|  |  |
| --- | --- |
| **bỘ GIÁO DỤC VÀ ĐÀO TẠO** | **BỘ NÔNG NGHIỆP VÀ PTNT** |

TRƯỜNG ĐẠI HỌC THỦY LỢI



nGUYỄN dUY TÂN

Dự đoán xếp hạng mì ramen

bằng mô hình học máy

ĐỒ ÁN TỐT NGHIỆP

HÀ NỘI, NĂM 2024

|  |  |
| --- | --- |
| **bỘ GIÁO DỤC VÀ ĐÀO TẠO** | **BỘ NÔNG NGHIỆP VÀ PTNT** |

TRƯỜNG ĐẠI HỌC THỦY LỢI

NGUYỄN DUY TÂN

Dự đoán xếp hạng mì ramen

bằng mô hình học máy

|  |  |
| --- | --- |
| Ngành: | Hệ thống thông tin |
| Mã số: | 7480104 |

|  |  |
| --- | --- |
| NGƯỜI HƯỚNG DẪN | **TS. Tạ Quang Chiểu** |
|  |  |

HÀ NỘI, NĂM 2024

**GÁY BÌA ĐỒ ÁN TỐT NGHIỆP, KHÓA LUẬN TỐT NGHIỆP**

**NGUYỄN DUY TÂN ĐỒ ÁN TỐT NGHIỆP HÀ NỘI, NĂM 2024**

|  |  |
| --- | --- |
|  | CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM  **Độc lập - Tự do - Hạnh phúc**  ----------★----------  **NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP** |

|  |  |
| --- | --- |
| Họ tên sinh viên: **Nguyễn Duy Tân** | Hệ đào tạo : **Đại học chính quy** |
| Lớp: **61HT** | Ngành: **Hệ thống thông tin** |
| Khoa: **Công nghệ thông tin** |  |

1. TÊN ĐỀ TÀI:

DỰ ĐOÁN XẾP HẠNG MÌ RAMEN BẰNG MÔ HÌNH HỌC MÁY.

1. CÁC TÀI LIỆU CƠ BẢN:
2. [Machine Learning cơ bản (machinelearningcoban.com)](https://machinelearningcoban.com/2018/06/22/deeplearning/)
3. [Deep Learning cơ bản (nttuan8.com)](https://nttuan8.com/bai-12-image-segmentation-voi-u-net/)
4. NỘI DUNG CÁC PHẦN THUYẾT MINH VÀ TÍNH TOÁN:

* Chương 1: Giới thiệu (10%)
  + Lý do chọn đề tài
  + Mục tiêu đề tài
  + Đối tượng và phạm vi nghiên cứu
* Chương 2: Các kỹ thuật dùng trong bài toán (25%)
  + Học máy, học sâu
  + Hồi quy Logistic (Logistic Regression)
  + Cây quyết định (Decision Tree Classifier)
  + Rừng ngẫu nhiên (RandomForestClassifier)
* Chương 3: Ứng dụng phương pháp và xây dựng mô hình (60%)
  + Phân tích dữ liệu
  + Công cụ và các thư viện
  + Xây dựng mô hình
  + Kết quả
* Kết Luận (5%)
* Kết luận
* Định hướng phát triển
* GIÁO VIÊN HƯỚNG DẪN TỪNG PHẦN

Giáo viên hướng dẫn toàn bộ quá trình thực hiện đồ án: **TS. Tạ Quang Chiểu**

1. NGÀY GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Ngày .... tháng …. năm 202....

|  |  |  |
| --- | --- | --- |
| **Trưởng Bộ môn**  (Ký và ghi rõ Họ tên) |  | **Giáo viên hướng dẫn chính**  (Ký và ghi rõ Họ tên) |

Nhiệm vụ Đồ án tốt nghiệp đã được Hội đồng thi tốt nghiệp của Khoa thông qua.

|  |
| --- |
| Ngày …. tháng …. năm 202…  **Chủ tịch Hội đồng**  (Ký và ghi rõ Họ tên) |

Sinh viên đã hoàn thành và nộp bản Đồ án tốt nghiệp cho Hội đồng thi.

|  |
| --- |
| Ngày .... tháng …. năm 202….  **Sinh viên làm Đồ án tốt nghiệp**  (Ký và ghi rõ Họ tên) |

|  |  |
| --- | --- |
| logo | TRƯỜNG ĐẠI HỌC THUỶ LỢI  **KHOA CÔNG NGHỆ THÔNG TIN**  BẢN TÓM TẮT ĐỀ CƯƠNG ĐỒ ÁN TỐT NGHIỆP |

**Tên đề tài:** **Dự đoán xếp hạng mì ramen bằng mô hình học máy**

Sinh viên thực hiện: Nguyễn Duy Tân

Lớp: 61HT

Email: Nguyentan7799@gmail.com

Điện thoại: 0394422547

Giáo viên hướng dẫn: TS. Tạ Quang Chiểu

**TÓM TẮT ĐỀ TÀI**

Mì ramen hay được biết tới với cái tên khác là mì ăn liền, đã trở thành một phần không thể thiếu trong ẩm thực hàng ngày của nhiều người trên khắp thế giới. Tuy nhiên, mì ramen hay loại mì ăn liền được yêu thích vì sự thuận tiện, giá cả phải chăng và hương vị đa dạng. Để lựa chọn giữa các thương hiệu mì khác nhau thường là việc khó khăn cho người tiêu dùng.

Từ đây đặt ra một câu hỏi quan trọng: Điều gì tạo nên sự khác biệt của một loại mì ramen so với các loại mì khác? Liệu có thể xác định các yếu tố cụ thể nào làm cho một thương hiệu mì ramen nổi bật hơn và có hương vị tốt hơn so với các thương hiệu khác không?

Trong đồ án này, em tiến hành nghiên cứu các kỹ thuật học máy để dự đoán xem một loại mì ramen đó sẽ được xem là "tốt" hay "không tốt" dựa trên các yếu tố như loại sản phẩm, kiểu mì, thương hiệu và quốc gia sản xuất. Vì vậy em thực hiện ĐATN về đề tài: “**Dự đoán xếp hạng mì ramen bằng mô hình học máy**”. Từ đó cung cấp thông tin hữu ích cho người tiêu dùng, giúp họ có thể lựa chọn mì ăn liền tốt nhất và phù hợp nhất với sở thích và nhu cầu của mình. Đồng thời giúp các nhà sản xuất chủ động cách cải thiện sản phẩm ramen để đạt được điểm đánh giá cao hơn của người sử dụng. Trong đồ án này, em sử dụng ngôn ngữ lập trình Python kết hợp các thư viện hỗ trợ để phân tích, trực quan hóa và cài đặt một số kỹ thuật học máy giải quyết bài toán.

**CÁC MỤC TIÊU CHÍNH**

* Mục tiêu 1: Tìm hiểu bài toán dự đoán xếp hạng mì ramen.
* Mục tiêu 2: Tìm hiểu, nghiên cứu một số kỹ thuật học máy.
* Mục tiêu 3: Áp dụng các kỹ thuật học máy dự đoán xếp hạng mì ramen.
* Mục tiêu 4: Đánh giá mô hình, xem xét tính hiệu quả của thuật toán.
* Mục tiêu 5: Kết luận.

**KẾT QUẢ DỰ KIẾN**

* Hoàn thành các mục tiêu đề ra.
* Hiểu, phân tích bộ dữ liệu bài toán.
* Tìm hiểu các thuật toán học máy và ứng dụng xây dựng được kỹ thuật học máy cho bài toán dự đoán xếp hạng mì ramen.
* Đánh giá kết quả dự đoán.
* Báo cáo tổng kết ĐATN

LỜI CAM ĐOAN

Tác giả xin cam đoan đây là Đồ án tốt nghiệp của bản thân. Tất cả các kết quả trong Đồ án tốt nghiệp này là trung thực, và không sao chép từ bất kỳ một nguồn nào và dưới bất kỳ hình thức nào. Việc tham khảo các nguồn tài liệu đã được thực hiện trích dẫn và ghi nguồn tài liệu tham khảo đúng quy định.

|  |  |
| --- | --- |
|  | **Tác giả ĐATN**    **Nguyễn Duy Tân** |

LỜI CÁM ƠN

Trước hết, em muốn bày tỏ lòng biết ơn chân thành đến đội ngũ giảng viên tại Khoa Công nghệ thông tin của trường Đại học Thủy lợi. Sự chỉ dẫn và hỗ trợ nhiệt tình từ phía thầy cô trong suốt quãng thời gian học đã giúp em tích lũy đủ kiến thức và kỹ năng cần thiết để hoàn thành khóa học cũng như đồ án tốt nghiệp của mình.

Em muốn gửi lời biết ơn chân thành đến tất cả những người đã đồng hành và hỗ trợ em trong suốt quá trình học tập. Đặc biệt, em muốn gửi lời cảm ơn sâu sắc đến Thầy TS. Tạ Quang Chiểu với sự tận tâm và hướng dẫn sát sao đã giúp em vượt qua những khó khăn và hoàn thành đồ án tốt nghiệp một cách thành công. Em rất trân trọng và biết ơn sự hỗ trợ nhiệt tình từ Thầy.

Tiếp theo, em muốn gửi lời biết ơn sâu sắc đến tất cả bạn bè, anh chị em đã luôn đồng hành và hỗ trợ em trong suốt quãng thời gian học tập và nghiên cứu. Sự chia sẻ kiến thức và kinh nghiệm, cùng những ý kiến đóng góp quý báu từ mọi người, thực sự là nguồn động viên không thể thiếu giúp em vượt qua mọi thách thức.

Mặc dù em đã cố gắng tập trung để hoàn thành đồ án tốt nghiệp theo khả năng của mình, nhưng vẫn tồn tại nhiều thiếu sót. Em chân thành nhận được sự đồng cảm và những đóng góp xây dựng từ phía các thầy cô và bạn bè, giúp em có cơ hội hiểu biết sâu hơn và phát triển từ những trải nghiệm học tập này.

Em xin chân thành cảm ơn!

MỤC LỤC

[DANH MỤC CÁC HÌNH ẢNH ix](#_Toc171523110)

[DANH MỤC BẢNG BIỂU x](#_Toc171523111)

[DANH MỤC CÁC TỪ VIẾT TẮT VÀ GIẢI THÍCH CÁC THUẬT NGỮ xi](#_Toc171523112)

[CHƯƠNG 1 : GIỚI THIỆU BÀI TOÁN 1](#_Toc171523113)

[1.1 Lý do chọn đề tài 1](#_Toc171523114)

[1.2 Mục tiêu nghiên cứu 1](#_Toc171523115)

[1.3 Đối tượng và phạm vi nghiên cứu 1](#_Toc171523116)

[CHƯƠNG 2: CÁC KỸ THUẬT DÙNG TRONG BÀI TOÁN 3](#_Toc171523117)

[2.1 Cơ sở lý thuyết chung của học máy 3](#_Toc171523118)

[2.1.1 Học có giám sát (Supervised Learning) 4](#_Toc171523119)

[2.1.2 Học không giám sát (Unsupervised Learning) 5](#_Toc171523120)

[2.1.3 Học sâu (Deep learning) 6](#_Toc171523121)

[2.2 Kỹ thuật dùng trong bài toán 7](#_Toc171523122)

[2.2.1 Hồi quy Logistic (Logistic Regression) 7](#_Toc171523123)

[2.2.2 Cây quyết định (Decision Tree Classifier) 9](#_Toc171523124)

[2.2.3 Rừng ngẫu nhiên (Random Forest Classifier) 11](#_Toc171523125)

[2.3 Đánh giá mô hình 13](#_Toc171523126)

[2.3.1. Confusion Matrix 13](#_Toc171523127)

[2.3.2. Accuracy 14](#_Toc171523128)

[2.3.3. Precision 15](#_Toc171523129)

[2.3.4. Recall 15](#_Toc171523130)

[2.3.5. F1-score 15](#_Toc171523131)

[CHƯƠNG 3: ỨNG DỤNG PHƯƠNG PHÁP VÀ XÂY DỰNG MÔ HÌNH 17](#_Toc171523132)

[3.1 Mô tả bài toán 17](#_Toc171523133)

[3.2 Phân tích dữ liệu 17](#_Toc171523134)

[3.2.1 Tiền xử lý dữ liệu 18](#_Toc171523135)

[3.2.2 Phân tích dữ liệu 27](#_Toc171523136)

[3.3 Công cụ sử dụng 30](#_Toc171523137)

[3.4 Xây dựng mô hình 32](#_Toc171523138)

[3.4.1 Huấn luyện mô hình 32](#_Toc171523139)

[3.4.2 Đánh giá chất lượng mô hình 37](#_Toc171523140)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 42](#_Toc171523141)

[TÀI LIỆU THAM KHẢO 43](#_Toc171523142)

DANH MỤC CÁC HÌNH ẢNH

[Hình 2.1 Sơ đồ lý thuyết học máy 4](#_Toc171499790)

[Hình 2.2 Logistic Regression sử dụng hàm phi tuyến để xác định xác suất của hai lớp 0 và 1 7](#_Toc171499791)

[Hình 2.3 Sơ đồ hoạt động hồi quy logistic 8](#_Toc171499792)

[Hình 2.4 Cấu trúc của thuật toán Cây quyết định 10](#_Toc171499793)

[Hình 2.5 Minh họa về thuật toán Cây quyết định 10](#_Toc171499794)

[Hình 2.6 Minh họa về cách hoạt động của thuật toán Rừng ngẫu nhiên 12](#_Toc171499795)

[Hình 2.7 Một ma trận nhầm lẫn (confusion matrix) 13](#_Toc171499796)

[Hình 3.1 Dữ liệu ban đầu 17](#_Toc171499797)

[Hình 3.2 Kiểm tra giá trị null trong từng cột 18](#_Toc171499798)

[Hình 3.3 Dữ liệu mới sau khi loại bỏ null và cột Review # 18](#_Toc171499799)

[Hình 3.4 Các giá trị không xác định trong cột Stars 19](#_Toc171499800)

[Hình 3.5 Dữ liệu sau khi tạo xong cột IsSpicy 21](#_Toc171499801)

[Hình 3.6 Timeline sự phát triển chính của các mô hình ngôn ngữ 22](#_Toc171499802)

[Hình 3.7 Dữ liệu sau khi trích xuất token từ thuộc tính Variety 24](#_Toc171499803)

[Hình 3.8 Các biến giả được tạo ra từ cột Brand 25](#_Toc171499804)

[Hình 3.9 Các biến giả được tạo ra từ cột Style 26](#_Toc171499805)

[Hình 3.10 Các biến giả được tạo ra từ cột Country 27](#_Toc171499806)

[Hình 3.11 Dữ liệu cuối cùng thu được 27](#_Toc171499807)

[Hình 3.12 Biểu đồ boxplot giữa 2 cột Styles và Stars 28](#_Toc171499808)

[Hình 3.13 Biểu đồ cột về số lượng mì của các quốc gia 29](#_Toc171499809)

[Hình 3.14 Biểu đồ về rating (Stars) của các quốc gia 30](#_Toc171499810)

[Hình 3.15 Biểu đồ so sánh về rating và số lượng của mì cay và không cay 30](#_Toc171499811)

[Hình 3.16 Bảng các thuộc tính quan trọng với mô hình Hồi quy logistic 37](#_Toc171499812)

[Hình 3.17 Bảng các thuộc tính quan trọng với mô hình Cây quyết định 38](#_Toc171499813)

[Hình 3.18 Bảng các thuộc tính quan trọng với mô hình Rừng ngẫu nhiên 38](#_Toc171499814)

[Hình 3.19 Độ chính xác của mô hình Hồi quy Logistic 39](#_Toc171499815)

[Hình 3.20 Độ chính xác của mô hình Cây quyết định 39](#_Toc171499816)

[Hình 3.21 Độ chính xác của mô hình Rừng ngẫu nhiên 40](#_Toc171499817)

[Hình 3.22 Độ chính xác của mô hình Hồi quy Logistic với Grid Search 41](#_Toc171499818)

[Hình 3.23 Độ chính xác của mô hình Cây quyết định với Grid Search 41](#_Toc171499819)

[Hình 3.24 Độ chính xác của mô hình Rừng ngẫu nhiên với Grid Search 42](#_Toc171499820)

DANH MỤC BẢNG BIỂU

[Bảng 3.1 Một số token có tần xuất xuất hiện lớn 21](#_Toc171521972)

[Bảng 3.2 Bảng tổng hợp widget của Streamlit 31](#_Toc171521973)

[Bảng 3.3So sánh độ chính xác của các thuật toán qua các phương thức đánh giá 39](#_Toc171521974)

[Bảng 3.4 So sánh độ chính xác của các thuật toán sau khi sử dụng Grid Search 41](#_Toc171521975)

DANH MỤC CÁC TỪ VIẾT TẮT VÀ GIẢI THÍCH CÁC THUẬT NGỮ

|  |  |
| --- | --- |
| **Từ viết tắt** | **Viết đầy đủ** |
| **ĐATN** | Đồ án tốt nghiệp |
| **AI** | Artificial Intelligence |
| **FP-Growth** | Frequent Pattern Growth |

# : GIỚI THIỆU BÀI TOÁN

## Lý do chọn đề tài

Mì ramen hay được biết tới với cái tên khác là mì ăn liền, đã trở thành một phần không thể thiếu trong ẩm thực hàng ngày của nhiều người trên khắp thế giới. Tuy nhiên, mì ramen hay loại mì ăn liền được yêu thích vì sự thuận tiện, giá cả phải chăng và hương vị đa dạng. Để lựa chọn giữa các thương hiệu mì khác nhau thường là việc khó khăn cho người tiêu dùng.

Từ đây đặt ra một câu hỏi quan trọng: Điều gì tạo nên sự khác biệt của một loại mì ramen so với các loại mì khác? Liệu có thể xác định các yếu tố cụ thể nào làm cho một thương hiệu mì ramen nổi bật hơn và có hương vị tốt hơn so với các thương hiệu khác không?

Trong đồ án này, em tiến hành nghiên cứu các kỹ thuật học máy để dự đoán xem một loại mì ramen đó sẽ được xem là "tốt" hay "không tốt" dựa trên các yếu tố như loại sản phẩm, kiểu mì, thương hiệu và quốc gia sản xuất. Vì vậy em thực hiện ĐATN về đề tài: “Dự đoán xếp hạng mì ramen bằng mô hình học máy”.

## Mục tiêu nghiên cứu

*Mục tiêu tổng quát*: Ứng dụng các mô hình học máy để dự đoán xếp hạng mi ramen.

*Mục tiêu cụ thể*: Để thực hiện mục tiêu tổng quát thì cần:

- Nghiên cứu bài toán phân loại mì ramen

- Nghiên cứu lý thuyết học máy, các mô hình (Hồi quy logistic, Cây quyết định, Rừng ngẫu nhiên) học máy cho bài toán phân lớp.

- Ứng dụng các mô hình nghiên cứu ở trên để thực hiện dự đoán bài toán mì ramen

## Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu: Là xếp hạng của các loại mì ramen, tìm ra loại mì có stars(rating) cao

Phạm vi nghiên cứu: Các loại mì ramen của các nhà sản xuất trên khắp thế giới bao gồm tên, nhãn hàng sản xuất, xuất xứ.

CHƯƠNG 2: CÁC KỸ THUẬT DÙNG TRONG BÀI TOÁN

## 2.1 Cơ sở lý thuyết chung của học máy

Trí tuệ nhân tạo (AI) là một nhánh của khoa học máy tính chuyên về phát triển các hệ thống máy tính có khả năng thực hiện những nhiệm vụ đòi hỏi trí thông minh của con người. AI bao gồm nhiều lĩnh vực con khác nhau như học máy (machine learning), xử lý ngôn ngữ tự nhiên (natural language processing), thị giác máy tính (computer vision), và robot học (robotics). Mục tiêu chính của AI là phát triển các hệ thống có khả năng tư duy, học hỏi và đưa ra quyết định một cách tự động.

AI có thể được phân thành hai loại chính: AI yếu (narrow AI) và AI mạnh (general AI). AI yếu được thiết kế để thực hiện một nhiệm vụ cụ thể, chẳng hạn như nhận diện khuôn mặt hoặc dự đoán xu hướng thị trường. Đối với AI mạnh, mặc dù vẫn còn trong giai đoạn nghiên cứu nhưng được kỳ vọng có thể thực hiện bất kỳ nhiệm vụ sử dụng trí tuệ nào mà con người có thể làm. Khi nhắc tới AI thì không thể bỏ qua khái niệm cực kỳ nổi bật đó là học máy Học máy, hay tên gọi tiếng Anh là machine learning, là một nhánh con quan trọng của AI, tập trung vào việc phát triển các thuật toán và mô hình cho phép máy tính học hỏi từ dữ liệu. Thay vì được lập trình rõ ràng cho từng nhiệm vụ, các hệ thống học máy tự động cải thiện hiệu suất thông qua việc học từ kinh nghiệm và dữ liệu. Học máy là một phần cực kỳ quan trọng của AI, đóng vai trò đặc biệt trong việc hiện thực hóa các hệ thống AI thông minh. Trong nhiều ứng dụng AI hiện đại, học máy được sử dụng để tạo ra các mô hình có khả năng dự đoán, phân loại và ra quyết định tự động. Nhờ đó, AI có thể phân tích và hiểu dữ liệu phức tạp, từ đó thực hiện các nhiệm vụ mà trước đây chỉ con người mới có thể làm được. [1] [2]

Trong lĩnh vực học máy, có ba nhóm thuật toán học máy chính được sử dụng rộng rãi: học có giám sát (Supervised learning), học không giám sát (Unsupervised learning), và học tăng cường (Reinforcement learning). Mỗi nhóm đều có những đặc điểm và ứng dụng riêng, giúp giải quyết các vấn đề khác nhau.

Học có giám sát là nhóm thuật toán phổ biến nhất. Trong nhóm này, mô hình sẽ huấn luyện trên tập dữ liệu đã được gắn nhãn cho trước, tức là mỗi dữ liệu đầu vào (input) đi kèm với một nhãn đầu ra (output). Mục tiêu là học cách ánh xạ từ đầu vào đến đầu ra dựa trên các cặp (data, label).

Học không giám sát không yêu cầu dữ liệu phải được gắn nhãn. Thay vào đó, nó khám phá các cấu trúc tiềm ẩn trong dữ liệu. Học không giám sát thường được sử dụng cho các bài toán phân cụm (clustering) và giảm số chiều (dimensionality reduction).

Học tăng cường là nhóm thuật toán mà trong đó một tác nhân (agent) học cách hành động trong một môi trường để tối đa hóa phần thưởng tích lũy. Mô hình này học bằng cách tương tác với môi trường và tiếp nhận phản hồi dưới dạng phần thưởng hoặc hình phạt.

Trí tuệ nhân tạo (AI) và học máy (Machine Learning) đang mở ra những khả năng mới và thay đổi cách chúng ta tiếp cận các vấn đề phức tạp. Học máy, như một nhánh quan trọng của AI, đang ngày càng chứng tỏ vai trò không thể thiếu trong việc phân tích và khai thác dữ liệu. Từ việc hỗ trợ chẩn đoán y tế đến tối ưu hóa kinh doanh, học máy đang giúp cải thiện hiệu suất và hiệu quả trong nhiều lĩnh vực. Tuy nhiên, để khai thác tối đa tiềm năng của học máy, cần có sự đầu tư liên tục vào nghiên cứu, phát triển và quản lý dữ liệu một cách hiệu quả. Học máy không chỉ đóng vai trò hỗ trợ mà còn là động lực quan trọng thúc đẩy sự đổi mới và phát triển trong tương lai.[1]

Hình 2.1A diagram of a graph

Description automatically generated with medium confidence Sơ đồ lý thuyết học máy

### 2.1.1 Học có giám sát (Supervised Learning)

Học có giám sát là loại thuật toán thông dụng nhất trong học máy, ở học có giám sát bộ dữ liệu được gắn nhãn đưa vào mô hình có sẵn, sau khi huấn luyện ta thu được các kết quả đầu ra. Trong học có giám sát, mỗi mẫu dữ liệu đầu vào được đại diện bằng x và mục tiêu dự đoán được đại diện bằng y. Mục tiêu y thường là một nhãn hoặc giá trị mong muốn để dự đoán.

Học có giám sát thường được phân ra hai loại chính là Classification (Phân loại) và Regression (Hồi quy).

Phân loại (Classification): Trong bài toán phân loại, đó là quá trình gắn nhãn cho dữ liệu và từng phân lớp hoặc các nhóm khác nhau dự vào các đặc điểm của nó. Mục tiêu là xây dựng một mô hình học máy có khả năng dự đoán nhãn (lớp) của các dữ liệu mới dựa trên các đặc trưng đầu vào. Ví dụ, trong bài toán xác định xem một email có phải là spam hay không, thuật toán phân loại sẽ được sử dụng để dự đoán nhãn "spam" hoặc "non-spam" cho từng email.

Hồi quy (Regression): Hồi quy là một loại mô hình dự đoán giá trị của một biến liên tục dựa trên các biến độc lập. Nó được sử dụng để mô hình hóa và dự đoán mối quan hệ giữa các biến. Mục tiêu là xây dựng một mô hình học máy có khả năng dự đoán giá trị số cho các dữ liệu mới dựa trên các đặc trưng đầu vào. Ví dụ, trong bài toán dự đoán giá của một căn nhà dựa trên diện tích, số phòng ngủ và khoảng cách đến trung tâm thành phố, thuật toán hồi quy sẽ được sử dụng để dự đoán giá trị số là giá của căn nhà.[3]

### 2.1.2 Học không giám sát (Unsupervised Learning)

Học không giám sát (unsupervised learning) là một loại thuật toán quan trọng trong học máy, nơi mà mô hình được huấn luyện trên một tập dữ liệu không có nhãn. Điều này có nghĩa là các đầu ra mong muốn không được cung cấp trong quá trình huấn luyện, buộc mô hình phải tự tìm kiếm và xác định các mẫu và cấu trúc tiềm ẩn trong dữ liệu. Học không giám sát đóng vai trò then chốt trong việc khai thác thông tin từ dữ liệu phi cấu trúc và đã được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau như xử lý ngôn ngữ tự nhiên, phân tích thị trường, và sinh học.

Cụ thể, có hai loại chính trong học không giám sát là:

* Clustering (phân cụm): Phân cụm là quá trình chia dữ liệu thành các nhóm sao cho các điểm dữ liệu trong cùng một nhóm có nhiều điểm chung với nhau. Đây là một kỹ thuật mạnh mẽ để khám phá các cấu trúc ẩn trong dữ liệu. Ví dụ, trong phân tích thị trường, phân cụm có thể được sử dụng để phân loại khách hàng dựa trên hành vi mua sắm của họ, từ đó hỗ trợ việc phát triển chiến lược tiếp thị hiệu quả.

Không những thế phân cụm còn hỗ trợ trong việc giảm chiều dữ liệu và phát hiện các ngoại lệ. Chẳng hạn, trong sinh học, phân cụm có thể giúp xác định các nhóm gene có chức năng tương tự, hoặc trong bảo mật mạng, nó có thể được sử dụng để phát hiện các hoạt động bất thường.

* Association (Luật kết hợp): Luật kết hợp tập trung vào việc tìm kiếm các quy luật hoặc mối liên hệ giữa các biến trong tập dữ liệu. Thuật toán Apriori và FP-Growth là những công cụ phổ biến trong lĩnh vực này. Ví dụ: Từ trong giỏ hàng mua sắm của khách hàng, nơi các nhà bán lẻ có thể phát hiện ra những mặt hàng nào thường được mua cùng nhau, từ đó tối ưu hóa việc sắp xếp hàng hóa và các chương trình khuyến mãi. Ngoài ra, khai thác luật kết hợp còn có ứng dụng trong nhiều lĩnh vực khác như phân tích văn bản, nơi nó có thể giúp tìm ra các từ thường xuất hiện cùng nhau trong các tài liệu, hoặc trong y học, để xác định mối liên hệ giữa các triệu chứng và bệnh tật.

### 2.1.3 Học sâu (Deep learning)

Học sâu (Deep Learning) là một lĩnh vực then chốt trong học máy, được phát triển dựa trên mô hình và hoạt động của mạng nơ-ron nhân tạo (Artificial Neural Networks). Ý tưởng của học sâu là thiết lập các mạng nơ-ron với nhiều tầng ẩn (hidden layers) để thu nhận và xử lý thông tin từ dữ liệu đầu vào một cách hiệu quả.

Trong học sâu, các mạng nơ-ron được thiết kế với kiến trúc phức tạp, cho phép việc học hỏi dữ liệu thông qua nhiều lớp nơ-ron được kết nối liên tiếp. Mỗi lớp nơ-ron trong mạng có khả năng nhận diện các đặc trưng trừu tượng từ dữ liệu đầu vào, bắt đầu từ những đặc trưng cơ bản đến những đặc trưng phức tạp hơn.

Các mô hình học sâu như Convolutional Neural Networks (CNNs) và Recurrent Neural Networks (RNNs) đã chứng minh được hiệu quả cao trong việc giải quyết các bài toán nhận dạng hình ảnh, nhận dạng giọng nói, và xử lý ngôn ngữ tự nhiên với độ phức tạp và yêu cầu độ chính xác cao. Khả năng tự động học từ dữ liệu lớn mà không cần sự can thiệp quá nhiều từ con người đã mở ra nhiều ứng dụng mới cần thiết trong cuộc sống của con người, từ những ngành cần tới sự chuẩn xác cao như xe tự lái hay chẩn đoán y khoa. Tương lai của học sâu hứa hẹn sẽ tiếp tục mang lại những đột phá mới, góp phần nâng cao hiệu quả và độ chính xác trong nhiều lĩnh vực khác nhau.

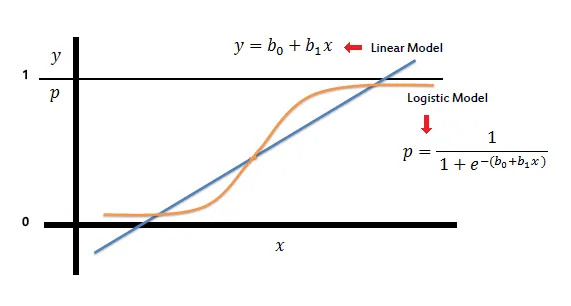
**2.2 Kỹ thuật dùng trong bài toán**

### 2.2.1 Hồi quy Logistic (Logistic Regression)

* Khái niệm

Hồi quy logistic là một kỹ thuật phân tích dữ liệu dựa trên toán học để khám phá mối liên hệ giữa hai yếu tố dữ liệu. Sau đó áp dụng mối quan hệ này để tiên đoán giá trị của một trong những yếu tố đó dựa trên yếu tố còn lại. Dự đoán này thường có một số kết quả hữu hạn, như là có hoặc không.

Hồi quy logistic sử dụng hàm sigmoid, một hàm phi tuyến, để chuyển đổi đầu vào thành xác suất của một trong hai lớp nhị phân.



Hình 2.2 Hồi quy logistic sử dụng hàm phi tuyến để xác định xác suất của hai lớp 0 và 1

Ta có phương trình tuyến tính của mô hình, hàm Sigmoid nhận một giá trị z bất kỳ làm đầu vào và trả về một giá trị xác suất nằm trong khoảng [0,1]:

(2‑1)

Trong đó: : đầu ra trong khoảng từ 0 đến 1 (giá trị xác suất ước lượng)

e: đầu vào của hàm (giá trị dự đoán của thuật toán)

z: hằng số Euler, là cơ số của logarit tự nhiên

* Sơ đồ hoạt động của Hồi quy logistic

A diagram of a graph

Description automatically generated

Hình 2.3 Sơ đồ hoạt động hồi quy logistic

Giải thích sơ đồ hoạt động:

**Bước 1**: Chúng ta tính giá trị đầu ra y bằng cách kết hợp tuyến tính các biến đầu vào sau khi đã nhân với các hệ số tương ứng.

* Mỗi node (hình tròn) đại diện cho một biến đầu vào.
* Các cạnh được biểu thị bằng mũi tên có hướng, thể hiện chiều hướng tính toán của đồ thị.
  + **Đầu vào**: node ở gốc mũi tên.
  + **Đầu ra**: node ở ngọn mũi tên.

**Bước 2**: Giá trị y được đưa qua hàm Sigmoid để biến đổi thành xác suất p, giá trị này nằm trong khoảng từ 0 đến 1. [3]

Quá trình huấn luyện mô hình tập trung vào việc xác định bộ trọng số 𝑤 để đầu ra dự đoán từ hàm Sigmoid khớp nhất với kết quả thực tế. Để làm được điều này, ta sử dụng hàm mất mát (loss function) để đánh giá hiệu năng của mô hình.

* Hàm mất mát

Hàm mất mát (loss function) là một hàm toán học được sử dụng để đo lường mức độ sai lệch giữa giá trị dự đoán và giá trị thực tế trong một bài toán học máy. Nhiệm vụ của nó là đưa ra một đánh giá chất lượng của mô hình dự đoán.

(2‑2)

Trong phương trình trên:

* L(w) là hàm mất mát, là đại lượng cần tối thiểu hóa
* w là tham số của mô hình hồi quy
* n là số lượng mẫu dữ liệu trong tập huấn luyện.
* 𝑦𝑖 là giá trị thực tế của đầu ra thứ i.
* 𝑝𝑖 là xác suất dự đoán thuộc lớp 1 của mô hình cho đầu vào thứ i.

Giá trị của hàm số L(w) càng bé thì mô hình hồi quy càng tốt. Để tối ưu hóa cho hàm số w thì ta có thể sử dụng thuật toán gradient descent. Thuật toán này hoạt động bằng cách bắt đầu với một giá trị ban đầu của tham số w và sau đó liên tục cập nhật tham số này cho đến khi hàm mất mát đạt đến một giá trị nhỏ nhất.

(2‑3)

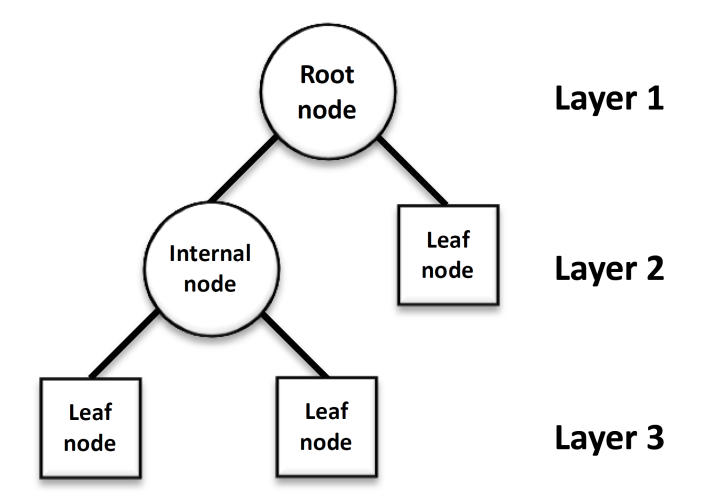
### 2.2.2 Cây quyết định (Decision Tree Classifier)

* **Khái niệm:**

Cây quyết định (Decision Tree Classifier) là một thuật toán học có giám sát được sử dụng trong học máy để phân loại dữ liệu. Thuật toán này hoạt động bằng cách phân chia tập dữ liệu thành các nhóm con nhỏ hơn dựa trên các thuộc tính của dữ liệu. Quy trình này được lặp lại đệ quy cho đến khi đạt được các nhóm con thuần nhất hoặc đáp ứng một số điều kiện dừng nhất định. Kết quả cuối cùng là một cấu trúc cây quyết định, trong đó mỗi nút biểu diễn một thuộc tính của dữ liệu, mỗi nhánh biểu diễn một kết quả có thể có của thuộc tính đó, và mỗi lá cây biểu diễn một nhãn hoặc giá trị quyết định.

Cấu trúc của cây quyết định bao gồm các thành phần chính sau:

* **Nút gốc (Root Node)**: Là nút đầu tiên của cây, đại diện cho toàn bộ dữ liệu đầu vào.
* **Nút trung gian (Internal Node)**: Đại diện cho các thuộc tính của dữ liệu, mỗi nút trung gian thực hiện một thử nghiệm trên một thuộc tính và chia dữ liệu thành các tập con dựa trên kết quả của thử nghiệm.
* **Nút lá (Leaf Node)**: Đại diện cho các nhãn hoặc giá trị dự đoán, là kết quả cuối cùng của các quyết định dựa trên các thuộc tính.



Hình 2.4 Cấu trúc của thuật toán Cây quyết định



Hình 2.5 Minh họa về thuật toán Cây quyết định

* Ưu điểm:
* Dễ hiểu và trực quan: Cây quyết định dễ dàng giải thích và trực quan hóa, giúp người dùng dễ dàng hiểu được cách thức hoạt động của mô hình.
* Không cần giả định về phân phối dữ liệu: Thuật toán không dựa vào các giả định về phân phối của các biến đầu vào.
* Có thể xử lý cả dữ liệu số và dữ liệu phân loại: Cây quyết định có thể làm việc với cả các thuộc tính dạng số và dạng phân loại mà không cần chuyển đổi.
* Nhược điểm:
* Dễ bị overfitting: Cây quyết định có thể dễ dàng bị overfitting nếu không được kiểm soát độ sâu hoặc không có các kỹ thuật làm giảm độ phức tạp của cây.
* Không ổn định với dữ liệu biến đổi: Một thay đổi nhỏ trong dữ liệu có thể dẫn đến sự thay đổi lớn trong cấu trúc của cây quyết định, làm cho mô hình trở nên không ổn định.
* Thiên vị các thuộc tính có nhiều mức: Cây quyết định có xu hướng ưa thích các thuộc tính có nhiều mức phân chia hơn, điều này có thể dẫn đến các cây không tối ưu.
* Hiệu suất kém với dữ liệu lớn: Khi làm việc với các tập dữ liệu rất lớn hoặc các thuộc tính có nhiều giá trị, cây quyết định có thể trở nên rất phức tạp và kém hiệu quả hơn so với các thuật toán khác.

### 2.2.3 Rừng ngẫu nhiên (Random Forest Classifier)

Đúng như tên gọi của thuật toán, Random là ngẫu nhiên, Forest là rừng, ở thuật toán này nó kết hợp nhiều cây quyết định (decision trees) thành một "rừng" (forest) và dựa trên nguyên tắc số đông để đưa ra dự đoán cuối cùng. Mỗi cây quyết định trong rừng được huấn luyện trên một mẫu ngẫu nhiên của dữ liệu, và cây đó đưa ra một dự đoán sau đó chọn dự đoán xuất hiện nhiều nhất trong số các dự đoán của các cây để đưa ra kết quả cuối cùng.

A diagram of a tree

Description automatically generated

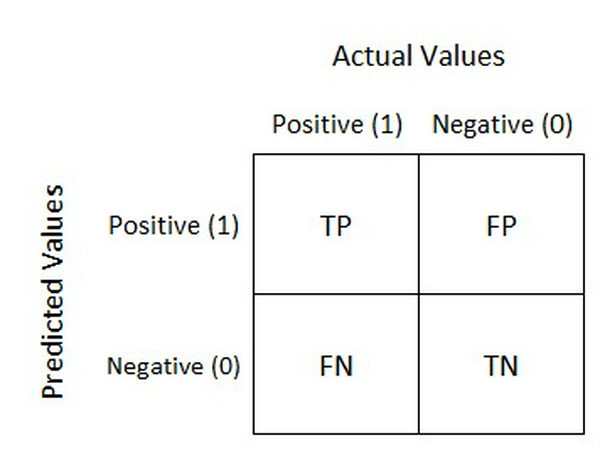
Hình 2.6 Minh họa về cách hoạt động của thuật toán Rừng ngẫu nhiên

* Ưu Điểm:
* **Hiệu suất cao và độ chính xác:** Thường có độ chính xác cao hơn so với một cây quyết định đơn lẻ.
* **Ít ảnh hưởng bởi overfitting:** Do kết hợp nhiều cây quyết định với nhau, Rừng ngẫu nhiên ít bị ảnh hưởng bởi overfitting.
* **Xử lý dữ liệu mất cân bằng:** Có thể xử lý tốt các tập dữ liệu có số lượng lớp không cân bằng.
* **Xử lý dữ liệu đa dạng:** Có khả năng làm việc với cả dữ liệu phân loại và dữ liệu liên tục.
* **Tính năng quan trọng:** Cung cấp thông tin về tầm quan trọng của các đặc trưng nào ảnh hưởng nhiều nhất đến kết quả dự đoán.
* Nhược Điểm:
* **Tính toán và tài nguyên:** Yêu cầu nhiều tài nguyên tính toán và bộ nhớ.
* **Diễn giải kết quả phức tạp:** Kết quả của Rừng ngẫu nhiên khó diễn giải hơn so với các thuật toán đơn giản như cây quyết định hoặc hồi quy tuyến tính.
* **Khả năng dự đoán không tốt trên dữ liệu mới:** Mặc dù giảm thiểu được overfitting, nhưng đôi khi Rừng ngẫu nhiên có thể không dự đoán tốt trên dữ liệu hoàn toàn mới nếu mẫu huấn luyện không đủ lớn.

## 2.3 Đánh giá mô hình

2.3.1. Confusion Matrix

Confusion Matrix, hay ma trận nhầm lẫn, là một công cụ phổ biến trong học máy để đánh giá hiệu suất của các mô hình phân loại. Nó cho phép chúng ta hiểu rõ hơn về cách mà mô hình của chúng ta đang hoạt động bằng cách trình bày số lượng các dự đoán đúng và sai theo từng loại. Ma trận nhầm lẫn gồm bốn ô: True Positive (TP), True Negative (TN), False Positive (FP), và False Negative (FN).



Hình 2.7 Một ma trận nhầm lẫn (confusion matrix)

Giả sử chúng ta có một mô hình phân loại dùng để chẩn đoán một bệnh nào đó:

True Positive (TP): số lượng các trường hợp mà mô hình dự đoán chính xác. Là khi mô hình dự đoán đúng một trường hợp bị bệnh.

True Negative (TN) là số lượng các trường hợp mà mô hình dự đoán chính xác gián tiếp. Là khi mô hình dự đoán đúng một trường hợp không bị bệnh, tức là việc không lựa chọn trường bị bệnh là chính xác.

False Positive (FP), hay còn gọi là lỗi loại I, là số lượng các trường hợp mà mô hình dự đoán sai lệch. Là khi mô hình dự đoán một người bị bệnh và người đó hoàn toàn khỏe mạnh.

False Negative (FN), hay còn gọi là lỗi loại II, là số lượng các trường hợp mà mô hình dự đoán sai một cách gián tiếp. Là khi mô hình dự đoán một người không bị bệnh nhưng người đó bị bệnh, tức là việc không chọn trường hợp bị bệnh là sai.

Dựa trên các giá trị này, chúng ta có thể tính toán một loạt các chỉ số hiệu suất khác nhau như độ chính xác (accuracy), độ nhạy (recall), và độ chính xác của từng loại (precision).

2.3.2. Accuracy

Đây là một phương pháp đánh giá độ chính xác của bài toán phân lớp đơn giản nhất. Tính độ chính xác bằng cách sử dụng số kết quả dự đoán chính xác chia cho toàn bộ số mẫu dự đoán. Có nhiều phương pháp để tìm ra độ chính xác của một model. Trong đó thường được sử dụng nhất là phương pháp Accuracy.

(2‑4)

Khi giá trị Accuracy cao cho thấy mô hình có khả năng phân loại chính xác các mẫu dữ liệu và ngược lại khi giá trị Accuracy thấpta thấy mô hình đang phân loại sai nhiều mẫu dữ liệu.

Ưu điểm của Accuracy là đơn giản và trực quan trong việc đo lường hiệu suất của một mô hình hay hệ thống bằng cách trực tiếp tính toán tỷ lệ đúng trên tổng số trường hợp xét. Tuy nhiên, nhược điểm của Accuracy là chỉ cung cấp một tỉ lệ phần trăm chính xác tổng thể của mô hình trên toàn bộ tập dữ liệu mà không chỉ ra được độ chính xác của mỗi mẫu riêng lẻ. Điều này có thể dẫn tới việc không phản ánh đúng về hiệu suất của mô hình trên các trường hợp cụ thể, đặc biệt khi tập dữ liệu mất cân bằng. Accuracy cũng không thể đo lường được sự chênh lệch của mô hình giữa các lớp dữ liệu, khiến cho một mô hình có độ chính xác cao tổng thể vẫn có thể dự đoán kém trên các lớp thiểu số.

2.3.3. Precision

Precision là phương pháp đánh giá mô hình thể hiện tỷ lệ các dự đoán đúng trong số các dự đoán được phân loại là positive. Nói cách khác, nó cho ta biết mức độ tin cậy của các kết quả positive mà mô hình đưa ra.

Công thức tính Precision:

(2‑5)

Ưu điểm lớn nhất của Precision là nó phù hợp cho các bài toán mà false positive gây hậu quả nghiêm trọng hơn false negative, ví dụ như phát hiện lừa đảo, ung thư. Tuy nhiên, nhược điểm của Precision là nó chỉ tập trung vào hiệu suất của lớp positive mà không xem xét lớp negative, do đó có thể dẫn đến những đánh giá không công bằng trong trường hợp dữ liệu không cân bằng giữa các lớp.

2.3.4. Recall

Cũng tương tự với Precision, chỉ khác ở mẫu số công thức, Recall là phương pháp đo lường tỷ lệ dự báo chính xác các trường hợp positive trên toàn bộ mẫu thuộc nhóm positive

Công thức tính Recall:

(2‑6)

Ví dụ, trong một bài toán chẩn đoán bệnh, nếu recall cao đồng nghĩa với việc True Positive Rate cao, điều đó có nghĩa là mô hình có khả năng phát hiện hầu hết các bệnh nhân mắc bệnh.

2.3.5. F1-score

Recall và precision thường có mối quan hệ ngược chiều nhau: khi tăng recall thì precision có thể giảm và ngược lại. Điều này xảy ra do khi chúng ta điều chỉnh mô hình để nhận diện nhiều trường hợp dương hơn (tăng recall), mô hình có thể tạo ra nhiều dự đoán dương sai hơn (giảm precision). Ngược lại, khi chúng ta điều chỉnh mô hình để giảm dự đoán dương sai (tăng precision), mô hình có thể bỏ sót nhiều trường hợp dương thực sự (giảm recall).

Từ mối quan hệ của Precision và Recall, F1-score chính là trung bình của 2 phương pháp trên, giúp đánh giá tổng hiệu suất của mô hình. Nó được sử dụng khi ta quan tâm đến **cả hai** việc dự đoán true positive và tránh dự đoán False Negative.

​ (2‑7)

F1-score **không phải lúc nào cũng là lựa chọn tốt nhất.** Trong một số trường hợp, ta có thể quan trọng hơn một chỉ số so với chỉ số kia, ví dụ như Recall cao hơn Precision. Khi đó, ta nên sử dụng chỉ số **phù hợp** với mục tiêu cụ thể của bài toán. [4]

# CHƯƠNG 3: ỨNG DỤNG PHƯƠNG PHÁP VÀ XÂY DỰNG MÔ HÌNH

## 3.1 Mô tả bài toán

Chương trình xếp hạng mì ramen là một phần mềm được thiết kế trong mục đích xác định chất lượng dựa trên những đặc điểm cơ bản của một loại mì như mùi vị, cách đóng gói,… Từ đó ta có thể lựa chọn ra loại mì có rating cao mà phù hợp với sở thích và khả năng tài chính của mình.

## 3.2 Phân tích dữ liệu

Bộ dữ liệu tổng cộng 4650 bản ghi và được tổng hợp bởi trang [THE RAMEN RATER](https://www.theramenrater.com/). Dữ liệu bao gồm 7 cột thuộc tính bao gồm:

1. Reviews #: Mã đánh giá (Integer)
2. Brand: Nhãn hiệu sản xuất món ramen (Object)
3. Style: Kiểu đóng gói của Ramen ['Cup','Box','Tray','Bowl','Pack'] (Categorical)
4. Variety: Tên của loại mì cùng với nhãn hiệu (object)
5. Country: Quốc gia xuất xứ (object)
6. Ratings: Đánh giá trên thang điểm từ 1-5 (object)
7. T: Rank của loại mì trong 1 năm nhất định (object)

A screenshot of a computer

Description automatically generated

Hình 3.18 Dữ liệu ban đầu

Nhiệm vụ là xếp hạng từng loại mì xem đó là mì có rating cao hay thấp từ đó có thể chọn ra được đó là loại mì ngon (rating cao) hay không ngon (rating thấp).

3.2.1 Tiền xử lý dữ liệu

3.2.1.1 Kiểm tra dữ liệu

Sử dụng câu lệnh df.isnull().sum() để kiểm tra xem dữ liệu có giá trị null nào hay không?

**A screenshot of a computer

Description automatically generated**

Hình 3.29 Kiểm tra giá trị null trong từng cột

Nhận thấy cột Stars có 2 dữ liệu null và cột T là cột chứa rank top 10 loại mì của các năm nên sẽ có rất nhiều dữ liệu null. Em thực hiện xóa các hàng giá trị null có trong cột Stars và xóa luôn cột không cần thiết là cột Review # (bởi đó chỉ là cột mang thứ tự)

A screenshot of a menu

Description automatically generated

Hình 3.310 Dữ liệu mới sau khi loại bỏ null và cột Review #

**Tiến hành làm sạch dữ liệu trong cột Stars**

Cột Stars là cột dữ liệu chứa rating cụ thể của từng loại mì. Do đó, cột này là cột dữ liệu cực kỳ quan trọng và cần được kiểm tra kỹ lưỡng để dữ liệu được đồng nhất. Em tiến hành kiểm tra dữ liệu có trong cột Stars.

A screenshot of a computer

Description automatically generated

Hình 3.411 Các giá trị không xác định trong cột Stars

Sau khi cho đếm các giá trị có trong cột Stars, em nhận thấy trong cột này không chỉ có các giá trị là dưới dạng số mà còn xuất hiện một số giá trị không xác định dưới dạng text. Điều này gây ra sự bất đồng bộ, ta cần toàn bộ dữ liệu trong cột này dưới dạng số.

Thực hiện chuyển dữ liệu trong cột về kiểu numeric và loại bỏ các giá trị không xác định:

df['Stars']=pd.to\_numeric(df['Stars'], errors='coerce')

df = df.dropna(subset=['Stars'])

df.to\_csv('ramen\_rating\_2023\_clean.csv', index=None)

Sau quá trình xử lý trên, toàn bộ dữ liệu đã được làm sạch, loại bỏ cột không cần thiết và xuất ra file dữ liệu csv mới để sẵn sàng cho quá trình tiếp theo.

3.2.1.2 Bổ sung các trường dữ liệu

Từ việc tìm hiểu ý nghĩa của các cột có trong dữ liệu, em nhận thấy chỉ có cột Stars là cột dữ liệu phù hợp cho việc sử dụng các mô hình học máy để phân loại các loại mì. Vì vậy, em đã nghĩ tới việc trích xuất các đặc điểm của từng loại mì được thực hiện bằng cách trích xuất thông tin từ những cột dữ liệu đã có sẵn thay vì chỉ sử dụng những dữ liệu đã có hiện tại để tạo mô hình. Mỗi loại mì sẽ có tên khác nhau và trong tên đó thường sẽ kèm theo tên hương vị của từng loại mì. Từ đó ta có thể xác định được hương vị của mì đó là hương vị như thế nào. Cụ thể, từ cột Variety ta có thể tìm ra được đó là loại mì cay hay không dựa vào các từ mang ý nghĩa “cay”. Ngoài ra, từ việc phân tích tần xuất xuất hiện của các từ ngữ trong tên các loại mì, ta có thể chọn ra các hương vị phổ biến để tạo nên các trường dữ liệu nổi bật khác. Thêm nữa ở các trường dữ liệu Brand và Country, ta cũng sẽ tìm những giá trị nổi bật, chọn các giá trị có tần xuất xuất hiện lớn, để tạo ra thêm nhiều cột dữ liệu khác để tăng độ lớn của dữ liệu.

Đầu tiên với cột Variety, em tiến hành trích xuất thông tin từ bằng cách tách toàn bộ các giá trị có trong cột dữ liệu này thành các token, mỗi token ứng với 1 từ có trong từng giá trị. Các loại mì thường có 2 loại đó là vị cay và không cay. Đó là lý do để em tạo thêm một trường dữ liệu mới với tên “IsSpicy”: có vị cay.

Để lọc ra được mì đó có là mì cay hay không cay, em sẽ thực hiện tách các tên loại mì có trong cột Variety thành các token, mỗi token ứng với một từ.

Bước đầu tiên, em kết hợp các phần tử có trong danh sách Variety bằng câu lệnh:

variety\_text = " ".join(variety)

Đồng thời chuyển chuỗi thu được về dạng lower(chữ thường):

variety\_text\_lower = variety\_text.lower()

Tiếp theo lọc ra những ký tự đặc biệt “! " # $ % & ' ( ) \* + , - . / : ; < = > ? @ [ \ ] ^ \_ ` { | } ~” để loại bỏ chúng:

punctuation\_list = list(string.punctuation)

variety\_lower\_clean = "".join([char for char in variety\_text\_lower if char not in punctuation\_list])

Tách chuỗi trên thành các token đơn lẻ:

variety\_token = nltk.tokenize.word\_tokenize(variety\_lower\_clean)

Sau khi phân tích các token đơn lẻ thì em dựa vào các từ có liên quan đến “vị cay” có trong tên các loại mì theo ngôn ngữ của các quốc gia khác nhau như: "spicy", "curry", "hot", "tom", "chilli", "chili", "kari", "pedas", "laksa", "sriracha", "pepper", "sichuan", "habanero", "kimchi", "mala", "picante", "kung", "szechuan", "ghost", "peppery". Và em đặt cho cột này tên IsSpicy và gán cho cột các giá trị nhị phân cụ thể với những mì có vị cay thì giá trị của mì đó sẽ là 1, và ngược lại nếu không có giá trị sẽ là 0.

is\_spicy = []

spicy\_flavor = ["spicy", "curry", "hot", "tom", "chilli", "chili", "kari", "pedas", "laksa", "sriracha", "pepper", "sichuan", "habanero", "kimchi", "mala", "picante", "kung", "szechuan", "ghost", "peppery"]

for variety in df["Variety"].str.lower():

    if any(flavor in variety for flavor in spicy\_flavor):

        is\_spicy.append(1)

    else:

        is\_spicy.append(0)

df["IsSpicy"] = is\_spicy

A screenshot of a menu

Description automatically generated

Hình 3.512 Dữ liệu sau khi tạo xong cột IsSpicy

Tương tự như vậy, tiếp tục dựa vào các token để thêm các trường dữ liệu khác. Ở đây ta sử dụng hàm collections.Counter để xem tần xuất xuất hiện của từng token. Từ đó có thể dựa vào các token có tần xuất xuất hiện lớn để trích xuất ra các thuộc tính quan trọng khác.

variety\_counters = collections.Counter(variety\_token)

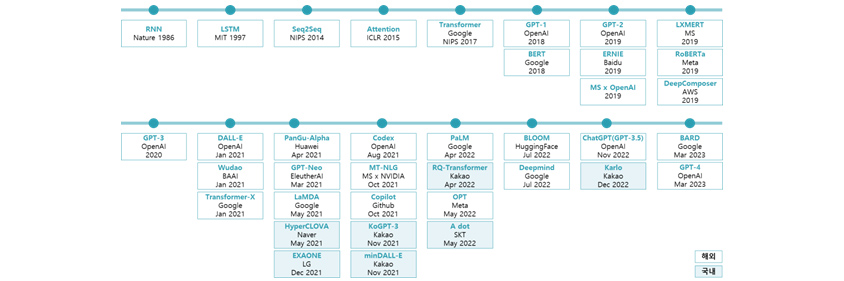
variety\_counters

Bảng 3.1 Một số token có tần xuất xuất hiện lớn

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| noodles | noodle | ramen | Flavor | instant | flavour | chicken | spicy | soup | beef | |
| 1047 | 976 | 897 | 657 | 637 | 580 | 533 | 532 | 427 | 359 | |
| cup | with | sauce | Hot | curry | rice | artificial | shrimp | mi | tom | seafood |
| 294 | 289 | 278 | 246 | 224 | 217 | 181 | 177 | 172 | 169 | 163 |

Ứng dụng ChatGPT vào xử lý token

ChatGPT là một mô hình ngôn ngữ tự nhiên được phát triển bởi OpenAI. Nó sử dụng trí tuệ nhân tạo để hiểu và tạo ra ngôn ngữ tự nhiên giống như con người. ChatGPT được huấn luyện trên một lượng lớn dữ liệu văn bản từ nhiều nguồn khác nhau, giúp nó có khả năng hiểu và tạo ra văn bản bằng nhiều ngôn ngữ khác nhau với độ chính xác cao.



Hình 3.613 Timeline sự phát triển chính của các mô hình ngôn ngữ

Ở phần này với số lượng token lớn và hầu hết đều là tiếng nước ngoài nên em đã áp dụng ChatGPT vào trong quá trình xử lý token. Việc này đã giúp đỡ rất nhiều bởi ChatGPT cung cấp các giải thích chi tiết và chính xác về ý nghĩa của các token được tìm thấy. Điều này không chỉ giúp em hiểu rõ hơn về ngữ cảnh và ý nghĩa của từng token mà còn giúp loại bỏ những token không cần thiết hoặc không chính xác. Hơn nữa, ChatGPT có khả năng phát hiện và sửa chữa các lỗi chính tả, đảm bảo rằng các token được xử lý một cách chính xác và hiệu quả. Nhờ vào khả năng này, quá trình chọn lọc token trở nên nhanh chóng và hiệu quả hơn rất nhiều so với việc sử dụng các phương pháp truyền thống. [5]

Em nhận thấy ‘chicken’, ‘beef’, ‘seafood’ có tần xuất xuất hiện lớn nên em sẽ tìm thêm các từ đồng nghĩa hoặc các từ mang ý nghĩa tương tự để trích ra thuộc tính quan trọng. Ba từ trên sẽ đại diện cho ba hương vị mà các loại mì này mang lại gồm vị gà, vị bò, vị hải sản. Ba hương vị này là ba hương vị cơ bản của mì ăn liền và xuất hiện nhiều nhất. Ở đây em sẽ lựa chọn tạo thêm 3 trường dữ liệu mới với tên và ý nghĩa chọn lọc như sau:

* HasChicken: có vị gà
* HasBeef: có vị bò
* HasSeafoods: có vị hải sản

Tạo thêm thuộc tính ‘HasChicken’ với các token như "artificial", "chicken", "ayam", "pollo", "gallina". Tất cả đều mang ý nghĩa là gà hoặc liên quan tới gà:

has\_chicken = []

chicken\_flavor = ["artificial", "chicken", "ayam", "pollo", "gallina"]

for variety in df["Variety"].str.lower():

    if any(flavor in variety for flavor in chicken\_flavor):

        has\_chicken.append(1)

    else:

        has\_chicken.append(0)

df["HasChicken"] = has\_chicken

Tạo thêm thuộc tính ‘HasBeef với các token như "beef", "cow", "meat", "steak", "rib", "barbecue", "brisket","sirloin","tenderloin", "ribeye", "shank". Tất cả đều mang ý nghĩa là bò hoặc liên quan tới bò:

beef\_flavor = ["beef", "cow", "meat", "steak", "rib", "barbecue", "brisket","sirloin","tenderloin", "ribeye", "shank"]

for variety in df["Variety"].str.lower():

    if any(flavor in variety for flavor in beef\_flavor):

        has\_beef.append(1)

    else:

        has\_beef.append(0)

df["HasBeef"] = has\_beef

Tạo thêm thuộc tính ‘HasSeafoods’ với các token như "seafood", "fish", "crab", "shrimp", "prawn", "tom", "lobster", "scallop", "clam", "abalone", "oyster", "squid", "seaweed", "sushi", "ebi", "caviar". Tất cả đều mang ý nghĩa là hải sản hoặc liên quan tới hải sản:

has\_seafoods = []

seafoods\_flavor = ["seafood", "fish", "crab", "shrimp", "prawn", "tom", "lobster", "scallop", "clam", "abalone", "oyster", "squid", "seaweed", "sushi", "ebi", "caviar"]

for variety in df["Variety"].str.lower():

    if any(flavor in variety for flavor in seafoods\_flavor):

        has\_seafoods.append(1)

    else:

        has\_seafoods.append(0)

df["HasSeafoods"] = has\_seafoods

Sau khi tạo được ba trường dữ liệu với các tiêu chí như trên. Em kết hợp 3 cột trên vào dữ liệu sau khi trích xuất token từ thuộc tính Variety thu được kết quả mới như sau:

A screenshot of a computer

Description automatically generated

Hình 3.714 Dữ liệu sau khi trích xuất token từ thuộc tính Variety

3.4.1.2 Tạo biến giả

Từ cột Brand chứa thông tin về các thương hiệu khác nhau. Mục tiêu hiện tại là tạo ra các cột mới phân loại các brand thành “TopBrand” và “Other” sau đó tạo các biến giả cho 2 loại này. Các biến giả được tạo cho các danh mục này để sử dụng trong phân tích hoặc mô hình hóa, với một danh mục bị loại bỏ để tránh Multicollinearity. Multicollinearity là hiện tượng xảy ra khi các biến độc lập trong một mô hình hồi quy tuyến tính có mối tương quan cao với nhau. Hiện tượng này có thể dẫn đến việc các khoảng tin cậy rộng hơn, làm giảm độ tin cậy của các ước tính về ảnh hưởng của các biến độc lập trong mô hình. Việc phát hiện và xử lý multicollinearity là rất quan trọng để đảm bảo mô hình hoạt động chính xác và hiệu quả. [6]

Không chỉ có vậy, việc tạo biến giả giúp biến đổi các giá trị phân loại thành một hoặc nhiều cột số, trong đó mỗi cột đại diện cho một giá trị duy nhất. Em chỉ muốn tập trung vào các thuộc tính phổ biến nhất, em đã nhóm tất cả các thuộc tính ít phổ biến hơn vào một nhóm "Other". Điều này giúp giảm độ phức tạp của mô hình và tránh tình trạng quá nhiều biến giả, mà mỗi biến chỉ đại diện cho một giá trị hiếm gặp, mô hình có thể tập trung vào các biến có ảnh hưởng lớn hơn đến xếp hạng mì, giúp tăng độ chính xác và hiệu quả của mô hình. Việc nhóm các nhãn hiệu ít phổ biến vào "Other" giúp đơn giản hóa dữ liệu và dễ dàng phân tích hơn.

Với cột dữ liệu Brand, em tiến hành phân loại các tên brand hiện tại thành TopBrand và Other. Top Brand ở đây em lựa chọn 20 Brand với tần xuất xuất hiện cao sẽ thuộc TopBrand và còn lại sẽ thuộc Other. Tạo các biến giả cho 2 loại này với các giá trị là 0 ứng với Other và 1 ứng với TopBrand:

top\_brand = df['Brand'].value\_counts()[:20]

new\_brand = []

for brand in df['Brand']:

    if brand in top\_brand:

        new\_brand.append(brand)

    else:

        new\_brand.append("Other")

df['TopBrand'] = new\_brand

brand\_dummy = pd.get\_dummies(df['TopBrand'], prefix="from")

brand\_dummy = brand\_dummy.astype(int)

A screenshot of a computer

Description automatically generated

Hình 3.815 Các biến giả được tạo ra từ cột Brand

Tiếp tục tạo biến giả với cột Style khi chia các giá trị có trong cột Style thành “TopStyle” và “Other”. Ở đây ta sẽ lựa chọn 4 Style có tần xuất đứng đầu là TopStyle. Các cột biến giả chứa các giá trị số nguyên (0 hoặc 1) thay vì giá trị kiểu float mặc định.

top\_style = df['Style'].value\_counts()[:4]

new\_style = []

for style in df['Style']:

    if style in top\_style:

        new\_style.append(style)

    else:

        new\_style.append("Other")

df['TopStyle'] = new\_style

style\_dummy = pd.get\_dummies(df['TopStyle'], prefix="With")

style\_dummy = style\_dummy.astype(int)

A screenshot of a table

Description automatically generated

Hình 3.916 Các biến giả được tạo ra từ cột Style

Tiếp tục tạo biến giả với cột Style khi chia các giá trị có trong cột Style thành “TopCountry” và “Other”. Ở đây ta sẽ lựa chọn 11 Country có tần xuất cao nhất là TopCountry và còn lại sẽ thuộc Other. Tạo các biến giả cho 2 loại này với các giá trị lần lượt là là 1 và 0:

top\_country = df['Country'].value\_counts()[:11]

new\_country = []

for country in df['Country']:

    if country in top\_country:

        new\_country.append(country)

    else:

        new\_country.append("Other")

df['TopCountry'] = new\_country

country\_dummy = pd.get\_dummies(df['TopCountry'], prefix="In")

country\_dummy = country\_dummy.astype(int)

A screenshot of a computer

Description automatically generated

Hình 3.1017 Các biến giả được tạo ra từ cột Country

Sau khi tạo được các cột biến giả trên, em tiến hành thêm những cột đó vào trong file csv dữ liệu. Cùng lúc xóa đi những cột không cần thiết còn lại, ta thu được bộ dữ liệu bao gồm 4632 dữ liệu với 44 cột.

df\_temp = df.drop(['Brand', 'Variety', 'Style', 'Country', 'TopBrand', 'TopStyle', 'TopCountry'], axis=1)

df\_final = pd.concat([df\_temp, style\_dummy,brand\_dummy, country\_dummy ], axis=1)

A screenshot of a computer

Description automatically generated

Hình 3.1118 Dữ liệu cuối cùng thu được

### 3.2.2 Phân tích dữ liệu

**Vẽ biểu đồ trực quan dữ liệu**

Phân tích mối liên hệ của Styles với Stars (Ramen Ratings)

A graph with numbers and lines

Description automatically generated

Hình 3.1219 Biểu đồ boxplot giữa 2 cột Styles và Stars

Dựa vào hình 3.12, hai giá trị Can và Bar có quá ít dữ liệu để hiển thị, ngược lại rating của Box và Pack lại rất cao. Với Box, phần lớn giá trị rating trải từ 4 đến 5, trong đó trung vị rất cao, tuy nhiên cũng có những loại mì ngoại lệ trong khoảng từ 0 đến 2 rating. Điều này cho thấy rằng các sản phẩm mì đóng hộp Box thường đạt được mức độ hài lòng cao từ người tiêu dùng, tuy nhiên, cũng có những sản phẩm không đạt được chất lượng mong muốn, dẫn đến các rating thấp hơn.

Với giá trị Pack, phần lớn giá trị rating nằm trong khoảng 3.5 đến 4.75 với trung vị 4. Điều này cho thấy rằng các sản phẩm mì đóng gói Pack cũng được đánh giá cao, tuy nhiên, có sự phân bố rộng hơn về mức độ hài lòng của người tiêu dùng. Trung vị 4 cho thấy phần lớn các sản phẩm mì đóng gói đạt được sự hài lòng khá cao từ người tiêu dùng, nhưng không có nhiều sản phẩm đạt được mức tuyệt đối như các sản phẩm đóng hộp Box.

Phân tích trường dữ liệu Country

Country là cột thể hiện xuất xứ quốc gia của các loại mì có trong dữ liệu.



Hình 3.1320 Biểu đồ cột về số lượng mì của các quốc gia

Dựa vào hình 3.13, biểu đồ thể hiện số lượng mì có ở các quốc gia, số loại mì trên thế giới chủ yếu đến từ 4 quốc gia Japan, US, South Korea, Taiwan. Đáng chú ý số lượng lớn đến từ các quốc gia châu Á, thể hiện sự đa dạng trong ẩm thực châu Á so với thế giới. Các loại mì từ những quốc gia châu Á không chỉ đơn thuần là thực phẩm nhanh gọn mà còn là sự kết hợp tinh tế của nghệ thuật ẩm thực, nơi mà mỗi gói mì đều mang một câu chuyện, một phong vị riêng biệt. Điều này không chỉ làm phong phú thêm thực đơn hàng ngày mà còn mở ra cánh cửa cho nhiều người khám phá và thưởng thức nền ẩm thực đa dạng của châu Á.



Hình 3.1421 Biểu đồ về rating (Stars) của các quốc gia

Dựa vào hình 3.14 mì đến từ các quốc gia châu Á Taiwan, Malaysia, Indonesia, Singapore là những loại mì được ưa chuộng và có rating cao chủ yếu từ 3.5 đến 5. Các thương hiệu mì nổi tiếng từ những quốc gia này thường xuyên được người tiêu dùng đánh giá cao nhờ vào hương vị độc đáo, sự đa dạng trong thành phần và chất lượng vượt trội.

A diagram of a chart

Description automatically generated with medium confidenceA graph with purple and blue squares

Description automatically generated

Hình 3.1522 Biểu đồ so sánh về rating và số lượng của mì cay và không cay

Dựa vào hình 3.15, mức rating mà mì có vị cay “IsSpicy” cao hơn rõ rệt so với mì không cay. Tuy nhiên, mì không cay lại có số lượng vượt trội, gần như gấp đôi so với mì cay. Với biểu đồ cột bên phải, ta dễ dàng nhận thấy lượng mì không cay cao hơn nhiều so với mì cay, lượng mì đạt rating cao cũng rất nhiều nhưng lượng rating thấp cũng khá nhiều kéo theo việc làm giảm đi trung vị của mì không cay (ở mức 3.75), thấp hơn so với mì cay (mức 4).

## 3.3 Công cụ sử dụng

**Môi trường thực nghiệm:**

* Ngôn ngữ: Python
* Nền tảng phát triển: Jupyter Notebook
* Công cụ: Visual Studio Code
* Thiết bị: Window 11, Ram 16GB, Intel(R) Core (TM) i5-11400H

**Thư viện sử dụng chính:**

**Pandas và Numpy:**

Pandas là một thư viện Python cung cấp các cấu trúc dữ liệu và công cụ phân tích dữ liệu mạnh mẽ, đặc biệt là cho việc thao tác và phân tích dữ liệu dạng bảng (DataFrame). Numpy là một thư viện toán học cung cấp thao tác với các mảng (arrays) và ma trận (matrices) cùng với các hàm toán học để vận hành trên chúng.

**Sklearn:**

Sklearn Là một thư viện học máy trong Python cung cấp các công cụ học máy hiệu quả và dễ sử dụng để phân tích dữ liệu và xây dựng mô hình dự đoán.

**Matplotlib và Seaborn:**

Cả Matplotlib và Seaborn đều là những thư viện dùng để trực quan hóa dữ liệu trong Python. Matplotlib cung cấp các biểu đồ cơ bản, trong khi Seaborn xây dựng trên Matplotlib, cung cấp các giao diện đơn giản hơn và các kiểu biểu đồ phong phú hơn, giúp dễ dàng tạo ra các biểu đồ trực quan hóa dữ liệu phức tạp và đẹp mắt hơn.

**Nltk:**

Nltk là một thư viện xử lý ngôn ngữ tự nhiên của python cung cấp nhiều kho dữ liệu văn bản khác nhau cùng nhiều chức năng để xử lý văn bản phù hợp với nhiều mục đích sử dụng từ đó nâng cao hiệu quả, chất lượng của các mô hình thuật toán. [7]

**Streamlit:**

Streamlit là một framework web được viết bằng Python giúp người dùng tạo ứng dụng web một cách nhanh chóng và dễ dàng. Người dùng chỉ cần viết code Python và sử dụng các thành phần có sẵn của Streamlit để xây dựng giao diện người dùng và xử lý dữ liệu.

Một số widget của Streamlit trong Python

Bảng 3.2 Bảng tổng hợp widget của Streamlit [8]

|  |  |
| --- | --- |
| Widget | Mô tả |
| st.button() | Tạo một nút nhấn. |
| st.checkbox() | Tạo một ô checkbox. |
| st.selectbox() | Tạo một hộp dropdown cho việc chọn từ một danh sách các lựa chọn. |
| st.multiselect() | Tạo một hộp dropdown cho phép chọn nhiều lựa chọn từ một danh sách. |
| st.text\_input() | Tạo một ô nhập liệu văn bản. |
| st.number\_input() | Tạo một ô nhập liệu cho các giá trị số. |
| st.date\_input() và st.time\_input() | Tạo ô nhập liệu cho ngày và thời gian. |
| st.file\_uploader() | Tạo một khung tải tệp lên. |
| st.spinner() | Hiển thị một spinner khi ứng dụng đang xử lý. |
| MessageBox | Nó được sử dụng để hiển thị hộp thông báo trong các ứng dụng desktop. |
| st.progress() | Hiển thị thanh tiến trình. |
| st.text\_area() | Tạo một ô nhập liệu lớn cho văn bản đa dòng. |

## 3.4 Xây dựng mô hình

3.4.1 Huấn luyện mô hình

Chuẩn bị dữ liệu:

Khởi tạo tập test và tập train: Chia dữ liệu thành 2 phần là các biến đầu vào X và biến mục tiêu y.

X = df.drop(['binStars', 'Stars','T'], axis=1)

y = df['binStars']

Ta tiến hành thực hiện split data để bắt đầu huấn luyện mô hình học máy.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, stratify=y, random\_state=42)

Trong đó:

* X\_train, X\_test: Tập huấn luyện và tập kiểm tra của các đặc trưng.
* y\_train, y\_test: Tập huấn luyện và tập kiểm tra của các giá trị mục tiêu.
* test\_size=0.2: 20% dữ liệu sẽ được sử dụng để kiểm tra, và 80% còn lại sẽ được sử dụng để huấn luyện.
* random\_state=42: Đảm bảo quá trình chia dữ liệu sẽ cho ra kết quả nhất quán mỗi khi chạy lại đoạn mã. Số 42 chỉ là một con số ngẫu nhiên (random seed).

3.4.1.1 Huấn luyện mô hình với giá trị tham số ban đầu

Thực hiện huấn luyện mô hình với thuật toán Hồi quy Logistic:

#LogisticRegression

logr = LogisticRegression()

logr.fit(X\_train, y\_train)

logr\_pred = logr.predict(X\_test)

Đoạn code trên sử dụng thư viện thuật toán Hồi quy Logistic có sẵn của python để huấn luyện mô hình.

Thực hiện huấn luyện mô hình với thuật toán Cây quyết định:

#decisiontreeclassifier

tree = DecisionTreeClassifier()

tree.fit(X\_train, y\_train)

tree\_pred = tree.predict(X\_test)

Đoạn code trên sử dụng thư viện thuật toán Cây quyết định có sẵn của python để huấn luyện mô hình.

Thực hiện huấn luyện mô hình với thuật toán Rừng ngẫu nhiên:

#RandomForestClassifier

forest = RandomForestClassifier()

forest.fit(X\_train, y\_train)

forest\_pred = forest.predict(X\_test)

Đoạn code trên sử dụng thư viện thuật toán Rừng ngẫu nhiên có sẵn của python để huấn luyện mô hình.

3.4.1.2 Huấn luyện mô hình với Grid search

* Hồi quy logistic

logistic\_model = LogisticRegression()

param\_grid = {

        'penalty': ['l2', 'none'],

        'C': [0.01, 0.1, 1, 10, 100],

        'solver': ['newton-cg', 'lbfgs', 'sag'],

        'max\_iter': [100, 200, 300]

    }

grid\_search = GridSearchCV(estimator=logistic\_model, param\_grid=param\_grid, cv=5, n\_jobs=-1)

grid\_search.fit(X\_train, y\_train)

y\_pred\_acc = grid\_search.predict(X\_test)

print(classification\_report(y\_test, y\_pred\_acc))

Tối ưu giá trị tham số với Grid Search

Trong đó:

* penalty: Các loại quy tắc (regularization) được sử dụng để tránh overfitting. regularization L2 và none là không sử dụng regularization.
* C: Các giá trị cho tham số điều chỉnh mức độ regularization (inverse of regularization strength). Giá trị nhỏ hơn thể hiện regularization mạnh hơn.
* solver: Các thuật toán tối ưu hóa được sử dụng để tìm các tham số tốt nhất. newton-cg, lbfgs, và sag là các thuật toán khác nhau.
* max\_iter: Số lượng vòng lặp tối đa cho thuật toán tối ưu hóa để hội tụ.

Khởi tạo Grid Search :

* cv=5: là dữ liệu sẽ được chia thành 5 phần và mỗi phần sẽ được sử dụng làm dữ liệu kiểm tra một lần.
* n\_jobs=-1: Sử dụng tất cả các lõi CPU để thực hiện tìm kiếm, giúp tăng tốc độ tính toán.
* grid\_search.fit(X\_train, y\_train): Huấn luyện mô hình bằng cách sử dụng các tham số từ param\_grid và tập dữ liệu huấn luyện X\_train và nhãn y\_train.
* grid\_search.predict(X\_test): Dự đoán nhãn cho dữ liệu kiểm tra X\_test sử dụng mô hình đã được tối ưu hóa.
* Cây quyết định

tree\_model = DecisionTreeClassifier()

param\_grid = {

    'criterion': ['gini', 'entropy'],

    'splitter': ['best', 'random'],

    'max\_depth': [None, 10, 20, 30, 40, 50],

    'min\_samples\_split': [2, 5, 10],

    'min\_samples\_leaf': [1, 2, 4],

    'max\_features': [None, 'auto', 'sqrt', 'log2']

}

grid\_search = GridSearchCV(estimator=tree\_model, param\_grid=param\_grid, cv=5, n\_jobs=-1)

grid\_search.fit(X\_train, y\_train)

y\_pred\_dec = grid\_search.predict(X\_test)

print(classification\_report(y\_test, y\_pred\_dec))

Trong đó lưới tham số param\_grid:

* criterion: hàm đánh giá để đo "chất lượng" của một split.
* splitter: Chiến lược để chọn điểm tách. best chọn điểm tách tốt nhất và random chọn điểm tách ngẫu nhiên.
* max\_depth: Độ sâu tối đa của cây. Các giá trị thử nghiệm là 10, 20, 30 và không giới hạn (None).
* min\_samples\_split: Số lượng mẫu tối thiểu cần thiết để chia một nút. Các giá trị thử nghiệm là 2, 5, và 10.
* min\_samples\_leaf: Số lượng mẫu tối thiểu cần thiết để ở lại một lá. Các giá trị thử nghiệm là 1, 2, và 4.
* max\_features: Số lượng đặc trưng tối đa được xem xét khi tìm điểm tách tốt nhất. None nghĩa là sử dụng tất cả các đặc trưng, auto, sqrt, và log2 là các phương pháp khác để giới hạn số lượng đặc trưng.
* Rừng ngẫu nhiên

forest\_model = RandomForestClassifier(random\_state=42)

param\_grid = {

    'n\_estimators': [100, 200, 300],

    'max\_features': ['sqrt', 'log2'],

    'max\_depth': [10, 20, 30, None],

    'min\_samples\_split': [2, 5, 10],

    'min\_samples\_leaf': [1, 2, 4]

}

grid\_search = GridSearchCV(estimator=forest\_model, param\_grid=param\_grid, cv=5, n\_jobs=-1)

grid\_search.fit(X\_train, y\_train)

yRFC\_pred = grid\_search.predict(X\_test)

print(classification\_report(y\_test, yRFC\_pred))

Có thêm chỉ số ‘n\_estimators’: Số lượng cây trong rừng. Các giá trị thử nghiệm là 100, 200, và 300.

3.4.1.3 Kiểm tra các hệ số thể hiện mức độ ảnh hưởng của mỗi đặc trưng đến dự đoán của mô hình

Ở đây, em thực hiện 2 thao tác:

* Tạo một DataFrame chứa tên các đặc trưng và tầm quan trọng tương ứng của chúng từ mô hình hồi quy logistic đã được huấn luyện.
* Sắp xếp DataFrame này sao cho các đặc trưng có tầm quan trọng cao nhất (hệ số dương lớn nhất) được liệt kê trước.

Hệ số (Coefficient):

Mỗi đặc trưng trong mô hình sẽ có một hệ số tương ứng. Hệ số này biểu thị mức độ thay đổi của xác suất dự đoán khi đặc trưng thay đổi một đơn vị, giữ các đặc trưng khác không đổi. Nếu hệ số dương lớn thì khi giá trị của đặc trưng này tăng, xác suất của biến mục tiêu (thường là xác suất thuộc về lớp 1 trong mô hình nhị phân) cũng tăng lên. Nếu hệ số của một đặc trưng là âm và lớn về độ lớn (ví dụ: -2, -3), thì khi giá trị của đặc trưng này tăng, xác suất của biến mục tiêu giảm đi.

Tầm quan trọng (Importance):

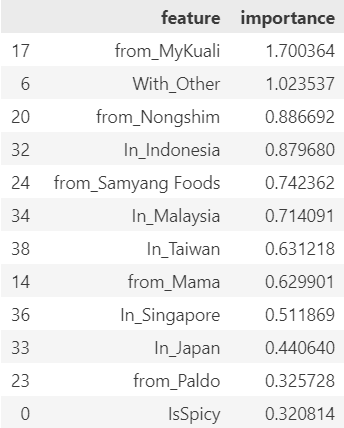
Đặc trưng có tầm quan trọng lớn là đặc trưng có hệ số (tuyệt đối) lớn, tức là nó có tác động mạnh đến kết quả dự đoán của mô hình. Và ngược lại đặc trưng có tầm quan trọng nhỏ sẽ có hệ số gần bằng 0, tức là nó có ít hoặc không có tác động đến kết quả dự đoán.

Hồi quy logistic (Logistic Regression):

logr\_feature = pd.DataFrame({"feature": X.columns,

                             "importance": logr.coef\_[0]})

logr\_feature.sort\_values('importance', ascending=False)



Hình 3.1623 Bảng các thuộc tính quan trọng với mô hình Hồi quy logistic

Với thuật toán này, cột from\_MyKuali lại có tính quan trọng nhất trong bộ dữ liệu, ảnh hưởng tới kết quả xử lý của mô hình.

Cây quyết định (Decision Tree Classifier):

tree\_feature = pd.DataFrame({"feature": X.columns,

                             "importance": tree.feature\_importances\_})

tree\_feature.sort\_values('importance', ascending=False)

A screenshot of a computer

Description automatically generated

Hình 3.17 24 Bảng các thuộc tính quan trọng với mô hình Cây quyết định

Với thuật toán này, cột In\_Other lại có tính quan trọng nhất trong bộ dữ liệu, ảnh hưởng tới kết quả xử lý của mô hình.

Rừng ngẫu nhiên (Random Forest Classifier):

forest\_feature = pd.DataFrame({"feature": X.columns,

                             "importance": forest.feature\_importances\_})

forest\_feature.sort\_values('importance', ascending=False)

A screenshot of a table

Description automatically generated

Hình 3.1825 Bảng các thuộc tính quan trọng với mô hình Rừng ngẫu nhiên

Với thuật toán này, cột IsSpicy lại có tính quan trọng nhất trong bộ dữ liệu, ảnh hưởng tới kết quả xử lý của mô hình.

3.4.2 Đánh giá chất lượng mô hình

3.4.2.1 Sử dụng giá trị tham số ban đầu

* Hồi quy Logistic (Logistic Regression)

A screenshot of a computer screen

Description automatically generated

Hình 3.1926 Độ chính xác của mô hình Hồi quy Logistic

Dựa vào hình 3.19, với mô hình này thì độ chính xác của thuật toán còn thấp, chỉ đạt mức cao nhất là 67% và 61% kết quả dự đoán đúng với đánh giá recall. Với bài toán có độ cân bằng thấp, các dữ liệu đầu vào chênh lệch nhau rất nhiều thì việc độ chính xác đạt 67% là ở mức tạm chấp nhận được và cần cải thiện thêm rất nhiều để tăng độ chính xác lên cao hơn.

* Cây quyết định (Decision Tree Classifier)

A screenshot of a computer

Description automatically generated

Hình 3.2027 Độ chính xác của mô hình Cây quyết định

Dựa vào hình 3.20, với mô hình này thì độ chính xác của thuật toán còn thấp, chỉ đạt mức 67% và 57% kết quả dự đoán đúng ở hai giá trị 0 và 1 với đánh giá recall. Như nhận xét về bộ dữ liệu ở trên thì việc độ chính xác đạt 67% và 57% là ở mức tạm chấp nhận được tuy nhiên cần cải thiện thêm để tăng độ chính xác lên cao hơn.

* Rừng ngẫu nhiên (Random Forest Classifier)

A screenshot of a computer

Description automatically generated

Hình 3.2128 Độ chính xác của mô hình Rừng ngẫu nhiên

Dựa vào hình 3.21, với mô hình này thì độ chính xác của thuật toán cũng ở mức thấp, chỉ đạt mức 66% và 60% kết quả dự đoán đúng hai giá trị 0 và 1 ở đánh giá precision. Mức độ chính xác này không khác biệt nhiều so với 2 mô hình phía trên.

Để so sánh kỹ hơn về độ chính xác của ba mô hình trên, ta có thể dựa vào bảng dưới đây để tham khảo:

Bảng 3.3So sánh độ chính xác của các thuật toán qua các phương thức đánh giá

|  |  |  |  |
| --- | --- | --- | --- |
|  | Hồi quy logistic | Cây quyết định | Rừng ngẫu nhiên |
| Accuracy | **0.64** | 0.62 | 0.63 |
| Precision | **0.61** | 0.59 | 0.60 |
| recall | 0.60 | 0.57 | **0.61** |
| F1-score | **0.61** | 0.58 | 0.60 |

Dựa vào bảng trên, nhận thấy mô hình hồi quy logistic có độ chính xác cao nhất dù không có khác biệt quá nhiều với hai phương pháp còn lại, chỉ có sự chênh lệch khoảng 1%. Điều này vẫn quan trọng trong các ứng dụng thực tế: một mô hình có độ chính xác cao hơn, dù chỉ một chút, có thể dẫn đến kết quả đáng tin cậy hơn và giảm thiểu rủi ro trong các quyết định quan trọng.

Tóm lại, mặc dù sự chênh lệch về độ chính xác không lớn, mô hình hồi quy logistic vẫn chứng tỏ được giá trị của mình trong việc phân loại dữ liệu. Việc lựa chọn mô hình phù hợp nên dựa trên nhiều yếu tố khác nhau, nhưng hồi quy logistic luôn là một phương án đáng cân nhắc trong nhiều trường hợp nhờ vào sự cân bằng giữa độ chính xác, hiệu quả tính toán và tính dễ hiểu.

3.4.2.2 Sử dụng Grid Search

* Hồi quy Logistic (Logistic Regression)

A screenshot of a computer screen

Description automatically generated

Hình 3.2229 Độ chính xác của mô hình Hồi quy Logistic với Grid Search

Dựa vào hình 3.22, với mô hình này thì độ chính xác của thuật toán còn thấp, chỉ đạt mức cao nhất là 68% và 58% kết quả dự đoán đúng với đánh giá recall. Với việc áp dụng Grid search thì độ chính xác của mô hình tăng lên không nhiều.

* Cây quyết định (Decision Tree Classifier)

A screenshot of a computer

Description automatically generated

Hình 3.2330 Độ chính xác của mô hình Cây quyết định với Grid Search

Dựa vào hình 3.23, với mô hình này thì độ chính xác của thuật toán còn thấp, chỉ đạt mức 67% và 62% kết quả dự đoán đúng ở hai giá trị 0 và 1 với đánh giá precision và f1-score. Như nhận xét về bộ dữ liệu ở trên thì việc độ chính xác đạt 67% và 62% là ở mức chưa cao.

* Rừng ngẫu nhiên (Random Forest Classifier)

A number of numbers on a white background

Description automatically generated

Hình 3.2431 Độ chính xác của mô hình Rừng ngẫu nhiên với Grid Search

Dựa vào hình 3.24, với mô hình này thì độ chính xác của thuật toán cũng ở mức thấp, chỉ đạt mức 68% và 64% kết quả dự đoán đúng hai giá trị 0 và 1 ở đánh giá Recall và f1-score. Mức độ chính xác này không khác biệt nhiều so với 2 mô hình phía trên.

3.4.2.3 Đánh giá kết quả tổng hợp

Để so sánh kỹ hơn về độ chính xác của ba mô hình trên sau khi đã áp dụng grid search, ta có thể dựa vào bảng dưới đây để so sánh:

Bảng 3.4 So sánh độ chính xác của các thuật toán sau khi sử dụng Grid Search

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Accuracy | Precision | recall | F1-score |
| Sử dụng giá trị tham số ban đầu | Hồi quy logistic | **0.64** | **0.61** | 0.60 | **0.61** |
| Cây quyết định | 0.62 | 0.59 | 0.57 | 0.58 |
| Rừng ngẫu nhiên | 0.63 | 0.60 | **0.61** | 0.60 |
| Sử dụng Grid Search | Hồi quy logistic | 0.64 | 0.61 | 0.58 | 0.60 |
| Cây quyết định | 0.65 | 0.62 | 0.62 | 0.62 |
| Rừng ngẫu nhiên | **0.66** | **0.63** | **0.64** | **0.64** |

Dựa vào bảng trên, sau khi sử dụng Grid Search thì độ chính xác đã có cải thiện so với sử dụng giá trị tham số ban đầu, mô hình Rừng ngẫu nhiên có độ chính xác cao nhất dù không có khác biệt quá nhiều với hai phương pháp còn lại, chỉ có sự chênh lệch khoảng 1%.

**KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

Xếp hạng mì ramen là một chủ đề thú vị và được không ít người quan tâm, nhưng việc xác định xếp hạng của mì ramen là vô cùng khó bởi phụ thuộc vào nhiều yếu tố. Do đó trong đồ án này, em mong muốn xây dựng được mô hình học máy hiệu quả trong việc dự đoán xếp hạng mì ramen. Để thực hiện được điều này, em đã thu thập thành công hơn 4.000 bản ghi dữ liệu từ trang thống kê uy tín TheRamenRater. Sau đó xử lý làm sạch và phân tích dữ liệu, rồi tiến hành xây dựng các mô hình học máy. Nhìn chung, hồi quy logistic luôn là mô hình cho kết quả dự đoán tốt nhất với độ chính xác 64%. Sau đó em sử dụng thêm Grid search thì kết quả mô hình tốt nhất là Rừng ngẫu nhiên với 66%. Con số 66% không phải là con số đáng mong đợi và nó thể hiện rằng em sẽ cần phải làm nhiều việc hơn để cải thiện con số này trong tương lai.

Qua quá trình phân tích dữ liệu, em cũng phát hiện ra những yếu tố nào quan trọng và ảnh hưởng nhiều nhất đến xếp hạng mì ramen. Các yếu tố đó bao gồm: thương hiệu, hương vị, phương thức chế biến, quốc gia xuất xứ. Điều này là phù hợp với thực tế, cho thấy bộ dữ liệu là tin cậy và phản ánh được các yếu tố của sản phẩm mì ramen trong việc định xếp hạng.

Tuy nhiên, trong thực tế còn có nhiều yếu tố khác cũng ảnh hưởng đến xếp hạng mì ramen, như xu hướng tiêu dùng, chiến dịch marketing, v.v. Do đó, trong tương lai em sẽ cố gắng cải tiến đồ án này bằng cách: thêm các nguồn dữ liệu khác để có được thông tin đầy đủ và chi tiết hơn, như các đánh giá từ người tiêu dùng qua các nền tảng và các nguồn khảo sát khác nhau, áp dụng các mô hình học sâu tiên tiến hơn để nâng cao độ chính xác, tìm hiểu sâu về cách tối ưu tham số hiệu quả.

Như vậy, đồ án này đã đạt được các mục tiêu đề ra ban đầu. Tuy nhiên, do thời gian có hạn và kinh nghiệm của bản thân còn hạn chế, nên đồ án không thể tránh khỏi các sai sót trong quá trình thực hiện. Do đó, em rất mong nhận được những ý kiến đóng góp quý báu của quý thầy cô để hoàn thiện hơn đồ án trong tương lai.

Một lần nữa em xin chân thành cảm ơn!

**TÀI LIỆU THAM KHẢO**

|  |  |
| --- | --- |
| [1] | A. Amazon, "What’s the Difference Between Supervised and Unsupervised Learning?," [Online]. Available: https://aws.amazon.com/compare/the-difference-between-machine-learning-supervised-and-unsupervised/?nc1=h\_ls. |
| [2] | B. Vinbigdata, "Supervised Learning và Unsupervised Learning: Khác biệt là gì?," [Online]. Available: https://blog.vinbigdata.org/supervised-learning-va-unsupervised-learning-khac-biet-la-gi/. |
| [3] | P. D. Khanh, "Hồi qui Logistic," [Online]. Available: https://phamdinhkhanh.github.io/deepai-book/ch\_ml/classification.html. |
| [4] | Phamdinhkhanh, "Đánh giá mô hình phân loại trong ML," [Online]. Available: https://phamdinhkhanh.github.io/2020/08/13/ModelMetric.html. |
| [5] | N. d. v. p. t. d. l. S. SDS, "Sách trắng phân tích công nghệ ChatGPT - Phần 1: Chat GPT là gì," [Online]. Available: https://www.samsungsds.com/vn/insights/chatgpt\_whitepaper1.html. |
| [6] | A. HAYES, "Multicollinearity: Meaning, Examples, and FAQs," [Online]. Available: https://www.investopedia.com/terms/m/multicollinearity.asp. |
| [7] | D. D. Hung, "Tiền xử lí dữ liệu văn bản với NLTK," [Online]. Available: https://viblo.asia/p/tien-xu-li-du-lieu-van-ban-voi-nltk-Az45b0LgZxY. |
| [8] | "Streamlit Documentation," [Online]. Available: https://docs.streamlit.io/develop/api-reference/widgets. |
| [9] | Trituenhantao.io, "Cây Quyết Định (Decision Tree)," [Online]. Available: https://trituenhantao.io/kien-thuc/decision-tree/. |
| [10] | T. Nguyễn, "Random Forest algorithm," [Online]. Available: https://machinelearningcoban.com/tabml\_book/ch\_model/random\_forest.html. |
| [11] | V. H. Tiệp, Machine Learning cơ bản, 2018. |