**THÔNG TIN NHÓM**

|  |  |  |
| --- | --- | --- |
| **Tên đề tài** | TÍCH HỢP ELASTICSEARCH,  SPARK CHO BÀI TOÁN  MOVIE RECOMMENDATION | Đóng góp của thành viên |
| **Họ và tên thành viên 1-**  **MSSV** | Lô Minh Hiếu – 1050080098 | -Phân tích yêu cầu  -Thu thập và xử lý dữ liệu  -Xây dựng triển khai các thuật toán  -Phân tích và trình bày dữ liệu  -Phân tích và tối ưu hóa  Điểm tự đánh giá:5 |
| **Họ và tên thành viên 2-MSSV** | Hoàng Minh Chiến – 1050080091 | -Phân tích yêu cầu  -Thu thập và xử lý dữ liệu  -Xây dựng triển khai các thuật toán  -Phân tích và trình bày dữ liệu  -Phân tích và tối ưu hóa  Điểm tự đánh giá:5 |
| **Họ và tên thành viên 3-MSSV** | Đoàn Công Đức Dương – 1050080093 | -Phân tích yêu cầu  -Thu thập và xử lý dữ liệu  -Xây dựng triển khai các thuật toán  -Phân tích và trình bày dữ liệu  -Phân tích và tối ưu hóa  Điểm tự đánh giá:5 |

**BÁO CÁO CHI TIẾT**

**Lý do chọn đề tài:**

Nhóm 07 chọn đề tài Tích hợp Elasticsearch và Spark cho bài toán Movie Recommendation có một số lý do quan trọng:

1.Độ phức tạp của bài toán: Bài toán Movie Recommendation là một trong những bài toán phức tạp trong lĩnh vực học máy và khoa học dữ liệu. Để đưa ra các gợi ý phim hiệu quả, cần sử dụng nhiều dữ liệu khác nhau như lịch sử xem phim, thông tin về người dùng, thông tin về phim, v.v. Elasticsearch và Spark là hai công nghệ mạnh mẽ có thể được sử dụng để xử lý và truy vấn dữ liệu lớn một cách hiệu quả, giúp nâng cao chất lượng của các hệ thống gợi ý.

2.Elasticsearch cho tìm kiếm và lập chỉ mục: Elasticsearch là một hệ thống tìm kiếm và lập chỉ mục mạnh mẽ, giúp tìm kiếm dữ liệu nhanh chóng và hiệu quả. Bằng cách tích hợp Elasticsearch vào hệ thống, bạn có thể dễ dàng tìm kiếm thông tin về các phim, người dùng và lịch sử xem phim.

3.Spark cho xử lý dữ liệu lớn: Apache Spark là một framework xử lý dữ liệu lớn có khả năng xử lý hàng terabyte hoặc petabyte dữ liệu một cách hiệu quả. Việc tích hợp Spark cho phân tích và xử lý dữ liệu trong bài toán Movie Recommendation giúp bạn xử lý các tập dữ liệu lớn một cách nhanh chóng và hiệu quả hơn.

4.Kết hợp sức mạnh của hai công nghệ: Khi kết hợp Elasticsearch và Spark, bạn có thể tận dụng sức mạnh của cả hai công nghệ để xây dựng hệ thống gợi ý phim mạnh mẽ. Elasticsearch giúp bạn tìm kiếm và truy xuất thông tin nhanh chóng, trong khi Spark giúp bạn xử lý và phân tích dữ liệu một cách hiệu quả.

5.Cải thiện trải nghiệm người dùng: Khi kết hợp Elasticsearch và Spark, bạn có thể xây dựng hệ thống gợi ý phim có khả năng đưa ra các gợi ý chính xác và nhanh chóng, cải thiện trải nghiệm người dùng và tạo ra giá trị thực sự cho ứng dụng của mình.

Hơn nữa đề tài trên có tính ứng dụng cao, Movies Recommendation là một bài toán phổ biến trong lĩnh vực Tư vấn (Recommendation) và có độ ứng dụng cao trong thực tế, như Netflix, IMDb, hay Amazon Prime. Hiểu rõ về cách thức xây dựng một hệ thống đề xuất phim có thể mang lại nhiều.

Nhóm 07 chọn đề tài Movies recommendation với elasticsearch và spark, việc chọn đề tài "movies recommendation" với Elasticsearch và Spark mang lại nhiều lợi ích và cung cấp cơ hội để hiểu và áp dụng các công nghệ phân tán mạnh mẽ vào lĩnh vực recommendation.

Tóm lại, chọn đề tài Tích hợp Elasticsearch và Spark cho bài toán Movie Recommendation có lý do là hai công nghệ này cung cấp các công cụ mạnh mẽ để xử lý và truy vấn dữ liệu lớn, đáp ứng độ phức tạp của bài toán và cải thiện trải nghiệm người dùng.

**Tầm quan trọng:**

Tích hợp Elasticsearch và Spark cho bài toán Movie Recommendation có tầm quan trọng đáng kể trong nhiều khía cạnh:

Hiệu suất tìm kiếm và truy vấn: Elasticsearch là một hệ thống tìm kiếm và lập chỉ mục mạnh mẽ, cho phép bạn nhanh chóng tìm kiếm và truy vấn thông tin liên quan đến phim, người dùng và lịch sử xem phim. Điều này cải thiện hiệu suất và thời gian đáp ứng của hệ thống gợi ý phim, giúp người dùng nhận được các gợi ý nhanh chóng.

Xử lý dữ liệu lớn: Apache Spark là một framework xử lý dữ liệu lớn, giúp bạn xử lý và phân tích dữ liệu từ nhiều nguồn khác nhau. Trong bài toán Movie Recommendation, có rất nhiều dữ liệu, bao gồm thông tin về phim, người dùng và lịch sử xem phim. Spark cho phép bạn xử lý và phân tích dữ liệu lớn một cách hiệu quả.

Personalization: Mục tiêu của bài toán Movie Recommendation là cung cấp các gợi ý phim cá nhân dựa trên hành vi và sở thích của mỗi người dùng. Elasticsearch và Spark giúp bạn tạo ra các mô hình gợi ý cá nhân bằng cách phân tích dữ liệu lịch sử xem phim và các thông tin cá nhân của người dùng.

Độ chính xác: Tích hợp Elasticsearch và Spark có thể giúp tăng cường độ chính xác của hệ thống gợi ý phim. Elasticsearch giúp tìm kiếm thông tin phim một cách chính xác, trong khi Spark giúp tạo ra các mô hình gợi ý phim có khả năng đưa ra các gợi ý tốt hơn.

Skalability: Cả Elasticsearch và Spark đều có khả năng mở rộng, cho phép bạn xây dựng hệ thống có khả năng mở rộng ngang để xử lý dữ liệu lớn và số lượng người dùng lớn hơn mà không mất đi hiệu suất.

Cải thiện trải nghiệm người dùng: Bằng cách kết hợp Elasticsearch và Spark, bạn có thể cung cấp cho người dùng các gợi ý phim chính xác và thú vị, cải thiện trải nghiệm của họ và thúc đẩy sự tương tác với ứng dụng hoặc dịch vụ của bạn.

Các nội dung và kết quả:

* Mô tả bài toán.
* Giải pháp, mô hình.
* Các công nghệ sử dụng.
* Cài đặt Spark và Elasticsearch trên host local và trên cluster ambari.
* Demo.
* Đánh giá.
* Báo cáo chi tiết
* Giới thiệu big data

**Đinh nghĩa:**



Big data là thuật ngữ được sử dụng để miêu tả lượng lớn các dữ liệu phức tạp, đa dạng và có tốc độ nhanh, mà không thể được xử lý bởi các phương tiện và công cụ truyền thống. Big data có ba yếu tố chính, được gọi là 3V:

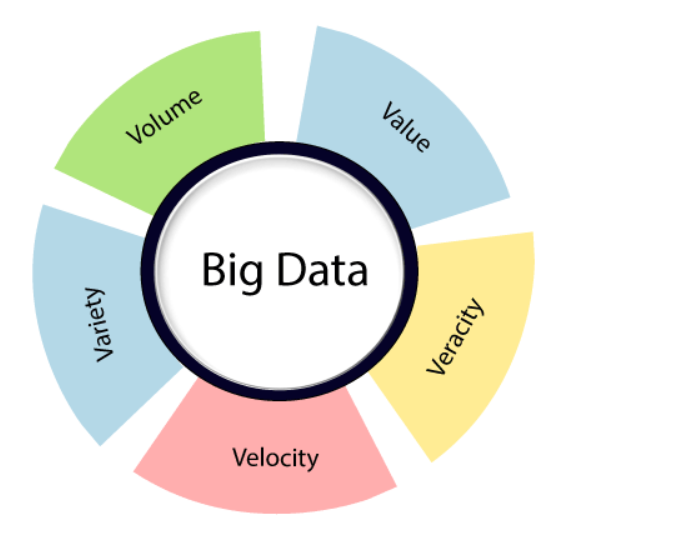
* Volume: Khối lượng - dữ liệu lớn không có mẫu; nó chỉ thực hiện và lần theo những gì diễn ra.
* Velocity: Tốc độ - dữ liệu lớn thường được xử lý thời gian thực.
* Variety: Đa dạng - dữ liệu lớn có thể thu thập từ văn bản, hình ảnh, âm thanh, video, cộng với nó hoàn thành các phần dữ liệu thiếu thông qua tổng hợp dữ liệu.

Các đặc điểm quan trọng khác của big data bao gồm tính thay đổi (dữ liệu liên tục được cập nhật và thay đổi liên tục), tính phức tạp (chứa các mối quan hệ phức tạp giữa các dữ liệu), tính chính xác (đảm bảo tính chính xác cao trong việc xử lý và phân tích dữ liệu) và tính mô hình (thường cho phép việc xây dựng các mô hình phân tích và dự đoán).

Big data có tiềm năng lớn để đem lại giá trị lớn cho các tổ chức và xã hội. Việc phân tích big data có thể cung cấp thông tin quan trọng và những mẫu cần thiết để đưa ra quyết định và thực hiện các hành động hiệu quả. Nghiên cứu big data có thể giúp tìm ra các xu hướng, mô hình và những thông tin tiềm ẩn trong dữ liệu, từ đó ứng dụng chúng vào các lĩnh vực như kinh doanh, y tế, khoa học, marketing và nhiều lĩnh vực khác.

Tuy nhiên, big data cũng đặt ra thách thức lớn đối với việc quản lý và xử lý dữ liệu. Vì lượng dữ liệu lớn, việc lưu trữ, truy xuất và phân tích big data yêu cầu sự đầu tư về hạ tầng phần cứng và phần mềm phức tạp. Ngoài ra, big data cũng đặt vấn đề về quyền riêng tư thông tin, bảo mật và đạo đức trong việc sử dụng các dữ liệu lớn.

**Đặt trưng của Big Data:**



Thông thường, Big Data được đặc trưng bởi bốn V, bao gồm:

**Volume** (Dung lượng)

Big data là thuật ngữ nói về khối lượng dữ liệu lớn, kích thước lớn. Xác định giá trị của dữ liệu và kích thước dữ liệu là rất quan trọng và cần thiết, nếu khối lượng lớn, đó chính là Big data.

Volume là khối lượng dữ liệu được các doanh nghiệp thu thập từ các nguồn khác nhau, như IoT (Internet of Things), video, giao dịch kinh doanh, các phương tiện truyền thông xã hội,...

Khi công nghệ chưa có sự phát triển vượt bậc, việc lưu trữ lượng lớn dữ liệu là một thách thức lớn. Tuy nhiên ngày nay, các nền tảng lưu trữ giá thành rẻ như Hadoop và Data lake xuất hiện, việc lưu trữ đã trở nên dễ dàng hơn nhiều.

**Variety** (Tính đa dạng)

Đặc trưng tiếp theo của Big data chính là tính đa dạng, linh hoạt, ở dạng cấu trúc và phi cấu trúc, bao gồm dữ liệu số, Email, Video, âm thanh, giao dịch tài chính,... Tính đa dạng ảnh hưởng đến hiệu suất, đây là một trong những vấn đề chính mà lĩnh vực Big data cần phải giải quyết.

**Velocity** (tốc độ)

Dựa vào tốc độ xử lý của luồng dữ liệu để xác định đó có phải là Big data hay không. Thường thì tốc độ của luồng dữ liệu trực tiếp vào bộ nhớ cao hơn so với khi được ghi vào đĩa. Đặc biệt là ngày nay, với sự phát triển của IoT, các luồng dữ liệu truyền tải với tốc độ cực nhanh và chúng phải được xử lý kịp thời.

Ví dụ: Trên mạng xã hội Facebook, các thông báo như status, tweet,... đã cũ sẽ không được người dùng quan tâm và bị quên lãng nhanh chóng. Dữ liệu giờ đây được tính gần như vào thời gian thực và tốc độ cập nhật thông tin dường như giảm xuống đơn vị mili giây.

\**Veracity** (Tính xác thực)

Tính xác thực của Big Data đề cập đến việc đảm bảo rằng dữ liệu được xử lý và sử dụng là chính xác, tin cậy và không bị biến đổi một cách không mong muốn. Điều này rất quan trọng trong việc đảm bảo sự tin cậy của thông tin trong môi trường dữ liệu lớn, nơi có rất nhiều nguồn dữ liệu khác nhau và quy trình xử lý phức tạp.

Dữ liệu phải được xử lý bằng các công cụ tiên tiến (phân tích và thuật toán) để cho

ra các thông tin có ý nghĩa. Ví dụ, để quản lý một nhà máy phải xem xét cả hai vấn

đề hữu hình và vô hình với các thành phần khác nhau. Các thuật toán tạo thông tin

phải phát hiện và giải quyết các vấn đề không nhìn thấy được như sự xuống cấp của

máy, mài mòn linh kiện, vv. trong nhà máy.

Big data được tạo thành từ các nguồn bao gồm các trang web, phương tiện truyền

thông xã hội, máy tính để bàn và ứng dụng di động, thí nghiệm khoa học, các cảm

biến và các thiết bị khác trên internet (IoT).

**Những vấn đề liên quan về big data**

**Phân tích dữ liệu Big Data**

Phân tích dữ liệu big data là quá trình xử lý, tìm hiểu và rút ra những thông tin hữu ích từ các nguồn dữ liệu lớn và phức tạp. Với Big data, thông tin được thu thập từ nhiều nguồn khác nhau như các trang web, mạng xã hội, cơ sở dữ liệu, hệ thống thông tin địa lý, dữ liệu từ cảm biến, vv. Dữ liệu big data có các đặc điểm chính như lượng lớn, tốc độ chập hành cao và đa dạng.

Phân tích dữ liệu big data bao gồm các bước sau:

\* Thu thập dữ liệu: Dữ liệu big data được thu thập từ nhiều nguồn khác nhau như cơ sở dữ liệu, hệ thống vị trí địa lý, cảm biến, mạng xã hội, vv.

\* Lưu trữ dữ liệu: Dữ liệu được lưu trữ trong các hệ thống lưu trữ khối lượng lớn như hệ thống File Hadoop, cơ sở dữ liệu phi cấu trúc, kho dữ liệu theo hướng phân tán, vv.

\* Tiền xử lý dữ liệu: Dữ liệu big data thường không được chuẩn hoá và xử lý trước khi phân tích. Quá trình tiền xử lý bao gồm việc làm sạch, chuyển đổi, chuẩn hoá và tạo các biểu diễn dữ liệu thích hợp.

\* Phân tích dữ liệu: Trong quá trình phân tích dữ liệu big data, các kỹ thuật phân tích dữ liệu được áp dụng để tìm ra các mẫu, gợi ý và thông tin hữu ích từ dữ liệu. Các phương pháp phổ biến bao gồm: phân tích đồ thị, phân tích cụm, phân tích dạng tụ, học máy, khai phá dữ liệu, vv.

\* Triển khai kết quả: Kết quả của quá trình phân tích được triển khai trong các ứng dụng thực tế. Điều này có thể bao gồm công cụ dự báo, ứng dụng trí tuệ nhân tạo hoặc hệ thống quản lí thông tin.

Phân tích dữ liệu big data đóng vai trò quan trọng trong việc đưa ra quyết định kinh doanh, dự báo xu hướng, phát hiện gian lận, tối ưu hóa quá trình sản xuất và cung ứng, phân tích kỹ thuật và nhiều ứng dụng khác trong nhiều lĩnh vực.

**Xây dựng cơ sở hạ tầng Big Data**

Xây dựng cơ sở hạ tầng big data là quá trình tạo ra một hệ thống vật lý và logic để lưu trữ, xử lý và quản lý dữ liệu lớn (big data). Để xây dựng một cơ sở hạ tầng big data, quy trình sau đây có thể được thực hiện:

\* Xác định yêu cầu và mục tiêu: Xác định mục tiêu và yêu cầu kỹ thuật của hệ thống big data, bao gồm khả năng lưu trữ, xử lý và quản lý dữ liệu lớn.

\* Lựa chọn công nghệ: Lựa chọn các công nghệ, công cụ và framework phù hợp để xây dựng hệ thống big data, bao gồm công nghệ lưu trữ dữ liệu, cơ sở dữ liệu phân tán, công cụ xử lý dữ liệu và công nghệ truy vấn.

\* Thiết kế kiến trúc: Thiết kế kiến trúc hệ thống big data, bao gồm các thành phần như máy chủ lưu trữ, mạng kết nối, hệ thống xử lý dữ liệu, hệ thống truy vấn và giao diện người dùng. Triển khai hệ thống: Triển khai cơ sở hạ tầng big data bằng cách cài đặt và cấu hình phần mềm và phần cứng cần thiết, bao gồm cài đặt và cấu hình máy chủ lưu trữ, mạng kết nối, hệ thống xử lý dữ liệu và giao diện người dùng.

\* Đảm bảo an ninh và bảo mật: Đảm bảo an ninh và bảo mật cho cơ sở hạ tầng big data bằng cách triển khai các biện pháp bảo mật như xác thực và ủy quyền, mã hóa dữ liệu và giám sát an ninh.

\* Kiểm tra và tối ưu hóa: Kiểm tra và đánh giá hiệu suất của hệ thống big data và thực hiện các bước tối ưu hóa để nâng cao hiệu suất và khả năng mở rộng của hệ thống.

\* Theo dõi và duy trì: Theo dõi và duy trì hệ thống big data để đảm bảo hoạt động ổn định và khắc phục các vấn đề kỹ thuật khi cần thiết.

Quá trình xây dựng cơ sở hạ tầng big data là một công việc phức tạp và đòi hỏi kiến thức vững chắc về các công nghệ và kiến thức về big data.

**Sử dụng phần mềm và công cụ hỗ trợ**

**Hệ sinh thái Hadoop**

Hadoop được coi là xương sống trong các dự án Big Data. Thư viện phần mềm

Hadoop là một nền tảng cho phép các tập dữ liệu lớn được xử lý phân tán thông qua

các cụm máy tính sử dụng mô hình lập trình đơn giản. Nó được thiết kế để mở rộng

từ một máy chủ lên hàng nghìn máy, mỗi máy chủ có khả năng cung cấp năng lực

tính toán và lưu trữ cục bộ.

Dự án bao gồm một số mô-đun:

Hadoop Common, các tiện ích phổ biến hỗ trợ các mô-đun Hadoop khác Hadoop Distributed File System, cung cấp quyền truy cập thông lượng cao vào dữ liệu ứng dụng

Hadoop YARN, một nền tảng cho việc lập kế hoạch công việc và quản lý tài nguyên cụm

Hadoop MapReduce, một hệ thống dựa trên YARN để xử lý song song các tập dữ liệu lớn.

**Apache Spark**

Apache Spark là một nền tảng hệ thống tính toán mã nguồn mở phục vụ như một công cụ để xử lý Big Data trong Hadoop. Spark đã trở thành một trong những nền tảng xử lý phân tán dữ liệu quan trọng và có thể được triển khai theo nhiều cách khác nhau. Nó cung cấp các ràng buộc nguyên gốc cho Java, Scala, Python (đặc biệt là Python Anaconda) và ngôn ngữ lập trình R (R đặc biệt phù hợp với dữ liệu lớn), đồng thời cũng hỗ trợ SQL, dữ liệu luồng, machine learning và xử lý đồ thị.

**Cơ sở dữ liệu**

Cơ sở dữ liệu NoSQL lưu trữ và quản lý dữ liệu theo cách linh hoạt và xử lý tốc độ

cao. Không giống như cơ sở dữ liệu SQL, nhiều cơ sở dữ liệu NoSQL có thể được

thu nhỏ theo chiều ngang trên hàng trăm hoặc hàng nghìn máy chủ.

**Cơ sở dữ liệu bộ nhớ trong**

Cơ sở dữ liệu bộ nhớ trong (IMDB) là một hệ thống quản lý cơ sở dữ liệu chủ yếu

dựa vào bộ nhớ chính, thay vì đĩa, để lưu trữ dữ liệu. Cơ sở dữ liệu bộ nhớ trong

nhanh hơn so với cơ sở dữ liệu trên đĩa được tối ưu hóa, một lựa chọn quan trọng

cho việc sử dụng phân tích Big Data tạo kho dữ liệu.

**Các trường hợp cần sử dụng Big Data**

**Quản lý Chính phủ:**

Việc sử dụng các dữ liệu lớn trong các quy trình của chính phủ cho phép tăng hiệu quả về mặt chi phí, năng suất và sự đổi mới, nhưng không phải là không có sai sót của nó. Phân tích dữ liệu thường yêu cầu nhiều bộ phận của chính phủ (trung ương và địa phương) hợp tác và tạo ra các quy trình mới và sáng tạo để mang lại kết quả mong muốn.

**Sự phát triển quốc tế:**

Nghiên cứu về việc sử dụng hiệu quả các công nghệ thông tin và truyền thông cho mục đích phát triển (hay còn gọi là ICT4D) cho thấy công nghệ dữ liệu lớn có thể có nhiều đóng góp quan trọng nhưng cũng là thách thức đối với sự phát triển của quốc tế. Những tiến bộ trong phân tích dữ liệu lớn giúp giảm chi phí cho việc ra quyết định trong các lĩnh vực quan trọng như chăm sóc sức khoẻ, việc làm, năng suất kinh tế, tội phạm, an ninh, thiên tai và quản lý tài nguyên. Tuy nhiên, những thách thức đối với các nước đang phát triển như cơ sở hạ tầng công nghệ không đầy đủ và sự khan hiếm về kinh tế và nguồn nhân lực sẽ làm nghiêm trọng thêm các mặt trái của dữ liệu lớn như sự riêng tư hoặc các vấn đề khác.

**Tài chính:**

Việc sử dụng các dữ liệu lớn dưới dạng lịch sử các giao dịch tài chính được gọi là

phân tích kỹ thuật. Sử dụng dữ liệu phi tài chính để dự đoán thị trường đôi khi được

gọi là dữ liệu thay thế.

**Sản Xuất:**

Theo bài Nghiên cứu xu hướng toàn cầu TCS 2013, sự cải tiến trong kế hoạch sản xuất và chất lượng sản phẩm là lợi ích lớn nhất của dữ liệu lớn cho ngành sản xuất. Dữ liệu lớn cung cấp cơ sở hạ tầng cho ngành công nghiệp sản xuất, đó là khả năng cải thiện năng suất và tính khả dụng. Việc lên kế hoạch sản xuất chính là một cách tiếp cận dữ liệu lớn cho phép giảm thời gian chết về gần như bằng không và cụ thể hóa số lượng lớn dữ liệu và các công cụ dự đoán khác cho phép tạo ra một quá trình nhằm hệ thống hóa dữ liệu thành các thông tin hữu ích. Khái niệm về việc dự báo sản xuất bắt đầu bằng việc thu thập dữ liệu cảm quan khác nhau như âm thanh, chuyển động, áp suất, điện áp... Số lượng lớn các dữ liệu cảm quan cộng với dữ liệu lịch sử sản xuất tạo thành dữ liệu lớn trong sản xuất. Các dữ liệu lớn này như là đầu vào cho các công cụ dự báo và các chiến lược phòng ngừa tương tự như việc dự báo trong lĩnh vực Quản lý Y tế.

**Chăm sóc sức khỏe:**

Phân tích dữ liệu lớn đã giúp cải thiện việc chăm sóc sức khoẻ bằng cách cá nhân hóa các phương pháp trị liệu và chẩn đoán lâm sàng, làm giảm thiểu chi phí và thời gian khám bệnh, tự động báo cáo và lưu trữ thông tin sức khỏe và dữ liệu bệnh nhân trong nội bộ cũng như mở rộng ra bên ngoài, chuẩn hóa các thuật ngữ y học và chống phân mảnh trong lưu trữ dữ liệu và thông tin của bệnh. Một số lĩnh vực có sự cải tiến mang tính hướng dẫn hơn là thực hành. Lượng dữ liệu được tạo ra trong các hệ thống chăm sóc sức khoẻ là không nhỏ. Với sự bổ sung thêm của mHealth, eHealth và các thiết bị công nghệ theo dõi sức khỏe được thì khối lượng dữ liệu sẽ tiếp tục gia tăng. Điều này bao gồm dữ liệu ghi chép sức khoẻ điện tử, dữ liệu hình ảnh, dữ liệu được tạo ra của bệnh nhân, dữ liệu cảm biến và các dạng dữ liệu khó xử lý khác. Hiện nay, nhu cầu lớn hơn đối với các môi trường như vậy là chú ý nhiều hơn đến chất lượng dữ liệu và thông tin. "Dữ liệu lớn rất thường có nghĩa là dữ liệu chưa được xử lý và một phần số liệu không chính xác tăng lên khi có sự tăng trưởng khối lượng dữ liệu." Việc theo dõi bằng con người ở quy mô dữ liệu lớn là không thể và có một nhu cầu cấp thiết về các công cụ thông minh để kiểm soát chính xác và xử lý thông tin bị mất trong dịch vụ y tế. Mặc dù dữ liệu trong lĩnh vực chăm sóc sức khoẻ hiện nay thường được lưu trữ dưới dạng điện tử, nhưng nó nằm ngoài phạm vi của dữ liệu lớn vì hầu hết không có cấu trúc và khó sử dụng.

**Giáo dục:**

Một nghiên cứu của Viện nghiên cứu toàn cầu McKinsey cho thấy, ngành dữ liệu lớn đang thiếu hụt 1,5 triệu chuyên gia cũng như nhà quản lý dữ liệu, và một số trường đại học bao gồm Đại học Tennessee và UC Berkeley đã tạo ra các chương trình thạc sĩ để đáp ứng nhu cầu này. Các khóa huấn luyện tư nhân cũng phát triển các chương trình để đáp ứng nhu cầu đó, bao gồm các chương trình miễn phí như The Data Incubator hoặc chương trình trả tiền như General Assembly.

**Truyền thông:**

Để hiểu cách thức các phương tiện truyền thông sử dụng dữ liệu lớn như thế nào, trước tiên cần hiểu rõ một số ngữ cảnh trong cơ chế sử dụng cho quá trình truyền thông. Nick Couldry và Joseph Turow đề xuất rằng các học viên trong ngành Truyền thông và Quảng cáo cần tiếp cận dữ liệu lớn như là nhiều điểm thông tin về hàng triệu cá nhân. Ngành công nghiệp dường như đang chuyển hướng từ cách tiếp cận truyền thống bằng cách sử dụng các môi trường truyền thông cụ thể như báo chí, tạp chí hoặc chương trình truyền hình và thay vào đó là những người tiêu dùng với công nghệ tiếp cận những người này được nhắm mục tiêu vào những thời điểm tối ưu ở những vị trí tối ưu. Mục đích cuối cùng là để phục vụ hoặc truyền tải, một thông điệp hoặc nội dung (theo cách thống kê) phù hợp với suy nghĩ của người tiêu dùng.

**Ví dụ**: môi trường xuất bản ngày càng làm cho các thông điệp (quảng cáo) và nội

dung (bài viết) được cải thiện để thu hút người tiêu dùng đã được thu thập độc quyền

thông qua các hoạt động khai thác dữ liệu khác nhau.

Nhắm đến người tiêu dùng mục tiêu (đối với quảng cáo của các nhà tiếp thị) thu thập dữ liệu

Dữ liệu trong báo chí: nhà xuất bản và nhà báo sử dụng các công cụ dữ liệu lớn để cung cấp thông tin chi tiết và các bản đồ họa chi tiết độc đáo và sáng tạo. Kênh 4, một kênh phát thanh truyền hình công cộng của Anh, đang dẫn đầu trong lĩnh vực dữ liệu lớn và phân tích dữ liệu.

**Mạng lưới vạn vật kết nối Internet (IoT)**

Dữ liệu lớn có thể kết hợp với công nghệ Mạng lưới vạn vật kết nối Internet. Dữ liệu được chiết xuất từ các thiết bị IoT cung cấp một bản đồ kết nối giữa các thiết bị. Những sự kết nối này đã được ngành công nghiệp truyền thông, các công ty và chính phủ sử dụng để nhắm mục tiêu chính xác hơn đối tượng của họ và tăng hiệu quả của phương tiện truyền thông. IoT cũng ngày càng được chấp nhận như một phương tiện thu thập dữ liệu cảm giác, và dữ liệu cảm giác này đã được sử dụng trong các ngành như y học và sản xuất.

Kevin Ashton, chuyên gia đổi mới kỹ thuật số người được cho là người tạo ra thuật ngữ định nghĩa Internet vạn vật đã phát biểu: "Nếu chúng ta có máy tính biết tất cả mọi thứ - nó sẽ sử dụng dữ liệu mà nó thu thập được mà không có sự trợ giúp từ chúng ta - chúng ta sẽ có thể theo dõi và kiểm soát mọi thứ, giảm đáng kể lượng chất thải, tổn thất và chi phí. Chúng ta sẽ biết khi nào cần thay thế, sửa chữa hoặc thu hồi lại, và liệu rằng thức ăn chúng ta đang ăn có tươi hay không."

**Công nghệ:**

Từ năm 2015, dữ liệu lớn trở nên nổi bật trong hoạt động kinh doanh như một công cụ để giúp nhân viên làm việc hiệu quả hơn cũng như tối ưu hóa việc thu thập và chia sẻ thông tin. Việc sử dụng dữ liệu lớn để giải quyết các vấn đề thu thập dữ liệu và CNTT trong một doanh nghiệp được gọi là IT Operations Analytics (ITOA). Bằng cách áp dụng các nguyên tắc dữ liệu lớn vào các khái niệm về trí thông minh của máy móc và tính toán sâu, các bộ phận CNTT có thể dự đoán các vấn đề tiềm ẩn và đưa ra các giải pháp trước khi vấn đề xảy ra. Vào thời điểm này, các doanh nghiệp ITOA cũng bắt đầu đóng vai trò quan trọng trong việc quản lý hệ thống bằng cách cung cấp các nền tảng mang các dữ liệu cá nhân riêng biệt và tạo ra những hiểu biết sâu sắc từ toàn bộ hệ thống chứ không phải từ các dữ liệu riêng lẻ.

\* EBay sử dụng hai kho dữ liệu với tốc độ 7.5 petabyte và 40PB cũng như một cụm 40PB Apache Hadoop để tìm kiếm, khuyến nghị người tiêu dùng và bán hàng.

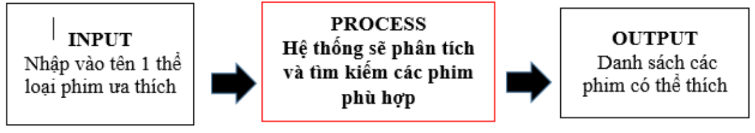
\*Amazon.com xử lý hàng triệu hoạt động back-end hàng ngày, cũng như các truy vấn từ hơn nửa triệu người bán hàng bên thứ ba. Công nghệ cốt lõi mà Amazon hoạt động dựa trên Linux và đến năm 2005 họ có ba cơ sở dữ liệu Linux lớn nhất thế giới, với dung lượng 7,8 TB, 18,5 TB và 24,7 TB.

\*Facebook xử lý 50 tỷ hình ảnh từ cơ sở người dùng của nó.

Google đã xử lý khoảng 100 tỷ lượt tìm kiếm mỗi tháng vào tháng 8 năm 2012.

Cơ sở dữ liệu Oracle NoSQL đã được kiểm tra để vượt qua mốc 1 triệu xử lý mỗi giây với 8 nhân và đạt tốc độ 1.2 triệu xử lý mỗi giây với 10 nhân.

**Bài toán**

****

Một người làm việc cả ngày về chỉ có thời gian giải trí 120 phút và anh ta muốn xem phim. Nhưng khi vào kho phim thì không biết nên xem phim nào hay. Tốn thời gian tìm kiếm khiến anh ta không thể xem hết 1 bộ phim, chọn ngẫu nhiên 1 bộ thì lại gặp phim không hay. Hãy nghĩ cách giúp anh ấy.

**Giải pháp**

**Apache Spark & Elaticsearch**

**Giới thiệu Apache Spark:**

Apache Spark là một open source cluster computing framework được phát triển sơ khởi vào năm 2009 bởi AMPLab tại đại học California, Berkeley. Sau này, Spark đã được trao cho Apache Software Foundation vào năm 2013 và được phát triển cho đến nay. Spark cho phép xây dựng và phân tích nhanh các mô hình dự đoán. Spark cung cấp khả năng truy xuất toàn bộ dữ liệu cùng lúc, và còn cung cấp tính năng streaming, được dùng để xây dựng các mô hình real-time bằng cách nạp toàn bộ dữ liệu vào bộ nhớ

Spark cho phép phân chia tác vụ này thành những phần dễ quản lý hơn. Sau đó, Spark sẽ chạy các tác vụ này trong bộ nhớ, trên các cluster của nhiều server khác nhau để khai thác tốc độ truy xuất nhanh từ RAM. Spark sử dụng API Resilient Distributed Dataset (RDD) để xử lý dữ liệu.

**Đặc điểm các thành phần Spark:**

\* Apache Spark Core: Spark Core là thành phần cốt lõi thực thi cho tác vụ cơ bản làm nền tảng cho các chức năng khác. Nó cung cấp khả năng tính toán trên bộ nhớ và datase trong bộ nhớ hệ thống lưu trữ ngoài.

\* Spark SQL: Là một thành phần nằm trên Spark Core nó cung cấp một sự ảo hóa mới cho dữ liệu là SchemaRDD, hỗ trợ các dữ liệu có cấu trúc và bán cấu trúc. Spark Streaming: Cho phép thực hiện phân tích xử lý trực tuyến xử lý theo lô.

\*MLlib (Machine Learning Library): MLlib là một nền tảng học máy phân tán bên trên Spark do kiến trúc phân tán dựa trên bộ nhớ. Theo các so sánh benchmark Spark MLlib nhanh hơn chín lần so với phiên bản chạy trên Hadoop (Apache Mahout)

\* GrapX: Grapx là nền tảng xử lý đồ thị dựa trên Spark. Nó cung cấp các Api để diễn tả các tính toán trong đồ thị

\* Apache Spark có các tính năng đặc trưng sau đây.

\* Tốc độ: Spark có thể chạy trên cụm Hadoop và có thể chạy nhanh hơn 100 lần khi chạy trên bộ nhớ RAM, và nhanh hơn 10 lần khi chạy trên ổ cứng. Bằng việc giảm số thao tác đọc nghi lên đĩa cứng. Nó lưu trữ trực tiếp dữ liệu xử lý lên bộ nhớ Hỗ trợ đa ngôn ngữ: Spark cung cấp các API có sẵn cho các ngôn ngữ Java, Scala, hoặc Python. Do đó, bạn có thể viết các ứng dụng bằng nhiều các ngôn ngữ khác nhau. Spark đi kèm 80 truy vấn tương tác mức cao. Phân tích nâng cao: Spark không chỉ hỗ trợ ‘Map’ và ‘Reduce’. Nó còn hỗ trợ truy vấn SQL, xử lý theo Stream, học máy, và các thuật toán đồ thị (Graph)

\* Resilient Distributed Datasets (RDD) là một cấu trúc dữ liệu cơ bản của Spark. Nó là một tập hợp bất biến phân tán của một đối tượng. Mỗi dataset trong RDD được chia ra thành nhiều phần vùng logical. Có thể được tính toán trên các node khác nhau của một cụm máy chủ (cluster). RDDs có thể chứa bất kỳ kiểu dữ liệu nào của Python, Java, hoặc đối tượng Scala, bao gồm các kiểu dữ liệu do người dùng định nghĩa

**Giới thiệu Elaticsearch**

Elasticsearch được phát triển bởi Shay Banon vào năm 2010 và là hệ thống phân tán theo thời gian thực là một hệ thống tìm kiếm full-text, phân tích mã nguồn mở. Sử dụng thông qua RESTfull sử dụng JSON (JavaScript Object Notation) để chứa dữ liệu. Được viết bằng ngôn ngữ Java điều này cho phép Elasticsearch có thể chạy trên nhiều nền tảng khác nhau, cho phép sử dụng truy vấn dữ liệu lớn với tốc độ cao.

\* Elasticsearch có thể được mở rộng lên đến Petabyte dữ liệu có cấu trúc và không có cấu trúc.

\* Elasticsearch có thể được sử dụng như một thay thế cho các lưu trữ tài liệu như MongoDb hay RavenDb.

\* Elasticsearch được sử dụng để cải thiện hiệu năng tìm kiếm, đặc biệt là tìm kiếm full-text.

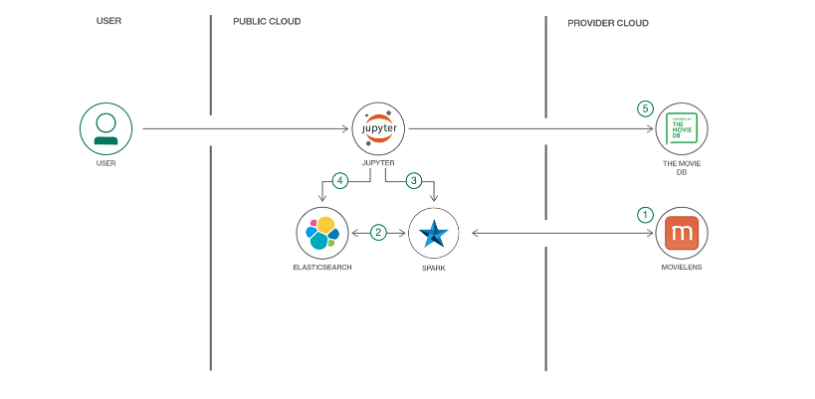
\* Elasticsearch là một máy tìm kiếm phổ biến nhất được sử dụng bởi nhiều tổ chức lớn như Wikipedia, The Guardian, StakOverflow, GitHub, ...

\* Elasticsearch là một hệ thống tìm kiếm và phân tích trực tuyến có khả năng mở rộng theo chiều ngang, dữ liệu trong Elasticsearch được chia thành các Shard, mỗi Shard là một thành phần độc lập với nhau, mỗi khi có dữ liệu mới được index, đầu tiên dữ liệu sẽ được gửi đến máy Master. Master sẽ kiểm tra thông tin về các Shard như kích thước, nơi lưu trữ sau đó sẽ phân bổ tài nguyên giao task chia công việc index ra thành nhiều Shard khác nhau nhằm tăng tốc thời gian đánh chỉ mục cho dữ liệu. Khi có yều cầu truy vấn trên Elasticsearch truy vấn sẽ được gửi về máy Master sau đó máy này sẽ đưa truy vấn đến tất cá chác Shard chứa tài liệu liên quan đến truy vấn và thực thi tìm kiếm xếp hạng kết quả trả về trên mỗi Shard, sau đó toàn bộ liệu sẽ được hợp nhất lại để đưa ra kết quả cuối cùng.

\* Apache Spark thực hiện tính toán song song nên sẽ có rất nhiều module để truy cập đến nhiều nguồn chứa dữ liệu khác nhau như HDFS thao tác dữ liệu trên hệ thống tập tin của Hadoop. Spark SQL là một module cho phép thao tác lấy dữ liệu song song trong các cơ sở dữ liệu. Với Elasticsearch thì có module Apache Spark connector giúp cho phép tạo và sử dụng đối tượng Spark Context có thể đánh chỉ mục và truy vấn dữ liệu song song bộ thự viện này được cung cấp thông qua phần mở rộng là elasticsearchhadoop Việc tính toán song song dựa trên 2 quá trình: Tìm kiếm song song trên Elasticsearch và tính toán, áp dụng mô hình học máy xếp hạng song song với Apache Spark.

**Mô hình giải pháp:**

Mô hình của IBM sử dụng apache spark và elasticsearch



1.Load tập dữ liệu movie vào Spark.

2.Sử dụng Spark DataFrame để làm sạch dữ liệu và load vào Elasticsearch.

3.Sử dụng Spark Mllib để trainning.

4.Lưu kết quả mô hình vào Elasticsearch.

5.Sử dụng Elasticsearch queries và một custom vector scoring plugin, để tìm ra phim đề xuất. Movie Database API để hiển thị poster phim.

**Cài đặt Thử nghiệm:**

Chạy thử code của tác giả trên 1 host local (HĐH Ubuntu 22 .04.03) với bộ dữ liệu ml-latest -small có kích thước 3.14MB

**DATASET**

Quá trình thực nghiệm học máy xếp hạng gồm các bước chính sau đây:

Thu thập dữ liệu: thu thập toàn bộ dữ liệu về phim và dữ liệu lịch sử của người dùng trong hệ thống tìm kiếm Google Chrome.

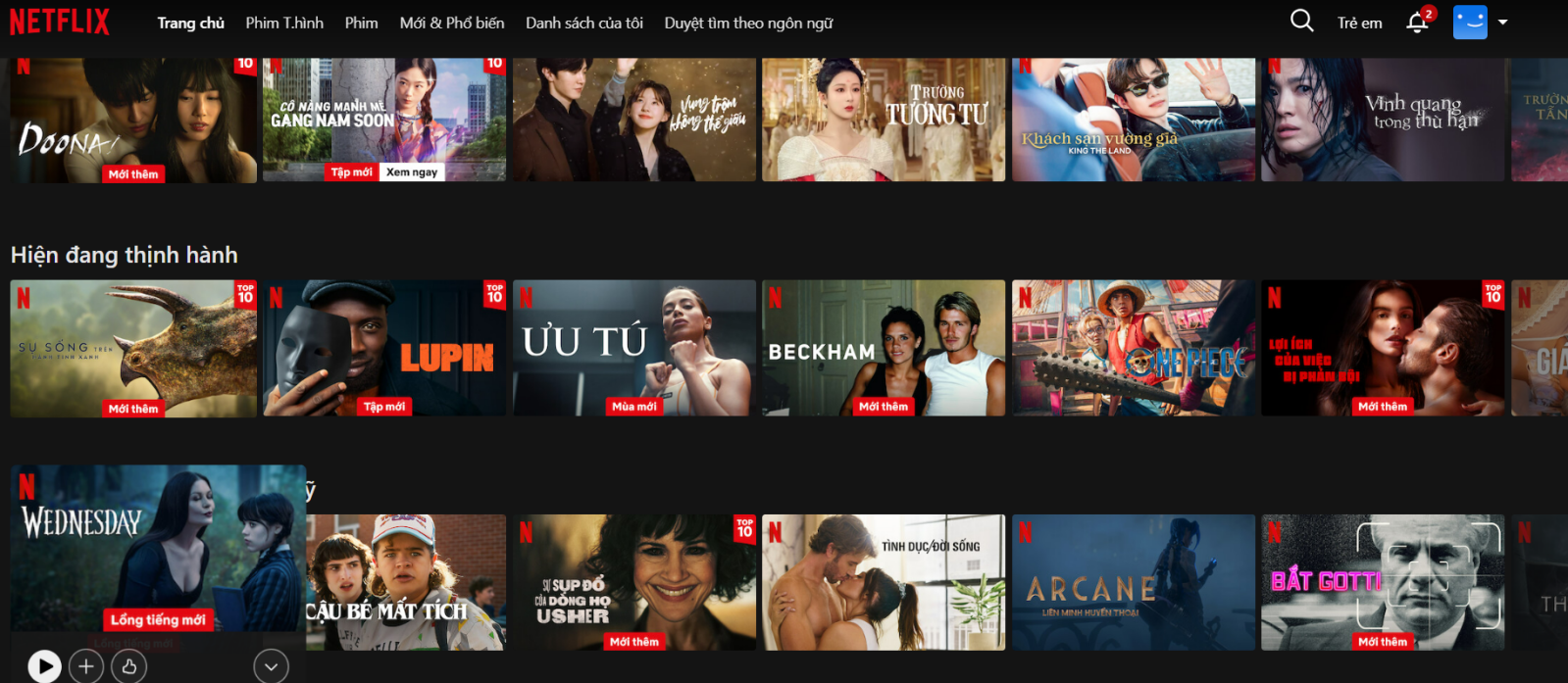
Xử lý dữ liệu: tiền xử lý dữ liệu, đánh chỉ mục cho dữ liệu, xây dựng tập dữ liệu.

**Thu thập dữ liệu phim**

Tất các các dữ liệu sẽ được thu thập từ nhiều trang web và thông tin của người dùng từ hệ thống crawler và search của Google Chrome hệ thống được chạy hàng ngày ngay khi có tất cả các dữ liệu thêm mới, bộ phân tích sẽ tự động bóc tách (sử dụng Jsoup để bóc tách dữ liệu html đây là công cụ cho phép dùng cú pháp css để chọn các thẻ và thuộc tính html) và lưu trữ vào cơ sở dữ liệu.

**Thu thập dữ liệu phim NETFLIX**

Đầu tiên là hệ thống sẽ trích xuất các thông tin từ trang web đánh giá phim NETFLIX (Internet Movie Database). Dưới đây là thông tin một bộ phim được trích xuất từ NETFLIX. (Nguồn <https://www.netflix.com/vn/>)

Hình: Thông tin phim trên trang NETFLIX

NETFLIXlà một website trực tuyến nó đóng vai trò như một thư viện, nơi lưu trữ những thông tin chi tiết về các tác phẩm điện ảnh nổi tiếng, ngoài ra NETFLIXcòn là website uy tín đóng vai trò như một nhà phê bình. NETFLIXcũng là nơi tổng hợp những ý kiến đánh giá, xếp hạng của mộttác phẩm điện ảnh dựa trên các yếu tố như kịch bản, công tác đạo diễn, bối cảnh, hiệu quả hìnhảnh, kỹ thuật quay phim...NETFLIX rất có uy tín với giới độc giả Internet, cũng như các tín đồ củamôn nghệ thuật thứ 7. Ngoài nội dung phê bình đánh giá về các tác phẩm thuộc lĩnh vực điệnảnh, NETFLIX còn đánh giá những tác phẩm truyền hình hay những ngôi sao điện ảnh, nhà sảnxuất phim... Tên phim, năm sản xuấtĐạo diễn, diễn viên. Nội dung phim, thể loại, điểm số rating. Bước này thu thập được 117.094 thông tin phim NETFLIX dữ liệu ban đầu được chứa vào cơsở dữ liệu MySQL, và được chứa theo định dạng sau.

**Bảng định dạng trường dữ liệu thông tin phim NETFLIXtrong cơ sở dữ liệu**

|  |  |
| --- | --- |
| Tên trường | Miêu tả |
| id | Định danh của IMDb |
| director | Đạo diễn |
| genre | Thể loại |
| Image link | Poster |
| link | Link trên IMDb |
| name | Tên phim |
| outline | Nội dung |
| year | Năm |
| release\_date | Ngày phát hành |
| actor | Diễn viên |
| runtime | Thời lượng |
| ratingCount | Tống số đánh giá |
| rate | Điểm đánh giá trung |

**Thử nghiệm trên 1 Host**

**Các bước cài đặt**

**Cài đặt Elasticsearch 5.3.0**

$ wget <https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-5.3.0.tar.gz>

$ tar xfz elasticsearch-5.3.0.tar.gz



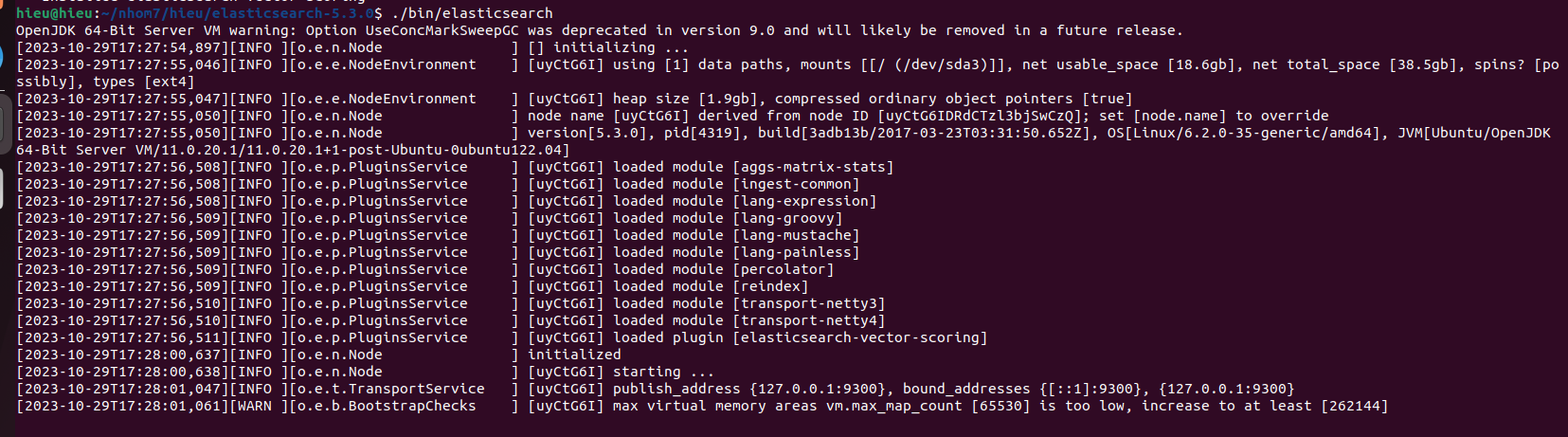
**Cài đặt Elasticsearch vector scoring plugin**

****

$ cd elasticsearch-5.3.0

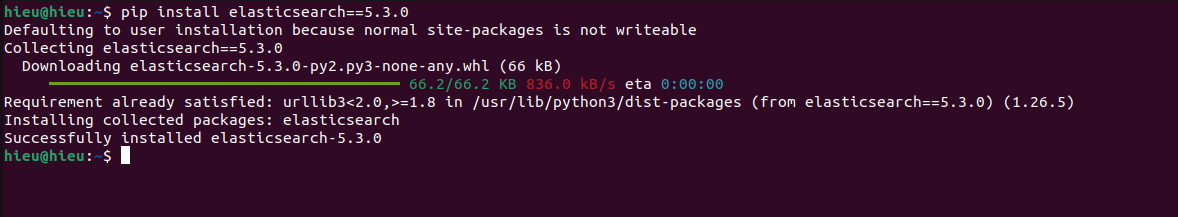
$ ./bin/elasticsearch-plugin install https://github.com/MLnick/elasticsearch-vector-scoring/releases/download/v5.3.0/elasticsearch-vector-scoring-5.3.0.zip

**Khởi động Elasticsearch**

****

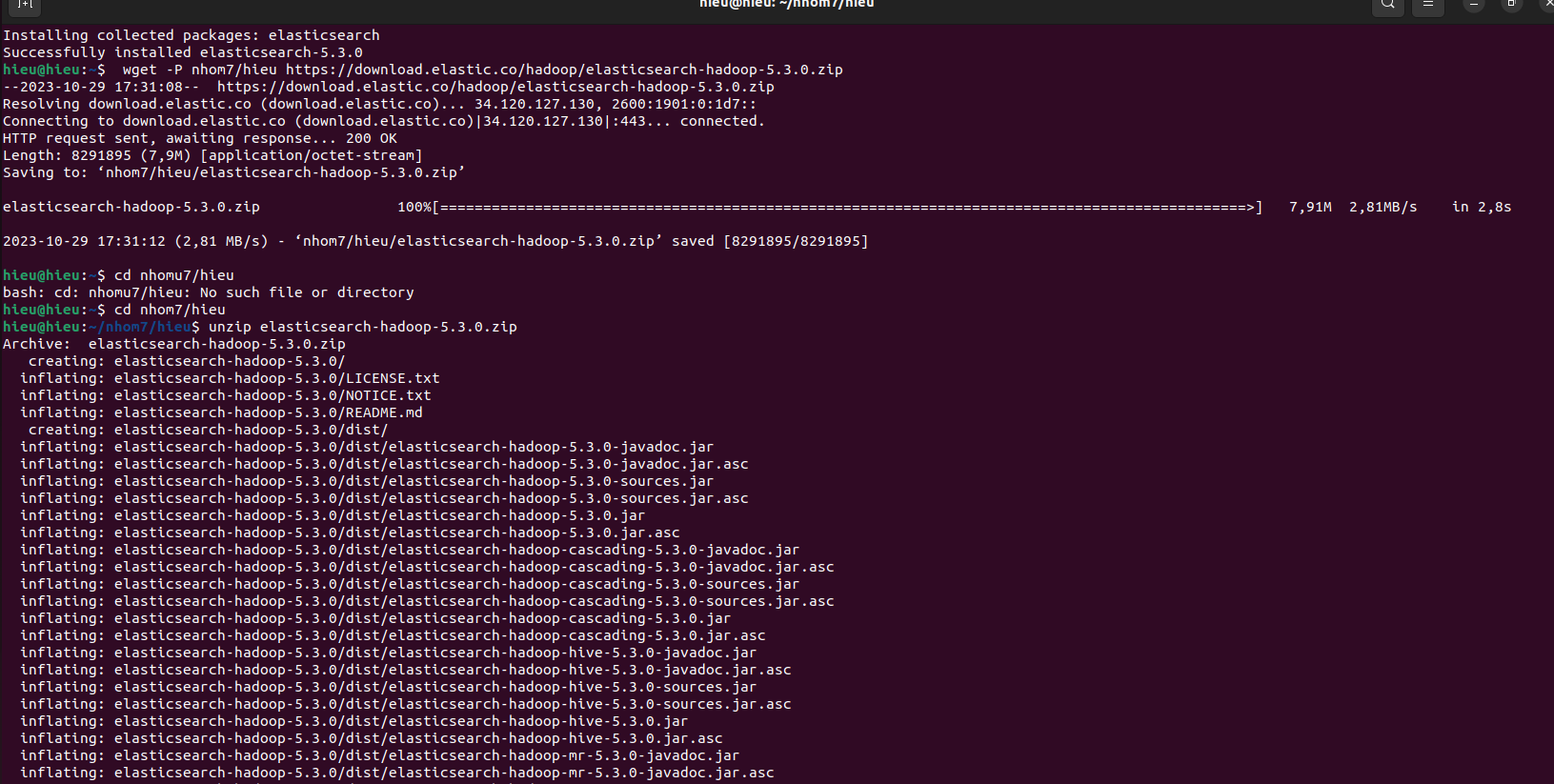
$ ./bin/elasticsearch

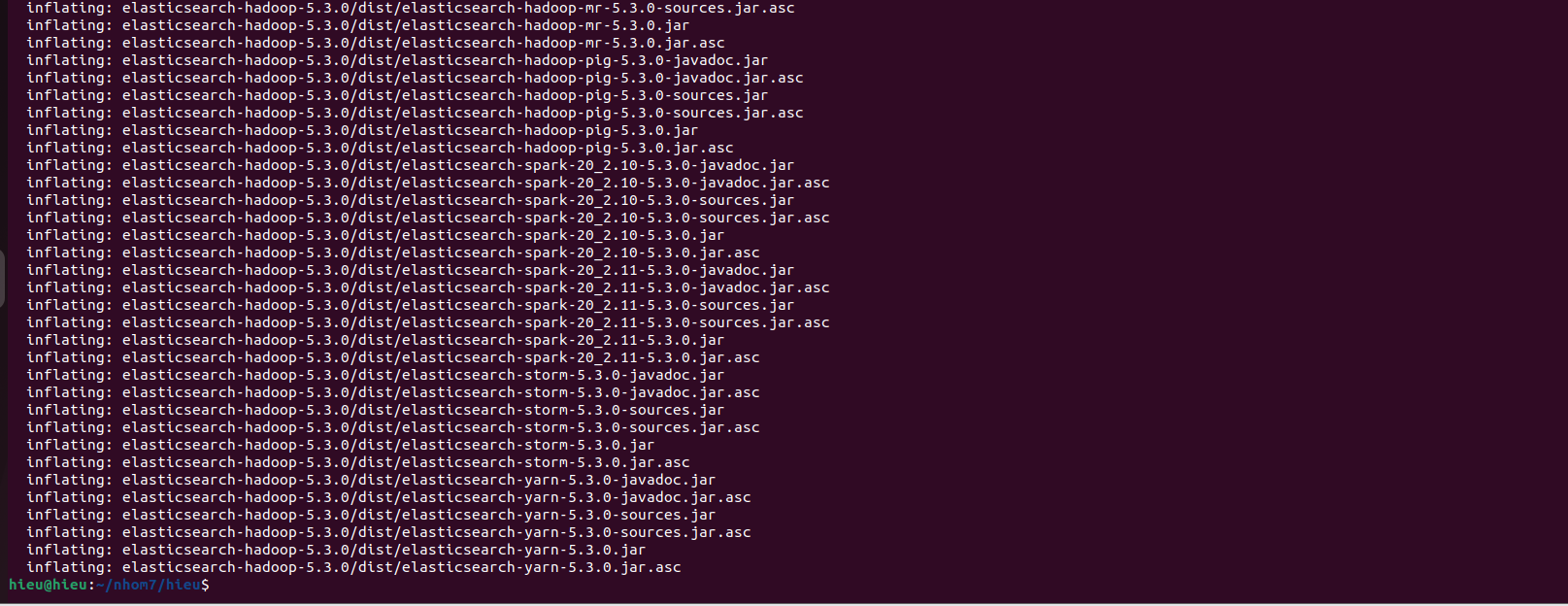
**Cài đặt Elasticsearch Python client**

****

$ pip install elasticsearch==5.3.0

**Tải Elasticsearch Spark connector**

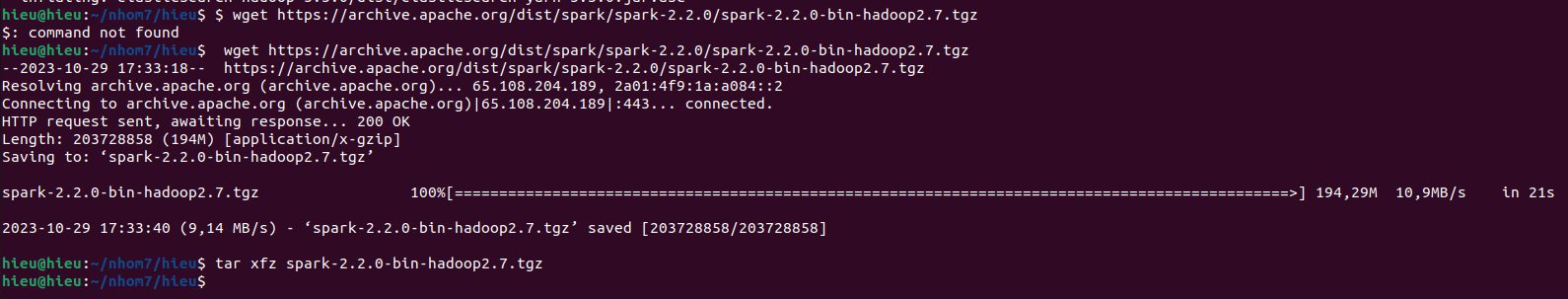
****



$ wget <https://download.elastic.co/hadoop/elasticsearch-hadoop-5.3.0.zip>

$ unzip elasticsearch-hadoop-5.3.0.zip

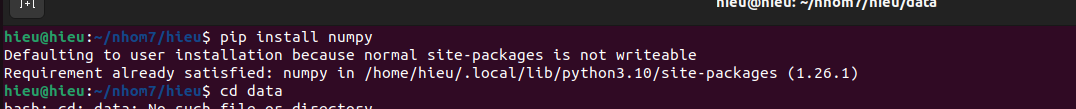
**Tải Apache Spark:**

****

$ wget <https://archive.apache.org/dist/spark/spark-2.2.0/spark-2.2.0-bin-hadoop2.7.tgz>

$ tar xfz spark-2.2.0-bin-hadoop2.7.tgz

**Cài đặt Numpy**



$ pip install numpy

**Tải bộ dữ liệu Movielens**

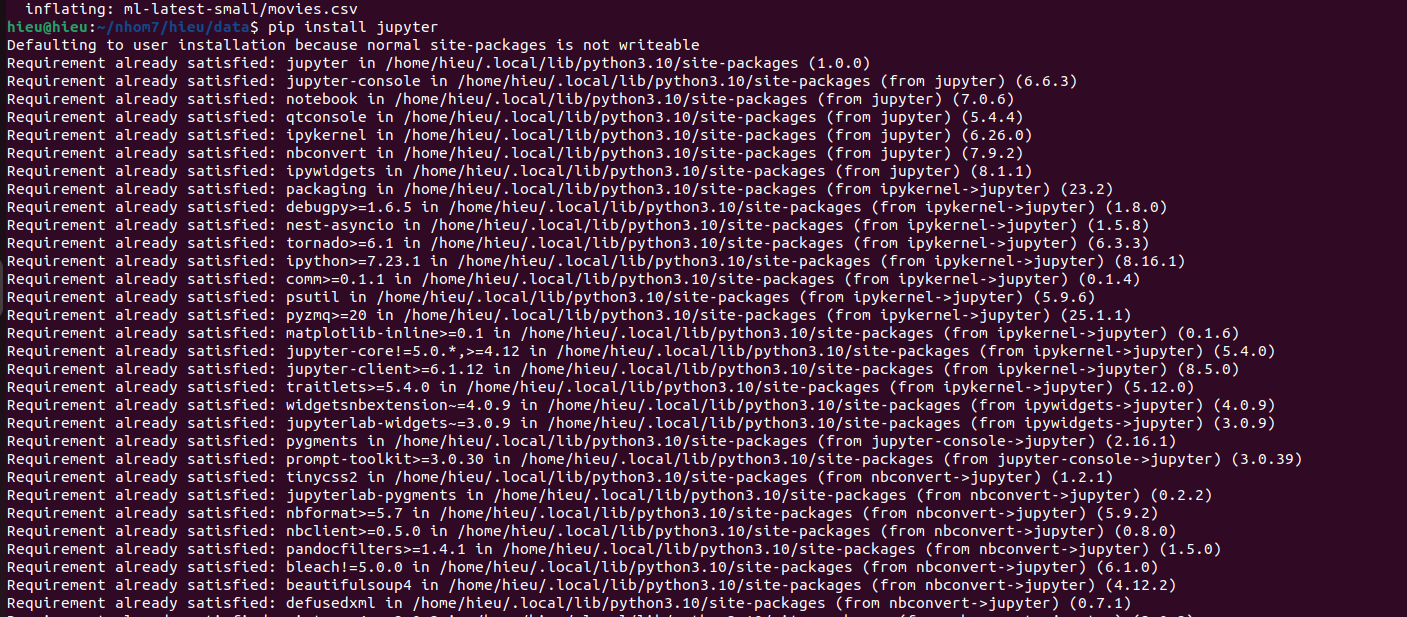
****

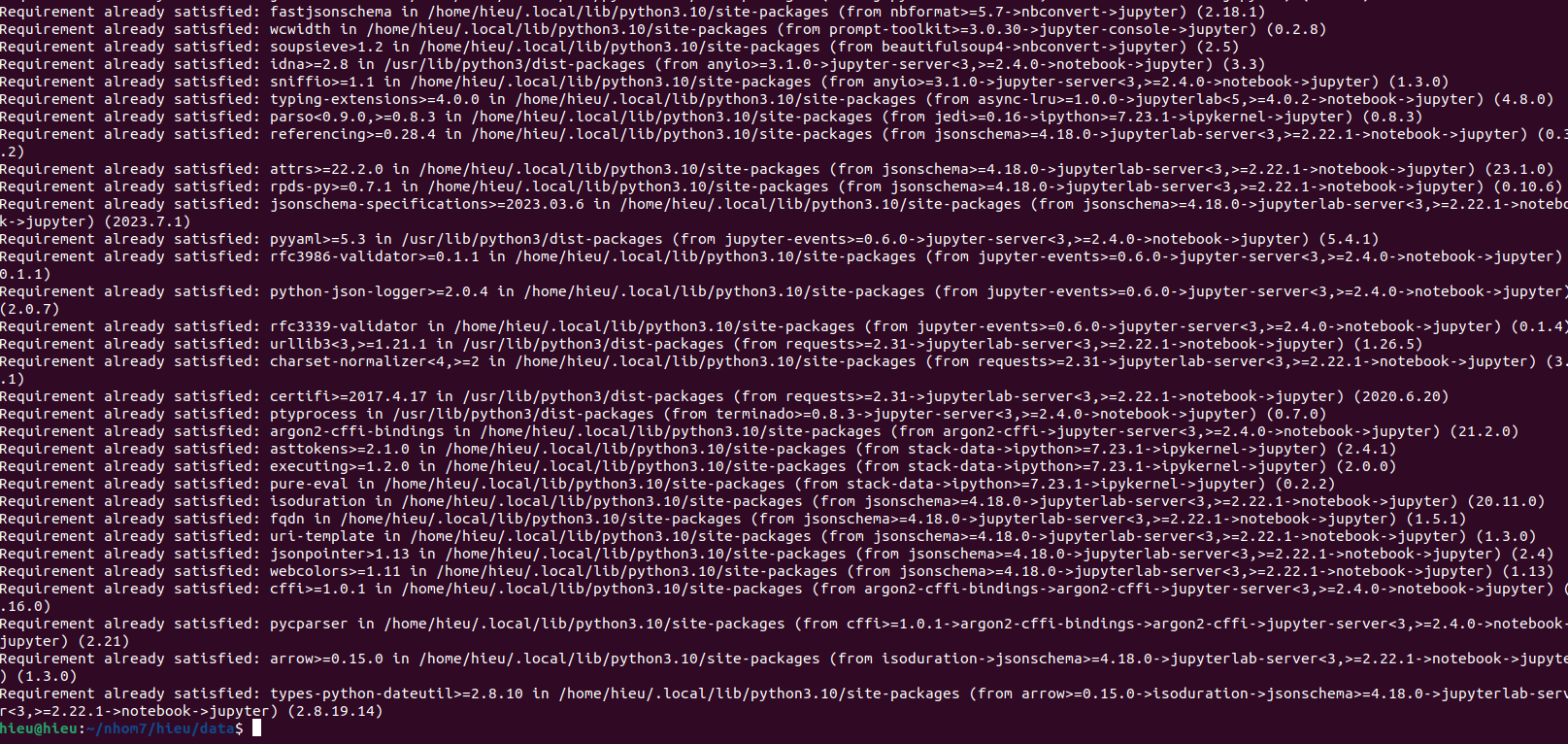
$ cd data

$ wget <http://files.grouplens.org/datasets/movielens/ml-latest-small.zip>

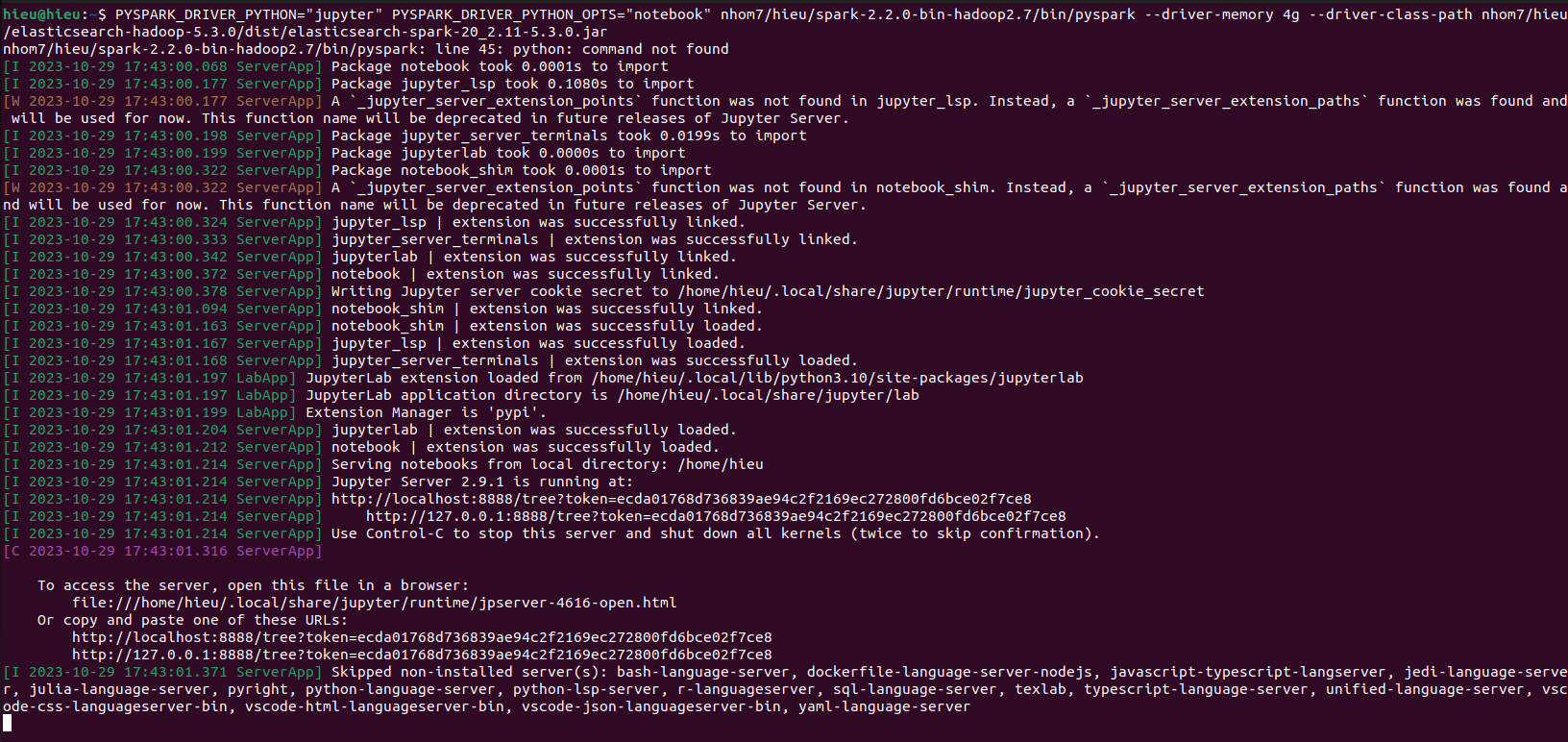
$ unzip ml-latest-small.zip

**Cài đặt Jupyter notebook**

$ pip install jupyter



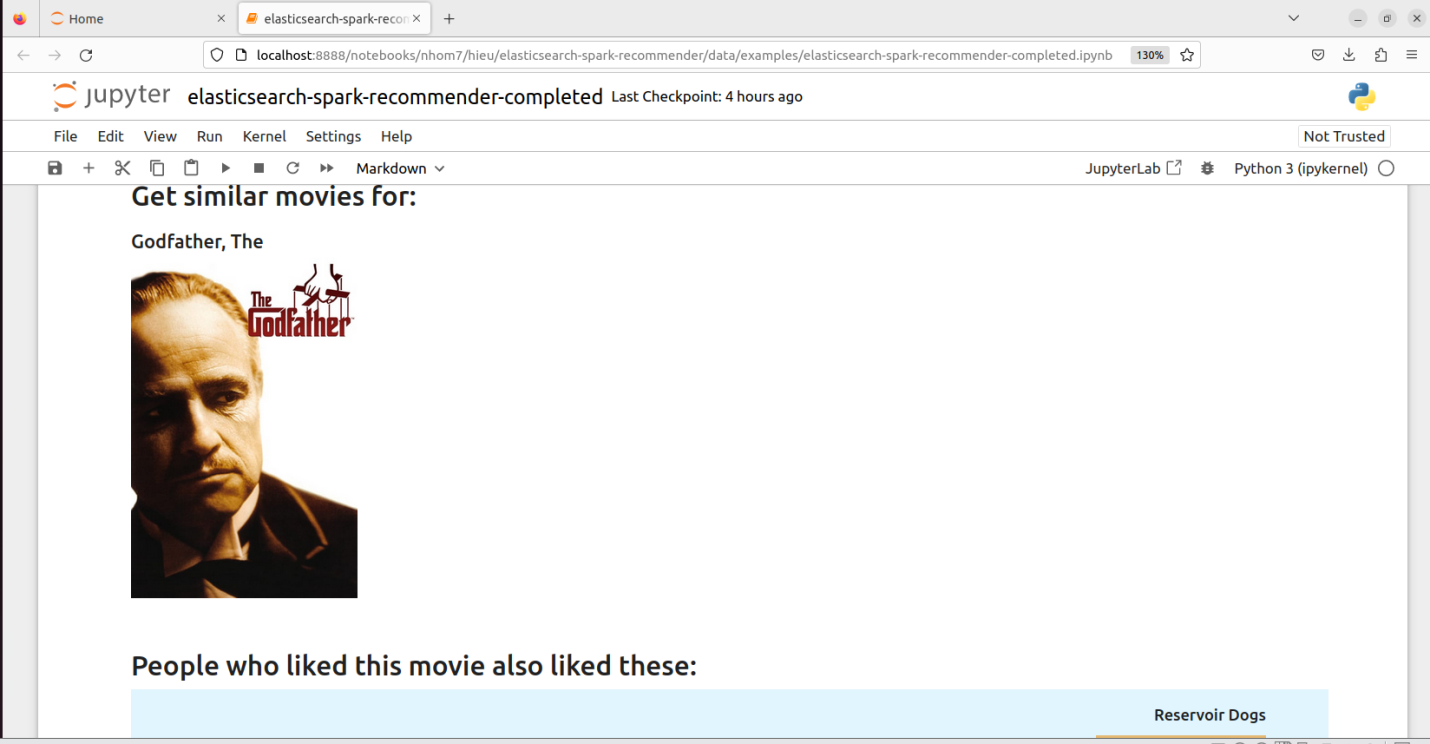
**Chạy jupyter notebook:**

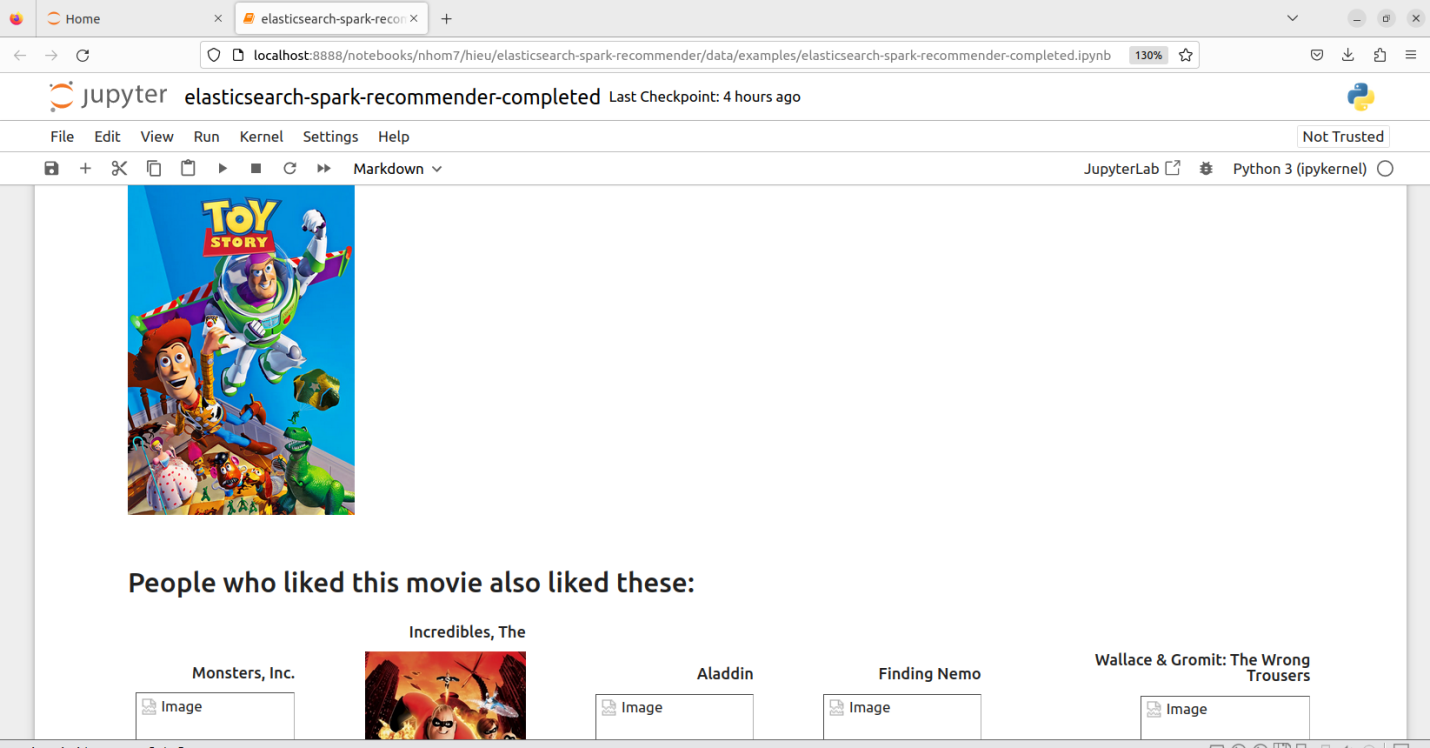
****

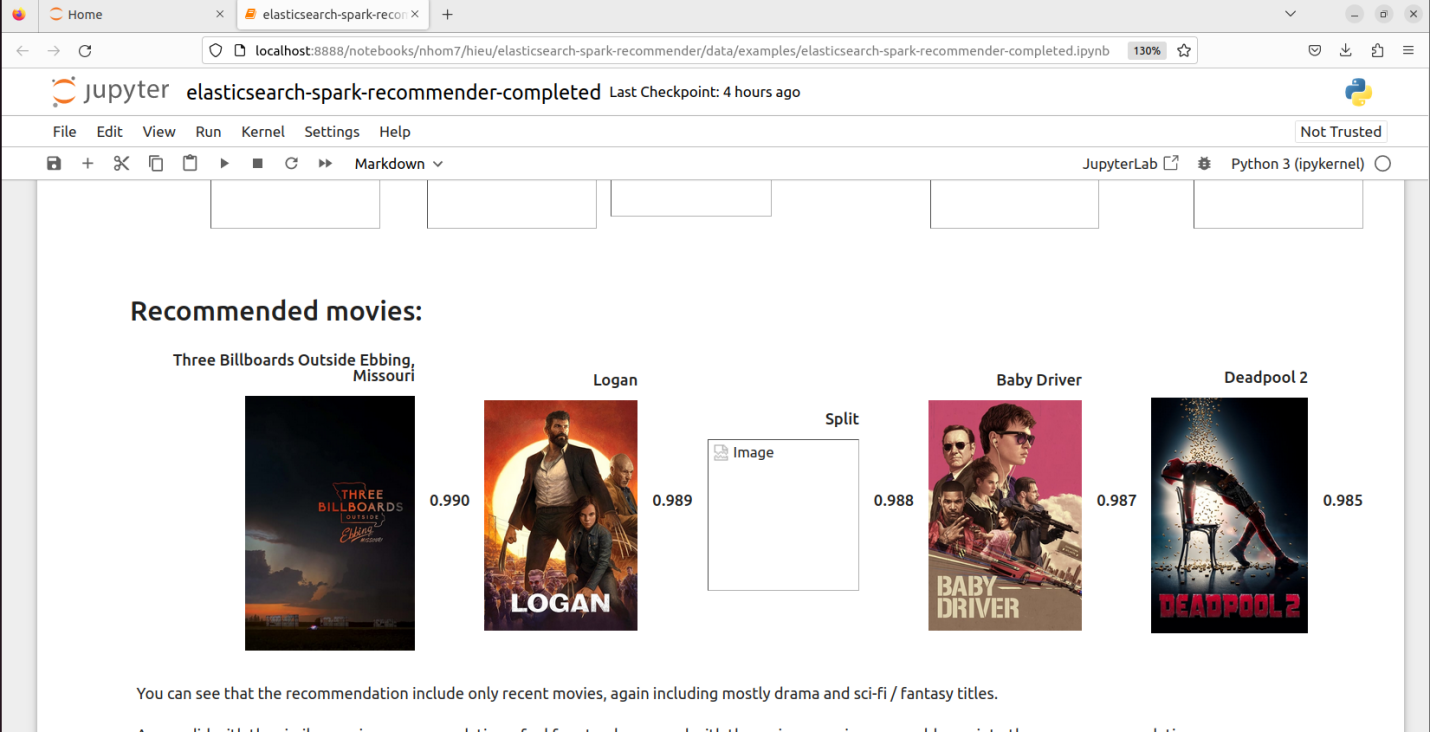
****

PYSPARK\_DRIVER\_PYTHON="jupyter" PYSPARK\_DRIVER\_PYTHON\_OPTS="notebook" nhom7/hieu/spark-2.2.0-bin-hadoop2.7/bin/pyspark --driver-memory 4g --driver-class-path nhom7/hieu/elasticsearch-hadoop-5.3.0/dist/elasticsearch-spark-20\_2.11-5.3.0.jar

**Kết Quả Đạt Được:**







**Phân tích và quan sát dữ liệu trước khi đưa vào sử dụng.**

**Bước 1:**

Sử dụng biểu thức chính quy (regular expression) để tìm kiếm xem tiêu đề có khớp với định dạng tiêu đề chuẩn hay không. Biểu thức chính quy được sử dụng là ^(.\*), The \((\d{4})\)<span class="math-inline">\, trong đó: \* `(.\*)`: Nhóm bắt đầu (capturing group) thứ nhất, dùng để bắt lấy tên bộ phim. \* `, The (`: Dấu phẩy và dấu ngoặc đơn mở. \* `(\d{4})`: Nhóm bắt đầu thứ hai, dùng để bắt lấy năm ra mắt của bộ phim. \* `)`: Dấu ngoặc đơn đóng.

Nếu tiêu đề khớp với định dạng tiêu đề chuẩn, hàm sẽ trả về tiêu đề đã được sửa thành dạng The Movie Name (Year).

Nếu tiêu đề không khớp với định dạng tiêu đề chuẩn, hàm sẽ trả về tiêu đề gốc.

Tác dụng của đoạn code này là để sửa lại tiêu đề của các bộ phim thành dạng chuẩn, giúp cho việc quản lý và tìm kiếm phim trở nên dễ dàng hơn.

A screenshot of a computer

Description automatically generated

**Bước 2:**

Đoạn code này thực hiện phân tích và trực quan hóa dữ liệu từ tệp CSV (movies.csv) để hiển thị top 10 thể loại phim phổ biến nhất. Dưới đây là phân tích từng bước của đoạn code:

`import pandas as pd`: Import thư viện Pandas để làm việc với dữ liệu.

`import matplotlib.pyplot as plt` và `import seaborn as sns`: Import thư viện Matplotlib và Seaborn để vẽ biểu đồ.

`movies = pd.read\_csv('movie.csv')`: Đọc dữ liệu từ tệp CSV 'movie.csv' và lưu vào biến `movies`. Đây là dữ liệu về các bộ phim.

`genres\_count = movies['genres'].str.split('|', expand=True).stack().value\_counts().head(10)`: Tạo một Series (danh sách các giá trị) với số lượng mỗi thể loại phim. Đầu tiên, cột 'genres' của DataFrame `movies` được tách thành các thể loại phim riêng lẻ bằng dấu '|' và mỗi thể loại được chuyển thành một dòng. Sau đó, `value\_counts()` đếm số lượng xuất hiện của mỗi thể loại và `head(10)` chỉ lấy 10 thể loại phim phổ biến nhất.

`plt.figure(figsize=(10, 6))`: Tạo một hình vẽ với kích thước là 10x6.

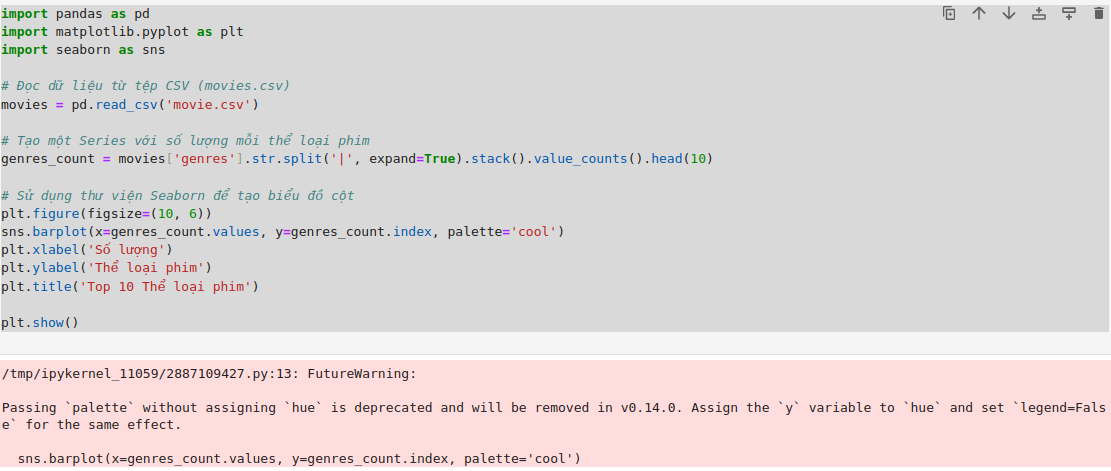
`sns.barplot(x=genres\_count.values, y=genres\_count.index, palette='cool')`: Sử dụng thư viện Seaborn để vẽ biểu đồ cột (bar chart). Trục x là số lượng (values) và trục y là tên thể loại phim (index) từ biến `genres\_count`. Màu sắc của biểu đồ được chọn từ bảng màu 'cool'.

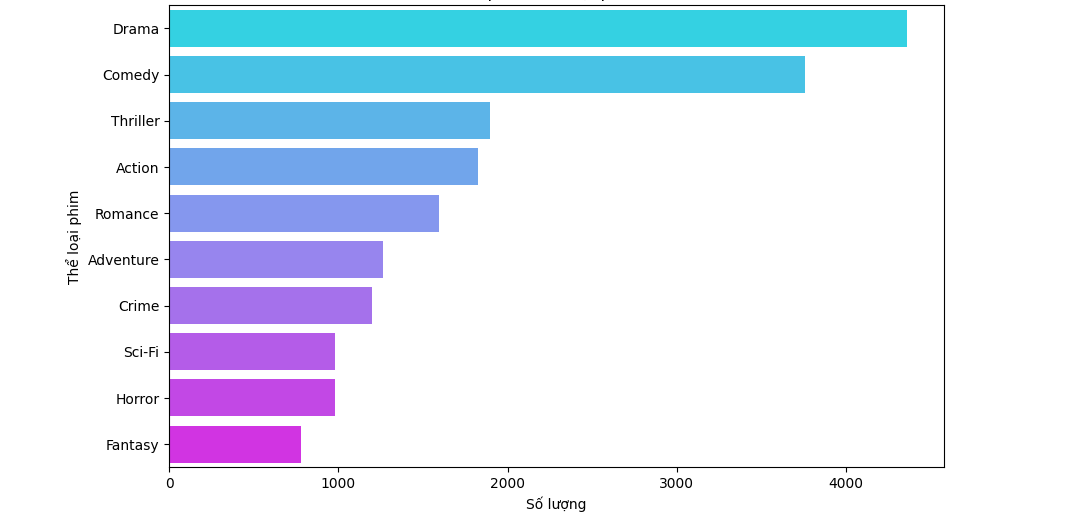
`plt.xlabel('Số lượng')`: Đặt nhãn cho trục x là "Số lượng".

`plt.ylabel('Thể loại phim')`: Đặt nhãn cho trục y là "Thể loại phim".

`plt.title('Top 10 Thể loại phim')`: Đặt tiêu đề cho biểu đồ.

`plt.show()`: Hiển thị biểu đồ cột đã tạo.





**Bước 3:**

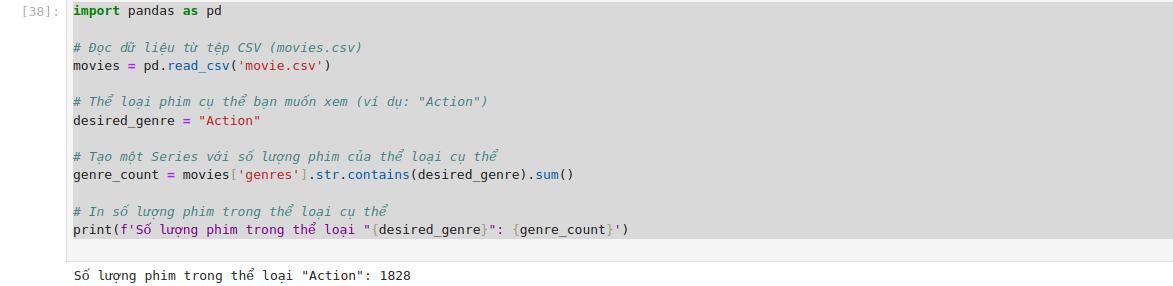
1. `import pandas as pd`: Import thư viện Pandas để làm việc với dữ liệu.

2. `movies = pd.read\_csv('movie.csv')`: Đọc dữ liệu từ tệp CSV 'movie.csv' và lưu vào biến `movies`. Đây là dữ liệu về các bộ phim.

3. `desired\_genre = "Action"`: Đặt thể loại phim cụ thể bạn muốn xem. Trong ví dụ này, thể loại là "Action".

4. `genre\_count = movies['genres'].str.contains(desired\_genre).sum()`: Tạo một Series (danh sách các giá trị) với kết quả của việc kiểm tra xem cột 'genres' của DataFrame `movies` có chứa thể loại cụ thể (ví dụ: "Action") không. Sau đó, sử dụng `sum()` để đếm số lượng trường hợp thỏa mãn điều kiện.

5. `print(f'Số lượng phim trong thể loại "{desired\_genre}": {genre\_count}')`: In số lượng phim trong thể loại cụ thể lên màn hình.



**Bước 4:**

Đoạn code này thực hiện việc đọc dữ liệu xếp hạng từ tệp CSV 'ratings.csv', sau đó đếm số lượng đánh giá có xếp hạng là 5 sao. Dưới đây là phân tích từng bước của đoạn code

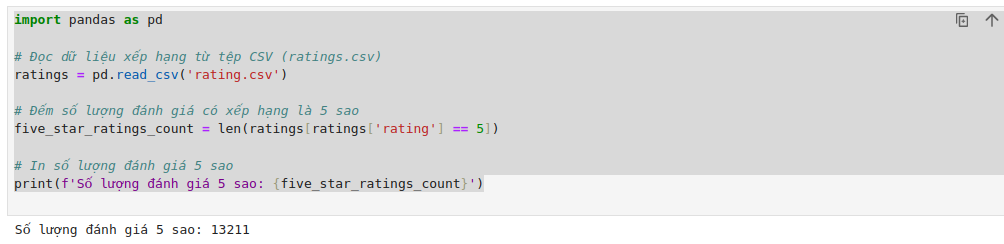
1. `import pandas as pd`: Import thư viện Pandas để làm việc với dữ liệu.

2. `ratings = pd.read\_csv('rating.csv')`: Đọc dữ liệu từ tệp CSV 'rating.csv' và lưu vào biến `ratings`. Đây là dữ liệu về xếp hạng của các bộ phim.

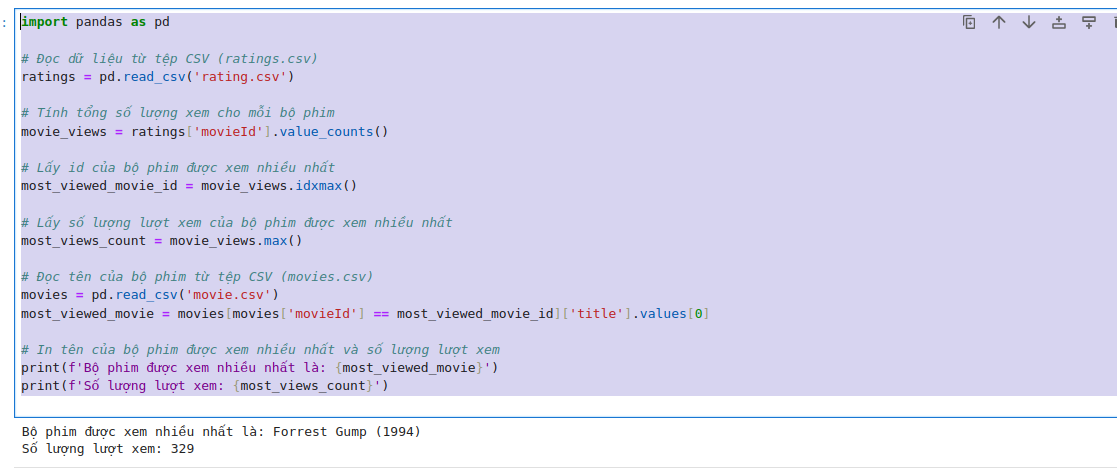
3. `five\_star\_ratings\_count = len(ratings[ratings['rating'] == 5])`: Tạo một Series (danh sách các giá trị) bằng cách chọn các dòng trong DataFrame `ratings` mà có xếp hạng bằng 5 (5 sao) và sau đó sử dụng `len()` để đếm số lượng dòng thỏa mãn điều kiện này.

4. `print(f'Số lượng đánh giá 5 sao: {five\_star\_ratings\_count}')`: In số lượng đánh giá có xếp hạng 5 sao lên màn hình.

Đoạn code này cho phép bạn đếm và in ra số lượng đánh giá có xếp hạng là 5 sao trong dữ liệu xếp hạng của các bộ phim.



**Bước 5:**



:

1. `import pandas as pd`: Import thư viện Pandas để làm việc với dữ liệu.

2. `ratings = pd.read\_csv('rating.csv')`: Đọc dữ liệu từ tệp CSV 'rating.csv' và lưu vào biến `ratings`. Đây là dữ liệu về xếp hạng và số lượng lượt xem của các bộ phim.

3. `movie\_views = ratings['movieId'].value\_counts()`: Tạo một Series (danh sách các giá trị) với số lượng lượt xem cho mỗi bộ phim. Dùng hàm `value\_counts()` để đếm số lượt xem cho mỗi bộ phim dựa trên cột 'movieId'.

4. `most\_viewed\_movie\_id = movie\_views.idxmax()`: Tìm id của bộ phim có số lượng lượt xem nhiều nhất bằng cách sử dụng `idxmax()`.

5. `most\_views\_count = movie\_views.max()`: Lấy số lượng lượt xem của bộ phim được xem nhiều nhất bằng cách sử dụng `max()`.

6. `movies = pd.read\_csv('movie.csv')`: Đọc tên của bộ phim từ tệp CSV 'movie.csv'.

7. `most\_viewed\_movie = movies[movies['movieId'] == most\_viewed\_movie\_id]['title'].values[0]`: Tìm và lấy tên của bộ phim được xem nhiều nhất bằng cách so sánh id bộ phim với `most\_viewed\_movie\_id` và truy xuất tên của bộ phim.

8. `print(f'Bộ phim được xem nhiều nhất là: {most\_viewed\_movie}')`: In tên của bộ phim được xem nhiều nhất.

9. `print(f'Số lượng lượt xem: {most\_views\_count}')`: In số lượng lượt xem của bộ phim được xem nhiều nhất.

**Bước 6:**

Đoạn code này thực hiện việc phân tích và gợi ý 10 bộ phim có xếp hạng trung bình cao nhất. Dưới đây là phân tích từng bước của đoạn code:

1. `import pandas as pd`: Import thư viện Pandas để làm việc với dữ liệu.

2. `movies = pd.read\_csv('movie.csv')` và `ratings = pd.read\_csv('rating.csv')`: Đọc dữ liệu từ tệp CSV 'movie.csv' và 'rating.csv' và lưu vào biến `movies` và `ratings`. `movies` chứa thông tin về các bộ phim, `ratings` chứa thông tin về xếp hạng của các bộ phim.

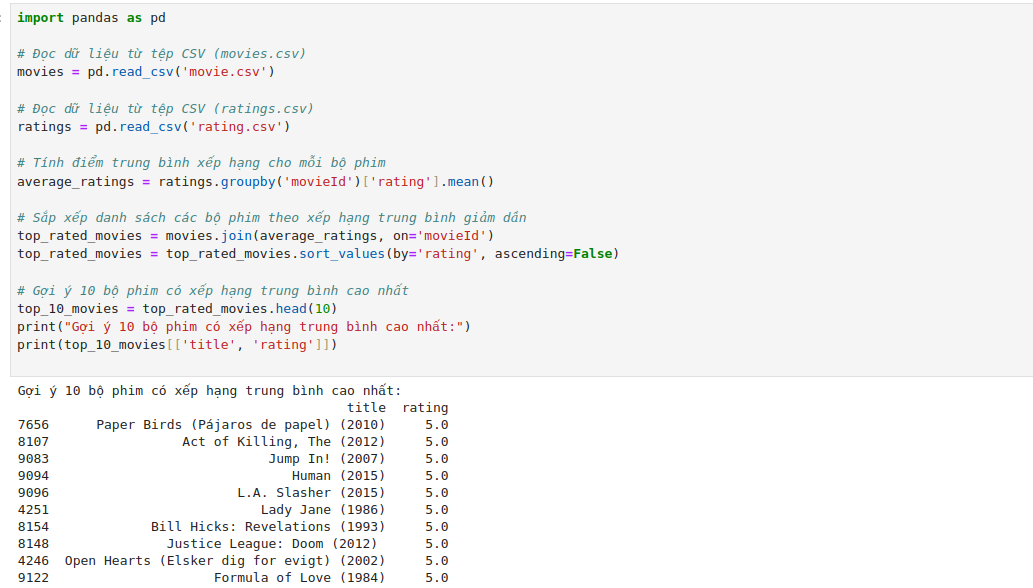
3. `average\_ratings = ratings.groupby('movieId')['rating'].mean()`: Tính điểm trung bình xếp hạng cho mỗi bộ phim bằng cách nhóm dữ liệu trong `ratings` theo cột 'movieId' và tính trung bình xếp hạng của từng nhóm.

4. `top\_rated\_movies = movies.join(average\_ratings, on='movieId')`: Kết hợp thông tin về bộ phim từ `movies` và điểm trung bình xếp hạng từ `average\_ratings` dựa trên cột 'movieId'.

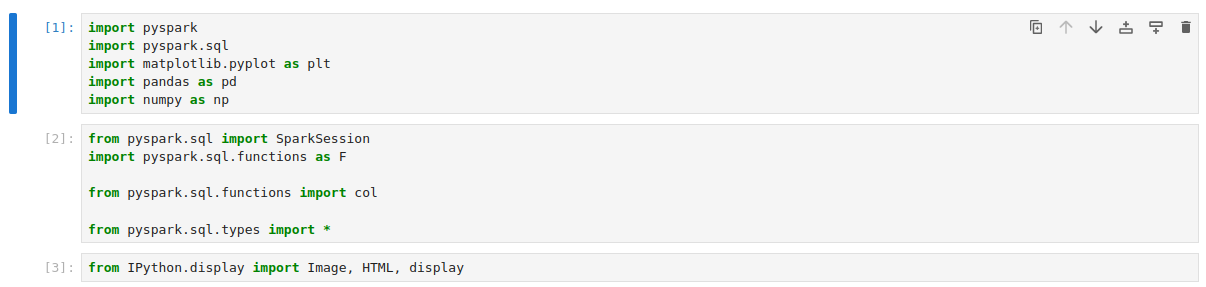
5. `top\_rated\_movies = top\_rated\_movies.sort\_values(by='rating', ascending=False)`: Sắp xếp danh sách các bộ phim theo xếp hạng trung bình giảm dần (từ cao đến thấp) dựa trên cột 'rating'.

6. `top\_10\_movies = top\_rated\_movies.head(10)`: Lấy 10 bộ phim ở đầu danh sách, đó là 10 bộ phim có xếp hạng trung bình cao nhất.

7. `print("Gợi ý 10 bộ phim có xếp hạng trung bình cao nhất:")` và `print(top\_10\_movies[['title', 'rating']])`: In ra màn hình danh sách 10 bộ phim có xếp hạng trung bình cao nhất, bao gồm tên của bộ phim và điểm trung bình xếp hạng của nó.

****

**Bước 1:** Import các thư viên Spark cũng như những công cụ phân tích sẽ được sử dụng trong quá trình xây dựng hệ thông phân tích đề xuát phim cho người dung đã được cài đặt trước đó. Đảm bảo rằng các thư viện được cài đặt thành công từ trước đó.



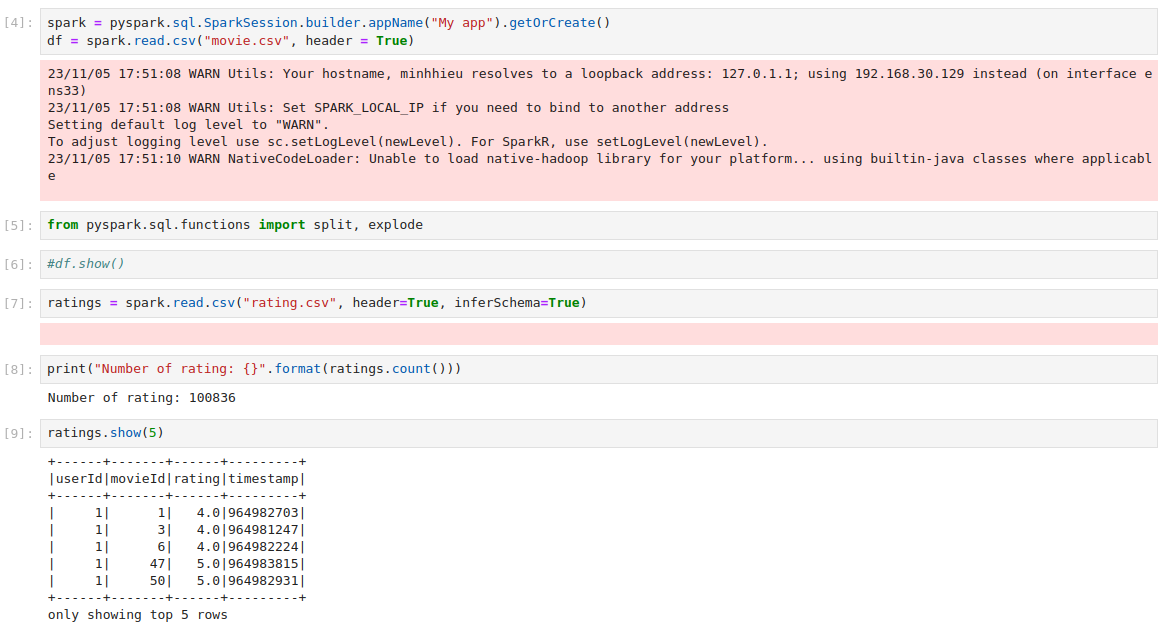
**Bước 2:**

spark = pyspark.sql.SparkSession.builder.appName("My app").getOrCreate() tạo một đối tượng SparkSession, đây là đối tượng được sử dụng để kết nối với Spark Cluster và thực hiện các tác vụ xử lý dữ liệu.

df = spark.read.csv("movie.csv", header = True) đọc dữ liệu từ tệp CSV movie.csv và lưu vào một đối tượng DataFrame. Tham số header = True cho biết rằng tệp CSV có chứa hàng tiêu đề.

Sử dụng spark đọc dữ liệu từ file rating.csv

Bước trên được gọi là bước đọc dữ liệu (data loading). Đây là bước đầu tiên trong quá trình xử lý dữ liệu, nhằm mục đích đưa dữ liệu từ nguồn dữ liệu ban đầu vào môi trường Spark để xử lý.



**Bước 3:**

Kiểm tra và lọc dữ liệu từ file movie cũng như kiểm tra dữ liệu từ file movie xem đã được clear hay chưa.

Cụ thể, hàm filter() sẽ lọc dữ liệu trong DataFrame dựa trên một điều kiện được chỉ định. Trong trường hợp này, điều kiện là df["genres"].contains("Comedy|Dramar"), nghĩa là bộ phim phải thuộc thể loại Comedy|Drama.

Sau khi lọc dữ liệu, ta sẽ có một DataFrame mới comedydr\_show chỉ chứa những bộ phim thuộc thể loại Comedy|Drama.



**Bước 4:**

* rom pyspark.sql.functions import udf và from pyspark.sql.types import \* import các hàm và lớp cần thiết để định nghĩa và sử dụng UDF.
* extract\_genres = udf(lambda x: x.lower().split("|"), ArrayType(StringType())) định nghĩa một UDF có tên là extract\_genres(), UDF này có tác dụng chuyển đổi chuỗi thể loại thô (raw genres string) thành một mảng các thể loại (array of genres) với các giá trị được viết thường. Tham số ArrayType(StringType()) cho biết kiểu dữ liệu trả về của UDF là một mảng các chuỗi.
* test it out là một dòng bình luận, không ảnh hưởng đến việc thực thi chương trình.
* raw\_movies.select("movieId", "title", extract\_genres("genres").alias("genres")).show(5, False) sử dụng UDF extract\_genres() để chuyển đổi chuỗi thể loại thô trong cột genres của DataFrame raw\_movies thành một mảng các thể loại với các giá trị được viết thường. Sau đó, chọn các cột movieId, title, và genres và hiển thị 5 hàng đầu tiên của DataFrame kết quả.

Mục đích của đoạn code trên là để chuyển đổi chuỗi thể loại thô thành một mảng các thể loại với các giá trị được viết thường. Điều này có thể hữu ích cho nhiều mục đích khác nhau, chẳng hạn như:

Để lọc dữ liệu dựa trên thể loại phim.

Để tính toán thống kê về các thể loại phim.

Để tạo các tính năng mới cho mô hình machine learning.





**Bước 5:**

* import re import thư viện re để sử dụng các hàm và lớp liên quan đến việc biểu thức chính quy (regular expression).
* # define a UDF to extract the release year from the title, and return the new title and year in a struct type là dòng bình luận, giải thích mục đích của UDF.
* def extract\_year\_fn(title): định nghĩa một UDF có tên là extract\_year\_fn(), UDF này có tác dụng trích xuất năm phát hành từ tiêu đề phim và trả về tiêu đề mới và năm phát hành trong một struct type.
* result = re.search("\(\d{4}\)", title) sử dụng biểu thức chính quy để tìm kiếm năm phát hành trong tiêu đề phim.
* try::
  + if result:: nếu tìm thấy năm phát hành, thì trích xuất năm phát hành và vị trí bắt đầu của năm phát hành trong tiêu đề phim.
  + title = title[:start\_pos-1] loại bỏ năm phát hành khỏi tiêu đề phim.
  + return (title, year) trả về tiêu đề mới và năm phát hành trong một struct type.
* except:: nếu không tìm thấy năm phát hành, thì trả về tiêu đề phim gốc và năm 1970.
* extract\_year = udf(extract\_year\_fn,\ : định nghĩa một UDF có tên làextract\_year()dựa trên hàmextract\_year\_fn(), với kiểu dữ liệu trả về là một struct type gồm hai trường:title(kiểu dữ liệu StringType) vàrelease\_date` (kiểu dữ liệu StringType).
* # test out our function là dòng bình luận, giải thích cách sử dụng UDF.
* s = "Jumanji (1995)" tạo một biến s chứa tiêu đề phim Jumanji (1995).
* extract\_year\_fn(s) gọi UDF extract\_year\_fn() với tham số là biến s.

Tác dụng của đoạn code trên là để định nghĩa và sử dụng một UDF để trích xuất năm phát hành từ tiêu đề phim và trả về tiêu đề mới và năm phát hành trong một struct type. UDF này có thể được sử dụng cho nhiều mục đích khác nhau, chẳng hạn như:

* Lọc dữ liệu dựa trên năm phát hành.
* Tính toán thống kê về năm phát hành.
* Tạo các tính năng mới cho mô hình machine learning.



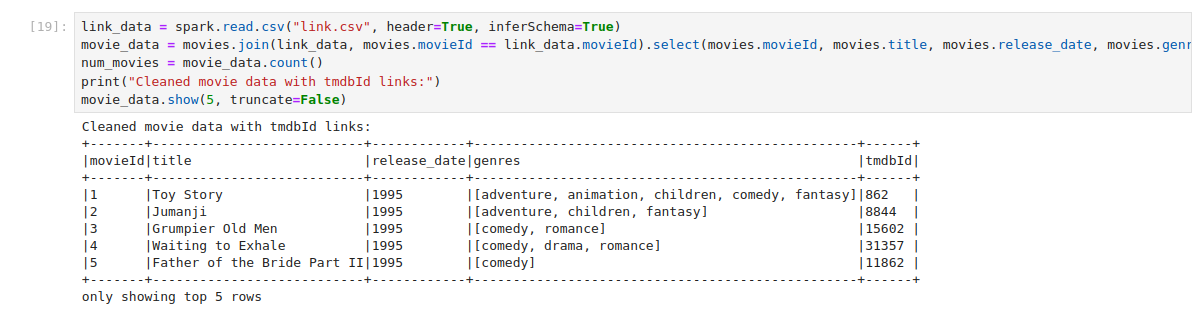


**Bước 6:**

* link\_data = spark.read.csv("link.csv", header=True, inferSchema=True): đọc dữ liệu từ tệp CSV link.csv và lưu vào một đối tượng DataFrame link\_data. Tham số header=True cho biết rằng tệp CSV có chứa hàng tiêu đề. Tham số inferSchema=True cho biết Spark sẽ tự động suy luận kiểu dữ liệu của các cột trong DataFrame.
* movie\_data = movies.join(link\_data, movies.movieId == link\_data.movieId).select(movies.movieId, movies.title, movies.release\_date, movies.genres, link\_data.tmdbId): kết hợp dữ liệu của hai DataFrame movies và link\_data dựa trên cột movieId. Sau đó, chọn các cột movieId, title, release\_date, genres, và tmdbId từ DataFrame kết hợp và lưu vào một đối tượng DataFrame movie\_data.
* num\_movies = movie\_data.count(): tính toán số lượng phim trong DataFrame movie\_data.
* print("Cleaned movie data with tmdbId links:"): in ra thông báo "Cleaned movie data with tmdbId links:".
* movie\_data.show(5, truncate=False): hiển thị 5 hàng đầu tiên của DataFrame movie\_data mà không cắt ngắn dữ liệu.

Tác dụng của đoạn code trên là để đọc dữ liệu từ hai tệp CSV link.csv và movie.csv, kết hợp dữ liệu của hai tệp này và tính toán số lượng phim có liên kết đến TMDb. Dữ liệu kết hợp sau đó có thể được sử dụng cho nhiều mục đích khác nhau, chẳng hạn như:

* Đề xuất phim cho người dùng dựa trên các bộ phim mà họ đã xem và đánh giá.
* Tạo các tính năng mới cho mô hình machine learning để dự đoán sở thích của người dùng đối với các bộ phim khác nhau.
* Hiểu rõ hơn về xu hướng xem phim của người dùng.



**Bước 7:**

* try::
  + import tmdbsimple as tmdb: import thư viện tmdbsimple.
  + import json: import thư viện json để xử lý dữ liệu JSON.
  + from requests.exceptions import HTTPError: import lớp HTTPError từ thư viện requests để xử lý các lỗi HTTP.
  + tmdb.API\_KEY = 'e9e9d8da18ae29fc430845952232787c': thiết lập khóa API cho TMDb.
  + print("Successfully imported tmdbsimple!"): in ra thông báo "Successfully imported tmdbsimple!".
  + IMAGE\_URL = 'https://image.tmdb.org/t/p/w500': tạo biến IMAGE\_URL chứa URL cơ sở cho ảnh poster của TMDb.
  + movie\_id = movie\_data.first().tmdbId: lấy ID phim đầu tiên trong DataFrame movie\_data.
  + movie\_info = tmdb.Movies(movie\_id).info(): lấy thông tin về phim từ TMDb dựa trên ID phim.
  + movie\_poster\_url = IMAGE\_URL + movie\_info['poster\_path']: tạo biến movie\_poster\_url chứa URL ảnh poster của bộ phim.
  + display(Image(movie\_poster\_url, width=200)): hiển thị ảnh poster của bộ phim với chiều rộng là 200 pixel.
* except ImportError::
  + print("Cannot import tmdbsimple as it is not installed, no movie posters will be displayed!"): in ra thông báo "Cannot import tmdbsimple as it is not installed, no movie posters will be displayed!".
* except HTTPError as e::
  + if e.response.status\_code == 401::
    - j = json.loads(e.response.text): tải dữ liệu JSON từ phản hồi HTTP.
    - print("TMdb API call failed: {}".format(j['status\_message'])): in ra thông báo "TMdb API call failed: {}".format(j['status\_message'])".

Tác dụng của đoạn code trên là để lấy thông tin về phim từ TMDb và hiển thị ảnh poster của bộ phim. Đoạn code này có thể được sử dụng cho nhiều mục đích khác nhau, chẳng hạn như:

* Hiển thị ảnh poster của phim trong trang web hoặc ứng dụng.
* Tạo bộ dữ liệu gồm thông tin về phim và ảnh poster của phim để sử dụng cho các mục đích khác, chẳng hạn như huấn luyện mô hình machine learning.
* Hiểu rõ hơn về các bộ phim mà người dùng quan tâm.



**Bước 8:**

Import elasticsearch để sử dụng để xây dựng hệ thống tích hợp cho Spark và Elasticseach.

* es = Elasticsearch("http://localhost:9200"): tạo một đối tượng es để kết nối với cụm Elasticsearch có địa chỉ là http://localhost:9200.
* es.info(pretty=True): lấy thông tin về cụm Elasticsearch và in ra thông tin này theo định dạng đẹp.

Tác dụng của đoạn code trên là để kết nối với cụm Elasticsearch và lấy thông tin về cụm. Thông tin này bao gồm:

* Phiên bản Elasticsearch
* Cấu hình cụm
* Số lượng node trong cụm
* Số lượng index trong cụm
* Số lượng document trong cụm

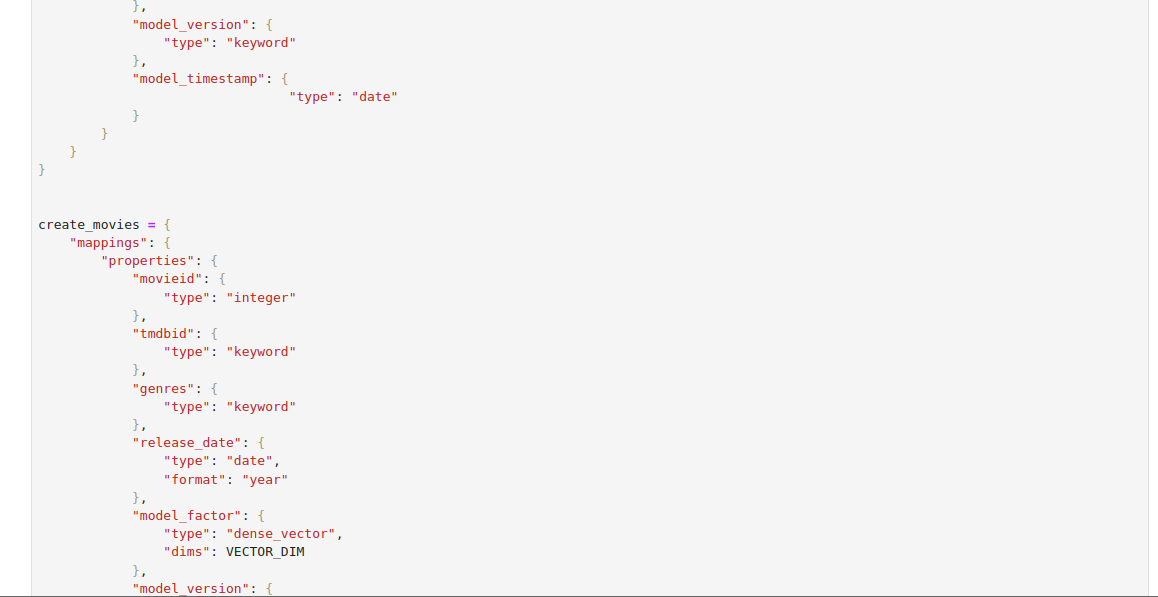


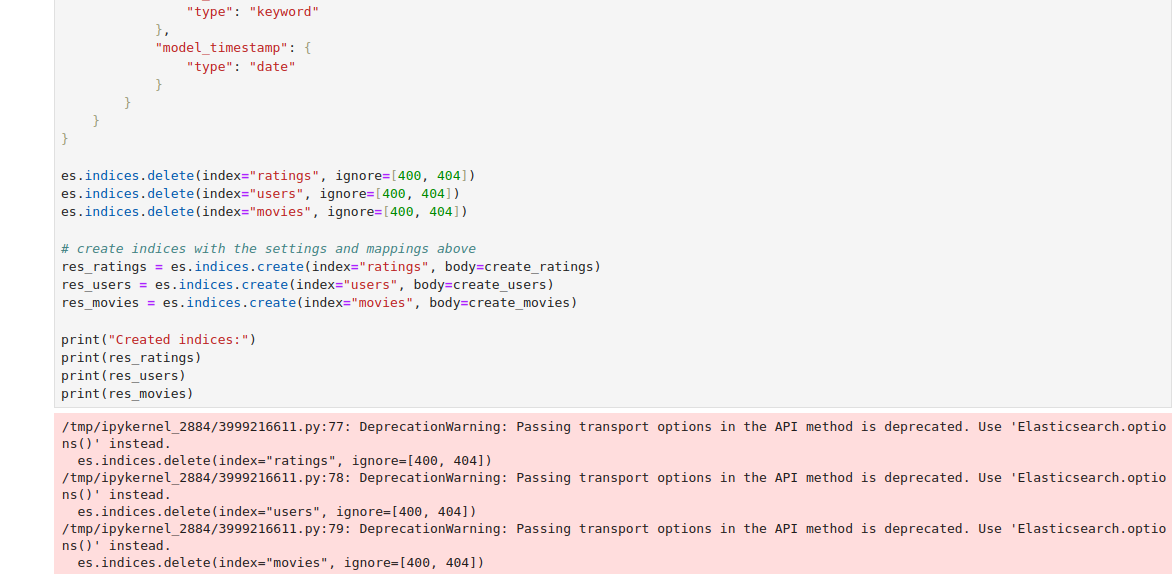
**Bước 9:**

* # set the factor vector dimension for the recommendation model định nghĩa biến VECTOR\_DIM để thiết lập kích thước của vector đặc trưng cho mô hình đề xuất phim.
* create\_ratings, create\_users, và create\_movies định nghĩa cấu trúc của ba index ratings, users, và movies.
* es.indices.delete(index="ratings", ignore=[400, 404]), es.indices.delete(index="users", ignore=[400, 404]), và es.indices.delete(index="movies", ignore=[400, 404]) xóa các index ratings, users, và movies nếu có.
* res\_ratings = es.indices.create(index="ratings", body=create\_ratings), res\_users = es.indices.create(index="users", body=create\_users), và res\_movies = es.indices.create(index="movies", body=create\_movies) tạo các index ratings, users, và movies với cấu trúc đã định nghĩa ở trên.
* print("Created indices:") và print(res\_ratings), print(res\_users), và print(res\_movies) in ra các index đã được tạo.

Tác dụng của đoạn code trên là để tạo các index ratings, users, và movies trong Elasticsearch để lưu trữ dữ liệu về xếp hạng phim, người dùng, và phim. Dữ liệu này sau đó có thể được sử dụng để huấn luyện và triển khai mô hình đề xuất phim.







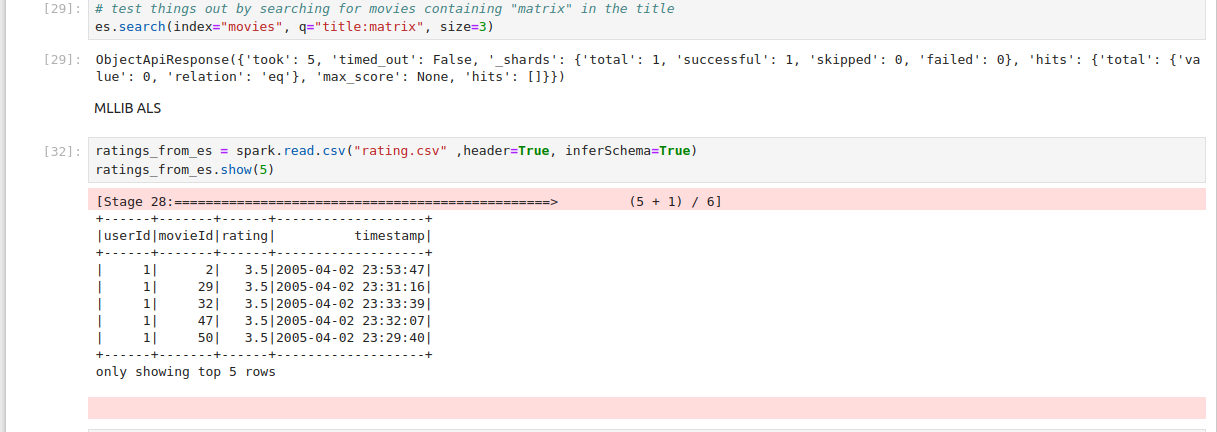
**Bước 10:**

* spark.read.csv("rating.csv" ,header=True, inferSchema=True):
  + spark.read.csv(): tạo một đối tượng DataFrameReader để đọc dữ liệu từ tệp CSV.
  + "rating.csv": tham số đầu tiên của hàm spark.read.csv() là đường dẫn đến tệp CSV.
  + header=True: tham số thứ hai của hàm spark.read.csv() cho biết rằng tệp CSV có chứa hàng tiêu đề.
  + inferSchema=True: tham số thứ ba của hàm spark.read.csv() cho biết rằng Spark sẽ tự động suy luận kiểu dữ liệu của các cột trong DataFrame.

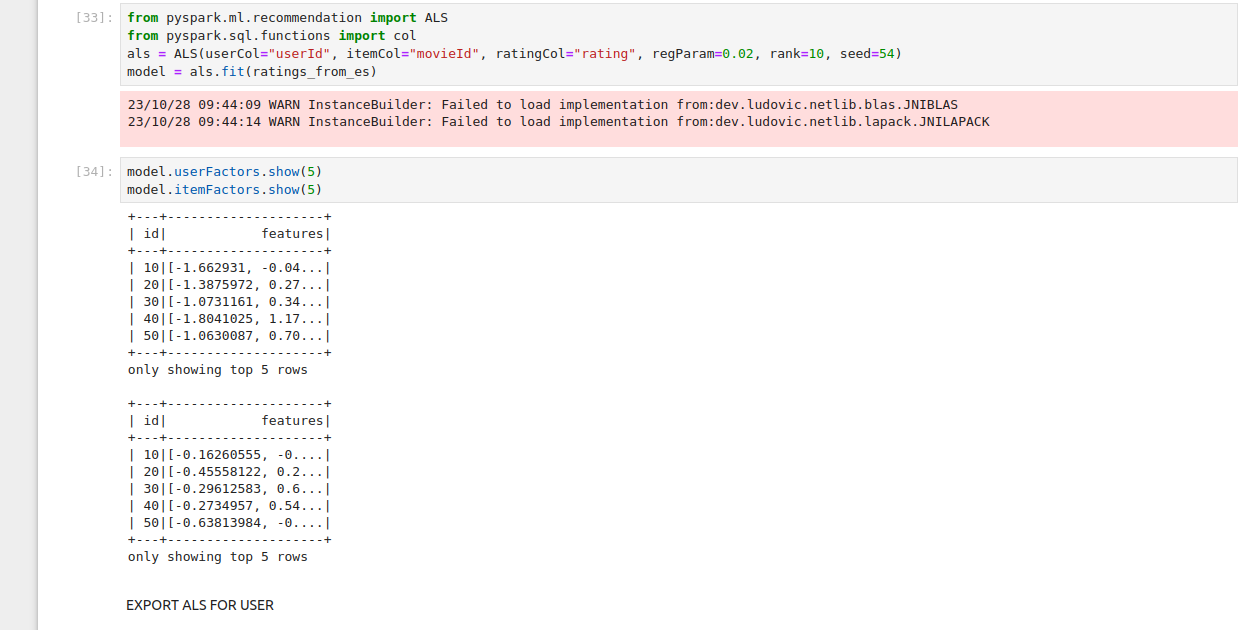
Sau khi đọc dữ liệu xếp hạng phim vào DataFrame Spark, ta có thể sử dụng các hàm và phương thức của Spark để xử lý dữ liệu và huấn luyện mô hình đề xuất phim.

* from pyspark.ml.recommendation import ALS: import lớp ALS từ thư viện Spark ML.
* from pyspark.sql.functions import col: import hàm col() từ thư viện Spark SQL. Hàm col() được sử dụng để truy cập các cột trong DataFrame.
* als = ALS(userCol="userId", itemCol="movieId", ratingCol="rating", regParam=0.02, rank=10, seed=54): tạo một đối tượng ALS với các tham số sau:
  + userCol: tên cột chứa ID người dùng.
  + itemCol: tên cột chứa ID bộ phim.
  + ratingCol: tên cột chứa xếp hạng của người dùng đối với bộ phim.
  + regParam: tham số điều chỉnh độ mạnh của thuật toán ALS.
  + rank: số lượng yếu tố ẩn được sử dụng trong thuật toán ALS.
  + seed: hạt giống được sử dụng để khởi tạo thuật toán ALS.
* model = als.fit(ratings\_from\_es): huấn luyện mô hình đề xuất phim sử dụng dữ liệu xếp hạng phim trong DataFrame ratings\_from\_es.

Sau khi mô hình được huấn luyện, ta có thể sử dụng nó để dự đoán xếp hạng của người dùng đối với các bộ phim khác nhau và để đề xuất phim cho người dùng.



**Bước 11:**

* model.userFactors: trả về DataFrame chứa vector đặc trưng của người dùng. Mỗi hàng trong DataFrame này đại diện cho một người dùng và mỗi cột đại diện cho một yếu tố ẩn.
* model.itemFactors: trả về DataFrame chứa vector đặc trưng của bộ phim. Mỗi hàng trong DataFrame này đại diện cho một bộ phim và mỗi cột đại diện cho một yếu tố ẩn.
* show(5): hiển thị 5 hàng đầu tiên của DataFrame.
* 

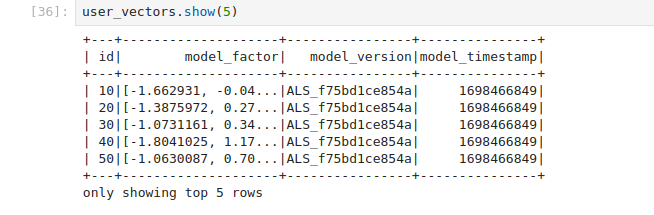
**Bước 12:**

* from pyspark.sql.functions import lit, current\_timestamp, unix\_timestamp: import các hàm lit(), current\_timestamp(), và unix\_timestamp() từ thư viện Spark SQL.
* ver = model.uid: lưu ID của mô hình vào biến ver.
* ts = unix\_timestamp(current\_timestamp()): lấy timestamp hiện tại và chuyển đổi thành timestamp Unix.
* movie\_vectors = model.itemFactors.select("id",\ col("features").alias("model\_factor"),\ lit(ver).alias("model\_version"),\ ts.alias("model\_timestamp")): tạo DataFrame movie\_vectors chứa các cột sau:
  + id: ID của bộ phim.
  + model\_factor: vector đặc trưng của bộ phim.
  + model\_version: phiên bản của mô hình đề xuất phim.
  + model\_timestamp: timestamp mà mô hình đề xuất phim được huấn luyện.
* movie\_vectors.show(5): hiển thị 5 hàng đầu tiên của DataFrame movie\_vectors.
* user\_vectors = model.userFactors.select("id",\ col("features").alias("model\_factor"),\ lit(ver).alias("model\_version"),\ ts.alias("model\_timestamp")): tạo DataFrame user\_vectors chứa các cột sau:
  + id: ID của người dùng.
  + model\_factor: vector đặc trưng của người dùng.
  + model\_version: phiên bản của mô hình đề xuất phim.
  + model\_timestamp: timestamp mà mô hình đề xuất phim được huấn luyện.

Sau khi tạo DataFrame movie\_vectors và user\_vectors, ta có thể lưu chúng vào Elasticsearch để sử dụng trong mô hình đề xuất phim.

Tác dụng của đoạn code này là để tạo hai DataFrame movie\_vectors và user\_vectors, chứa vector đặc trưng của bộ phim và người dùng sau khi mô hình đề xuất phim được huấn luyện. Hai DataFrame này có thể được sử dụng để lưu trữ thông tin về mô hình đề xuất phim và vector đặc trưng của bộ phim và người dùng, từ đó giúp cải thiện hiệu suất và khả năng mở rộng của mô hình đề xuất phim.





**Bước 13:**

* get\_poster\_url(id): lấy URL hình ảnh poster của bộ phim có ID id từ TMDb API.
* vector\_query(query\_vec, vector\_field, q="\*", cosine=False): tạo một truy vấn Elasticsearch sử dụng điểm số kịch bản để tìm các tài liệu có vector đặc trưng tương tự với vector query\_vec. Tham số vector\_field là tên của trường chứa vector đặc trưng trong tài liệu. Tham số q là chuỗi truy vấn cho truy vấn tìm kiếm (mặc định là '\*' để tìm kiếm trên tất cả các tài liệu). Tham số cosine xác định liệu có sử dụng độ tương tự cosine để tính toán điểm số hay không.
* get\_similar(the\_id, q="\*", num=10, index="movies", vector\_field='model\_factor'): tìm các bộ phim tương tự với bộ phim có ID the\_id bằng cách sử dụng truy vấn tìm kiếm điểm số kịch bản. Tham số num xác định số lượng bộ phim trả về. Tham số index là tên của chỉ mục Elasticsearch chứa dữ liệu phim. Tham số vector\_field là tên của trường chứa vector đặc trưng của bộ phim trong chỉ mục.
* get\_user\_recs(the\_id, q="\*", num=10, users="users", movies="movies", vector\_field='model\_factor'): tìm các bộ phim được đề xuất cho người dùng có ID the\_id bằng cách sử dụng truy vấn tìm kiếm điểm số kịch bản. Tham số num xác định số lượng bộ phim trả về. Tham số users là tên của chỉ mục Elasticsearch chứa dữ liệu người dùng. Tham số movies là tên của chỉ mục Elasticsearch chứa dữ liệu phim. Tham số vector\_field là tên của trường chứa vector đặc trưng của người dùng trong chỉ mục.
* get\_movies\_for\_user(the\_id, num=10, ratings="ratings", movies="movies"): lấy danh sách các bộ phim được đánh giá bởi người dùng có ID the\_id, được sắp xếp từ cao nhất đến thấp nhất theo xếp hạng. Tham số num xác định số lượng bộ phim trả về. Tham số ratings là tên của chỉ mục Elasticsearch chứa dữ liệu xếp hạng. Tham số movies là tên của chỉ mục Elasticsearch chứa dữ liệu phim.
* display\_user\_recs(the\_id, q="\*", num=10, num\_last=10, users="users", movies="movies", ratings="ratings"): hiển thị các đề xuất phim cho người dùng có ID the\_id. Tham số num xác định số lượng đề xuất phim trả về. Tham số num\_last xác định số lượng bộ phim được đánh giá gần đây nhất của người dùng được hiển thị. Tham số users là tên của chỉ mục Elasticsearch chứa dữ liệu người dùng. Tham số movies là tên của chỉ mục Elasticsearch chứa dữ liệu phim. Tham số ratings là tên của chỉ mục Elasticsearch chứa dữ liệu xếp hạng.
* display\_similar(the\_id, q="\*", num=10, movies="movies"): hiển thị danh sách các bộ phim tương tự với bộ phim có ID the\_id. Tham số num xác định số lượng bộ phim trả về. Tham số movies là tên của chỉ mục Elasticsearch chứa dữ liệu phim.

Tác dụng của đoạn code trên là để hiển thị các đề xuất phim cho người dùng hoặc tìm các bộ phim tương tự với một bộ phim đã cho. Điều này có thể được sử dụng để xây dựng các hệ thống đề xuất phim hoặc tìm kiếm phim.

