập được lưu trữ dưới dạng từng file riêng tương ứng với mỗi bộ phim nên cần phải xử lý đồng loạt các file dữ liệu reviews để tiết kiệm thời gian, công sức và tài nguyên máy. Ở đây, nhóm thực hiện một vòng lặp cho phép đọc và xử lý dữ liệu hàng loạt file reviews và cuối cùng là cho ra kết quả, gộp lại các file dữ liệu riêng thành một file lớn và lưu trữ dưới dạng csv như sau:

```

filename = movie15['MovieID']
reviewcount = 0
#run for loop to process
for i in range(len(filename)):
    try:
        # Read file csv review
        df = pd.read_csv('Review/f'review{filename[i]}.csv')
        # Get data without NaN values in User_ID, Review_rate column
        df1 = df.loc[df['User_ID']!='NaN']
        df2=df1.loc[df1['Review_rate']!='NaN']
        # Processing Data: User_ID column
        df2["User_ID"] = df2["User_ID"].str.split("/").str.get(2)
        # Processing Data: Review_body column
        df2["Review_body"] = df2["Review_body"].str.split("\n").str.get(1)
        # Processing Data: Review_date column
        df2["Review_date"] = pd.to_datetime(df2["Review_date"], format='%d %B %Y').dt.strftime('%d/%m/%Y')
        # Processing Data: Review_rate column
        df2["Review_rate"] = df2["Review_rate"].str.split("/").str.get(0).astype(int)
        # Drop unknown column
        df2 = df2.drop('Unnamed: 0', axis=1)
        print(df2)
        reviewcount += df2.shape[0]
    except:
        pass
print(reviewcount)

```

*Code xử lý và lưu trữ các file dữ liệu Reviews riêng lẻ*

## Kết quả:

```
      Review_ID    User_ID      User_name \
0      rw3835213  ur20552756  TheLittleSongbird
1      rw3097874  ur2898520   SnoopyStyle
2      rw3106157  ur3947986   brando647
3      rw3062536  ur5876717   kosmasp
4      rw2979053  ur28528605  trublu215
..      ...        ...
508     rw5507601  ur64360477  remzisiyer
509     rw3768042  ur71977754  BryanVigier
510     rw4154995  ur72332484  ramacriz
511     rw4075421  ur85886780  afroninjaen
512     rw4281029  ur91135415  brandontard

                           Review_title  Review_rate \
0                  Definitely does stay with you      7
1                  weirdly intriguing            7
2  An Addicting Mind-Bender from the Director of ...      8
3                  Keep em close ...            10
4          A disturbing psychological thriller            9

      Review_date           Review_body      MovieID
0      01/07/2008  I was completely astonished th...  tt0832266
1      13/04/2008  One guy and the four loves of ...  tt0832266
2      10/07/2010  "Definitely, Maybe" was market...  tt0832266
3      09/04/2008  I just saw this movie today an...  tt0832266
4      21/03/2008  When his ten-year-old daughter...  tt0832266
..      ...
207     24/07/2008  Definitely, MaybeI just watche...  tt0832266
208     14/02/2009  I liked the lead character pla...  tt0832266
209     11/02/2009  I would likely have never seen...  tt0832266
210     19/07/2016  I had no expectations with thi...  tt0832266
211     21/02/2019  This movie is about Will Hayes...  tt0832266

[212 rows x 8 columns]
```

Kết quả xử lý và lưu trữ các file dữ liệu Reviews riêng lẻ

- **Xử lý và gộp các file dữ liệu Reviews thành một file lớn**

Để thuận tiện cho việc chuẩn xá dữ liệu, tạo mô hình và đánh giá mô hình, nhóm chúng em đã quyết định gộp nhiều file nhỏ đã xử lý ở trên thành một file lớn.

```

movie15 = movie14.toPandas()
filename = movie15['MovieID']
# Create initial index
reviewcount = 0
moviefull = []
moviefull = pd.DataFrame(data=moviefull)
#run for loop to process
for i in range(len(filename)):
    try:
        # Read file csv review
        df = pd.read_csv('f'review{filename[i]}.csv')
        # Get data without Nan values in User_ID, Review_rate column
        df1 = df.loc[df["User_ID"] != 'NaN']
        df2 = df1.loc[df1['Review_rate'] != 'NaN']
        # Processing Data: User_ID column
        df2["User_ID"] = df2["User_ID"].str.split("/").str.get(2)
        # Processing Data: Review_body column
        df2["Review_body"] = df2["Review_body"].str.split("\n").str.get(1)
        # Processing Data: Review_date column
        df2["Review_date"] = pd.to_datetime(df2["Review_date"], format='%d %B %Y').dt.strftime('%d/%m/%Y')
        # Processing Data: Review_rate column
        df2["Review_rate"] = df2["Review_rate"].str.split("/").str.get(0).astype(int)
        # Drop unknow column
        df2 = df2.drop('Unnamed: 0', axis=1)
        # Merge review movie file
        moviefull = pd.concat([moviefull, df2], axis=0, ignore_index=True)
    except:
        pass
#print & Save file csv
print(moviefull)
print(moviefull.shape[0])
moviefull.to_csv('ReviewFixed/f'ReviewFull.csv', header=True, index=False)

```

*Code xử lý và gộp các file dữ liệu Reviews thành một file lớn*

### Kết quả:

	Review_ID	User_ID	User_name	\
0	rw3835213	ur20552756	TheLittleSongbird	
1	rw3097874	ur2898520	SnoopyStyle	
2	rw3106157	ur3947986	brando647	
3	rw3062536	ur5876717	kosmasp	
4	rw2979053	ur28528605	trublu215	
...	...	...	...	...
808756	rw5325128	ur81544839	mikdel-45286	
808757	rw4987325	ur41953923	jm3251	
808758	rw4447649	ur23252629	vipul-chawla09	
808759	rw5100334	ur24017656	andrea-kucis	
808760	rw6307271	ur54284136	john-67-180222	
		Review_title	Review_rate	\
0		Definitely does stay with you	7	
1		weirdly intriguing	7	
2	An Addicting Mind-Bender from the Director of ...		8	
3		Keep em close ...	10	
4		A disturbing psychological thriller	9	
...		...	...	...
808756		One of the best detective show ever	10	
808757		Very good series but....	8	
808758		A Hidden gem on Netflix	10	
808759		Solid crime TV show	7	
808760		Hooked!	9	

```

808758 09/11/2018      This show is a perfect example...
808759 04/09/2019      Watchable british crime detect...
808760 25/11/2020      We've watched dozens of US pol...

      MovieID
0      tt2316411
1      tt2316411
2      tt2316411
3      tt2316411
4      tt2316411
...
808756  tt2303687
808757  tt2303687
808758  tt2303687
808759  tt2303687
808760  tt2303687

[808761 rows x 8 columns]
808761

```

*Kết quả xử lý và gộp các file dữ liệu Reviews thành một file lớn*

## 4.6 Khai phá và trực quan hóa dữ liệu

### 4.5.1 Ratings của phim

```

[5] movies['Rating'].value_counts()

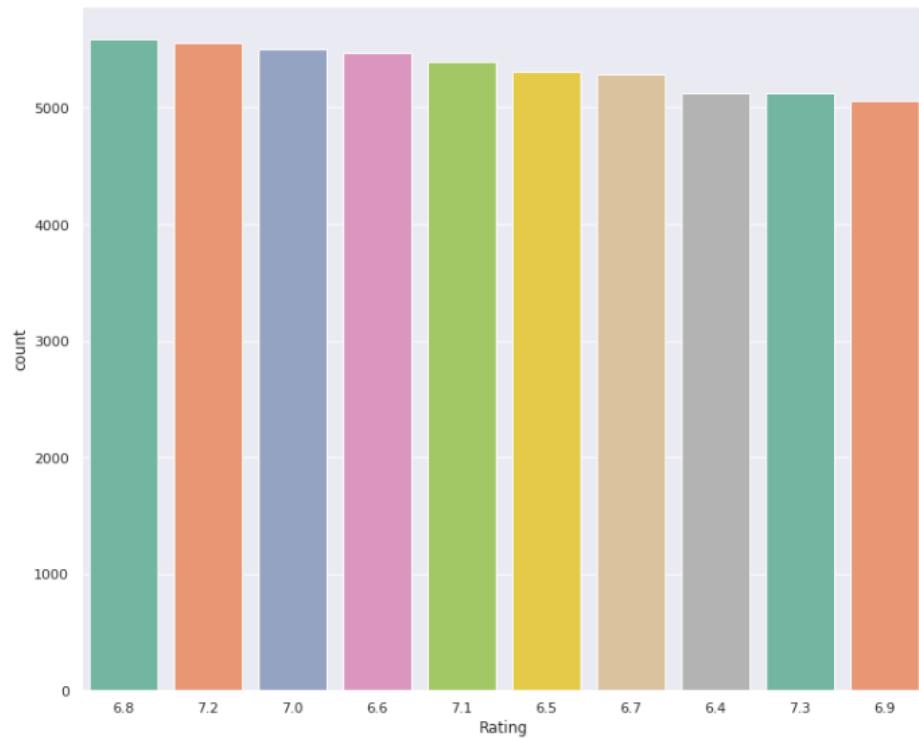
6.8    5593
7.2    5557
7.0    5506
6.6    5473
7.1    5398
...
9.9     63
1.3     57
1.1     39
1.2     39
1.0     36
Name: Rating, Length: 91, dtype: int64

```

*Số lượng các giá trị ratings*

Những bộ phim trong tệp dữ liệu được đánh giá ở 91 giá trị ratings khác nhau, và trải dài từ 1.0 đến 10.0

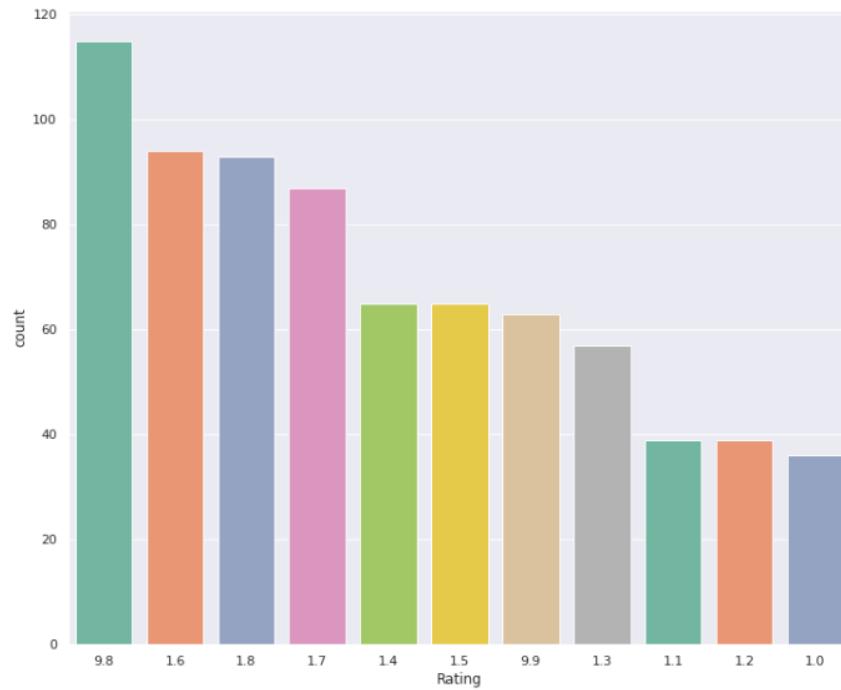
```
[57] plt.figure(figsize=(12,10))
sns.set(style="darkgrid")
ax = sns.countplot(x="Rating", data=movies, palette="Set2", order=movies['Rating'].value_counts().index[0:10])
```



### *Top 10 chỉ số ratings nhiều nhất*

Biểu đồ trên cho chúng ta thấy được top 10 chỉ số ratings ít nhất và trong đó chỉ số rating nhiều nhất là 6.8 với 5593 bộ phim có rating này.

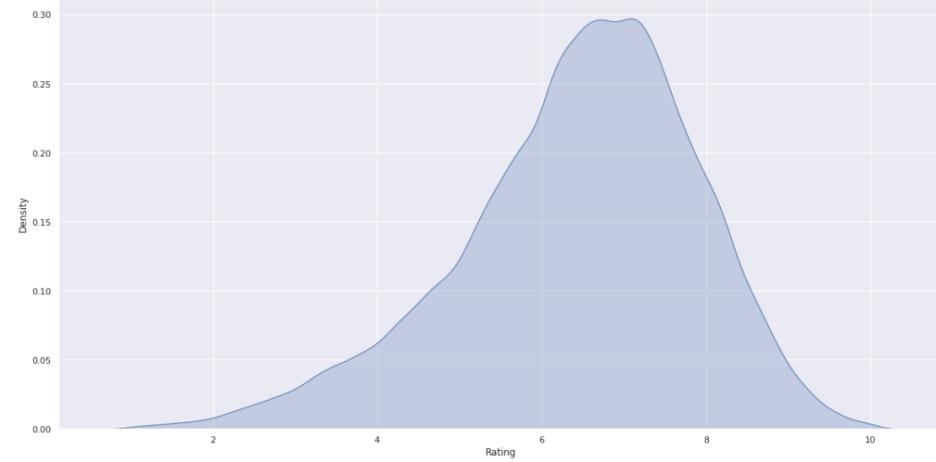
```
[59] plt.figure(figsize=(12,10))
sns.set(style="darkgrid")
ax = sns.countplot(x="Rating", data=movies, palette="Set2", order=movies['Rating'].value_counts().index[80:91])
```



*Top 10 chỉ số ratings ít nhất*

Biểu đồ trên cho chúng ta thấy được top 10 chỉ số ratings ít nhất và trong đó chỉ số rating ít nhất là 1.0 với 36 bộ phim có rating này

```
plt.figure(figsize=(20,10))
sns.set(style="darkgrid")
sns.kdeplot(data=movies['Rating'], shade=True)
```



*Tổng quan về chỉ số ratings của những bộ phim trên tệp dữ liệu*

Những bộ phim trong tệp dữ liệu này có mức độ ratings tập trung cao từ khoảng 6.5 đến 7.5. Điều đó cho thấy những bộ phim này chỉ được đánh giá ở mức “trung bình - khá” là chủ yếu.

#### 4.5.2 Thể loại phim

```
genre_popularity = (movies.Genre.str.split(',')
                     .explode()
                     .value_counts()
                     .sort_values(ascending=False))
genre_popularity.head(10)
```

Drama	37289
Comedy	33900
Drama	32582
Documentary	26024
Action	16254
Romance	15974
Short	15164
Comedy	13156
Thriller	13149
Animation	10592

Name: Genre, dtype: int64

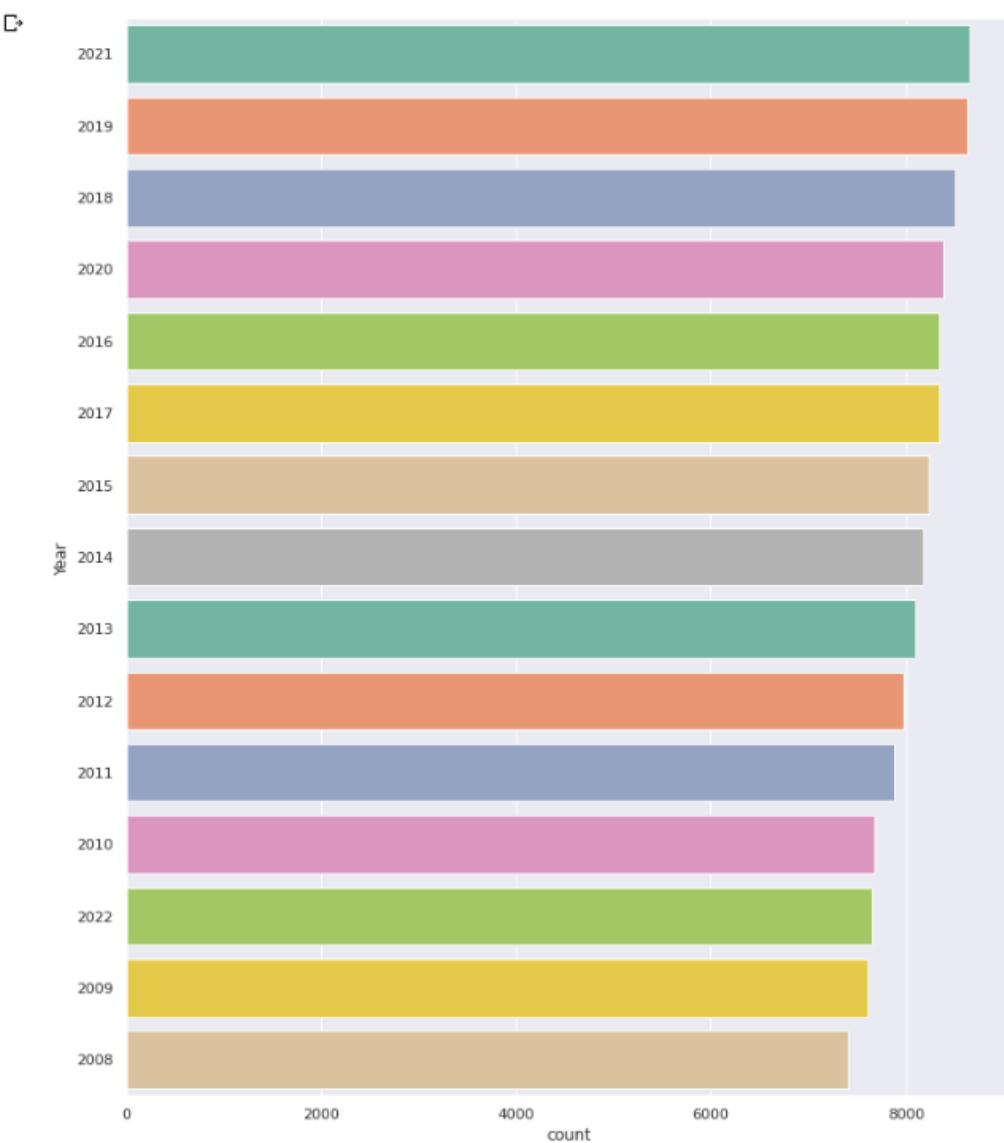
*Code tách thẻ loại phim - In ra top 10 thẻ loại phổ biến nhất*

Top 10 chủ đề phim phổ biến nhất là Drama, Comedy, Documentary, Action, Romance, Short, Comedy, Thriller và Animation.

## *Word Cloud cho những thể loại*

#### 4.5.3 Năm sản xuất phim

```
plt.figure(figsize=(12,15))
sns.set(style="darkgrid")
ax = sns.countplot(y="Year", data=movies, palette="Set2", order=movies['Year'].value_counts().index[0:15])
```



*Top 15 năm có nhiều bộ phim được sản xuất nhất*

Biểu đồ trên cho thấy top 15 năm có nhiều bộ phim được sản xuất nhất và cụ thể hóa từng năm. Năm 2021 là năm sản xuất nhiều nhất với hơn 8500 bộ phim.

#### 4.5.4 Đạo diễn sản xuất phim

```
[13] director_popularity = (movies.Director.str.split(' | ')
     .explode()
     .value_counts()
     .sort_values(ascending=False))
director_popularity.head(15)
```

```
124756
Kevin Dunn           219
David DeCoteau        105
Laurent Bouzereau      91
Fred Olen Ray          78
Michael Feifer          69
Glenn Weiss             63
Joel Lamangan            59
Bill Zebub              56
Terry Ingram              55
Jim Wynorski              55
Takashi Miike             51
Louis J. Horvitz            51
Josh Oreck                50
Steven R. Monroe             50
Name: Director, dtype: int64
```

*Code tách tên đạo diễn - In ra top 15 đạo diễn sản xuất nhiều phim nhất*

Mỗi bộ phim có thể có sự tham gia của nhiều đạo diễn khác nhau. Dữ liệu trên là top 15 những đạo diễn sản xuất nhiều bộ phim nhất từ năm 2002 đến năm 2022. Đạo diễn Kevin Dunn sở hữu cho mình nhiều dự án điện ảnh nhất, lên đến 219 bộ phim.

```
#plot a word-cloud with the Director
director_wc = WordCloud(width=1300,height=600,background_color='white')
director_wc.generate_from_frequencies(director_popularity.to_dict())
plt.figure(figsize=(16, 8))
plt.imshow(director_wc, interpolation="bilinear")
plt.axis('off')
```



*Word Cloud đạo diễn*

#### 4.5.5 Diễn viên tham gia

```
[15] star_popularity = (movies.Star.str.split(',')
                           .explode()
                           .value_counts()
                           .sort_values(ascending=False))
    star_popularity.head(15)
```

Eric Roberts	216
Prakash Raj	151
Grey Griffin	140
Brahmanandam	132
Michael Madsen	132
Steve Blum	127
Dee Bradley Baker	123
Laura Bailey	120
Troy Baker	116
Tom Sizemore	116
Danny Trejo	115
Vivica A. Fox	112
John Cena	108
Dean Cain	108
Takahiro Sakurai	103
Name: Star, dtype: int64	

*Code tách tên diễn viên - In ra top 15 diễn viên tham gia nhiều phim nhất*

Mỗi diễn viên có thể tham gia vào nhiều bộ phim khác nhau. Dữ liệu trên là top 15 những diễn viên tham gia nhiều bộ phim nhất từ năm 2002 đến năm 2022. Diễn viên Eric Roberts tham gia nhiều bộ phim nhất, 216 bộ.

```
[35] #plot a word-cloud with the Star
star_wc = WordCloud(width=1300,height=600,background_color='white')
star_wc.generate_from_frequencies(star_popularity.to_dict())  
#input data: star_popularity.to_dict()
```



*Word Cloud diễn viên*

#### 4.5.6 Chỉ số Metascore ảnh hưởng đến Rating

```
[64] Metascore_ratings=pd.read_csv('MovieFullPandas.csv',usecols=['Metascore'])
    Metascore_Name=pd.read_csv('MovieFullPandas.csv', usecols=['MovieID','MovieName','Genre'])
    ratings = pd.DataFrame({'MovieName':Metascore_Name.MovieName,
                           'Rating': Metascore_ratings.Metascore,
                           'Genre':Metascore_Name.Genre})
    ratings.drop_duplicates(subset=['MovieName', 'Rating'], inplace=True)
    ratings.shape

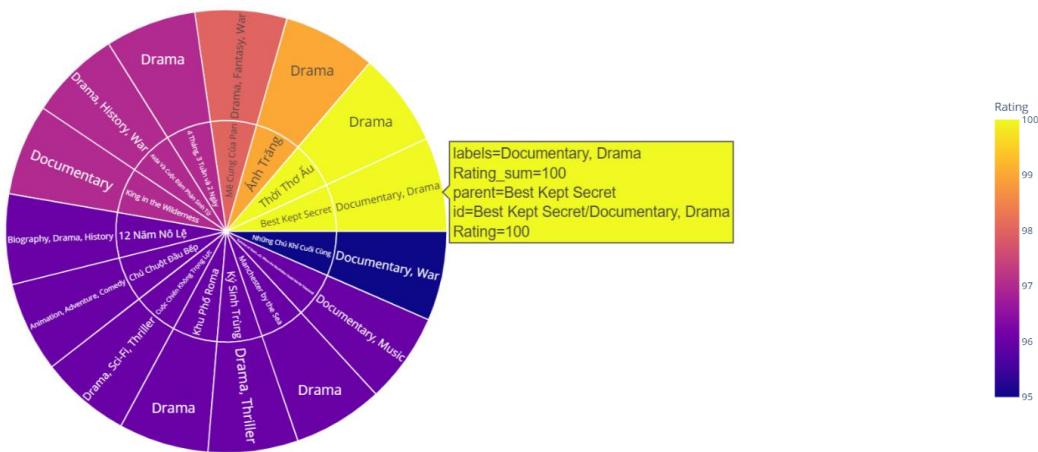
(169623, 3)

[65] ratings.dropna()
    joint_data=ratings.sort_values(by='Rating', ascending=False)

    import plotly.express as px
    top_rated=joint_data[0:15]
    fig =px.sunburst(
        top_rated,
        path=[ 'MovieName','Genre'],
        values='Rating',
        color='Rating')
    fig.show()
```

Code loại bỏ những giá trị null - Tạo biểu đồ tròn biểu diễn theo màu sắc mức rating và thể loại

Loại bỏ các giá trị Null và sắp xếp theo Rating. Tạo biểu đồ tròn biểu diễn theo màu sắc mức rating và thể loại



Biểu đồ tròn biểu diễn theo màu sắc mức rating và thể loại

Biểu đồ thê hiện top 15 bộ phim có rating cao nhất và được biểu diễn theo màu sắc từ xanh lam đậm đến vàng. Bộ phim “Best Kept Secret” đạt rating cao nhất ( 100 - màu vàng), Thể loại là Documentary và Drama.

#### 4.5.7 Những bộ phim có lượt vote nhiều nhất

```
[34] #Top 15 Highest number of votes
top_votes=movies.sort_values(by='Votes')
top_votes=top_votes[181453:181468]

import plotly.graph_objects as go

fig = go.Figure(data=[go.Table(header=dict(values=['Movie Name', 'Number of Votes']),
                                cells=dict(values=[top_votes['MovieName'],top_votes['Votes']],fill_color='lavender'))])
fig.show()
```

Movie Name	Number of Votes
Đảo Kinh Hoàng	1305595
Điệp Vụ Boston	1319246
Áo Thuật Gia Đầu Trí	1327048
Avengers: Biệt Đội Siêu Anh Hùng	1386728
Sói Già Phố Wall	1397929
Định Mệnh	1440178
Người Dơi Xuất Hiện	1464679
Hành Trình Django	1543743
Chúa Tể Của Những Chiếc Nhẫn: Hai Tòả Tháp	1658555
Kỵ Sĩ Bóng Đêm Trỗi Dậy	1697312
Hổ Đen Tứ Thần	1810970
Chúa Tể Của Những Chiếc Nhẫn: Sự Quay Trở Lại Của Nhà Vua	1836797
Tập Làm Người Xấu	1866698
Trò Chơi Vương Quyền	2085514
Kẻ Đánh Cắp Giác Mơ	2336958

*Code sắp xếp và in ra top 15 bộ phim có lượt vote cao nhất*

Bảng dữ liệu trên là top 15 bộ phim có lượt vote cao nhất. Trong đó, số lượt vote cao nhất là 2.336.958 thuộc về bộ phim Kẻ Đánh Cắp Giác Mơ.

#### 4.5.8 Nhữn group thẻ loại phim phổ biến năm 2002

```

movies_fr=movies[movies['Year']=='2002']
nannef=movies_fr.dropna()

import plotly.express as px
fig = px.treemap(nannef, path=['Year', 'Genre'],
                  color='Genre', hover_data=['Genre', 'MovieName'],color_continuous_scale='Purples')
fig.show()

```



Những group thẻ loại phim phổ biến năm 2002

Phần trên, nhóm chúng em đã phân tích về những thẻ loại phim phổ biến trong suốt 20 từ 2002 đến 2022. Ở phần này, nhóm em sẽ chọn ra một năm cụ thể (năm 2002) để tiến hành trực quan hóa dữ liệu, xem nhóm thẻ loại phim nào là phổ biến nhất. Năm 2002, phổ biến nhất là thẻ loại phim drama, với 62 bộ phim thuần Drama. Tiếp theo đó là group thẻ loại “Comedy, Drama, Romance”,...

## 5. Đề xuất mô hình

### 5.1 Mô hình áp dụng TF-IDF



*Mô hình áp dụng TF-IDF vào hệ thống đề xuất phim dựa trên nội dung*

### 5.2 Mô hình áp dụng SVD



*Mô hình áp dụng SVD vào hệ thống phim dựa trên lượt đánh giá*

## 6. Kết quả thực nghiệm

### 6.1 Kết quả thực nghiệm mô hình TF-IDF

#### a, Ý tưởng chung

Bộ dữ liệu mà nhóm thu thập được về thông tin các bộ phim gồm có các trường thông tin: MovieID, MovieName, Year, Genre, Content, Star, Rating, Votes, Metascore. Trong đó có các trường về tên, nội dung, thể loại và diễn viên là loại dữ liệu văn bản. Nhóm quyết định lựa chọn trường thông tin “Genre” để thử nghiệm mô hình TF-IDF vì nội dung ngắn gọn, xúc tích của trường thông tin và tính phổ biến của hệ thống đề xuất dựa trên thể loại.

Sau khi phân tích trường “Genre” thành các vector TF-IDF, mô hình sử dụng ma trận độ tương tự Cosine Similarity để tính điểm tương tự giữa các bộ phim với nhau. Từ ma trận này, với mỗi bộ phim được yêu cầu đề xuất, mô hình sẽ lọc top phim có độ tương tự cao nhất so với bộ phim được yêu cầu. Từ đó, mô hình đưa ra danh sách phim đề xuất theo số lượng mà người dùng yêu cầu.

## b. Các bước thực hiện

Bước 1: Nhập dữ liệu và thư viện

Bước 2: Tách các từ trong trường “Genre” (Tokenization)

Bước 3: Tính các trọng số TF, IDF

Bước 4: Chuẩn hóa về điểm TF-IDF

Bước 5: Tạo ma trận tính điểm tương đồng Cosine (Cosine Similarity Matrix)

Bước 6: Viết hàm đưa ra đề xuất phim

Bước 7: Đưa ra đề xuất phim bằng hàm đã viết

## c. Thư viện sử dụng

```
from pyspark.ml.feature import Tokenizer
```

Tokenizer đơn giản cung cấp chức năng lấy văn bản (chẳng hạn như một câu) và chia nó thành các thuật ngữ riêng lẻ (thường là các từ). (*Extracting, transforming and selecting features*, 2017)

```
from pyspark.ml.feature import HashingTF, IDF
```

Đối với chỉ số TF: Cả HashingTF và CountVectorizer đều có thể được sử dụng để tạo ra các vectơ tần số. HashingTF là một Transformer nhận các tập hợp thuật ngữ và chuyển đổi các tập hợp đó thành các vectơ đặc trưng có độ dài cố định. Trong khi đó, CountVectorizer chuyển đổi tài liệu văn bản thành vectơ đếm thuật ngữ. Ở trong bài, nhóm chọn sử dụng HashingTF. (*Extracting, transforming and selecting features*, 2017)

IDF: IDF là một Estimator phù hợp với tập dữ liệu và tạo ra IDFModel. IDFModel lấy các vectơ đặc trưng (thường được tạo từ HashingTF hoặc CountVectorizer) và chia tỷ lệ

từng đặc trưng. Nói một cách đơn giản, nó giúp giảm trọng lượng các tính năng xuất hiện thường xuyên trong kho văn bản. (*Extracting, transforming and selecting features*, 2017)

```
from pyspark.ml.feature import Normalizer
```

Normalizer là một Transformer biến đổi tập dữ liệu gồm các hàng Vector, chuẩn hóa từng Vector để có unit norm. Nó nhận tham số p, chỉ định p-norm được sử dụng để chuẩn hóa ( $p=2$  theo mặc định.) Quá trình chuẩn hóa này có thể giúp chuẩn hóa dữ liệu đầu và cải thiện hiệu quả của thuật toán học. (*Extracting, transforming and selecting features*, 2017)

```
import pyspark.sql.functions as psf  
  
from pyspark.sql.types import DoubleType
```

Module pyspark.sql.functions là một bộ sưu tập các chức năng có sẵn. (*Pyspark.sql module*)

Kiểu dữ DoubleType, trả về số thực có độ chính xác kép. (*Pyspark.sql module*)

#### d. Kết quả

Hàm để xuất phim được viết với các thông số đầu vào bao gồm: tên bộ phim yêu cầu danh sách đề xuất tương tự, ma trận độ tương tự Cosine, ma trận thông tin về phim (gồm ID và tên phim), số lượng phim mong muốn để xuất).

Từ đó với bộ phim “Đường dây tội phạm” ta thu được kết quả 10 đề xuất với độ tương tự về thể loại như sau:

### Create function

```
: def genre_recommendations(i, M, items, k=20):
    """
    Recommends movies based on a similarity dataframe
    Parameters
    -----
    i : str - MovieName
        Movie title (index of the similarity dataframe)
    M : pd.DataFrame - final
        Similarity dataframe, symmetric, with movies as indices and columns
    items : pd.DataFrame - movies
        Contains both the title and some other features used to define similarity
    k : int
        Amount of recommendations to return
    """
    a = items.select('MovieID').where(items.MovieName == i).collect()[0]
    list1 = M.select('j','dot').where(items.MovieID == a[0])
    finalist = list1.join(items, list1.j == items.MovieID, "inner")
    finalist = finalist.select('MovieID','MovieName','Genre','dot')
    finalist = finalist.selectExpr("MovieID","MovieName","Genre","dot as Similarity")
    finalist.orderBy(finalist.Similarity.desc()).show(k)

: # Test function
genre_recommendations("Đường Dây Tội Phạm", final, movies, k=10)
22/12/16 17:25:26 WARN DAGScheduler: Broadcasting large task binary with size 4.0 MiB
22/12/16 17:25:26 WARN DAGScheduler: Broadcasting large task binary with size 4.1 MiB
[Stage 25:>                                     (0 + 1) / 1]
+-----+
| MovieID|      MovieName|       Genre|  Similarity|
+-----+
|tt0363589| Elephant|Crime, Drama, Thr...|      1.0|
|tt0375679|       Ô Võ|Crime, Drama, Thr...|      1.0|
|tt0315733|Những Mảnh Dời Bâ...|Crime, Drama, Thr...|      1.0|
|tt0313542|Bôl Thâm Đoàn Châ...|Crime, Drama, Thr...|      1.0|
|tt0349903| 12 Tên Cướp Thê'Ký|Crime, Thriller|0.8989279823518066|
|tt0361862|      Gã Thủ Máy|Drama, Thriller|0.8155930051332752|
|tt0317740|Phi Vũ Cuối Cùng ...|Action, Crime, Th...|0.8808715174550896|
|tt0378194| Cô Dâu Bảo Thủ 2|Action, Crime, Th...|0.8808715174550896|
|tt0322259|Quá Nhanh Quá Ngu...|Action, Crime, Th...|0.8808715174550896|
|tt0314353|   Nước Mát Mát Trời|Action, Drama, Th...|0.7184988225401572|
+-----+
only showing top 10 rows
```

Kết quả đề xuất của mô hình TF-IDF

## 6.2 Kết quả thực nghiệm mô hình SVD

### a, Ý tưởng chung

Bộ dữ liệu mà nhóm thu thập được về thông tin các bộ phim gồm có các trường thông tin: userId, movieId, rating, timestamp, title, genres. Nhóm quyết định lựa chọn trường “userId”, “movieId”, “rating” để tiến hành thiết lập ma trận sử dụng trong thuật toán SVD.

“Ma trận tương tác giữa người dùng và mục” có cấu trúc ma trận trong đó mỗi hàng đại diện cho một người dùng và mỗi cột đại diện cho một mục. Các yếu tố của ma trận này là xếp hạng mà người dùng đưa ra cho các mục. Sử dụng SVD, ma trận ban đầu A với M hàng, N cột và hạng R có thể được phân tách thành tích của ba ma trận.

- Ma trận 1: Đại diện mối quan hệ giữa người dùng và các yếu tố tiềm ẩn
- Ma trận 2: Mô tả sức mạnh của từng yếu tố tiềm ẩn

- Ma trận 3: Mức độ liên quan của từng tính năng đối với từng bộ phim

**b. Các bước thực hiện:**

- Bước 1: Nhập dữ liệu
- Bước 2: Tải thư viện sử dụng
- Bước 3: Đọc dữ liệu (Sử dụng Pandas)
- Bước 4: Tạo ma trận tương tác giữa người dùng và mục (Sử dụng Pivot Table trong pandas)
- Bước 5: Chuẩn hóa ma trận
- Bước 6: Tính toán SVD, phân tách ma trận (Sử dụng hàm scipy svds)
- Bước 7: Tìm ma trận dự đoán
- Bước 8: Viết hàm đưa ra đề xuất phim
- Bước 9: Đưa ra đề xuất phim bằng hàm đã viết

**c. Thư viện sử dụng:**

```
import numpy as np
```

```
import pandas as pd
```

Pandas là thư viện được sử dụng để quản lý và phân tích dữ liệu, đặc biệt còn cung cấp những phép tính toán để chúng ta có thể thực hiện thao tác với các chuỗi, dữ liệu và những bảo số. Trong bài toán trên, chúng tôi sử dụng pandas để đọc và phân tích dữ liệu lớn, bên cạnh đó cũng sử dụng phép toán để phân tách ma trận. Ưu điểm khi sử dụng pandas là chúng có thể xử lý ở tốc độ cao với những bộ dữ liệu lớn mà chúng tôi nhập vào.

```
from scipy.sparse.linalg import svds
```

Thư viện scipy được hình thành dựa trên thư viện numpy, đây là nguồn mở cho toán học, kỹ thuật và khoa học, scipy cung cấp cho chúng ta thao tác tạo mảng nhiều chiều một cách nhanh chóng và tiện lợi.

```
from scipy.sparse.linalg import svds
```

Thư viện scipy được hình thành dựa trên thư viện numpy. Thư viện này được đánh giá là dễ sử dụng và được ưa chuộng bởi nhiều kỹ sư trên thế giới. Cụ thể hơn, đây là nguồn mở cho toán học, kỹ thuật và khoa học và nó cung cấp cho chúng ta thao tác tạo mảng nhiều chiều một cách nhanh chóng và tiện lợi. Trong bài toán trên, chúng tôi sử dụng spicy để tính toán ma trận.

#### d. Kết quả:



	movieId	title	genres	Top Ratings
18	608	Fargo (1996)	Comedy Crime Drama Thriller	2.495263
0	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	2.421927
9	32	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	Mystery Sci-Fi Thriller	2.220835
2	6	Heat (1995)	Action Crime Thriller	1.653115
22	648	NaN	NaN	1.573177
23	780	NaN	NaN	1.564679
24	733	NaN	NaN	1.478041
13	95	Broken Arrow (1996)	Action Adventure Thriller	1.470994
7	25	Leaving Las Vegas (1995)	Drama Romance	1.450276
25	736	NaN	NaN	1.439050

*Kết quả đề xuất phim sử dụng SVD*

### 6.3 Đánh giá kết quả

#### 6.3.1 Tổng quan phương pháp đánh giá

##### a) Confusion matrix

Confusion matrix là phương pháp đánh giá hiệu quả, đo lường hiệu suất của các mô hình được sử dụng phổ biến nhất.

Về cơ bản, ma trận Confusion là một ma trận vuông mà trong đó, các hàng của ma trận đại diện cho các lớp thực tế mà kết quả có, các cột của ma trận đại diện cho các dự đoán mà mô hình đã thực hiện. Nhờ vậy mà ma trận confusion này có thể giúp chỉ ra được cụ thể mỗi loại được phân loại như thế nào, lớp nào được phân loại đúng nhiều nhất, và dữ liệu thuộc lớp nào thường bị phân loại nhầm vào lớp khác. (Huu, 2018)

## Confusion Matrix

		Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)	
	False Negatives (FNs)	True Negatives (TNs)	
Predicted Negative (0)			

*Minh họa Confusion Matrix*

### b) Các chỉ số RMSE & MAE

RMSE và MAE là các chỉ số đo lường được áp dụng rộng rãi trong nhiều hệ thống để xuất hiện nay nhằm mục đích đo lường sự khác biệt giữa các điểm dự đoán với xếp hạng thực tế của người dùng.

Trong đó, Root Mean Square Error (RMSE) là căn bậc hai của sai số bình phương trung bình dùng để đo sự khác biệt giữa các giá trị dự đoán và giá trị thực tế.

$$\text{RMSE} = \sqrt{\frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} (r_{ui} - \hat{r}_{ui})^2}$$

Mean Absolute Error (MAE) là sai số tuyệt đối trung bình của các lỗi trong một tập hợp các dự đoán để giải thích sự khác biệt tuyệt đối giữa các dự đoán và quan sát thực tế. Chỉ số MAE được tính bằng cách lấy tổng chênh lệch tuyệt đối giữa các giá trị dự đoán và giá trị thực rồi chia cho tổng số giá trị trong bộ thử nghiệm.

$$\text{MAE} = \frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} |r_{ui} - \hat{r}_{ui}|$$

### 6.3.2 Thực thi đánh giá

- Đánh giá kết quả SVD**

#### Bước 1: Nhập các thư viện hỗ trợ

```
!pip install surprise
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting surprise
  Downloading surprise-0.1-py2.py3-none-any.whl (1.8 kB)
Collecting scikit-surprise
  Downloading scikit-surprise-1.1.3.tar.gz (771 kB)
    ██████████ | 771 kB 5.0 MB/s
Requirement already satisfied: joblib>=1.0.0 in /usr/local/lib/python3.8/dist-packages (from scikit-surprise->surprise) (1.2.0)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.8/dist-packages (from scikit-surprise->surprise) (1.21.6)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.8/dist-packages (from scikit-surprise->surprise) (1.7.3)
Building wheels for collected packages: scikit-surprise
  Building wheel for scikit-surprise (setup.py) ... done
    Created wheel for scikit-surprise: filename=scikit_surprise-1.1.3-cp38-cp38-linux_x86_64.whl size=2626522 sha256=a51e8004dfd3f85a3ebf
    Stored in directory: /root/.cache/pip/wheels/af/db/86/2c18183a80ba05da35bf0fb7417aac5cddb93bc1b92fd3ea
Successfully built scikit-surprise
Installing collected packages: scikit-surprise, surprise
Successfully installed scikit-surprise-1.1.3 surprise-0.1

from surprise import Dataset, Reader, SVD, accuracy
from surprise.model_selection import train_test_split, cross_validate
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
```

Nhập thư viện

Để sử dụng các chỉ số RMSE, MAE, chúng ta cần cài đặt và nhập vào các thư viện và mô đun hỗ trợ như Surprise, train\_test\_split, Reader, Dataset, SVD, accuracy, cross\_validate, pandas, numpy, matplotlib,...

## Bước 2: Thiết lập dữ liệu

Trong đồ án này, nhóm sẽ tiến hành thiết lập hai file dữ liệu về movie và rating. Sau đó, nhóm bắt đầu tạo ra một tập dữ liệu lớn bằng cách gộp hai file dữ liệu movie và rating lại và sử dụng phương thức train\_test\_split để chia tập dữ liệu thành hai phần, một phần là training model và phần còn lại là test xem mô hình của nhóm đang hoạt động như thế nào. Ở đây, nhóm quyết định chia dữ liệu thành 0.7 dữ liệu train tương ứng với 70% dữ liệu sẽ dùng để training model và 0.3 dữ liệu test tương ứng với 30% dữ liệu sẽ dùng để test, kiểm tra mô hình. Trong đó, tập training sẽ có 70585 dòng dữ liệu và tập testing sẽ có 30251 dòng dữ liệu.

```
movies1 = pd.read_csv('movies.csv')
ratings1 = pd.read_csv('ratings.csv')
df = pd.merge(movies1, ratings1, how='inner', on='movieId')

reader = Reader(rating_scale=(0.5, 5))
data = Dataset.load_from_df(df[['userId', 'title', 'rating']], reader)
trainSet, testSet = train_test_split(data, test_size=.30, random_state=0)
print("Size of trainset: ", trainSet.n_ratings)
print("Size of testset: ", len(testSet))

Size of trainset: 70585
Size of testset: 30251
```

## Thiết lập dữ liệu

## Bước 3: Training model

Ở bước này, nhóm sử dụng mô hình SVD trong Surprise package để dự đoán và training model dựa trên tập dữ liệu train đã chia ở bước trên (chiếm 70% dataset ban đầu). Sau đó là dự đoán đầu ra của tập dữ liệu test thử nghiệm (chiếm 30% dataset ban đầu).

```

algo = SVD(random_state=0)

algo.fit(trainSet)
test_predictions = algo.test(testSet)
train_predictions = algo.test(trainSet.build_testset())

```

### *Training mô hình SVD*

#### **Bước 4: Tính toán các chỉ số đánh giá**

Trước hết, nhóm tính toán và cho in ra màn hình các chỉ số RMSE và MAE trên các tập dữ liệu thử nghiệm và training.

```

def MAE(predictions):
    return accuracy.mae(predictions, verbose=False)
def RMSE(predictions):
    return accuracy.rmse(predictions, verbose=False)

print("RMSE on training data : ", RMSE(train_predictions))
print("RMSE on test data : ", RMSE(test_predictions))
print("MAE on training data : ", MAE(train_predictions))
print("MAE on test data : ", MAE(test_predictions))

RMSE on training data :  0.637060884620494
RMSE on test data :  0.8752648509444276
MAE on training data :  0.4947182918554523
MAE on test data :  0.6730137009807416

```

### *Tính toán chỉ số RMSE, MAE*

Ngoài ra, nhóm còn sử dụng hàm cross\_validate() để chạy quy trình xác thực chéo theo đối số cv tính ra được các chỉ số RMSE và MAE để đo độ chính xác của mô hình tốt hơn. Ở đây, nhóm sẽ xét đến kết quả của các giá trị chỉ số đánh giá MAE và RMSE thu được từ mô hình SVD với 5 lần xác thực chéo

```

cv = cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
cv

Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

      Fold 1  Fold 2  Fold 3  Fold 4  Fold 5    Mean    Std
RMSE (testset)  0.8751  0.8690  0.8760  0.8694  0.8723  0.8723  0.0029
MAE (testset)   0.6721  0.6697  0.6712  0.6668  0.6698  0.6699  0.0018
Fit time        2.32    1.55    1.48    1.49    1.51    1.67    0.32
Test time       0.28    0.13    0.14    0.14    0.13    0.16    0.06
{'test_rmse': array([0.87507631, 0.86899578, 0.87596694, 0.86936113, 0.87231997]),
 'test_mae': array([0.67206522, 0.66974407, 0.67116721, 0.66682999, 0.66975007]),
 'fit_time': (2.3165273666381836,
  1.5515360832214355,
  1.4835774898529053,
  1.4867050647735596,
  1.5097010135650635),
 'test_time': (0.2795073986053467,
  0.12825584411621094,
  0.1400911808013916,
  0.14038801193237305,
  0.13323211669921875)}

```

### *Xác thực chéo mô hình SVD*

#### **Bước 5: Đánh giá kết quả và kết luận**

Cuối cùng, nhóm tạo ra các biểu đồ biểu diễn các chỉ số đã tính toán ra được để dễ dàng quan sát và so sánh.

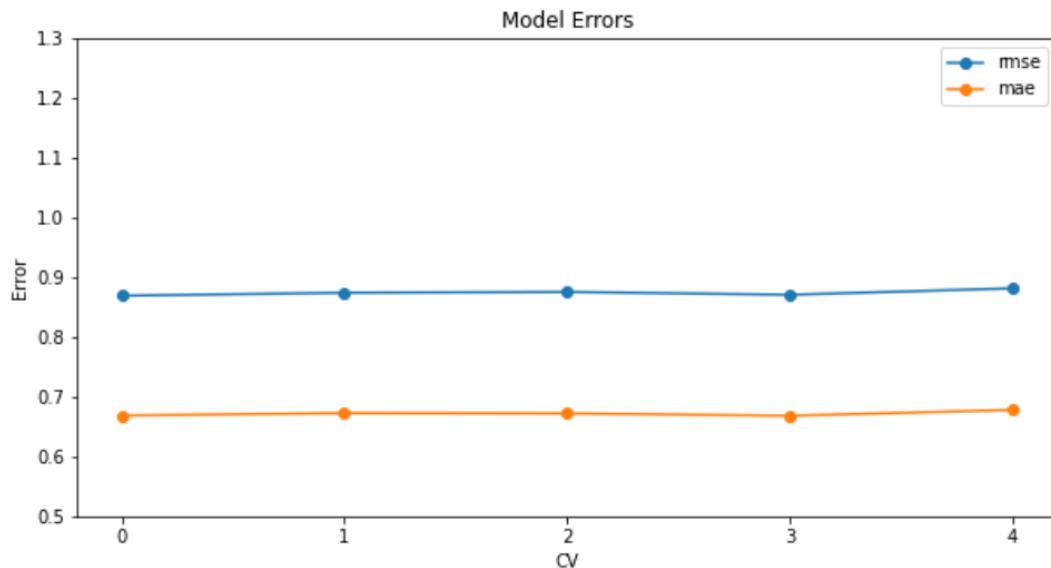
```

# Get data
rmse = cv['test_rmse']
mae = cv['test_mae']
x = np.arange(len(rmse))

# Set up the matplotlib figure
fig, ax = plt.subplots(figsize = (10, 5))
plt.xticks(np.arange(min(x), max(x) + 1, 1.0))
plt.ylim(0.5, 1.3)
ax.plot(x, rmse, marker='o', label="rmse")
ax.plot(x, mae, marker='o', label="mae")

# Chart setup
plt.title("Model Errors", fontsize = 12)
plt.xlabel("CV", fontsize = 10)
plt.ylabel("Error", fontsize = 10)
plt.legend()
plt.show()

```



*Biểu đồ biểu diễn so sánh các chỉ số RMSE và MAE đo được*

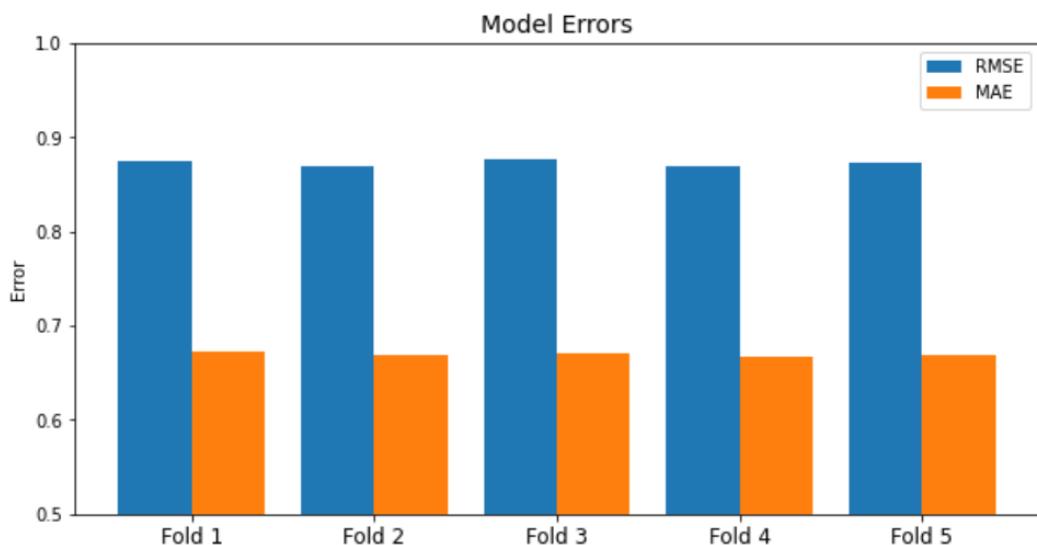
```

# Get data
rmse = cv['test_rmse']
mae = cv['test_mae']
x = np.arange(len(rmse))
width = 0.4
# Set up the matplotlib figure
fig, ax = plt.subplots(figsize = (10, 5))
rects1 = ax.bar(x - width/2, rmse, width,
                 label='RMSE')
rects2 = ax.bar(x + width/2, mae, width,
                 label='MAE')

# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Error')
ax.set_ylim(0.5, 1.0)
ax.set_xticks(np.arange(min(x), max(x) + 1, 1.0))
ax.set_title('Model Errors', fontsize = 14)
ax.set_xticklabels(('Fold 1', 'Fold 2', 'Fold 3', 'Fold 4', 'Fold 5'), fontsize = 12)
ax.legend()

plt.show()

```



*Biểu đồ biểu diễn so sánh các chỉ số RMSE và MAE đo được*

Nhìn vào kết quả, có thể thấy đối với các giá trị MAE và RMSE từ thử nghiệm mô hình SVD không có sự thay đổi đáng kể giữa các lần xác thực chéo. Trong mô hình này, giá trị MAE thấp nhất đạt được trong lần 4 với giá trị 0,6668 trong khi giá trị cao nhất đạt được trong lần 1 với giá trị 0,6721. Trong thử nghiệm này, giá trị RMSE cao hơn giá trị MAE thu được. Giá trị RMSE thấp nhất thu được trong thử nghiệm này là trong lần 2 với giá trị là

0.8690, trong khi giá trị RMSE cao nhất thu được ở lần 3 với giá trị 0,8760. Dựa trên các giá trị MAE và RMSE mà chúng ta có thể thấy trước và sau khi thử nghiệm kết quả cho ra các chỉ số có giá trị không quá xa nhau, khác biệt cũng không đáng kể cho thấy mô hình đang hoạt động khá tốt và ổn định.

### b. Đánh giá kết quả TF-IDF

Nhóm thực hiện thử nghiệm ma trận nhầm lẫn (Confusion matrix) để đánh giá mô hình TF-IDF bằng cách gán nhãn “positive” và “negative”.

#### Bước 1: Nhập các thư viện hỗ trợ

```
import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

import pandas as pd
import seaborn as sns

from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

import re

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.naive_bayes import MultinomialNB
```

*Minh họa mã nguồn*

Để sử dụng các chỉ số Confusion matrix chúng ta cần cài đặt và nhập vào các thư viện và mô đun hỗ trợ như train\_test\_split, confusion\_matrix, classification\_report, TfidfVectorizer, MultinomialNB,...

## Bước 2: Thiết lập dữ liệu

Trong đồ án này, nhóm sẽ tiến hành thiết lập dữ liệu về IMDB movie. Sau đó, sử dụng phương thức train\_test\_split để chia tập dữ liệu thành hai phần, một phần là training model và phần còn lại là test xem mô hình của nhóm đang hoạt động như thế nào. Ở đây, nhóm quyết định chia dữ liệu thành 0.7 dữ liệu train tương ứng với 70% dữ liệu sẽ dùng để training model và 0.3 dữ liệu test tương ứng với 30% dữ liệu sẽ dùng để test, kiểm tra mô hình. Trong đó, tập training sẽ có 92981 dòng dữ liệu và tập testing sẽ có 40000 dòng dữ liệu.

```
lemmatizer = WordNetLemmatizer()

def cleantext(text):
    text= text.lower()
    text= re.sub(r"[^a-zA-Z?.!.,]+", " ", text)
    text= re.sub(r"http\S+", "",text)
    text= re.sub(r"http", "",text)

    punctuations= '@#!?+&*[ ]-%.:;/();$=><|{}^' + "``" + '_'
    for p in punctuations:
        text= text.replace(p, '')

    text= [word.lower() for word in text.split() if word.lower() not in sw]

    text= [lemmatizer.lemmatize(word) for word in text]

    text = " ".join(text)

    emoji_pattern = re.compile("["
                                u"\U0001F600-\U0001F64F"
                                u"\U0001F300-\U0001F5FF"
                                u"\U0001F680-\U0001F6FF"
                                u"\U0001F1E0-\U0001F1FF"
                                u"\U00002702-\U000027B0"
                                u"\U000024C2-\U0001F251"
                                "]+", flags=re.UNICODE)

    text= emoji_pattern.sub(r'',text)

    return text
```

```
X= df['review']
y= df['sentiment_positive']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.2, random_state= 26)
```

```
print("n_samples: %d, n_features: %d" % tfidf_train_vectors.shape)
```

```
n_samples: 40000, n_features: 92981
```

*Minh họa mã nguồn*

### Bước 3: Training model

Ở bước này, nhóm sử dụng TfidfVectorizer để dự đoán và training model dựa trên tập dữ liệu train đã chia ở bước trên (chiếm 70% dataset ban đầu). Sau đó là dự đoán đầu ra của tập dữ liệu test thử nghiệm (chiếm 30% dataset ban đầu)

```
tfidf_vectorizer = TfidfVectorizer()  
tfidf_train_vectors = tfidf_vectorizer.fit_transform(X_train) #applying tf idf to training data  
tfidf_test_vectors = tfidf_vectorizer.transform(X_test) #applying tf idf to training data
```

```
print("n_samples: %d, n_features: %d" % tfidf_train_vectors.shape)
```

```
n_samples: 40000, n_features: 92981
```

```
naive_bayes_classifier = MultinomialNB()  
naive_bayes_classifier.fit(tfidf_train_vectors, y_train)  
MultinomialNB()
```

```
y_pred = naive_bayes_classifier.predict(tfidf_test_vectors)
```

*Minh họa mã nguồn*

#### Bước 4: Tạo ma trận

```
print(classification_report(y_test,y_pred))

precision    recall    f1-score    support

          0       0.86      0.88      0.87      5067
          1       0.88      0.85      0.86      4933

   accuracy                           0.87      10000
macro avg       0.87      0.87      0.87      10000
weighted avg    0.87      0.87      0.87      10000
```

```
cnf_matrix = confusion_matrix(y_test,y_pred)
```

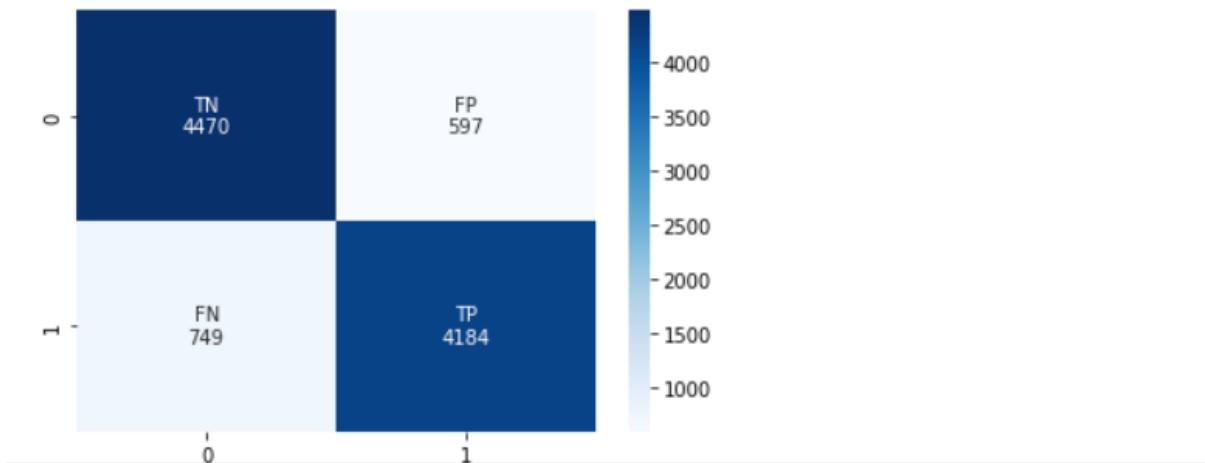
```
cnf_matrix
```

```
array([[4470,  597],
       [ 749, 4184]])
```

*Minh họa ma trận nguồn*

#### Bước 5: Đánh giá kết quả và kết luận

```
group_names = ['TN','FP','FN','TP']
group_counts = ["{0:0.0f}".format(value) for value in cnf_matrix.flatten()]
labels = [f'{v1}\n{v2}' for v1, v2 in zip(group_names,group_counts)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(cnf_matrix, annot=labels, fmt='', cmap='Blues');
```



Nhìn vào ma trận có thể thấy, các phần tử TN, TP trên đường chéo có giá trị cao, cao hơn hẳn so với các giá trị còn lại là FP và FN. Vậy nên có thể nói, đây là một mô hình tốt.

Đối chiếu với mô hình mà nhóm xây dựng, mô hình sử dụng điểm TF-IDF thông qua ma trận tương tự Cosine (Cosine Similarity Matrix) để đưa ra danh sách đề xuất. Vì vậy mô hình đánh giá bằng ma trận nhầm lẫn mà nhóm thử nghiệm chưa phù hợp. Sau khi tham khảo và phân tích, nhóm đề xuất hai hướng đi để có thể xây dựng được mô hình đánh giá cho mô hình TF-IDF trong hệ thống đề xuất hiện tại: Hướng thứ nhất, xây dựng mô hình đề xuất bằng TF-IDF dựa trên ratings. Từ đó có thể đánh giá mô hình bằng cách so sánh ratings dự đoán và ratings thực tế thu thập được của từng bộ phim. Hướng thứ hai là gán nhãn phân loại cho kết quả đề xuất thành bốn loại là “đề xuất đúng”, “đề xuất sai”, “bỏ sót đề xuất đúng”, “bỏ sót đề xuất sai” để áp dụng ma trận nhầm lẫn.

## 7. Kết luận

### 7.1 Đóng góp của dự án

- Dự án đóng góp một phần vào hệ thống lý thuyết về các mô hình đề xuất phim ở trong nước cũng như thế giới. Đặc biệt là xây dựng mô hình TF-IDF cho hệ thống đề xuất hoàn toàn bằng Pyspark.
- Dự này cũng là một nguồn tham khảo đáng tin cậy trong lĩnh vực đề xuất phim trên các website dành cho người mới bắt đầu tìm hiểu.
- Những kết luận của dự án này góp một phần làm rõ hơn các kiến thức liên quan đến hai mô hình chính SVD và TF-IDF.

### 7.2 Hạn chế của dự án và đề xuất

#### 7.2.1. Hạn chế

Trong quá trình thực hiện dự án xây dựng hệ thống mô hình đề xuất phim, nhóm gặp phải những hạn chế sau:

- Khó khăn trong quá trình cào và thu thập dữ liệu, vì lần đầu nhóm thực hiện dữ liệu cào dữ liệu nên nguồn dữ liệu thu được vẫn chưa đầy đủ và rõ ràng như mục tiêu ban đầu đề ra.
- Hạn chế về nguồn tham khảo về các ứng dụng của Pyspark. Nhóm chưa thể xây dựng mô hình SVD hoàn toàn bằng Pyspark. Từ đó cũng dẫn đến việc chưa thể kết hợp 2 phương pháp mà nhóm nghiên cứu là SVD và TF-IDF thành 1 để tăng độ chính xác cho hệ thống đề xuất phim.
- Hạn chế về thời gian và kiến thức của thành viên dẫn đến có nhiều vấn đề nhóm phải tìm hiểu và thử nghiệm trong khoảng thời gian khá lâu để có thể tìm ra hướng đi.
- Lượng dữ liệu nhóm cào khá lớn, trong khi đó cấu hình máy chưa cho phép, vì vậy thời gian xử lý dữ liệu cũng kéo dài, việc chạy mô hình chia ma trận cũng gặp khó khăn.
- Nhóm chưa thể hoàn thiện mô hình đánh giá dành cho mô hình đề xuất bằng TF-IDF bằng Pyspark và so sánh hiệu quả của 2 mô hình với nhau.

### 7.2.2. Đề xuất

Để khắc phục những hạn chế trên, nhóm đưa ra một số đề xuất sau:

- Thu thập kiến thức và dành nhiều thời gian hơn để nghiên cứu, tìm hiểu về quy trình cào dữ liệu.
- Tìm hiểu rõ kiến thức nền tảng về hệ thống đề xuất và mô hình nhóm lựa chọn để quá trình chạy diễn ra mượt mà hơn.
- Phân tích kỹ càng, nghiên cứu và chắt lọc lượng dữ liệu cần cào để nhóm không mất nhiều thời gian trong việc tải dữ liệu.
- Kết hợp 2 phương pháp để đo lường độ chính xác của hệ thống đề xuất phim và tiện hơn trong việc so sánh từng phương pháp.

- Nghiên cứu thêm về Pyspark và cách kết nối các máy con với máy master để thực hiện đúng bản chất của Big Data, không bị hạn chế về mặt dung lượng.
- Thực nghiệm để xây dựng mô hình đánh giá cho mô hình TF-IDF và tiến hành so sánh giữa mô hình TF-IDF và SVD.

## PHẦN II: MÃ NGUỒN

### 1. Tiết xử lý dữ liệu

```
# Drop null values in Rating, Votes, MovieID column
movie1 = moviedata.na.drop(subset=["Rating","Votes","MovieID"])
movie1.show()
print(movie1.count())
```

MovieID	MovieName	Year	Genre	Content	Star	Rating	Votes
NaN	Black Tree Forest...	(2012)	Horror	Add a Plot Director:Dustin F...	2.5	23	
NaN	A Night at the Si...	(2012)	Comedy	A raucous comedy ... Director:Tim Russ...	7.6	21	
NaN	The Daniel Project	(2012 TV Movie)	Documentary	The Daniel Projec... Director:Stewart ...	5.0	166	
NaN	No te enamores de mí	(2012)	Comedy, Drama, Ro...	A collection of s... Director:Federico...	6.1	126	
NaN	Andamio	(2012)	Short, Comedy, Drama	Eduardo, a snob p... Director:Juanma C...	6.7	91	
68	They Call It Myan...	(2012)	Documentary, Hist...	Shot clandestinel... Director:Robert H...	7.0	311	
NaN	Secret Eaters	(2012- )	Reality-TV	Secret Eaters put... Stars:Anna Richar...	6.8	129	
NaN	Passenger: Let He...	(2012 Music Video)	Music	Music video for "... Directors:Dave Je...	7.0	29	
NaN	Bloopers	(2012- )	Comedy	Bloopers is a loo... Stars:Dean Cain,J...	5.3	69	
NaN	Combat Countdown	(2012- )	Documentary	Series showcases ... Stars:Mike New,Mi...	6.4	36	
NaN	Dr. Fubalous	(2012- )	Comedy, Musical	Dr. Fubalous and ... Stars:Donnivin Jo...	5.8	52	
NaN	The Luckiest Man ...	(2012)	Short, Comedy	Everything that c... Director:Matthew ...	6.5	6	
NaN	Airport 24/7: Miami	(2012- )	Reality-TV	Features a behind... Stars:Lauren Stov...	7.2	102	
NaN	Loon Lake	(2012)	Drama, Family, Th...	Madeleine and Ros... Director:Mary Sel...	7.4	9	
NaN	A Thousand Cuts	(2012)	Thriller	A stranger with a... Director:Charles ...	4.5	814	

```
# Filter out data without NaN value in Rating, Votes, MovieID column
movie2 = movie1.filter(movie1.MovieID != 'NaN')
movie3 = movie2.filter(movie2.Rating != 'NaN')
movie4 = movie3.filter(movie3.Votes != 'NaN')
movie5 = movie4.distinct()
print(movie5.count())
movie5.show()
```

181469

MovieID	MovieName	Year	Genre	Content	Star	Rating	Votes
Metascore							
tt6796652	Christina Aguilera... (2012 Music Video)		Music	music video for "... Director:Melina M...	7.1	67	
NaN							
tt2479848	Goodnight Mr. Foot	(2012)	Animation, Short,...	Bigfoot checks in... Director:Genndy T...	5.3	470	
NaN							
tt1525580	Magic Camp	(2012)	Documentary, Family	Welcome to the re... Director:Judd Ehr...	6.2	241	
NaN							
tt2343495	FOX 25th Annivers... (2012 TV Special)			Nan  Add a Plot Director:Louis J....	6.3	54	
NaN							
tt2366218	Sound of Heimat -...	(2012)	Documentary	Add a Plot Directors:Arne Bi...	7.3	61	
NaN							
tt1961334	In the Shadows of...	(2012)	Documentary, Biog...	A documentary tha... Director:J.C. Bur...	7.9	12	
NaN							
tt2415946	Oglum Bak Git	(2012)	Comedy	During the life o... Director:Kamil Ce...	2.0	1995	
NaN							
tt1989486	Dead Money	(2012)	Action	Four drug kingpin... Directors:Frank E...	8.1	10	
NaN							
tt2188867	Ray Bradbury's Ka...	(2012)	Short, Drama, Sci-Fi	We were hurdling ... Director:Eric Toz...	8.1	87	
NaN							
tt2223820	David Bowie & the...	(2012 TV Movie)	Documentary, Music	Both a visual fla... Director:James Ha...	7.4	229	
NaN							
tt2137351	The Bitter Buddha	(2012)	Documentary, Comedy	FEATURING ZACH GA... Director:Steven F...	7.0	294	
72							
tt5879400	The Inside Line	(2012- )	Sport	THE POWER OF FORM...  null	6.9	9	
NaN							
tt3265530	Jingang Jing	(I) (2012)	Short	Kang-sheng Lee's ... Director:Ming-lia...	6.5	12	
NaN							
tt2249634	Rousseaus Children	(2012 TV Movie)	Documentary, History	A reality-check i... Director:Monika S...	8.8	9	
NaN							
tt2342088	Crimes of Mike Re...	(2012)	Drama	A failed real est... Director:Bruce Sw...	5.6	16	
NaN							

```
# Processing Data: Split Star Column into Director Column
movie6 = movie5.withColumn("Director", split(col("Star"), 'Stars:').getItem(0))
# Processing Data: Star & Director Column
movie7 = movie6.withColumn("Director", regexp_replace(col("Director"), 'Director:', ''))
movie8 = movie7.withColumn("Star", split(col("Star"), 'Stars:').getItem(1))
movie8.show()
```

MovieID	MovieName	Year	Genre	Content	Star	Rating	Votes
Metascore	Director						
tt6796652	Christina Aguilera... (2012 Music Video)		Music	music video for "... Christina Aguiler...	7.1	67	
NaN	Melina Matsoukas						
tt2479848	Goodnight Mr. Foot	(2012)	Animation, Short,...	Bigfoot checks in... Rose Abdoo,Corey ...	5.3	470	
NaN	Genndy Tartakovsky						
tt1525580	Magic Camp	(2012)	Documentary, Family	Welcome to the re... Jonah Conlin,Zach...	6.2	241	
NaN	Judd Ehrlich						
tt2343495	FOX 25th Annivers... (2012 TV Special)			Nan  Add a Plot Ryan Seacrest,Chr...	6.3	54	
NaN	Louis J. Horvitz						
tt2366218	Sound of Heimat -...	(2012)	Documentary	Add a Plot  null	7.3	61	
NaN	Directors:Arne Bi...						
tt1961334	In the Shadows of...	(2012)	Documentary, Biog...	A documentary tha... J.C. Burdine,Lois...	7.9	12	
NaN	J.C. Burdine						
tt2415946	Oglum Bak Git	(2012)	Comedy	During the life o... Yavuz Seçkin,Orha...	2.0	1995	
NaN	Kamil Cetin						
tt1989486	Dead Money	(2012)	Action	Four drug kingpin... Chaka Balamani,Ba...	8.1	10	
NaN	Directors:Frank E...						
tt2188867	Ray Bradbury's Ka...	(2012)	Short, Drama, Sci-Fi	We were hurdling ... Brett Stimely,Nat...	8.1	87	
NaN	Eric Tozzi						
tt2223820	David Bowie & the...	(2012 TV Movie)	Documentary, Music	Both a visual fla... Jarvis Cocker,Dav...	7.4	229	
NaN	James Hale						
tt2137351	The Bitter Buddha	(2012)	Documentary, Comedy	FEATURING ZACH GA... Eddie Pepitone,Za...	7.0	294	
72	Steven Feinartz						
tt5879400	The Inside Line	(2012- )	Sport	THE POWER OF FORM...  null	6.9	9	
NaN	null						
tt3265530	Jingang Jing	(I) (2012)	Short	Kang-sheng Lee's ...  null	6.5	12	
NaN	Ming-liang Tsai,S...						
tt2249634	Rousseaus Children	(2012 TV Movie)	Documentary, History	A reality-check i... Wurtilla Kilcher-H...	8.8	9	
NaN	Monika Schärer						
tt2342088	Crimes of Mike Re...	(2012)	Drama	A failed real est... Nicholas Lea,Gabri...	5.6	16	
NaN	Bruce Sweeney						

```

# Processing Data: Year Column
movie9 = movie8.withColumn("Year", regexp_replace(col("Year"), '_ ', ''))
movie10 = movie9.withColumn("Year", when(length(col("Year"))>=13, substring("Year",1,5)).when(col("Year").contains(") ("),
                                         |split(col("Year"), " ").getItem(1)).otherwise(col("Year"))))
movie11 = movie10.withColumn("Year", when(length(col("Year"))<=6, substring("Year",2,4)).otherwise(substring("Year",2,9)))
movie11.show()
movie11.printSchema()

+-----+-----+-----+-----+-----+-----+
| MovieID|    MovieName|Year|   Genre|Content|  Star|Rating| Votes|Metascore|
|Director|
+-----+-----+-----+-----+-----+-----+
|tt6796652|Christina Aguilera...|2012|    Music|music video for "...|Christina Aguilera...| 7.1| 67|   NaN|
|Melina Matsoukas|||
|tt2479848| Goodnight Mr. Foot|2012|Animation, Short,...|Bigfoot checks in...|Rose Abdo,Corey ...| 5.3| 470|   NaN|
|Genndy Tartakovsky|||
|tti525580|      Magic Camp|2012| Documentary, Family|Welcome to the re...|Jonah Conlin,Zach...| 6.2| 241|   NaN|
|Judd Ehrlich|||
|tt2343495|FOX 25th Annivers...|2012|        NaN|Add a Plot|Ryan Seacrest,Chr...| 6.3| 54|   NaN|
|Louis J. Horvitz|||
|tt2366218|Sound of Heimat -...|2012| Documentary|Add a Plot|           null| 7.3| 61|   NaN|
|irectors:Arne Bi...|
|tti1961334|In the Shadows of...|2012|Documentary, Biog...|A documentary tha...|J.C. Burdine,Lois...| 7.9| 12|   NaN|


# Convert DataType
movie12 = movie11.withColumn("Rating", col("Rating").cast(DoubleType()))
movie13 = movie12.withColumn("Votes", col("Votes").cast(IntegerType()))
movie14 = movie13.withColumn("Metascore", col("Metascore").cast(IntegerType()))
movie14.show()
movie14.printSchema()

+-----+-----+-----+-----+-----+-----+
| MovieID|    MovieName|Year|   Genre|Content|  Star|Rating| Votes|Metascore|
|Director|
+-----+-----+-----+-----+-----+-----+
|tt6796652|Christina Aguilera...|2012|    Music|music video for "...|Christina Aguilera...| 7.1| 67|   null|
|Melina Matsoukas|||
|tt2479848| Goodnight Mr. Foot|2012|Animation, Short,...|Bigfoot checks in...|Rose Abdo,Corey ...| 5.3| 470|   null| Ge
|nndy Tartakovsky||| | | | | | |
|tti525580|      Magic Camp|2012| Documentary, Family|Welcome to the re...|Jonah Conlin,Zach...| 6.2| 241|   null|
|Judd Ehrlich|||
|tt2343495|FOX 25th Annivers...|2012|        NaN|Add a Plot|Ryan Seacrest,Chr...| 6.3| 54|   null|
|Louis J. Horvitz|||
|tt2366218|Sound of Heimat -...|2012| Documentary|Add a Plot|           null| 7.3| 61|   null|Dir
|ectors:Arne Bi...|
|tti1961334|In the Shadows of...|2012|Documentary, Biog...|A documentary tha...|J.C. Burdine,Lois...| 7.9| 12|   null|
|J.C. Burdine|||
|tti2188867|Ray Bradbury's Ka...|2012|Short, Drama, Sci-Fi|We were hurdling ...|Brett Stimely,Nat...| 8.1| 87|   null|
|Eric Tozzi|||
|tt2223820|David Bowie & the...|2012| Documentary, Music|Both a visual fla...|Jarvis Cocker,Dav...| 7.4| 229|   null|
|James Hale|||
|tti2137351| The Bitter Buddha|2012| Documentary, Comedy|FEATURING ZACH GA...|Eddie Pepitone,Za...| 7.0| 294|   72|
|Steven Feinartz|||
|tti5879400|      The Inside Line|2012|       Sport|THE POWER OF FORM...|           null| 6.9| 9|   null|
|null|
|tt3265530|      Jingang Jing|2012|       Short|Kang-sheng Lee's ...|           null| 6.5| 12|   null|Min
|g-liang Tsai|S...| | | | | | | |
|tt2249634| Rousseau's Children|2012|Documentary, History|A reality-check i...|Wurtilla Kilcher-H...| 8.8| 9|   null|
|Monika Schäfer|||
|tti2342088|Crimes of Mike Re...|2012|       Drama|A failed real est...|Nicholas Lea,Gabri...| 5.6| 16|   null|
|Bruce Sweeney|||


# Print & Create file csv
movie14.write.option("header",True).option("delimiter","|")
                           .csv('MovieFixed/'f'MovieFullSpark.csv')
movie15 = movie14.toPandas()
movie15.to_csv('MovieFixed/'f'MovieFullPandas.csv')

```

```

filename = movie15['MovieID']
reviewcount = 0
#run for loop to process
for i in range(len(filename)):
    try:
        # Read file csv review
        df = pd.read_csv('Review/f'review{filename[i]}.csv')
        # Get data without NaN values in User_ID, Review_rate column
        df1 = df.loc[df['User_ID'] != 'NaN']
        df2= df1.loc[df1['Review_rate'] != 'NaN']
        # Processing Data: User_ID column
        df2["User_ID"] = df2["User_ID"].str.split("/").str.get(2)
        # Processing Data: Review_body column
        df2["Review_body"] = df2["Review_body"].str.split("\n").str.get(1)
        # Processing Data: Review_date column
        df2["Review_date"] = pd.to_datetime(df2["Review_date"], format='%d %B %Y').dt.strftime('%d/%m/%Y')
        # Processing Data: Review_rate column
        df2["Review_rate"] = df2["Review_rate"].str.split("/").str.get(0).astype(int)
        # Drop unknown column
        df2 = df2.drop('Unnamed: 0', axis=1)
        print(df2)
        reviewcount += df2.shape[0]
    except:
        pass
print(reviewcount)

      Review_ID   User_ID       User_name \
0   rw3835213  ur20552756  TheLittleSongbird
1   rw3097874  ur2898520      SnoopyStyle
2   rw3106157  ur3947986      brando647
3   rw3062536  ur5876717      kosmasp
4   rw2979053  ur28528605     trublu215
..      ...
508  rw5507601  ur64360477      remzisiyer
509  rw3768042  ur71977754      BryanVigier
510  rw4154995  ur72332484      ramacriz
511  rw4075421  ur85886780      afroninjaen
512  rw4281029  ur91135415      brandontard

      Review_title  Review_rate \
0           Definitely does stay with you      7
1           weirdly intriguing                  7
2  An Addicting Mind-Bender from the Director of ...
3           Keep em close ...                  8
4           A disturbing psychological thriller      10

      Review_date       Review_body      MovieID
0   01/07/2008  I was completely astonished th...  tt0832266
1   13/04/2008  One guy and the four loves of ...  tt0832266
2   10/07/2010  "Definitely, Maybe" was market...
3   09/04/2008  I just saw this movie today an...
4   21/03/2008  When his ten-year-old daughter...
..      ...
207  24/07/2008  Definitely, MaybeI just watche...
208  14/02/2009  I liked the lead character pla...
209  11/02/2009  I would likely have never seen...
210  19/07/2016  I had no expectations with thi...
211  21/02/2019  This movie is about Will Hayes...

[212 rows x 8 columns]

```

```

movie15 = movie14.toPandas()
filename = movie15['MovieID']
# Create initial index
reviewcount = 0
moviefull = []
moviefull = pd.DataFrame(data=moviefull)
#run for loop to process
for i in range(len(filename)):
    try:
        # Read file csv review
        df = pd.read_csv('Review/f' + review[filename[i]] + '.csv')
        # Get data without NaN values in User_ID, Review_rate column
        df1 = df.loc[df["User_ID"] != 'NaN']
        df2 = df1.loc[df1['Review_rate'] != 'NaN']
        # Processing Data: User_ID column
        df2["User_ID"] = df2["User_ID"].str.split("/").str.get(2)
        # Processing Data: Review_body column
        df2["Review_body"] = df2["Review_body"].str.split("\n").str.get(1)
        # Processing Data: Review_date column
        df2["Review_date"] = pd.to_datetime(df2["Review_date"], format='%d %B %Y').dt.strftime('%d/%m/%Y')
        # Processing Data: Review_rate column
        df2["Review_rate"] = df2["Review_rate"].str.split("/").str.get(0).astype(int)
        # Drop unknown column
        df2 = df2.drop('Unnamed: 0', axis=1)
        # Merge review movie file
        moviefull = pd.concat([moviefull, df2], axis=0, ignore_index=True)
    except:
        pass
#print & save file csv
print(moviefull)
print(moviefull.shape[0])
moviefull.to_csv('ReviewFixed/f' + 'ReviewFull.csv', header=True, index=False)

```

```

      Review_ID      User_ID      User_name \
0      rw3835213    ur20552756  TheLittleSongbird
1      rw3097874    ur2898520   SnoopyStyle
2      rw3106157    ur3947986   brando647
3      rw3062536    ur5876717   kosmasp
4      rw2979053    ur28528605  trublu215
...
808756  rw5325128  ur81544839   mikdel-45286
808757  rw4987325  ur41953923   jm3251
808758  rw4447649  ur23252629  vipul-chawla09
808759  rw5100334  ur24017656  andrea-kucis
808760  rw6307271  ur54284136  john-67-180222

                                         Review_title  Review_rate \
0                           Definitely does stay with you        7
1                           weirdly intriguing            7
2  An Addicting Mind-Bender from the Director of ...
3                           Keep em close ...
4                           A disturbing psychological thriller        9
...
808756  One of the best detective show ever        10
808757  Very good series but....                  8
808758  A Hidden gem on Netflix                   10
808759  Solid crime TV show                    7
808760  Hooked!                                9

808758  09/11/2018      This show is a perfect example...
808759  04/09/2019      Watchable british crime detect...
808760  25/11/2020      We've watched dozens of US pol...
                                         MovieID
0      tt2316411
1      tt2316411
2      tt2316411
3      tt2316411
4      tt2316411
...
808756  tt2303687
808757  tt2303687
808758  tt2303687
808759  tt2303687
808760  tt2303687

[808761 rows x 8 columns]
808761

```

## 2. Trực quan hóa và khám phá dữ liệu

### VISUALIZATION & EDA

```

import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt

```

```

# Libraries
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn import metrics

```

```

import io
movies = pd.read_csv('MovieFullPandas.csv')
print(movies)

```

```

      Unnamed: 0   MovieID          MovieName \
0           0   tt6796652    Christina Aguilera: Your Body
1           1   tt2479848        Goodnight Mr. Foot
2           2   tt1525580          Magic Camp
3           3   tt2343495  FOX 25th Anniversary Special
4           4   tt2366218     Sound of Heimat - Deutschland singt
...
181464     181464  tt2308711       Healed by Grace
181465     181465  tt2010939 One Track Heart: The Story of Krishna Das
181466     181466  tt1809365        The End of Our Lives
181467     181467  tt2583530        It Gets Better
181468     181468  tt1992268      Blutsbrüder teilen alles

      Year          Genre \
0   2012          Music
1   2012  Animation, Short, Comedy
2   2012  Documentary, Family
3   2012          NaN
4   2012          Documentary
...
181464  2012          ...
181465  2012  Documentary, Biography, Music
181466  2012          Drama
181467  2012  Drama, Romance
181468  2012  Drama, Music, War

      Content \
0  music video for "Your Body" by Christina Aguil...
1  Bigfoot checks into Dracula's resort,where he ...
2  Welcome to the real Hogwarts. To cope with the...
3  Add a Plot
4  Add a Plot
...
181464  Healed by Grace is a charming tale of a faith, ...
181465  Krishna Das is on a journey to India to discov...
181466  After attempting suicide, four people from dif...
181467  Add a Plot
181468  Set in the midst of World War 2 and interspers...

      Star  Rating  Votes \
0  Christina Aguilera,Ryan Paevey    7.1   67
1  Rose Abdo,Carey Burton    5.3  470
2  Jonah Conlin,Zach Ivins,Zoe Reiches,Reed Spool  6.2  241
3  Ryan Seacrest,Christina Applegate,Katey Sagal,...  6.3   54
4  NaN                           7.3   61
...
181464  Tommy Beardmore,Larry Bower,Mark S. Esch,Dija ...  5.9  162
181465  Krishna Das,Ram Dass,Rick Rubin,Neem Karoli Baba  7.1  192
181466  Carol Goans,Dave Huber,Sinsu Co,G. Lane    8.2   24
181467  Prama Imanotai,Nuntita Khampiranon,Penpak Siri...  7.1  123
181468  Lorenz Willkomm,Johannes Nussbaum,Benedikt Hös...

      Metascore          Director
0  NaN  Melina Matsoukas|
1  NaN  Genndy Tartakovsky|
2  NaN  Judd Ehrlich|
3  NaN  Louis J. Horvitz|
4  NaN  Directors:Arne Birkenstock,Jan Tengeler|Star:H...
...
181464  NaN  David Matthew Weese|
181465  49.0  Jeremy Frindell|
181466  NaN  Bryan Sandlin|
181467  NaN  Tanwarin Sukkaphisit|
181468  NaN  Wolfram Paulus|
```

[181469 rows x 11 columns]

## Movie rating Analysis

```

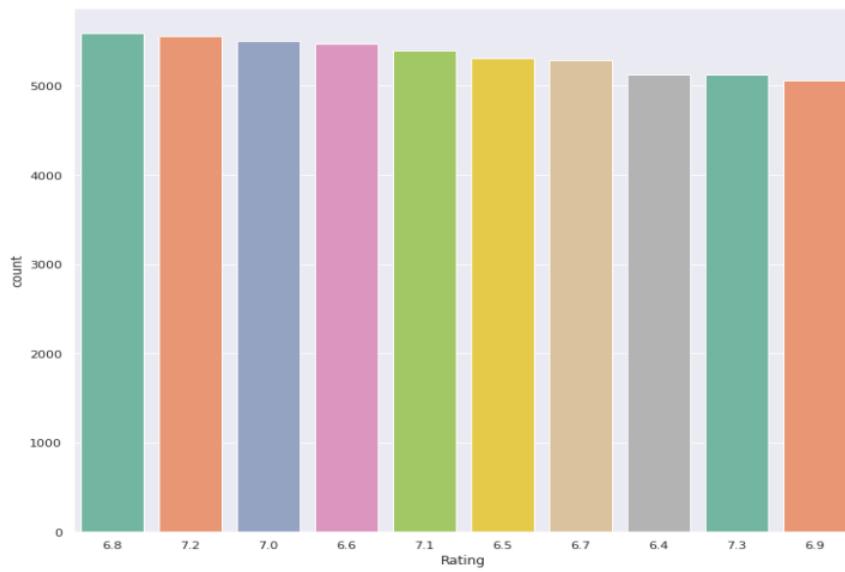
movies['Rating'].value_counts()

6.8    5593
7.2    5557
7.0    5506
6.6    5473
7.1    5398
...
9.9     63
1.3     57
1.1     39
1.2     39
1.0     36
Name: Rating, Length: 91, dtype: int64
```

```

plt.figure(figsize=(12,10))
sns.set(style="darkgrid")
ax = sns.countplot(x="Rating", data=movies, palette="Set2", order=movies['Rating'].value_counts().index[0:10])

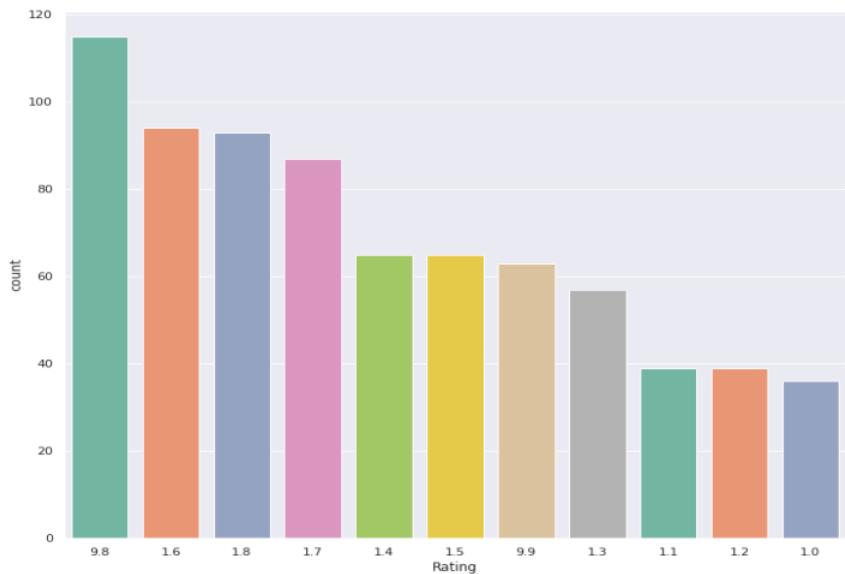
```

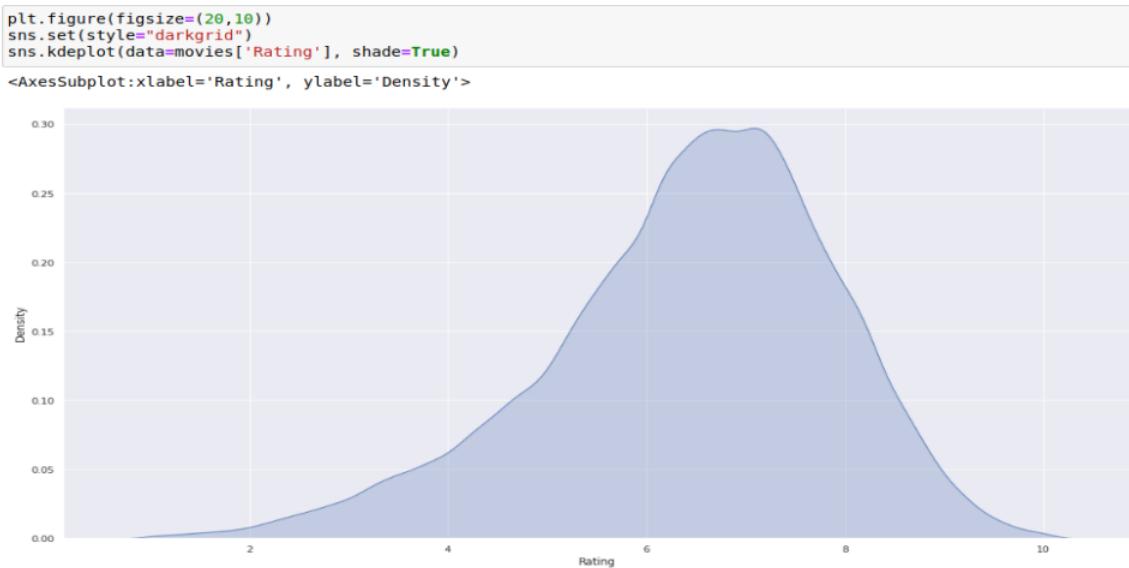


```

plt.figure(figsize=(12,10))
sns.set(style="darkgrid")
ax = sns.countplot(x="Rating", data=movies, palette="Set2", order=movies['Rating'].value_counts().index[80:91])

```





## Movie Genres Analysis

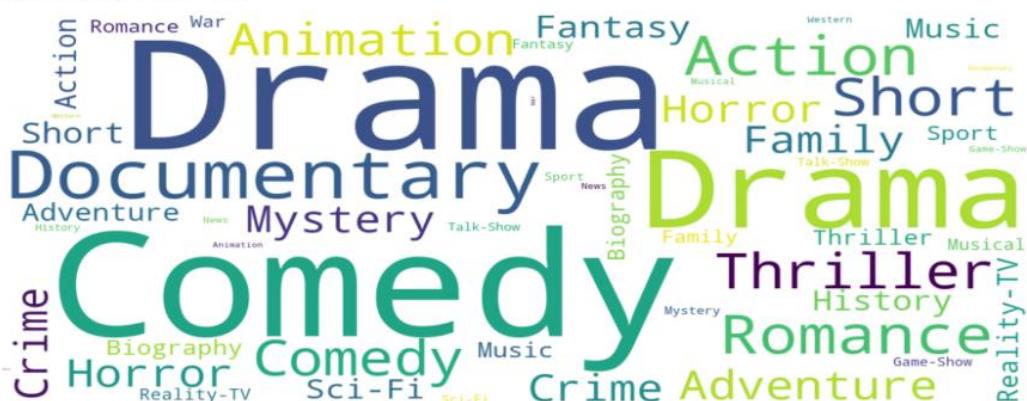
```
genre_popularity = (movies.Genre.str.split(',')
                     .explode()
                     .value_counts()
                     .sort_values(ascending=False))
genre_popularity.head(10)

Drama           37289
Comedy          33900
Drama           32582
Documentary     26024
Action           16254
Romance          15974
Short            15164
Comedy          13156
Thriller         13149
Animation        10592
Name: Genre, dtype: int64
```

```
#plot a word-cloud with the genres
genre_wc = WordCloud(width=1300,height=600,background_color='white')
genre_wc.generate_from_frequencies(genre_popularity.to_dict())  
#input data: genre_popularity.to_dict()

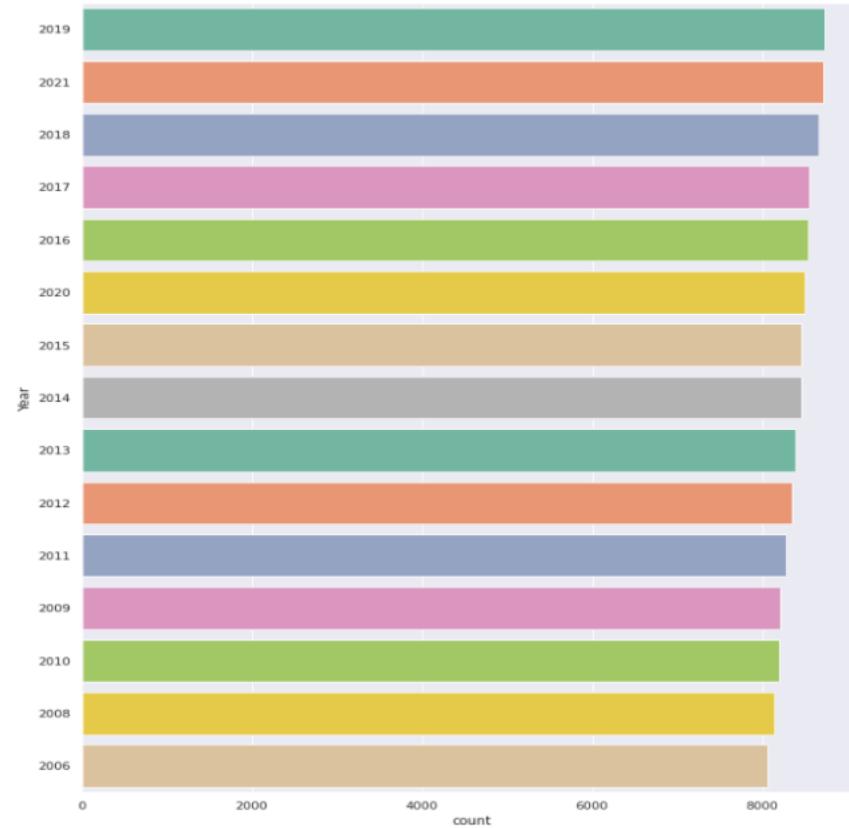
plt.figure(figsize=(16, 8))
plt.imshow(genre_wc, interpolation="bilinear")
plt.axis('off')

(-0.5, 1299.5, 599.5, -0.5)
```



## Year wise analysis

```
plt.figure(figsize=(12,15))
sns.set(style="darkgrid")
ax = sns.countplot(y="Year", data=movies, palette="Set2", order=movies['Year'].value_counts().index[0:15])
```

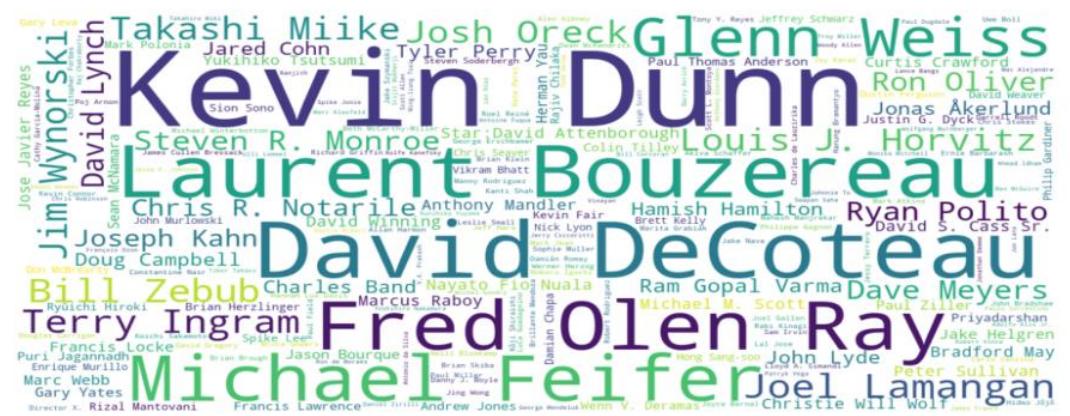


## Director Analysis

```
director_popularity = (movies.Director.str.split('|')
                        .explode()
                        .value_counts()
                        .sort_values(ascending=False))
director_popularity.head(15)
```

```
124756
Kevin Dunn      219
David DeCoteau   105
Laurent Bouzereau 91
Fred Olen Ray    78
Michael Feifer   69
Glenn Weiss      63
Joel Lamangan    59
Bill Zebub       56
Terry Ingram     55
Jim Wynorski     55
Takashi Miike    51
Louis J. Horvitz  51
Josh Oreck        50
Steven R. Monroe  50
Name: Director, dtype: int64
```

```
#plot a word-cloud with the Director
director_wc = WordCloud(width=1300,height=600,background_color='white')
director_wc.generate_from_frequencies(director_popularity.to_dict())
plt.figure(figsize=(16, 8)) #input data: director_popularity.to_dict()
plt.imshow(director_wc, interpolation="bilinear")
plt.axis('off')
```



### Star analysis

```
star_popularity = (movies.Star.str.split(',')
                   .explode()
                   .value_counts()
                   .sort_values(ascending=False))
star_popularity.head(15)
```

Eric Roberts	216
Prakash Raj	151
Grey Griffin	140
Brahmanandam	132
Michael Madsen	132
Steve Blum	127
Dee Bradley Baker	123
Laura Bailey	120
Troy Baker	116
Tom Sizemore	116
Danny Trejo	115
Vivica A. Fox	112
John Cena	108
Dean Cain	108
Takahiro Sakurai	103

Name: Star, dtype: int64

```
#plot a word-cloud with the Star
star_wc = WordCloud(width=1300,height=600,background_color='white')
star_wc.generate_from_frequencies(star_popularity.to_dict()) #input data: star_popularity.to_dict()
```



## Analysing Metascore ratings to get top rated movies

```

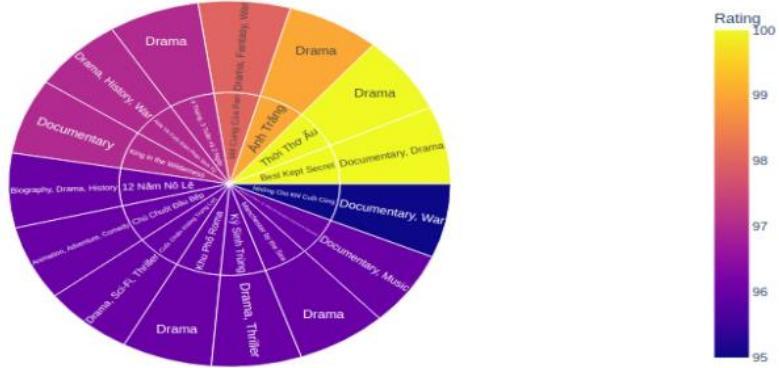
: Metascore_ratings=pd.read_csv('MovieFullPandas.csv',usecols=['Metascore'])
Metascore_Name=pd.read_csv('MovieFullPandas.csv', usecols=['MovieID','MovieName','Genre'],
                           'Rating': Metascore_ratings.Metascore,
                           'Genre':Metascore_Name.Genre)
ratings.drop_duplicates(subset=['MovieName','Rating'], inplace=True)
ratings.shape
: (169623, 3)

: ratings.dropna()
joint_data=ratings.sort_values(by='Rating', ascending=False)

: import plotly.express as px
top_rated=joint_data[0:15]
fig =px.sunburst(
    top_rated,
    path=['MovieName','Genre'],
    values='Rating',
    color='Rating')
fig.show()

/home/user/anaconda3/lib/python3.9/site-packages/plotly/express/_core.py:1637: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  df_all_trees = df_all_trees.append(df_tree, ignore_index=True)
/home/user/anaconda3/lib/python3.9/site-packages/plotly/express/_core.py:1637: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  df_all_trees = df_all_trees.append(df_tree, ignore_index=True)

```



## Highest number of votes

```

#Top 15 Highest number of votes
top_votes=movies.sort_values(by='Votes')
top_votes=top_votes[181453:181468]

import plotly.graph_objects as go

fig = go.Figure(data=[go.Table(header=dict(values=['Movie Name', 'Number of Votes']),
                                cells=dict(values=[top_votes['MovieName'],top_votes['Votes']],
                                           fill_color='lavender'))])
fig.show()

```

Movie Name	Number of Votes
Đảo Kinh Hoàng	1305595
Điệp Vũ Boston	1319246
Áo Thuật Gia Đầu Trí	1327048
Avengers: Biệt Đội Siêu Anh Hùng	1386728
Sói Già Phố Wall	1397929
Dịnh Mệnh	1440178
Người Dơi Xuất Hiện	1464679
Hành Trình Django	1543743
Chùa Tề Của Những Chiếc Nhẫn: Hai Tòa Tháp	1658555
Kỵ Sĩ Bóng Đêm Trỗi Dậy	1697312
Hổ Đen Tử Thần	1810970
Chùa Tề Của Những Chiếc Nhẫn: Sự Quay Trở Lại Của Nhà Vua	1836797
Tập Lãm Người Xấu	1866698
Trò Chơi Vương Quyền	2085514
Ké Đánh Cắp Giác Mơ	2336958

## **Group of Genres in 2002**

```
movies_fr=movies[movies['Year']=='2002']
nannef=movies_fr.dropna()

import plotly.express as px
fig = px.treemap(nannef, path=['Year', 'Genre'],
                  color='Genre', hover_data=['Genre', 'MovieName'],color_continuous_scale='Purples')
fig.show()

/home/user/anaconda3/lib/python3.9/site-packages/plotly/express/_core.py:1637: FutureWarning:
The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
/home/user/anaconda3/lib/python3.9/site-packages/plotly/express/_core.py:1637: FutureWarning:
The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
```



### 3. Mô hình TF-IDF

## TF-IDF

## Hoan toan bang Pyspark

```
movies = spark.read.csv('PandasMovieTest.csv',header=True, inferSchema=True)
movies.show(4)

22/12/16 17:13:28 WARN CSVHeaderChecker: CSV header does not conform to the schema.
Header: , MovieID, MovieName, Year, Genre, Content, Star, Rating, Votes, Metascore
Schema: _c0, MovieID, MovieName, Year, Genre, Content, Star, Rating, Votes, Metascore
Expected: _c0 but found:
CSV file: file:///usr/local/spark/FInal%20Project/PandasMovieTest.csv
+---+---+---+---+---+---+---+---+
|_c0| MovieID|      MovieName|      Year|      Genre|      Content|      Star|Rating|
+---+---+---+---+---+---+---+---+
| 0|tt0306414|  Dương Dây Tội Phạm| (2002–2008)|Crime, Drama, Thr...|The Baltimore dru...|Stars:Dominic Wes...|
9.3|339003|      NaN|          |          |          |          |          |
| 1|tt0145487|          Người Nhện| (2002)|Action, Adventure...|After being bitte...|Director:Sam Raim...|
7.4|815798|          73|          |          |          |          |          |
| 2|tt0304669| Ông Già Tuyết 2| (2002)|Comedy, Family, F...|Scott Calvin has ...|Director:Michael ...|
5.7|57807|          48|          |          |          |          |          |
| 3|tt0295297|Harry Potter và P...| (2002)|Adventure, Family...|An ancient prophe...|Director:Chris Co...|
7.4|633363|          63|          |          |          |          |          |
+---+---+---+---+---+---+---+---+
only showing top 4 rows
```

## Tokenization

```
from pyspark.ml.feature import HashingTF, IDF, Tokenizer

tokenizer = Tokenizer(inputCol="Genre", outputCol="GenRewords")
wordsData = tokenizer.transform(movies)
wordsData.show(truncate=False)

22/12/16 17:14:23 WARN CSVHeaderChecker: CSV header does not conform to the schema.
Header: , MovieID, MovieName, Year, Genre, Content, Star, Rating, Votes, Metascore
Schema: _c0, MovieID, MovieName, Year, Genre, Content, Star, Rating, Votes, Metascore
Expected: _c0 but found:
CSV file: file:///usr/local/spark/Final%20Project/PandasMovieTest.csv
+---+-----+-----+-----+-----+-----+-----+-----+
|_c0|MovieID |MovieName |Year |Genre |Content |
|Star |
|Rating|Votes |Metascore|GenRewords |
+---+-----+-----+-----+-----+-----+-----+-----+
+---+-----+-----+-----+-----+-----+-----+-----+
|0 |tt0306414|Đường Dây Tội Phạm |(2002–2008)|Crime, Drama, Thriller |The Baltimore |
+---+-----+-----+-----+-----+-----+-----+-----+
```

## Compute TF - IDF

```
from pyspark.ml.feature import HashingTF, IDF

hashingTF = HashingTF(inputCol="GenRewords", outputCol="tf")
tf = hashingTF.transform(wordsData)

idf = IDF(inputCol="tf", outputCol="feature").fit(tf)
tfidf = idf.transform(tf)
tfidf.show(4)

22/12/16 17:14:41 WARN DAGScheduler: Broadcasting large task binary with size 4.0 MiB
22/12/16 17:14:41 WARN CSVHeaderChecker: CSV header does not conform to the schema.
Header: , MovieID, MovieName, Year, Genre, Content, Star, Rating, Votes, Metascore
Schema: _c0, MovieID, MovieName, Year, Genre, Content, Star, Rating, Votes, Metascore
Expected: _c0 but found:
CSV file: file:///usr/local/spark/Final%20Project/PandasMovieTest.csv
+---+-----+-----+-----+-----+-----+-----+-----+-----+
|_c0| MovieID| MovieName| Year| Genre| Content| Star|Rat
ing| Votes|Metascore| GenRewords| tf| feature|
+---+-----+-----+-----+-----+-----+-----+-----+-----+
+---+-----+-----+-----+-----+-----+-----+-----+
| 0|tt0306414| Đường Dây Tội Phạm|(2002–2008)|Crime, Drama, Thr...|The Baltimore dru...|Stars:Dominic Wes...
9.3|339003| NaN|[crime,, drama,, ...|(262144,[31794,65...|(262144,[31794,65...
| 1|tt0145487| Người Nhện| (2002)|Action, Adventure...|After being bitte...|Director:Sam Raim...
7.4|815798| 73|[action,, adventu...|(262144,[142067,1...|(262144,[142067,1...
| 2|tt0304669| Ông Già Tuyết 2| (2002)|Comedy, Family, F...|Scott Calvin has ...|Director:Michael ...
5.7| 57807| 48|[comedy,, family,...|(262144,[94958,16...|(262144,[94958,16...
| 3|tt0295297|Harry Potter và P...| (2002)|Adventure, Family...|An ancient prophe...|Director:Chris Co...
7.4|633363| 63|[adventure,, fami...|(262144,[142067,1...|(262144,[142067,1...
+---+-----+-----+-----+-----+-----+-----+-----+
only showing top 4 rows
```

## Compute L2 norm

```
from pyspark.ml.feature import Normalizer

normalizer = Normalizer(inputCol="feature", outputCol="norm")
data = normalizer.transform(tfidf)
data.show(4)

22/12/16 17:15:05 WARN DAGScheduler: Broadcasting large task binary with size 4.1 MiB
22/12/16 17:15:05 WARN CSVHeaderChecker: CSV header does not conform to the schema.
Header: , MovieID, MovieName, Year, Genre, Content, Star, Rating, Votes, Metascore
Schema: _c0, MovieID, MovieName, Year, Genre, Content, Star, Rating, Votes, Metascore
Expected: _c0 but found:
CSV file: file:///usr/local/spark/Final%20Project/PandasMovieTest.csv
+---+-----+-----+-----+-----+-----+-----+-----+-----+
|_c0| MovieID| MovieName| Year| Genre| Content| Star|Rat
ing| Votes|Metascore| GenRewords| tf| feature| norm|
+---+-----+-----+-----+-----+-----+-----+-----+-----+
+---+-----+-----+-----+-----+-----+-----+-----+
| 0|tt0306414| Đường Dây Tội Phạm|(2002–2008)|Crime, Drama, Thr...|The Baltimore dru...|Stars:Dominic Wes...
9.3|339003| NaN|[crime,, drama,, ...|(262144,[31794,65...|(262144,[31794,65...
| 1|tt0145487| Người Nhện| (2002)|Action, Adventure...|After being bitte...|Director:Sam Raim...
7.4|815798| 73|[action,, adventu...|(262144,[142067,1...|(262144,[142067,1...
| 2|tt0304669| Ông Già Tuyết 2| (2002)|Comedy, Family, F...|Scott Calvin has ...|Director:Michael ...
5.7| 57807| 48|[comedy,, family,...|(262144,[94958,16...|(262144,[94958,16...
| 3|tt0295297|Harry Potter và P...| (2002)|Adventure, Family...|An ancient prophe...|Director:Chris Co...
7.4|633363| 63|[adventure,, fami...|(262144,[142067,1...|(262144,[142067,1...
+---+-----+-----+-----+-----+-----+-----+-----+
only showing top 4 rows
```

```

: #Get brief dataframe
datatfidf = data[["_c0","MovieID","norm"]
datatfidf.show(4)

22/12/16 17:15:11 WARN DAGScheduler: Broadcasting large task binary with size 4.0 MiB
22/12/16 17:15:11 WARN CSVHeaderChecker: CSV header does not conform to the schema.
Header: , MovieID, Genre
Schema: _c0, MovieID, Genre
Expected: _c0 but found:
CSV file: file:///usr/local/spark/Final%20Project/PandasMovieTest.csv
+-----+-----+
|_c0| MovieID|      norm|
+-----+-----+
| 0|tt0306414|(262144,[31794,65...|
| 1|tt0145487|(262144,[142067,1...|
| 2|tt0304669|(262144,[94958,16...|
| 3|tt0295297|(262144,[142067,1...|
+-----+-----+
only showing top 4 rows

```

### Create matrix for cosine similarity

```

import pyspark.sql.functions as psf
from pyspark.sql.types import DoubleType

dot_udf = psf.udf(lambda x,y: float(x.dot(y)), DoubleType())
final = data.alias("i").join(data.alias("j"), psf.col("i.MovieID") < psf.col("j.MovieID"))\
    .select(
        psf.col("i.MovieID").alias("i"),
        psf.col("j.MovieID").alias("j"),
        dot_udf("i.norm", "j.norm").alias("dot"))\
    .sort("i", "j")
final.show(10)
final.describe().show(4)

22/12/16 17:15:22 WARN DAGScheduler: Broadcasting large task binary with size 4.0 MiB
22/12/16 17:15:23 WARN DAGScheduler: Broadcasting large task binary with size 4.1 MiB

```

```

+-----+-----+-----+
|     i|     j|      dot|
+-----+-----+-----+
|tt0120679|tt0120804|      0.0|
|tt0120679|tt0120912|      0.0|
|tt0120679|tt0121765|      0.0|
|tt0120679|tt0133240|      0.0|
|tt0120679|tt0145487|      0.0|
|tt0120679|tt0159365|0.08751282205171346|
|tt0120679|tt0160984|      0.0|
|tt0120679|tt0164184|0.14154600294065997|
|tt0120679|tt0166813|      0.0|
|tt0120679|tt0167190|      0.0|
+-----+-----+-----+
only showing top 10 rows

```

22/12/16 17:15:30 WARN DAGScheduler: Broadcasting large task binary with size 4.0 MiB  
22/12/16 17:15:30 WARN DAGScheduler: Broadcasting large task binary with size 4.1 MiB

```

22/12/16 17:15:36 WARN DAGScheduler: Broadcasting large task binary with size 4.1 MiB
[Stage 12:>                               (0 + 1) / 1]
22/12/16 17:15:40 WARN DAGScheduler: Broadcasting large task binary with size 4.1 MiB
[Stage 14:>                               (0 + 1) / 1]

22/12/16 17:15:42 WARN DAGScheduler: Broadcasting large task binary with size 4.1 MiB
+-----+-----+-----+
|summary|     i|     j|      dot|
+-----+-----+-----+
| count| 44850| 44850| 44850|
| mean|   null|   null|0.12389136674888783|
| stddev|  null|   null|0.22871699754155403|
| min|tt0120679|tt0120804|      0.0|
+-----+-----+-----+
only showing top 4 rows

```

## Create function

```

: def genre_recommendations(i, M, items, k=20):
    """
    Recommends movies based on a similarity dataframe
    Parameters
    -----
    i : str - MovieName
        Movie title (index of the similarity dataframe)
    M : pd.DataFrame - final
        Similarity dataframe, symmetric, with movies as indices and columns
    items : pd.DataFrame - movies
        Contains both the title and some other features used to define similarity
    k : int
        Amount of recommendations to return
    """
    a = items.select('MovieID').where(items.MovieName == i).collect()[0]
    list1 = M.select('j','dot').where(items.MovieID == a[0])
    finalist = list1.join(items, list1.j == items.MovieID, "inner")
    finalist = finalist.select('MovieID','MovieName','Genre','dot')
    finalist = finalist.selectExpr("MovieID","MovieName","Genre","dot as Similarity")
    finalist.orderBy(finalist.Similarity.desc()).show(k)

: # Test function
genre_recommendations("Đường Dây Tội Phạm", final, movies, k=10)
22/12/16 17:25:26 WARN DAGScheduler: Broadcasting large task binary with size 4.0 MiB
22/12/16 17:25:26 WARN DAGScheduler: Broadcasting large task binary with size 4.1 MiB
[Stage 25:>                                         (0 + 1) / 1]
+-----+
| MovieID|      MovieName|       Genre|      Similarity|
+-----+
|tt0363589| Elephant|Crime, Drama, Thr...|          1.0|
|tt0375679|       Đô Võ|Crime, Drama, Thr...|          1.0|
|tt0315733|Những Mảnh Dời Bâ...|Crime, Drama, Thr...|          1.0|
|tt0313542|Bởi Thành Đoàn Châ...|Crime, Drama, Thr...|          1.0|
|tt0349903| 12 Tên Cuồng Thủ Ký|Crime, Thriller|0.8989279823518066|
|tt0361862|       Gã Thợ Máy|Drama, Thriller|0.8155930051332752|
|tt0317740|Phi Vũ Cuồng Cứng ...|Action, Crime, Th...|0.8880715174550896|
|tt0378194| Cô Dâu Bão Thủ 2|Action, Crime, Th...|0.8880715174550896|
|tt0322259|Quá Nhanh Quá Ngu...|Action, Crime, Th...|0.8880715174550896|
|tt0314353|   Nước Mát Mát Trời|Action, Drama, Th...|0.7184988225401572|
+-----+
only showing top 10 rows

```

## 4. Mô hình SVD

### SINGULAR VALUE DECOMPOSITION (SVD)

Using Pandas

```

import numpy as np
import pandas as pd

movies=pd.read_csv("movies.csv")
movies

      movieId          title           genres
0         1     Toy Story (1995)  Adventure|Animation|Children|Comedy|Fantasy
1         2        Jumanji (1995)  Adventure|Children|Fantasy
2         3  Grumpier Old Men (1995)  Comedy|Romance
3         4      Waiting to Exhale (1995)  Comedy|Drama|Romance
4         5  Father of the Bride Part II (1995)  Comedy
...
9737  193581  Black Butler: Book of the Atlantic (2017)  Action|Animation|Comedy|Fantasy
9738  193583  No Game No Life: Zero (2017)  Animation|Comedy|Fantasy
9739  193585            Flint (2017)  Drama
9740  193587  Bungo Stray Dogs: Dead Apple (2018)  Action|Animation
9741  193609  Andrew Dice Clay: Dice Rules (1991)  Comedy

```

9742 rows × 3 columns

```

movies.columns
Index(['movieId', 'title', 'genres'], dtype='object')

```

```
ratings=pd.read_csv("ratings.csv")
ratings
```

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931
...	...	...	...	...
100831	610	166534	4.0	1493848402
100832	610	168248	5.0	1493850091
100833	610	168250	5.0	1494273047
100834	610	168252	5.0	1493846352
100835	610	170875	3.0	1493846415

100836 rows × 4 columns

```
ratings.columns
```

```
Index(['userId', 'movieId', 'rating', 'timestamp'], dtype='object')
```

## Format Matrix

```
movies_ratings=pd.merge(ratings,movies,how='outer',on='movieId')
movies_ratings
```

	userId	movieId	rating	timestamp	title		genres
0	1.0	1	4.0	9.649827e+08	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	
1	5.0	1	4.0	8.474350e+08	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	
2	7.0	1	4.5	1.1066366e+09	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	
3	15.0	1	2.5	1.510578e+09	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	
4	17.0	1	4.5	1.3056966e+09	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	
...	...	...	...	...	...	...	...
100849	NaN	30892	NaN	NaN	In the Realms of the Unreal (2004)	Animation Documentary	
100850	NaN	32160	NaN	NaN	Twentieth Century (1934)	Comedy	
100851	NaN	32371	NaN	NaN	Call Northside 777 (1948)	Crime Drama Film-Noir	
100852	NaN	34482	NaN	NaN	Browning Version, The (1951)	Drama	
100853	NaN	85565	NaN	NaN	Chalet Girl (2011)	Comedy Romance	

100854 rows × 6 columns

```
matrix_format=ratings.pivot(index = 'userId', columns ='movieId', values = 'rating').fillna(0)
matrix_format
```

	movied	1	2	3	4	5	6	7	8	9	10	...	193565	193567	193571	193573	193579	193581	193583	193585	193587	193609
userId																						
1	4.0	0.0	4.0	0.0	0.0	4.0	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	
2	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	
3	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	
4	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	
5	4.0	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	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
606	2.5	0.0	0.0	0.0	0.0	0.0	2.5	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	
607	4.0	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	
608	2.5	2.0	2.0	0.0	0.0	0.0	0.0	0.0	4.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
609	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
610	5.0	0.0	0.0	0.0	0.0	5.0	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	

610 rows × 9724 columns

```
#Chuyển đổi thành ma trận
matrix=np.asmatrix(matrix_format)
matrix
```

```
matrix([[4. , 0. , 4. , ..., 0. , 0. , 0. ],
       [0. , 0. , 0. , ..., 0. , 0. , 0. ],
       [0. , 0. , 0. , ..., 0. , 0. , 0. ],
       ...,
       [2.5, 2. , 2. , ..., 0. , 0. , 0. ],
       [3. , 0. , 0. , ..., 0. , 0. , 0. ],
       [5. , 0. , 0. , ..., 0. , 0. , 0. ]])
```

## Normalize data

```
: user_mean_ratings=np.mean(matrix,axis=1)
user_mean_ratings
[ 0.01162073,
  0.01676265,
  0.04787125,
  0.03753599,
  0.04545455,
  0.19266763,
  0.18850267,
  0.08936652,
  0.14854998,
  0.03146853,
  0.04540313,
  0.04128599,
  0.01285479,
  0.00699301,
  0.04925956,
  0.1770362 ,
  0.03450226,
  0.01655697,
  0.02015631,
  0.03938708].
```

```
Demeaned_Data=matrix-user_mean_ratings
Demeaned_Data
matrix([[ 3.89582476, -0.10417524,  3.89582476, ..., -0.10417524,
         0.10417524, -0.10417524], [-0.01177499, -0.01177499, -0.01177499, ..., -0.01177499,
        -0.01177499, -0.01177499], [-0.00976964, -0.00976964, -0.00976964, ..., -0.00976964,
        -0.00976964, -0.00976964], ... , [ 2.23215755,  1.73215755,  1.73215755, ..., -0.26784245,
        -0.26784245, -0.26784245], [ 2.98755656, -0.01244344, -0.01244344, ..., -0.01244344,
        -0.01244344, -0.01244344], [ 4.50611888, -0.49388112, -0.49388112, ..., -0.49388112,
        -0.49388112, -0.49388112]])
```

## Compute matrix

```
#Matrix Factorisation using singular value decomposition
from scipy.sparse.linalg import svds
U, sigma, Vt = svds(Demeaned_Data, k = 50)
```

```
U
array([[-0.01716626,  0.00032694,  0.01194615, ...,  0.00335838,
       -0.06213084,  0.0596384 ], [-0.00501697, -0.0010387 ,  0.01505205, ..., -0.0012344 ,
       0.01767432,  0.00626322], [ 0.0065179,  0.00538272, -0.00649223, ...,  0.00067537,
       -0.00203417,  0.00064958], ... , [-0.16117225,  0.13376441, -0.05557292, ..., -0.01464468,
       -0.01227985,  0.11854893], [ 0.0089301 , -0.00652333, -0.00537419, ..., -0.04097903,
       -0.01400112,  0.00856716], [ 0.01039005, -0.00880112,  0.06333814, ...,  0.06183579,
       0.20316391,  0.12143586]])
```

```
sigma
array([ 67.86628347,  68.1967072 ,  69.02678246,  69.4170401 ,
       69.91863747,  70.02091789,  70.19468599,  71.67445157,
      72.43371861,  73.21879553,  73.43760593,  74.02644882,
      74.28978377,  74.9207733 ,  75.17528213,  75.59325141,
      76.70227225,  77.35717925,  78.39405157,  79.04344482,
      79.21217131,  80.56747647,  81.5467832 ,  82.1973482 ,
      83.04447645,  85.11688914,  85.74871886,  86.51711471,
      87.91550637,  90.33575237,  90.9348682 ,  92.26271695,
      93.39976829,  97.10067118,  99.28986754,  99.82361796,
     101.84794614, 105.97367358, 107.04782929, 109.20838712,
     112.80840902, 120.61532345, 122.64724436, 134.58721632,
     139.637245 , 153.93097112, 163.73084057, 184.86187801,
     231.22453421, 474.20606204])
```

```
#Sigma returned should be converted to diagonal matrix
sigma=np.diag(sigma)
sigma
array([[ 67.86628347,   0.          ,   0.          ,   0.          ,   0.          ,   0.          ],
       [   0.          ,  68.1967072 ,   0.          ,   0.          ,   0.          ,   0.          ],
       [   0.          ,   0.          ,  69.02678246,   0.          ,   0.          ,   0.          ],
       [   0.          ,   0.          ,   0.          ,  69.4170401 ,   0.          ,   0.          ],
       [   0.          ,   0.          ,   0.          ,   0.          ,  70.02091789,   0.          ],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  70.19468599],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  71.67445157],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  72.43371861],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  73.21879553],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  73.43760593],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  74.02644882],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  74.28978377],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  75.17528213],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  75.59325141],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  76.70227225],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  77.35717925],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  78.39405157],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  79.04344482],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  79.21217131],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  80.56747647],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  81.5467832 ],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  82.1973482 ],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  83.04447645],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  85.11688914],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  85.74871886],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  86.51711471],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  87.91550637],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  90.33575237],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  90.9348682 },
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  92.26271695],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  93.39976829],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  97.10067118],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  99.28986754],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          ,  99.82361796],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          , 101.84794614],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          , 105.97367358],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          , 107.04782929],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          , 109.20838712],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          , 112.80840902],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          , 120.61532345],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          , 122.64724436],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          , 134.58721632],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          , 139.637245 },
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          , 153.93097112],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          , 163.73084057],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          , 184.86187801],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          , 231.22453421],
       [   0.          ,   0.          ,   0.          ,   0.          ,   0.          , 474.20606204]])
```

```
array([[ 0.06053498e-02, -1.46261894e-03, -2.28232417e-03, ...  
       1.42764417e-03,  1.42764417e-03, -2.96452853e-03],  
      [-2.95078800e-02,  2.19714456e-02, -2.25072247e-02, ...  
       -2.92507189e-03, -2.92507189e-03,  9.95934144e-05],  
      [-6.65561487e-02, -1.43370497e-02,  2.64013814e-02, ...  
       -4.79377861e-04, -4.79377861e-04, -1.49239491e-03],  
      ...,  
      [-6.77263279e-02, -6.97142996e-02, -2.91611099e-02, ...  
       -2.24798857e-03, -2.24798857e-03, -2.06691001e-03],  
      [-2.84008740e-02, -2.36032577e-03, -2.47048049e-02, ...  
       7.01753154e-04,  7.01753154e-04,  1.36888991e-03],  
      [ 6.6098330e-02,  3.84874039e-02,  1.24439904e-02, ...  
       -5.10178162e-03, -5.10178162e-03, -4.81883637e-03]])
```

### **Make predictions from matrix**

```
# Predict the movie rating for every user
predicted_ratings=np.dot(np.dot(U, sigma), Vt) + user_mean_ratings.reshape(-1, 1)
predicted_ratings

matrix([[ 2.16732840e+00,  4.02750508e-01,  8.40183552e-01, ...,
       -2.34533753e-02, -2.34533753e-02, -5.87318552e-02],
       [ 2.11459069e-01,  6.65755884e-03,  3.34547997e-02, ...,
        1.94980595e-02,  1.94980595e-02,  3.22813825e-02],
       [ 3.58844848e-03,  3.05175179e-02,  4.63929239e-02, ...,
        5.90929301e-03,  5.90929301e-03,  8.00411072e-03],
       ...,
       [ 2.16136388e+00,  2.67091989e+00,  2.12845971e+00, ...,
        -4.40029476e-02, -4.40029476e-02,  7.18717825e-02],
       [ 7.80205947e-01,  5.33648654e-01,  9.64537701e-02, ...,
        4.35514249e-03,  4.35514249e-03, -1.34622131e-03],
       [ 5.36398127e+00, -3.40945139e-01, -1.75163291e-01, ...,
        -2.63577616e-02, -2.63577616e-02,  5.15415792e-02]])
```

## Make movie recommendation

```
#Construct predicted matrices
Predicted_Matrix= pd.DataFrame(data=predicted_ratings,columns=matrix_format.columns)
Predicted_Matrix
```

movield	1	2	3	4	5	6	7	8	9	10	...	193565	193567	193571	193
0	2.167328	0.402751	0.840184	-0.076281	-0.551337	2.504091	-0.890114	-0.026443	0.196974	1.593259	...	-0.023453	-0.019967	-0.026939	-0.026
1	0.211459	0.006658	0.033455	0.017419	0.183430	-0.062473	0.083037	0.024158	0.049330	-0.152530	...	0.019498	0.016777	0.022219	0.022
2	0.003588	0.030518	0.046393	0.008176	-0.006247	0.107328	-0.012416	0.003779	0.007297	-0.059362	...	0.005909	0.006209	0.005610	0.005
3	2.051549	-0.387104	-0.252199	0.087562	0.130465	0.270210	0.477835	0.040313	0.025858	-0.017365	...	0.004836	0.004172	0.005500	0.005
4	1.344738	0.778511	0.065749	0.111744	0.273144	0.584426	0.254930	0.128788	-0.085541	1.023455	...	-0.008042	-0.007419	-0.008664	-0.008
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
605	2.501444	-0.139015	-0.082080	0.079300	-0.158770	-0.587275	2.368039	-0.014790	-0.093695	-0.182211	...	-0.038935	-0.032500	-0.045369	-0.045
606	2.849138	1.368651	0.341869	0.000534	-0.272603	1.529573	-0.078889	-0.013913	0.075251	1.398731	...	0.006789	0.006108	0.007471	0.007
607	2.161364	2.670920	2.128460	0.036007	0.128314	3.684387	-0.028271	0.195936	0.073591	2.406330	...	-0.044003	-0.041123	-0.046882	-0.046
608	0.780206	0.533649	0.096454	0.029945	0.087756	0.209604	0.087704	0.061626	-0.064337	1.248235	...	0.004355	0.004268	0.004442	0.004442
609	5.363981	-0.340945	-0.175163	-0.032863	0.172089	5.394580	-0.024982	0.004594	0.087655	-0.237017	...	-0.026538	-0.021180	-0.031536	-0.031

610 rows x 9724 columns

```
#Creating a function that will return recommended Movies and User Data
def recommend_movies(preds_df, userID, movies_df, original_ratings_df, num_of_recomd=5):
    # Get and sort the user's predictions
    user_row_number = userID - 1 # As UserID starts at 1, not 0
    #Get the information from predicted Matrix of that particular User
    sorted_user_predictions = preds_df.iloc[user_row_number].sort_values(ascending=False)

    # Get the user's data from ratings dataset by comparing the userid and then merge in the movies dataframe on bas.
    user_data = original_ratings_df[original_ratings_df.userId == (userID)]
    user_full_Data= (user_data.merge(movies_df, how = 'left', left_on = 'movieId', right_on = 'movieId').
                     sort_values(['rating'], ascending=False))

    #Printing the predicted Movies
    print(f"User {userID} has already rated {user_full_Data.shape[0]} movies")
    print(f"Recommending the top {num_of_recomd} movies that you may like")
```

```
already_rated, recommend = recommend_movies(Predicted_Matrix, 200, movies ,ratings, num_of_recomd=10)
User 200 has already rated 334 movies
Recommendation 10 movies that you may like
```

already_rated					title	genres
	userid	movieId	rating	timestamp		
51	200	1196	5.0	1229886264	Star Wars: Episode V - The Empire Strikes Back...	Action Adventure Sci-Fi
307	200	51255	5.0	1229886899	Hot Fuzz (2007)	Action Comedy Crime Mystery
177	200	4308	5.0	1229886822	Moulin Rouge (2001)	Drama Musical Romance
226	200	6874	5.0	1229886192	Kill Bill: Vol. 1 (2003)	Action Crime Thriller
33	200	597	5.0	1229886223	Pretty Woman (1990)	Comedy Romance
...	...	...	...	...	...	...
89	200	1894	1.0	1229889198	Six Days Seven Nights (1998)	Adventure Comedy Romance
248	200	8638	1.0	1229877063	Before Sunset (2004)	Drama Romance
70	200	1438	1.0	1229889470	Dante's Peak (1997)	Action Thriller
98	200	2123	0.5	1229877257	All Dogs Go to Heaven (1989)	Animation Children Comedy Drama Fantasy
304	200	50601	0.5	1229878711	Bridge to Terabithia (2007)	Adventure Children Fantasy

334 rows × 6 columns

recommend						
	movieId	title	genres	Top Ratings		
19	356	Forrest Gump (1994)	Comedy Drama Romance War	5.488018		
190	4963	Ocean's Eleven (2001)	Crime Thriller	5.305434		
221	6539	Pirates of the Caribbean: The Curse of the Bla...	Action Adventure Comedy Fantasy	5.205945		
33	597	Pretty Woman (1990)	Comedy Romance	5.113426		
13	260	Star Wars: Episode IV - A New Hope (1977)	Action Adventure Sci-Fi	5.074770		
113	2571	Matrix, The (1999)	Action Sci-Fi Thriller	4.879106		
208	5816	Harry Potter and the Chamber of Secrets (2002)	Adventure Fantasy	4.772092		
131	2959	Fight Club (1999)	Action Crime Drama Thriller	4.608402		
54	1210	Star Wars: Episode VI - Return of the Jedi (1983)	Action Adventure Sci-Fi	4.568641		
189	4896	Harry Potter and the Sorcerer's Stone (a.k.a. ...	Adventure Children Fantasy	4.494244		

## 5. Đánh giá mô hình SVD

### EVALUATE SVD MODEL

```
!pip install surprise
Collecting surprise
  Downloading surprise-0.1-py2.py3-none-any.whl (1.8 kB)
Collecting scikit-surprise
  Downloading scikit-surprise-1.1.3.tar.gz (771 kB)
    |██████████| 771 kB 582 kB/s eta 0:00:01
Requirement already satisfied: joblib>=1.0.0 in /home/user/anaconda3/lib/python3.9/site-packages (from scikit-surprise->surprise) (1.1.0)
Requirement already satisfied: numpy>=1.17.3 in /home/user/anaconda3/lib/python3.9/site-packages (from scikit-surprise->surprise) (1.21.5)
Requirement already satisfied: scipy>=1.3.2 in /home/user/anaconda3/lib/python3.9/site-packages (from scikit-surprise->surprise) (1.7.3)
Building wheels for collected packages: scikit-surprise
  Building wheel for scikit-surprise (setup.py) ... done
    Created wheel for scikit-surprise: filename=scikit_surprise-1.1.3-cp39-cp39-linux_x86_64.whl size=1209884 sha256=0aa409cbf349b0a7a61c93ea52f3290e71aeff80f47d428dbebe926f9794f153
    Stored in directory: /home/user/.cache/pip/wheels/c6/3a/46/b17b3512bdf283c6cb84f59929cdd5199d4e754d596d22784
Successfully built scikit-surprise
Installing collected packages: scikit-surprise, surprise
Successfully installed scikit-surprise-1.1.3 surprise-0.1
```

```
from surprise import Dataset, Reader, SVD, accuracy
from surprise.model_selection import train_test_split, cross_validate
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
```

```
movies1 = pd.read_csv('movies.csv')
ratings1 = pd.read_csv('ratings.csv')
df = pd.merge(movies1, ratings1, how='inner', on='movieId')
```

```

reader = Reader(rating_scale=(0.5, 5))
data = Dataset.load_from_df(df[['userId', 'title', 'rating']], reader)
trainSet, testSet = train_test_split(data, test_size=.30, random_state=0)
print("Size of trainset: ", trainSet.n_ratings)
print("Size of testset: ", len(testSet))

Size of trainset: 70585
Size of testset: 30251

```

```

algo = SVD(random_state=0)

algo.fit(trainSet)
test_predictions = algo.test(testSet)
train_predictions = algo.test(trainSet.build_testset())

```

```

def MAE(predictions):
    return accuracy.mae(predictions, verbose=False)
def RMSE(predictions):
    return accuracy.rmse(predictions, verbose=False)

print("RMSE on training data : ", RMSE(train_predictions))
print("RMSE on test data : ", RMSE(test_predictions))
print("MAE on training data : ", MAE(train_predictions))
print("MAE on test data : ", MAE(test_predictions))

RMSE on training data : 0.637060884620494
RMSE on test data : 0.8752648509444276
MAE on training data : 0.4947182918554523
MAE on test data : 0.6730137009807415

```

```

cv = cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
cv

```

Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.8673	0.8760	0.8727	0.8732	0.8728	0.8724	0.0028
MAE (testset)	0.6683	0.6759	0.6689	0.6679	0.6701	0.6702	0.0029
Fit time	1.74	2.19	1.96	1.93	1.77	1.92	0.16
Test time	0.33	0.13	0.11	0.13	0.14	0.17	0.08

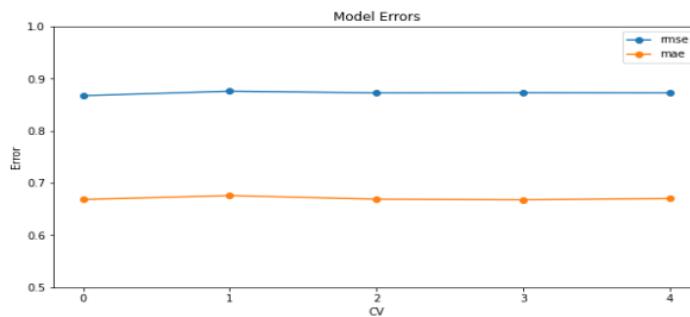
```

# Get data
rmse = cv['test_rmse']
mae = cv['test_mae']
x = np.arange(len(rmse))

# Set up the matplotlib figure
fig, ax = plt.subplots(figsize = (10, 5))
plt.xticks(np.arange(min(x), max(x) + 1, 1.0))
plt.ylim(0.5, 1.0)
ax.plot(x, rmse, marker='o', label="rmse")
ax.plot(x, mae, marker='o', label="mae")

# Chart setup
plt.title("Model Errors", fontsize = 12)
plt.xlabel("CV", fontsize = 10)
plt.ylabel("Error", fontsize = 10)
plt.legend()
plt.show()

```



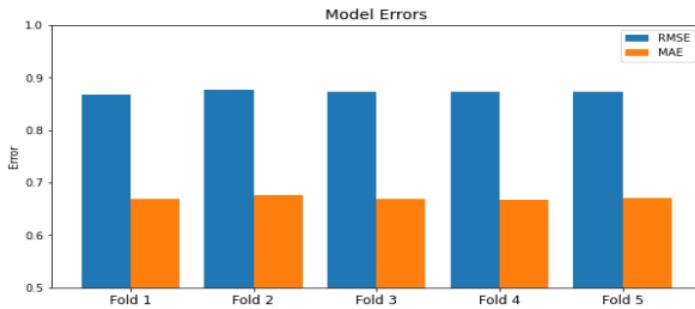
```

# Get data
rmse = cv['test_rmse']
mae = cv['test_mae']
x = np.arange(len(rmse))
width = 0.4
# Set up the matplotlib figure
fig, ax = plt.subplots(figsize = (10, 5))
rects1 = ax.bar(x - width/2, rmse, width,
                 label='RMSE')
rects2 = ax.bar(x + width/2, mae, width,
                 label='MAE')

# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Error')
ax.set_yticks(np.arange(0.5, 1.0))
ax.set_title('Model Errors', fontsize = 14)
ax.set_xticklabels(('Fold 1', 'Fold 2', 'Fold 3', 'Fold 4', 'Fold 5'), fontsize = 12)
ax.legend()

plt.show()

```



## 6. Đánh giá mô hình TF-IDF

### EVALUATION TF-IDF MODEL

```

import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

```

```

import pandas as pd
import seaborn as sns

from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

import re

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.naive_bayes import MultinomialNB

```

```

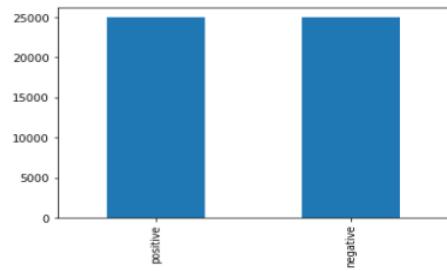
df = pd.read_csv('IMDB Dataset.csv')
df.shape
print(df)

review sentiment
0      One of the other reviewers has mentioned that ...  positive
1      A wonderful little production. <br /><br />The...  positive
2      I thought this was a wonderful way to spend ti...  positive
3      Basically there's a family where a little boy ...  negative
4      Petter Mattei's "Love in the Time of Money" is...  positive
...
49995  I thought this movie did a down right good job...  positive
49996  Bad plot, bad dialogue, bad acting, idiotic di...  negative
49997  I am a Catholic taught in parochial elementary...  negative
49998  I'm going to have to disagree with the previou...  negative
49999  No one expects the Star Trek movies to be high...  negative
[50000 rows x 2 columns]

```

```
df['sentiment'].value_counts().plot(kind='bar')
```

```
<AxesSubplot:>
```



## Data processing

```
lemmatizer = WordNetLemmatizer()

def cleantext(text):
    text= text.lower()
    text= re.sub(r"[^a-zA-Z?.!,?]+", " ", text)
    text= re.sub(r"http\S+", "",text)
    text= re.sub(r"http", "",text)

    punctuations= '@#!?+&*[]-%.:;/();$=><|{}^' + "'`" + '_'
    for p in punctuations:
        text= text.replace(p, '')

    text= [word.lower() for word in text.split() if word.lower() not in sw]
    text= [lemmatizer.lemmatize(word) for word in text]
    text = " ".join(text)

    emoji_pattern = re.compile("["
        u"\U0001F600-\U0001F64F"
        u"\U0001F300-\U0001F5FF"
        u"\U0001F680-\U0001F6FF"
        u"\U0001F1E0-\U0001F1FF"
        u"\U00002702-\U000027B0"
        u"\U000024C2-\U0001F251"
    "]+", flags=re.UNICODE)

    text= emoji_pattern.sub(r'',text)
    return text
```

## Train-test split

```
X= df['review']
y= df['sentiment']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.2, random_state= 26)
```

## TF-IDF Vectorize

```
tfidf_vectorizer = TfidfVectorizer()
tfidf_train_vectors = tfidf_vectorizer.fit_transform(X_train) #applying tf idf to training data
tfidf_test_vectors = tfidf_vectorizer.transform(X_test) #applying tf idf to training data
```

```
print("n_samples: %d n_features: %d" % tfidf_train_vectors.shape)
n_samples: 40000 n_features: 92981
```

```
naive_bayes_classifier = MultinomialNB()
naive_bayes_classifier.fit(tfidf_train_vectors, y_train)
MultinomialNB()
```

```

y_pred = naive_bayes_classifier.predict(tfidf_test_vectors)

print(classification_report(y_test,y_pred))
    precision    recall  f1-score   support

  negative      0.86      0.88      0.87     5067
  positive      0.88      0.85      0.86     4933

accuracy                           0.87
macro avg      0.87      0.87      0.87     10000
weighted avg    0.87      0.87      0.87     10000

cnf_matrix = confusion_matrix(y_test,y_pred)
cnf_matrix
array([[4470,  597],
       [ 749, 4184]])


group_names = ['TN','FP','FN','TP']
group_counts = ["{:0.0f}".format(value) for value in cnf_matrix.flatten()]
labels = [f'{v1}\n{v2}' for v1, v2 in zip(group_names,group_counts)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(cnf_matrix, annot=labels, fmt='', cmap='Blues');

```

## ĐÁNH GIÁ LÀM VIỆC NHÓM

MSSV	Họ Tên	Công việc chính	Mức độ đóng góp
K204110558	Vũ Thị Phương Anh (NT)	Phân công công việc, chạy mô hình TF-IDF	100%
K204111767	Nguyễn Thị Thanh Bình	Trực quan hóa dữ liệu, kiểm định mô hình TF-IDF	100%
K204111780	Lê Thị Thuỳ Linh	Chạy mô hình SVD	100%
K204111762	Đinh Thị Thúy An	Thu thập dữ liệu, tổng hợp báo cáo	100%
K204110587	Mai Lê Ngọc Trâm	Thu thập dữ liệu, tiền xử lý dữ liệu, kiểm định mô hình SVD	100%

## TÀI LIỆU THAM KHẢO

1. Hiếu, C. Đ. (2020). *Ứng dụng hệ thống gợi ý trong lĩnh vực thương mại điện tử*. Retrieved December 12, 2022 from [https://www.academia.edu/43464954/Bao\\_cao\\_do\\_an](https://www.academia.edu/43464954/Bao_cao_do_an)
2. Hrisav Bhowmick, A. C., Jaydip Sen. (2021). Comprehensive Movie Recommendation System. <https://arxiv.org/abs/2112.12463>
3. Dũng, T. A. (2020). *Ứng dụng các mô hình học sâu vào kỹ thuật lọc cộng tác dựa trên mô hình cho các hệ thống khuyến nghị thương mại*. Retrieved December 16, 2022 from <https://123docz.net/document/7429825-ung-dung-cac-mo-hinh-hoc-sau-vao-ki-thuat-loc-cong-tac-dua-tren-mo-hinh-cho-cac-he-thong-khuyen-nghi-thuong-mai.htm#fulltext-content>
4. Quang, B. N. L. (2019). Xây dựng một hệ thống gợi ý phim đơn giản với Python. Retrieved December 17, 2022 from <https://viblo.asia/p/xay-dung-mot-he-thong-goi-y-phim-don-gian-voi-python-eW65Ge1PZDO>
5. Triệu Vĩnh Viêm, T. Y. Y. v. N. T. N. (2013). Xây dựng hệ thống gợi ý phim dựa trên mô hình nhân tố láng giềng. (Số. CĐ Công nghệ TT 2013 (2013)), 170-179. Retrieved December 10, 2022, from <https://ctujsvn.ctu.edu.vn/index.php/ctujsvn/article/view/1918>
6. Nixon, A. E. (2020). *Building a movie content based recommender using tf-idf*. Retrieved December 15, 2022 from <https://towardsdatascience.com/content-based-recommender-systems-28a1dbd858f5>
7. Hrisav Bhowmick, A. C., Jaydip Sen. (2021). Comprehensive Movie Recommendation System. <https://arxiv.org/abs/2112.12463>
8. Trung, T. Đ. (2020). *Tìm hiểu chung về Web Scraping và các vấn đề cần quan tâm*. Retrieved December 12, 2022 from <https://viblo.asia/p/tim-hieu-chung-ve-web-scraping-vacac-van-de-can-quan-tam-djeZ1yXJZWz>
9. Dương, P. (2016). *TF-IDF ( term frequency – inverse document frequency)*. Retrieved December 16, 2022 from <https://viblo.asia/p/tf-idf-term-frequency-inverse-document-frequency-JQVkVZgKkyd>
10. Hiếu, N. V. (2019). *TF-IDF là gì?* Retrieved December 15, 2022 from <https://blog.luyencode.net/tf-idf-la-gi/>

11. Nam, L. (2020). *TF-IDF và vai trò của TF-IDF trong SEO* (2022). Retrieved December 16, 2022 from <https://vietmoz.edu.vn/tf-idf/>
12. Simha, A. (2021). *Understanding TF-IDF for Machine Learning*. Retrieved December 15, 2022 from <https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/>
13. *Extracting, transforming and selecting features*. (2017). Retrieved December 16, 2022 from <https://spark.apache.org/docs/latest/ml-features>
14. *Pyspark.sql module*. Retrieved December 16, 2022 from <https://spark.apache.org/docs/2.2.0/api/python/pyspark.sql.html#6>
15. Becker, N. (2016). *Matrix Factorization for Movie Recommendations in Python*. Retrieved December 15, 2022 from <https://beckernick.github.io/matrix-factorization-recommender/>
16. Rout, S. J. (2021). *Movie Recommendation System using SVD*. Retrieved December 17, 2022 from <https://github.com/Saijyoti52/Movie-Recommendation-System-using-SVD/blob/main/RECOMMENDATION%20SYSTEM.ipynb>
17. Thanh, N. Đ. *68 dòng code Python hay sử dụng xử lý dữ liệu trong Pandas*. [https://blog.hocexcel.online/68-dong-code-python-hay-su-dung-xu-ly-du-lieu-trong-pandas.html#Tim\\_gia\\_tri\\_do\\_lech\\_tieu\\_chuan\\_cho moi\\_cot](https://blog.hocexcel.online/68-dong-code-python-hay-su-dung-xu-ly-du-lieu-trong-pandas.html#Tim_gia_tri_do_lech_tieu_chuan_cho moi_cot)
18. Hegde, S. (2021). *Simple Movie Recommender Using SVD*. <https://pianalytix.com/simple-movie-recommender-using-svd/>
19. Huu, T. V. (2018). *Các phương pháp đánh giá một hệ thống phân lớp*. <https://machinelearningcoban.com/2017/08/31/evaluation/#-confusion-matrix>
20. vikashrajluhaniwal. (2020). Movie Recommendation using Surprise library. <https://www.kaggle.com/code/vikashrajluhaniwal/movie-recommendation-using-surprise-library>
21. AN, K. (2022). *Movie sentiment analysis with TF-IDF & Naive\_Bayes*. <https://www.kaggle.com/code/ankumagawa/movie-sentiment-analysis-with-tf-idf-naive-bayes/notebook>
22. Hug, N. (2022). *A Python scikit for recommender systems*. <https://surpriselib.com/>

23. Kaggle (2018), *IMDB Dataset of 50K Movie Reviews*.  
<https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews>
24. Kaggle (2022), *Movie Recommender System Dataset*,  
<https://www.kaggle.com/datasets/gargmanas/movierecommenderdataset>

## DỮ LIỆU TỰ THU THẬP

1. Dữ liệu thô:  
<https://drive.google.com/drive/u/0/folders/1GmnWneoWP44ry79zRegjeGZ4c6QthGGQ>
2. Dữ liệu qua xử lý:  
<https://drive.google.com/drive/folders/10gPKSFVIyhNEsDYWfaKmM0Mnx15U4ohn?usp=sharing>