**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**ĐẠI HỌC CẦN THƠ**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**



**ĐỒ ÁN**

**HỌC PHẦN MÁY HỌC ỨNG DỤNG**

**Đề tài**

**HỆ THỐNG DỰ ĐOÁN ĐIỂM VÀ GỢI Ý PHIM CHO NGƯỜI DÙNG**

**Sinh viên thực hiện: Đặng Nhật Duy - B2105568**

**Ung Ngọc Diễm Trinh - B2308397**

**Lê Minh Thư - B2308395**

**Giáo viên hướng dẫn: Mã Trường Thành**

**Tháng 07/2025**

# NHẬN XÉT CỦA GIẢNG VIÊN

**MỤC LỤC**

[PHÂN CÔNG CÔNG VIỆC 4](#_Toc204680796)

[PHẦN GIỚI THIỆU 5](#_Toc204680797)

[PHẦN NỘI DUNG 5](#_Toc204680798)

[1. Mô tả dữ liệu 5](#_Toc204680799)

[2. Ý nghĩa của dữ liệu 6](#_Toc204680800)

[3. Phân tích dữ liệu và lựa chọn mô hình 7](#_Toc204680801)

[3.1 KNN 7](#_Toc204680802)

[3.2 Decision Tree 10](#_Toc204680803)

[3.3 Linear Regression 13](#_Toc204680804)

[3.4 NCF 15](#_Toc204680805)

[3.5 LightGBM 19](#_Toc204680806)

[3.6 SVD 21](#_Toc204680807)

[4. Cấu hình môi trường 24](#_Toc204680808)

[5. Huấn luyện và kết quả thực nghiệm 24](#_Toc204680809)

[5.1 Quá trình xử lí dữ liệu 24](#_Toc204680810)

[5.2 Quá trình huấn luyện 24](#_Toc204680811)

[6. Đánh giá mô hình 25](#_Toc204680812)

[6.1 Các chỉ số đánh giá 25](#_Toc204680813)

[6.2 Kết quả đánh giá 26](#_Toc204680814)

[PHẦN KẾT LUẬN 27](#_Toc204680815)

[TÀI LIỆU THAM KHẢO 28](#_Toc204680816)

# PHÂN CÔNG CÔNG VIỆC

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Họ Tên | MSSV | Công việc |
| 1 | Đặng Nhật Duy | B2105568 | - mô hình: SVD, decision tree  - thuyết trình  - code giao diện |
| 2 | Lê Minh Thư | B2308395 | - mô hình: KNN, NCF  - file ppt thuyết trình  - viết báo cáo |
| 3 | Ung Ngọc Diễm Trinh | B2308397 | - mô hình:LightGBM, linear regression  - ghi chép  - viết báo cáo |

|  |  |
| --- | --- |
| Deadline | Công việc |
| 26/6 - 30/6 | - Cài đặt môi trường trên máy tính cá nhân  - Chạy thử mô hình trên dự án có sẵn (Recommenders) |
| 1/7 - 7/7 | - Tìm hiểu lý thuyết về các mô hình đã phân công  - Tải và nghiên cứu về tập dữ liệu MovieLens 100K |
| 10/7 | - Thực hiện huấn luyện và đánh giá mô hình đã được phân công trên tập dữ liệu đã chọn  - Xuất mô hình thành file .pkl |
| 11/7 - 17/7 | - Duy: Làm giao diện ứng dụng  - Thư: Chuẩn bị file thuyết trình và viết báo cáo  - Trinh: Viết báo cáo |
| 17/7 - 25/7 | Hoàn thiện và sửa lỗi của đề tài |
| 26/7 - 29/7 | Tổng kết |

# PHẦN GIỚI THIỆU

Hệ thống gợi ý (Recommender System – RS) là một kỹ thuật lọc thông tin tiên tiến, được ứng dụng rộng rãi trong nhiều lĩnh vực như thương mại điện tử, giải trí, giáo dục, nhằm dự đoán và đáp ứng nhu cầu, sở thích cá nhân của người dùng. Mục tiêu tổng quát của đề tài là nghiên cứu và xây dựng một hệ thống gợi ý phim cá nhân hóa, nhằm tối ưu hóa trải nghiệm người dùng trong quá trình tìm kiếm và lựa chọn nội dung. Hệ thống được thiết kế để tự động hóa quy trình đề xuất, cung cấp các khuyến nghị phù hợp với sở thích cá nhân của người dùng thông qua việc phân tích dữ liệu hành vi, đánh giá lịch sử và phản hồi trước đó. Đề tài tập trung khai thác và kết hợp hai phương pháp gợi ý phổ biến là lọc cộng tác (Collaborative Filtering) và lọc theo nội dung (Content-Based Filtering), nhằm nâng cao chất lượng đề xuất, giảm thiểu tình trạng quá tải thông tin và giúp người dùng nhanh chóng tìm thấy các nội dung quan tâm. Chức năng cốt lõi của hệ thống là phân tích hành vi và thói quen tiêu dùng để đưa ra các gợi ý, cụ thể là các bộ phim có khả năng phù hợp với sở thích cá nhân, đồng thời cải thiện hiệu suất tìm kiếm và tối ưu hóa khả năng tương tác giữa người dùng và hệ thống.

Trong đề tài lần này, nhóm tác giả sẽ áp dụng các mô hình máy học truyền thống và các mô hình học sâu chuyên về hệ thống gợi ý để đưa ra các nhận xét về mặt hiệu suất của từng mô hình. Đối với các mô hình máy học truyền thống, nhóm tác giả chọn các mô hình có thể dự đoán cho bài toán hồi quy.

# PHẦN NỘI DUNG

## Mô tả dữ liệu

MovieLens 100K là một bộ dữ liệu tiêu chuẩn được sử dụng phổ biến trong các bài toán hệ thống gợi ý. Bộ dữ liệu bao gồm 100.000 đánh giá (thang điểm 1–5) từ 943 người dùng trên 1.682 bộ phim, do GroupLens phát hành. Dữ liệu còn kèm theo thông tin cơ bản về người dùng và phim. Với kích thước vừa phải, MovieLens 100K thích hợp cho việc thử nghiệm, đánh giá và so sánh hiệu quả của các thuật toán máy học trong hệ thống đề xuất.

Dữ liệu của MovieLens 100K cân bằng theo điểm (rating) và theo tần suất đánh giá của phim/người dùng. Mức rating 4 và 3 chiếm đa số, các mức 1–2 rất ít, được minh họa trong Hình 1.

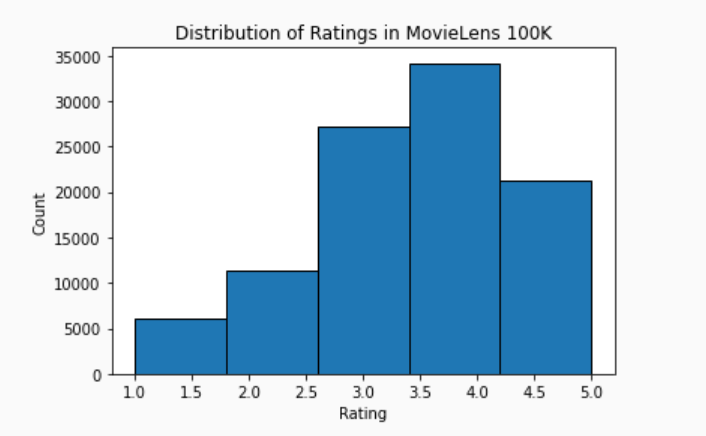
Các thành phần chính của tập dữ liệu bao gồm:

u.data: Tập tin chính chứa hơn 100.000 dòng đánh giá. Mỗi dòng gồm bốn trường: user id, item id (phim), rating (1–5) và timestamp (mốc thời gian đánh giá).

u.item: Thông tin về các phim. Mỗi phim được định danh bởi movie id, kèm theo tiêu đề, ngày phát hành, ngày video phát hành và liên kết IMDb. Cuối cùng là 19 cột nhị phân biểu diễn thể loại phim (Action, Drama, …).

u.user: Thông tin người dùng gồm user id, tuổi, giới tính, nghề nghiệp và mã ZIP. Các thuộc tính này cho biết hồ sơ nhân khẩu của từng người dùng.

u.occupation, u.genre: Tương ứng liệt kê các nghề nghiệp và thể loại phim.



Hình . Đồ thị phân phối số lượng đánh giá trong MovieLens 100K

## Ý nghĩa của dữ liệu

Tập dữ liệu MovieLens 100K cung cấp một nền tảng chuẩn để triển khai và kiểm thử các mô hình gợi ý. Nhờ có dữ liệu phân loại rõ ràng, hệ thống có thể huấn luyện các mô hình lọc cộng dựa trên đánh giá người dùng, đồng thời trích xuất đặc trưng nội dung để phục vụ cho lọc dựa trên nội dung. Đây là tập dữ liệu công khai và có tài liệu đi kèm đầy đủ, MovieLens 100K cho phép các nghiên cứu khác nhau tái lập và so sánh kết quả một cách khách quan. Điều này góp phần chuẩn hóa quá trình đánh giá hệ thống gợi ý, đảm bảo tính khoa học và minh bạch của nghiên cứu.

Mặc dù có quy mô vừa phải, MovieLens 100K vẫn thể hiện sự đa dạng trong sở thích người dùng và thể loại phim, tạo điều kiện thuận lợi cho việc kiểm thử hiệu suất mô hình trong các kịch bản khác nhau. Ngoài ra, việc sử dụng phiên bản 100K cũng giúp rút ngắn thời gian xử lý và huấn luyện mô hình trong các giai đoạn nghiên cứu ban đầu. Tóm lại, dữ liệu MovieLens 100K đóng vai trò là tập dữ liệu tham chiếu có độ tin cậy cao, vừa đảm bảo tính khả thi trong triển khai mô hình, vừa có giá trị học thuật trong việc so sánh và phân tích các phương pháp gợi ý khác nhau.

## Phân tích dữ liệu và lựa chọn mô hình

### KNN

#### Giới thiệu

K-nearest neighbors (KNN) là một thuật toán học máy giám sát, phi tham số (non-parametric) và dựa trên bộ nhớ (instance-based hoặc memory-based learning). Khác với các mô hình học truyền thống, KNN không huấn luyện theo nghĩa thông thường mà ghi nhớ toàn bộ tập dữ liệu huấn luyện và sử dụng trực tiếp để dự đoán.

KNN được sử dụng hiệu quả trong các bài toán phân loại, hồi quy, và đặc biệt là trong hệ thống gợi ý. Trong ngữ cảnh gợi ý, KNN giúp xác định người dùng hoặc sản phẩm tương đồng dựa trên các đặc trưng như lịch sử đánh giá, từ đó đưa ra đề xuất phù hợp.

Scikit-learn là một thư viện học máy mã nguồn mở nổi tiếng, hỗ trợ xây dựng các mô hình học có giám sát, bao gồm cả giải thuật KNN. Thư viện đã tách giải thuật thành hai phần riêng biệt, bao gồm: KneighborsClassifier dùng cho phân loại và KNeighborsRegressor dùng cho hồi quy.

Thư viện còn hỗ trợ các bước tiền xử lý như chuẩn hóa dữ liệu, mã hóa đặc trưng và tích hợp pipeline huấn luyện và đánh giá. Tuy nhiên, scikit-learn không được thiết kế chuyên biệt cho hệ thống gợi ý nên thường yêu cầu xử lý định dạng ma trận người dùng – sản phẩm trước khi áp dụng.

#### Quá trình đưa ra dự đoán

KNN hoạt động dựa trên giả định: các đối tượng tương tự nhau sẽ có hành vi giống nhau. Khi cần đưa ra dự đoán cho người dùng hoặc sản phẩm mới, mô hình tham khảo hành vi của các láng giềng gần nhất – những người dùng hoặc sản phẩm tương đồng.

Quá trình dự đoán gồm ba bước:

1. Tìm K láng giềng gần nhất. Có thể là người dùng giống người cần dự đoán (user-based) hoặc sản phẩm giống sản phẩm cần dự đoán (item-based). Độ tương đồng có thể được đo bằng cosine similarity, Pearson correlation hoặc Euclidean distance. Trong hệ thống gợi ý, cosine và Pearson thường được ưu tiên do dữ liệu thưa.
2. Tổng hợp thông tin từ các láng giềng. Nếu là bài toán dự đoán điểm số, mô hình sẽ lấy trung bình có trọng số từ các đánh giá của láng giềng. Nếu là phân loại nhị phân (ví dụ: thích/không thích), dùng biểu quyết đa số.
3. Xuất kết quả gợi ý. Kết quả có thể là điểm số dự đoán, nhãn phân loại, hoặc danh sách top-N sản phẩm gợi ý.

#### Các dữ liệu cần thiết

Đặc trưng (Features): Mỗi mẫu dữ liệu được biểu diễn thành một véc-tơ số trong không gian nhiều chiều: x ∈ ℝᵈ. Với bài toán phân loại hoặc hồi quy, mỗi mẫu xᵢ đi kèm một nhãn hoặc giá trị đầu ra yᵢ.Trong hệ thống gợi ý, đặc trưng có thể là: véc-tơ người dùng, ví dụ: [5, 3, 0, 4, ...], thể hiện điểm đánh giá của người đó cho các sản phẩm; véc-tơ sản phẩm, dựa trên các thuộc tính hoặc đánh giá từ nhiều người.

Nhãn / Đầu ra (Labels / Targets): chỉ cần thiết nếu mô hình dùng cho bài toán có giám sát: Phân loại nhãn là lớp; Hồi quy nhãn là giá trị thực. Trong hệ thống gợi ý: nếu dự đoán điểm đánh giá hoặc đề xuất sản phẩm, nhãn có thể là điểm số hoặc lượt mua, số lượt xem,… nếu chỉ dùng để tìm người dùng tương tự (user-based), không nhất thiết cần nhãn.

Dữ liệu dạng ma trận: Scikit-learn hỗ trợ ma trận đầy đủ: dùng NumPy array, ma trận thưa: dùng scipy.sparse phù hợp với dữ liệu người dùng – sản phẩm (user-item rating matrix). Các lớp như KNeighborsClassifier, KNeighborsRegressor hay NearestNeighbors đều hỗ trợ cả hai định dạng trên. Dữ liệu thưa vẫn hoạt động tốt với các công thức tính khoảng cách như cosine similarity hoặc Minkowski.

Chuẩn hóa dữ liệu: Do KNN tính toán khoảng cách giữa các điểm, các đặc trưng nên được chuẩn hóa để tránh hiện tượng “đặc trưng lớn át đặc trưng nhỏ”.Phương pháp chuẩn hóa phổ biến:Min-max scaling: đưa dữ liệu về khoảng [0, 1], Z-score: đưa về trung bình 0 và độ lệch chuẩn 1.

Bố cục dữ liệu huấn luyện: dữ liệu thường tổ chức dưới dạng: ma trận X: kích thước (số mẫu × số đặc trưng), vector y: độ dài bằng số mẫu (nếu có nhãn). Trong bài toán gợi ý: mỗi hàng đại diện cho một người dùng hoặc sản phẩm, các cột là sản phẩm hoặc thuộc tính sản phẩm, nhãn có thể là điểm đánh giá, lượt xem hoặc thứ tự mua hàng.

#### Bộ tham số

Sklearn cung cấp một bộ tham số rất đa dạng và có độ tùy chỉnh cao. Các tham số quan trọng khi khởi tạo sklearn.neighbors.KNeighborsRegressor bao gồm:

Bảng 3.1.1 Bảng tham số quan trọng cho sklearn.neighbors.KNeighborsRegressor

| **Tham số** | **Kiểu dữ liệu** | **Mặc định** | **Ý nghĩa** |
| --- | --- | --- | --- |
| n\_neighbors | int | 5 | Số lượng láng giềng gần nhất |
| weight | string / callable | uniform | Trọng số của láng giềng |
| metric | string / callable | minkowski | Hàm đo khoảng cách giữa các điểm |
| p | int | 2 | |  | | --- | |  |  |  | | --- | | Bậc của khoảng cách Minkowski | |

Ngoài ra Sklearn còn cung cấp các tham số nâng cao giúp việc tùy chỉnh cho mô hình trở nên dễ dàng hơn, các tham số bao gồm:

Bảng 3.1.2 Bảng tham số nâng cao cho sklearn.neighbors.KNeighborsRegressor

| **Tham số** | **Kiểu dữ liệu** | **Mặc định** | **Ý nghĩa** |
| --- | --- | --- | --- |
| algorithm | str | auto | Cách tìm láng giềng: ‘auto’, ‘ball\_tree’, ‘kd\_tree’, ‘brute’ |
| leaf\_size | int | 30 | Kích thước lá trong cây |
| metric\_params | dict hoặc None | None | Tham số bổ sung cho metric |
| n\_jobs | int hoặc None | None | |  | | --- | |  |  |  | | --- | | Số lượng CPU dùng song song | |

### Decision Tree

#### Giới thiệu

Cây quyết định (Decision Tree) là một phương pháp học máy có giám sát được sử dụng cho cả bài toán phân loại và hồi quy. Mô hình này biểu diễn tập dữ liệu dưới dạng một cấu trúc cây phân cấp gồm các nút quyết định và nút lá, tương tự như một lưu đồ của tập hợp các luật if-then đơn giản để dự đoán giá trị của biến mục tiêu từ các đặc trưng đầu vào.

Mặc dù các mô hình dựa trên cây quyết định đơn lẻ thường không đạt độ chính xác cao nhất trong hệ thống gợi ý so với các phương pháp hiện đại (ví dụ mô hình phân rã ma trận, mạng nơ-ron), chúng vẫn đóng vai trò nền tảng quan trọng. Nhiều hệ thống gợi ý thực tế sử dụng tổ hợp nhiều cây để cải thiện hiệu quả dự đoán.

Scikit-learn là một thư viện mã nguồn mở nổi tiếng trong lĩnh vực học máy bằng Python, cung cấp tập hợp các thuật toán học máy kinh điển và các công cụ hỗ trợ tiền xử lý, đánh giá, chọn mô hình cũng như xây dựng pipeline tự động. Thư viện hỗ trợ Decision Tree cho cả bài toán phân loại (sklearn.tree.DecisionTreeClassifier) và hồi quy (sklearn.tree.DecisionTreeRegressor).

#### Quá trình đưa ra dự đoán

Mô hình cây quyết định được xây dựng thông qua quá trình phân tách đệ quy trên tập huấn luyện. Ban đầu, toàn bộ dữ liệu huấn luyện nằm ở nút gốc (root). Thuật toán sẽ chọn một đặc trưng *X* cùng một ngưỡng hoặc tiêu chí phân chia tối ưu tại nút gốc để chia dữ liệu thành các nhánh con, sao cho các nhánh con “thuần nhất” hơn về nhãn mục tiêu so với nút cha. Việc chọn đặc trưng và ngưỡng phân chia dựa trên việc tối ưu tiêu chí phân chia – thường là các chỉ số đo độ bất thuần (impurity) của nhánh. Hai chỉ số phổ biến cho cây quyết định phân loại là Gini và Entropy (thông tin).

#### Các dữ liệu cần thiết

Tương tự như việc giải quyết bài toán cho mô hình học có giám sát, dữ liệu phải được tổ chức dưới dạng bảng với các hàng đại diện cho từng mẫu (sample) và các cột là các đặc trưng (feature) cùng với nhãn (label):

Table 3.2.1.bảng dữ liệu cần thiết

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| user\_id | item\_id | feature\_1 | feature\_2 | … | rating |
| Huy | Cars | … |  |  | 5 |
| … |  |  |  |  |  |

Trong đó, các đặc trưng không thể được đưa trực tiếp vào mô hình dưới dạng chuỗi, mà phải trải qua bước mã hóa và chuẩn hóa.

#### Bộ tham số

Mô hình DecisionTreeRegressor thuộc thư viện scikit-learn, được sử dụng để dự đoán giá trị liên tục bằng cách xây dựng một cây quyết định. Dưới đây là một vài tham số quan trọng:

Table 3.2.2.bảng tham số của Decision Tree

| **Tham số** | **Kiểu dữ liệu** | **Mặc định** | **Ý nghĩa** |
| --- | --- | --- | --- |
| criterion | String | 'squared\_error' | Hàm mất mát dùng để đo chất lượng phân chia. |
| splitter | String | ‘best’ | Chiến lược chọn điểm  phân chia, bao gồm: ‘best’ và ‘random’ |
| max\_depth | Int | None | Độ sâu tối đa của cây.  Giới hạn nó giúp giảm overfitting. |
| min\_samples\_split | Int | Float | 2 | Số lượng mẫu tối thiểu để chia một node. Tăng giá trị này giúp làm trơn cây (regularization). |
| max\_feature | Int | None | Số lượng đặc trưng tối đa được dùng khi tìm phân chia tốt nhất. Có thể dùng để random hóa như trong Random Forest. |
| random\_state | Int | None | Đặt hạt giống (seed) cho việc chọn phân chia ngẫu nhiên, giúp tái tạo kết quả |

### Linear Regression

#### Giới thiệu

Linear Regression là một trong những thuật toán học máy cơ bản và phổ biến nhất trong các bài toán dự đoán. Mục tiêu của mô hình là tìm ra một hàm tuyến tính có dạng:

Trong đó, xi là các đặc trưng đầu vào, wi là các hệ số (trọng số) mà mô hình cần học, và b là hệ số chệch (bias). Mô hình cố gắng điều chỉnh các trọng số sao cho sai số giữa giá trị dự đoán và giá trị thực là nhỏ nhất, thông qua tối ưu hàm mất mát thường là Mean Squared Error (MSE).

Dù đơn giản, Linear Regression vẫn rất hiệu quả trong các bài toán có mối quan hệ tuyến tính tương đối rõ ràng giữa đầu vào và đầu ra. Trong hệ thống gợi ý, Linear Regression có thể được dùng để dự đoán điểm đánh giá mà người dùng có khả năng gán cho một sản phẩm/dịch vụ chưa từng tương tác.

Thư viện hỗ trợ: Scikit-learn cung cấp lớp LinearRegression, hỗ trợ cả huấn luyện, dự đoán và đánh giá mô hình hồi quy một cách thuận tiện.

#### Làm sao để mô hình đưa ra dự đoán

Linear Regression dự đoán mức độ tương tác giữa người dùng và sản phẩm bằng một hàm tuyến tính.

1. **Biểu diễn đặc trưng người dùng và sản phẩm được mã hóa bằng LabelEncode, sau đó chuẩn hóa bằng StandarScaler, tạo thành vector đặc trưng liên tục để đưa vào mô hình**.
2. **Mô hình hóa tương tác** Vector đầu vào được đưa vào mô hình hồi quy tuyến tính:

|  |
| --- |
|  |

Trong đó x là vector đặc trưng, w và b là các tham số được học.

1. **Dự đoán đầu ra g**iá trị thể hiện mức độ tương tác.

#### Các dữ liệu cần thiết

Trong hệ thống gợi ý phim, Linear Regression sử dụng dữ liệu bảng (tabular), mỗi dòng là một tương tác (user, movie) kèm theo:

Đặc trưng người dùng: tuổi, giới tính (mã hóa), nghề nghiệp,...

Đặc trưng phim: thể loại (mã hóa one-hot), số lượt đánh giá, điểm trung bình phim,...

Đặc trưng hành vi: điểm trung bình người dùng, số phim đã đánh giá,...

Nhãn (Label): rating (1–5).

Trước khi huấn luyện, cần tiền xử lý dữ liệu: chuyển đổi các biến phân loại thành định dạng số (one-hot encoding hoặc embedding), chuẩn hóa các biến số đo (đưa về trung bình 0, phương sai 1), xử lý dữ liệu thiếu (loại bỏ hoặc thay thế giá trị trung bình), và chia dữ liệu thành tập huấn luyện – kiểm thử.

#### Bộ tham số

Table .. bảng tham số của linear regression

| **Tham số** | **Kiểu dữ liệu** | **Mặc định** | **Ý nghĩa** |
| --- | --- | --- | --- |
| fit\_intercept | Bool | True | Có học thêm hệ số chệch (b) hay không. Nếu False, mô hình không có hệ số tự do. |
| copy\_X | Bool | True | Sao chép dữ liệu đầu vào X trước khi fit, tránh thay đổi dữ liệu gốc. |
| n\_jobs | Int | None | Số lượng CPU sử dụng khi tính toán (song song hóa). |
| positive | Bool | False | Nếu True, các hệ số (weights) bị ràng buộc không âm (>=0). |

### NCF

#### Giới thiệu

Neural Collaborative Filtering (NCF) là một mô hình học sâu trong hệ thống gợi ý, được đề xuất bởi Heetal. (2017). Mục tiêu của NCF là thay thế mô hình Matrix Factorization (MF) tuyến tính truyền thống bằng một mạng nơ-ron sâu để khai thác các tương tác phi tuyến giữa người dùng và sản phẩm.

Khác với các mô hình cộng tác tuyến tính như MF (dùng tích vô hướng), NCF tận dụng mạng neural để học các hàm tương tác phức tạp giữa vector embedding của người dùng và sản phẩm.

Nhiều thư viện mã nguồn mở hỗ trợ triển khai mô hình Neural Collaborative Filtering (NCF) một cách thuận tiện. TensorFlow Recommenders (TFRS) cung cấp các lớp chuyên biệt cho hệ thống gợi ý, cho phép xây dựng mô hình embedding và mạng nơ-ron theo kiến trúc NeuMF một cách trực quan. PyTorch tuy không có module riêng cho gợi ý, nhưng rất linh hoạt trong việc hiện thực NCF bằng cách kết hợp các lớp Embedding, Linear, và các hàm tối ưu như Adam. LibRecommender là một thư viện Python chuyên dụng cho hệ thống gợi ý, hỗ trợ mô hình NCF thông qua lớp NCF với giao diện đơn giản, dễ cấu hình. Ngoài ra, các thư viện như Spotlight (dựa trên PyTorch) và Microsoft Recommenders cũng cung cấp các triển khai tham khảo của NCF phù hợp cho thử nghiệm và đánh giá.

#### Quá trình đưa ra dự đoán

Mô hình Neural Collaborative Filtering (NCF) thực hiện dự đoán bằng cách học một hàm ánh xạ phi tuyến từ cặp (người dùng, sản phẩm) đến một giá trị đầu ra thể hiện mức độ tương tác giữa hai thực thể này.

Quá trình dự đoán diễn ra theo ba bước chính:

1. Biểu diễn đặc trưng: mỗi người dùng và sản phẩm được mã hóa dưới dạng chỉ số, sau đó ánh xạ thành các vector đặc trưng (embedding) thông qua một lớp nhúng. Đây là bước đưa dữ liệu rời rạc vào không gian liên tục có thể học được.
2. Mô hình hóa tương tác: Hai vector embedding được kết hợp lại để mô hình hóa quan hệ giữa người dùng và sản phẩm. Trong kiến trúc NeuMF, sự kết hợp này diễn ra qua hai nhánh song song: Nhánh GMF học tương tác tuyến tính bằng phép nhân phần tử tương ứng và nhánh MLP học tương tác phi tuyến bằng cách nối hai vector và đưa qua nhiều tầng mạng nơ-ron có hàm kích hoạt phi tuyến (thường là ReLU).
3. Đầu ra của hai nhánh được nối lại và đưa qua một lớp kết nối cuối cùng, thường là một tầng tuyến tính có hàm kích hoạt sigmoid, để trả về một giá trị dự đoán cuối cùng trong khoảng [0,1][0,1][0,1]. Giá trị này biểu diễn xác suất người dùng sẽ tương tác với sản phẩm (trong bài toán implicit feedback), hoặc điểm đánh giá dự kiến (trong bài toán explicit rating).

#### Các dữ liệu cần thiết

Để huấn luyện và vận hành mô hình NCF, cần chuẩn bị các loại dữ liệu sau:

Dữ liệu tương tác người dùng – sản phẩm: Đây là thành phần cốt lõi của mô hình NCF. Dữ liệu gồm các bản ghi thể hiện lịch sử tương tác giữa người dùng và item trong hệ thống gợi ý, ví dụ: người dùng đã xem, click, mua một sản phẩm nào đó. Mỗi bản ghi thường bao gồm user\_id, item\_id, thời điểm tương tác và loại hành động. Trong bài toán phản hồi ẩn (implicit feedback), dữ liệu này sẽ được chuyển thành các cặp (user, item) kèm nhãn 1 hoặc 0, tương ứng với “đã tương tác” hoặc “chưa tương tác” và dùng để huấn luyện mô hình phân loại.

Mặc dù NCF không sử dụng nội dung sản phẩm hay hồ sơ người dùng, mô hình vẫn cần biết tổng số lượng người dùng và sản phẩm để khởi tạo kích thước embedding tương ứng. Đây là thông tin cần thiết để xác định số chiều của các lớp nhúng đầu vào.

Tập tương tác âm (negative samples): Vì dữ liệu implicit chỉ ghi nhận các tương tác dương (đã xảy ra), cần tạo thêm các tương tác âm để mô hình học phân biệt. Quá trình này gọi là negative sampling, thường thực hiện bằng cách lấy ngẫu nhiên các sản phẩm mà người dùng chưa từng tương tác, giả định là không yêu thích. Tỷ lệ lấy mẫu âm có thể điều chỉnh (ví dụ 1:1, 1:4) tùy thuộc vào yêu cầu bài toán.

Ngoài dữ liệu huấn luyện, cần tách riêng một phần dữ liệu làm tập validation và test để đánh giá chất lượng dự đoán của mô hình sau huấn luyện. Việc chia tách này nên đảm bảo tính thời gian (time-aware), ví dụ chia theo mốc thời gian để phản ánh tính tuần tự của hành vi người dùng.

#### Bộ tham số

Recommenders cung cấp một bộ tham số rất đa dạng và có độ tùy chỉnh cao. Các tham số quan trọng khi khởi tạo recommenders.models.ncf.ncf\_singlenode bao gồm:

Bảng 3.4.1 Bảng tham số quan trọng cho recommenders.models.ncf.ncf\_singlenode

| **Tham số** | **Kiểu dữ liệu** | **Mặc định** | **Ý nghĩa** |
| --- | --- | --- | --- |
| n\_user | int | Bắt buộc | Số lượng người dùng |
| n\_items | int | Bắt buộc | Số lượng item sản phẩm |
| model\_type | str | ‘NeuMF’ | Kiến trúc dùng: ‘GMF’, ‘MLP’, ‘NeuMF’ |
| embedding\_dim | int | 8 | |  | | --- | |  |  |  | | --- | | Kích thước embedding vector | |
| layers | list of in | [64, 32, 16, 8] | Cấu trúc các tầng ẩn của MLP |
| learning\_rate | float | 0.001 | Tốc độ học |
| batch\_size | int | 256 | Kích thước mỗi mini-batch |
| epochs | int | 20 | Số lần lặp toàn bộ dữ liệu |
| loss | str | ‘binary\_crossentropy’ | Hàm mất mát, thường dùng implicit feedback |

Ngoài ra Recommenders còn cung cấp các tham số nâng cao giúp việc tùy chỉnh cho mô hình trở nên dễ dàng hơn, các tham số bao gồm:

Bảng 3.4.2 Bảng tham số nâng cao cho sklearn.neighbors.KNeighborsRegressor

| **Tham số** | **Kiểu dữ liệu** | **Mặc định** | **Ý nghĩa** |
| --- | --- | --- | --- |
| optimizer | str | ‘adam’ | Thuật toán tối ưu: ‘adam’, ‘sgd’,… |
| verbose | int | 1 | Mức độ hiển thị log trong quá trình huấn luyện |
| seed | int | 42 | Thiết lập hạt giống để tái lập kết quả |
| dropout\_rate | float | 0.0 | |  | | --- | | Tỷ lệ dropout trong các tầng ẩn |  |  | | --- | |  | |
| use\_batch\_norm | bool | False | Dùng batch normalization sau mỗi tầng ẩn |
| use\_bias | bool | True | Có dùng bias trong các tầng mạng hay không |
| train\_only | bool | False | Chỉ huấn luyện không test |
| save\_model | bool | False | Có lưu mô hình sau khi huấn luyện không |
| model\_dir | str | ‘./checkpoints/’ | Đường dẫn lưu mô hình nếu save\_model = true |

### LightGBM

#### Giới thiệu

LightGBM là thư viện mã nguồn mở do Microsoft phát triển, áp dụng thuật toán Gradient Boosting Decision Tree (GBDT) để xử lý hiệu quả các bài toán học máy trên dữ liệu lớn. Đặc trưng của LightGBM là tập trung vào  tối ưu tốc độ huấn luyện và sử dụng bộ nhớ hiệu quả, sử dụng các kỹ thuật như chia thang giá trị, sinh cây theo lá, GOSS(Gradient-based One-Side Sampling) và EFB (Exclusive Feature Bundling) để giảm số lượng mẫu và đặc trưng cần xét. Thực nghiệm trên nhiều cho thấy LightGBM nhanh hơn gấp nhiều lần GBDT truyền thống mà vẫn duy trì độ chính xác. Trong hệ gợi ý, LightGBM được ưa chuộng nhờ khả năng xử lý dữ liệu lớn và hỗ trợ nhiều mục tiêu học như phân loại, hồi quy, và xếp hạng.

Thư viện hỗ trợ: LightGBM thư viện chính được mô tả, dùng cho huấn luyện mô hình học máy, scikit-learn cung cấp các lớp tương thích với thư viện này để dễ dàng tích hợp vào pipeline như LGBMClassifier, LGBMRegressor, LGBMRanker... và hỗ trợ tăng tốc với GPU (qua thư viện CUDA và OpenCL).

#### Làm sao để mô hình đưa ra dự đoán

Đối với bài toán phân loại, LightGBM thường học trực tiếp các giá trị log-odds (để phân loại nhị phân) hoặc giá trị điểm (scores) cho từng lớp; sau đó tính xác suất hoặc quyết định nhãn. Với hồi quy hoặc hệ thống gợi ý dự đoán điểm/điểm xếp hạng, mô hình trả về là giá trị dự đoán cuối cùng (có thể được quy về khoảng [0,1] nếu cần).

Nói chung, “điểm gợi ý” được tính bằng tổng các giá trị ở lá cây thu được từ các cây yếu (cộng với bias). Trong bài toán xếp hạng, LightGBM hỗ trợ các hàm mất mát dạng LambdaRank để huấn luyện trực tiếp trên mục tiêu thứ tự kết quả; mô hình đầu ra là một score (điểm sắp xếp) có thể dùng để xếp hạng các mục.

#### Các dữ liệu cần thiết

Trong hệ thống gợi ý, dữ liệu đầu vào LightGBM thường được xây dựng dưới dạng các bộ đặc trưng(features) của cặp (người dùng, đối tượng) kèm nhãn (rating, like hoặc score). Để áp dụng LightGBM cho hệ thống gợi ý, dữ liệu thường được tập hợp thành ma trận đặc trưng huấn luyệntrong đó mỗi dòng là một điểm (user, item) kèm giá trị mục tiêu (rating, 0/1). Ta chuẩn bị các biến số học (ví dụ embeddings, số lượt) và phân loại (ví dụ mã user, mã item, các thuộc tính phân loại), mã hóa phù hợp và để LightGBM tự xử lý phân chia tối ưu cho mô hình.

#### Bộ tham số

Table ..Bảng tham số của LightGBM

| **Tham số** | **Kiểu dữ liệu** | **Mặc định** | **Ý nghĩa** |
| --- | --- | --- | --- |
| Boosting-type | String | gbdt | Loại boosting, thường là Gradient Boosting Decision Tree |
| num\_leaves | Int | 31 | Số lượng lá tối đa của mỗi cây quyết định |
| max\_depth | Int | -1 | Độ sâu tối đa của cây, -1 nghĩa là không giới hạn |
| learning\_rate | Float | 0.1 | Tốc độ học, càng nhỏ thì học càng chậm nhưng có thể chính xác hơn |
| n\_estimators | Int | 100 | Số lượng cây quyết định (số vòng lặp boosting) |
| subsample | Float | 1.0 | Tỷ lệ mẫu được sử dụng cho mỗi cây (bagging) |
| colsample\_bytree | Float | 1.0 | Tỷ lệ cột (đặc trưng) được sử dụng cho mỗi cây |
| reg\_alpha | Float | 0.0 | Tham số regularization L1 (giúp giảm overfitting) |
| reg\_lambda | Float | 0.0 | Tham số regularization L2 (giúp giảm overfitting) |
| random\_state | Int hoặc none | None | Hạt giống ngẫu nhiên, giúp tái lập kết quả nếu đặt giá trị cố định |
| objective | String | Regression | Hàm mục tiêu, ở đây là hồi quy (dự đoán giá trị liên tục) |

### SVD

#### Giới thiệu

Trong đại số tuyến tính, Singular Value Decomposition (SVD) là một phép phân rã ma trận cổ điển: với một ma trận đầy đủ , luôn tồn tại , trong đó *U* và *V* là ma trận trực giao chứa các vector riêng (thành phần tương ứng gọi là vector kỳ dị), còn *Σ* là ma trận đường chéo chứa các giá trị kỳ dị không âm sắp xếp giảm dần. Thuật toán được gọi là SVD trong gợi ý thực chất là một mô hình học máy dựa trên ý tưởng phân rã ma trận, chứ không thực hiện phép SVD giải tích. Thông thường, SVD (CF) ám chỉ phương pháp Matrix Factorization do Simon Funk đề xuất trong cuộc thi Netflix Prize, sau này được Yehuda Koren và cộng sự phát triển thành thuật toán Regularized SVD cho bài toán dự đoán.

scikit-surprise (Surprise) là thư viện Python dành cho xây dựng và phân tích bài toán Collaborative Filtering trên dữ liệu rating tường minh (explicit ratings). Thư viện sẽ cung cấp lớp SVD đã cài đặt sẵn dành cho mô hình SVD với nhiều tùy chọn về mặt tham số cho người dùng.

#### Làm sao để mô hình đưa ra dự đoán

Mô hình đưa ra dự đoán tương tự theo phương pháp phân rã ma trận (Matrix Factorization). Cho phép xử lý bài toán hồi quy thông qua biểu diễn ma trận *Y* ∈ dưới dạng tích của hai ma trận hạng thấp, đại diện cho các đặc trưng tiềm ẩn (latent factors) thông qua phân rã ma trận.

Ngoài ra, để đưa ra dự đoán cho các cho phần dữ liệu còn thiếu, mô hình tập trung vào giảm thiểu sai số bình phương của hàm sau:

|  |  |
| --- | --- |
|  |  |

Ngoài các tham số đã nêu trên thì và là độ lệch của sản phẩm và người dùng. Còn và là các vector đặc trưng tiềm ẩn của sản phẩm và người dùng.

Việc giảm thiểu sai số sẽ được dựa trên thuật toán SGD, cập nhật từng bước nhỏ, dựa trên từng đánh giá một theo công thức sau đây:

|  |  |
| --- | --- |
|  |  |

Với . Phương pháp này sẽ lặp lại việc cập nhật 4 tham số , , và theo số lần epoch (n\_epochs) đã định trước (ví dụ: 20-50 lần), mỗi lần qua tất cả các đánh giá trong tập huấn luyện.

#### Các dữ liệu cần thiết

Là một mô hình vận hành theo phương pháp lọc cộng tác, một nhánh của hệ thống gợi ý. SVD cần bộ dữ liệu có các thuộc tính như user\_id, item\_id, rating (hoặc score).

Các dữ liệu này sẽ được phân rã thành ma trận đặc trưng của người dùng và sản phẩm. Từ đó có thể dễ dàng áp dụng các thuật toán cần thiết cho quá trình huấn luyện và dự đoán.

#### Bộ tham số

Surprise cung cấp một bộ tham số rất đa dạng và có độ tùy chỉnh cao. Các tham số quan trọng khi khởi tạo surprise.SVD bao gồm:

Table 3.6.1. bảng tham số quan trọng của SVD

| **Tham số** | **Kiểu dữ liệu** | **Mặc định** | **Ý nghĩa** |
| --- | --- | --- | --- |
| n\_factors | int | 100 | Số chiều của các vector đặc trưng tiềm ẩn |
| n\_epochs | int | 20 | Số lần lặp huấn luyện SGD |
| lr\_all | float | 0.005 | Learning rate cho tất cả các tham số |
| reg\_all | float | 0.02 | |  | | --- | |  |  |  | | --- | | Hệ số điều chuẩn (regularization) | |

Ngoài ra Surprise còn cung cấp các tham số nâng cao giúp việc tùy chỉnh cho mô hình trở nên dễ dàng hơn, các tham số bao gồm:

Table 3.6.2.bảng tham số nâng cao của SVD

| **Tham số** | **Kiểu dữ liệu** | **Mặc định** | **Ý nghĩa** |
| --- | --- | --- | --- |
| biased | bool | True | Có sử dụng các bias *b­u*, *bi* hay không |
| init\_means | float | 0 | Giá trị trung bình ban đầu cho phân phối khởi tạo các vector latent |
| random\_state | int, RandomState | None | Seed ngẫu nhiên hoặc trạng thái để tái tạo kết quả |
| verbose | bool | False | |  | | --- | |  |  |  | | --- | | In ra các thông tin trong quá trình huấn luyện hay không | |

## Cấu hình môi trường

Nhóm tác giả đã sử dụng công cụ Anaconda để cài đặt môi trường.

Phiên bản Python đang sử dụng là 3.9. Sau đó tạo kernel phù hợp.

Các thư viện có liên quan được nhóm tác giả liệt kê trong tập tin requirements.txt.

## Huấn luyện và kết quả thực nghiệm

### Quá trình xử lí dữ liệu

Trích đặc trưng và chuẩn hóa

Trích đặc trưng bằng cách mã hóa dữ liệu từ chuỗi sang số

Sử dụng MinMaxScaler để chuẩn hóa

### Quá trình huấn luyện

Theo dõi thời gian huấn luyện của từng mô hình, thay đổi bộ tham số của mô hình giúp tùy biến và đưa ra kết quả tốt hơn.Quá trình dự đoán

Quá trình dự đoán cũng được theo dõi về mặt thời gian và từ các kết quả dự đoán đưa ra được bộ chỉ số đánh giá trên từng mô hình.

Quá trình huấn luyện và dự đoán được lặp lại 10 lần, sau đó lấy giá trị trung bình của các chỉ số đánh giá, từ đó tăng độ chính xác và minh bạch trong quá trình đưa ra kết luận.

## Đánh giá mô hình

### Các chỉ số đánh giá

#### MAE

*MAE* đo lường sai số tuyệt đối trung bình giữa các giá trị dự đoán và thực tế, được xác định bằng công thức:

|  |  |
| --- | --- |
|  |  |

*MAE* không bình phương sai số nên phản ánh chính xác hơn về độ lệch trung bình không thiên vị, đồng thời ít bị ảnh hưởng bởi các giá trị ngoại lai.

#### MSE

*MSE* (Mean Squared Error) là một chỉ số đo lường mức độ chênh lệch trung bình bình phương giữa giá trị dự đoán và giá trị thực tế.

|  |  |
| --- | --- |
|  |  |

Do sử dụng bình phương của sai số, MSE phóng đại sai số lớn, giúp mô hình ưu tiên giảm các dự đoán sai nghiêm trọng.

#### RMSE

Chỉ số *RMSE* phản ánh độ lệch trung bình bình phương giữa giá trị dự đoán và giá trị thực tế. Chỉ số này được tính theo công thức:

|  |  |
| --- | --- |
|  |  |

Trong đó, là giá trị thực tế, là giá trị dự đoán, và *n* là tổng số điểm đánh giá trong tập kiểm thử. RMSE có đơn vị tương đương với đơn vị gốc của dữ liệu và nhạy cảm với sai số lớn, do đó thường được sử dụng để phát hiện các dự đoán lệch nhiều.

### Kết quả đánh giá

Thực hiện đánh giá sáu mô hình sau trên cùng tập dữ liệu MovieLens 100K: KNN, Decision Tree, Linear Regression, NCF, LightGBM, SVD. Dưới đây là bảng chi tiết kết quả đánh giá sau khi chạy thực nghiệm 10 lần:

Table 6.2.1.bảng đánh giá mô hình

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Mô hình** | **Phương pháp gợi ý** | **Thời gian huấn luyện (s)** | **Các chỉ số đánh giá** | | |
| **MAE** | **MSE** | **RMSE** |
| KNN  (k=20) | Collaborative Filtering | 0.336 | 0.873 | 1.173 | 1.083 |
| Linear  Regression | Content-based filtering | 0.005 | 0.912 | 1.224 | 1.106 |
| Decision Tree | Content-based filtering | 1.340 | 0.846 | 1.112 | 1.055 |
| lightGBM | Content-based filtering | 0.169 | 0.828 | 1.057 | 1.028 |
| SVD | Collaborative Filtering | 18.18 | 0.739 | 0.878 | 0.937 |
| NCF | Collaborative filtering | 78.536 | 3.000 | 10.203 | 3.194 |

# PHẦN KẾT LUẬN

Chúng tôi đã thử nghiệm huấn luyện các mô hình trên tập dữ liệu movielens 100k và đánh giá hiệu quả của các mô hình gợi ý khác nhau, bao gồm các mô hình đơn giản như KNN, Linear Regression, Decision Tree. Bên cạnh đó còn có có các hình hiện đại như LightGBM, SVD và NCF.

Quá trình thực hiện báo cáo bao gồm các bước: thu thập và xử lí dữ liệu đầu vào, lựa chọn và triển khai mô hình học máy, huấn luyện mô hình trên tập dữ liệu, đánh giá mô hình và xem xét hiệu quả.

Qua qua trình đánh giá, có thể thấy LightGBM và SVD là hai mô hình nổi bật về hiệu quả dự đoán qua các chỉ số MAE, MSE và RMSE. Cho thấy việc lựa chọn mô hình phù hợp với hệ thống rất quan trọng cần dựa vào các đặc trưng của dữ liệu, mục tiêu hệ thống, tài nguyên và khả năng mở rộng.

# TÀI LIỆU THAM KHẢO

[1]<https://medium.com/@jwu2/building-a-collaborative-filtering-model-with-decision-trees-56256b95cb03>

[2]<https://www.ibm.com/think/topics/knn>

[3]<https://www.geeksforgeeks.org/machine-learning/k-nearest-neighbours/>

[4]<https://www.tensorflow.org/recommenders/examples/movielens/movielens_implicit>

[5]<https://arxiv.org/abs/1708.05031>