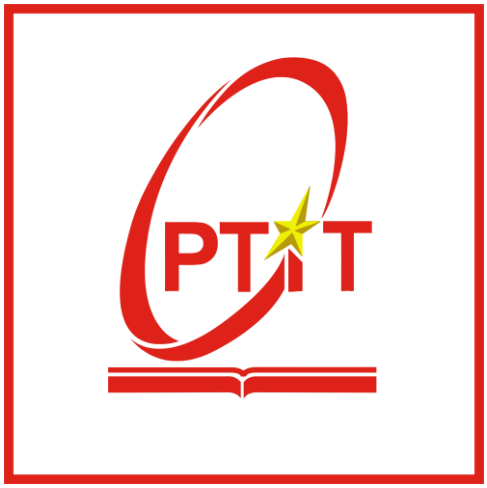
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



Báo cáo hàng tuần

Môn học: Thực tập cơ sở

Giảng viên: Kim Ngọc Bách

Họ tên: Trần Quang Anh

Mã SV: B22DCCN044

Lớp: E22CQCN04-B

Tuần: 3

Báo cáo thực tập cơ sở - Tuần 3: Thuật toán K-Nearest Neighbors

I. Phân tích thuật toán K-Nearest Neighbors (KNN)

1. Khái niệm cơ bản về KNN

K-Nearest Neighbors (KNN) là một thuật toán học máy thuộc nhóm học có giám sát (supervised learning), được sử dụng phổ biến trong các bài toán phân loại (classification) và hồi quy (regression). Đây là một phương pháp phi tham số (non-parametric), nghĩa là nó không giả định trước về phân phối của dữ liệu mà dựa trực tiếp vào chính dữ liệu huấn luyện để đưa ra dự đoán.

Nguyên lý cơ bản của KNN là: "Vật hợp theo loài, người hợp theo nhóm". Khi cần dự đoán một điểm dữ liệu mới, KNN tìm kiếm K điểm dữ liệu gần nhất (neighbors) trong tập huấn luyện dựa trên một thước đo khoảng cách (thường là khoảng cách Euclidean), sau đó đưa ra kết quả dựa trên thông tin của các điểm lân cận này.

Trong bài toán phân loại, mục tiêu là gán một nhãn lớp (label) cho một điểm dữ liệu dựa trên các đặc trưng (features) của nó. Có nhiều thuật toán phân loại, và một trong những cách phổ biến để đưa ra quyết định cuối cùng (khi có nhiều dự đoán hoặc ý kiến) là dựa trên bỏ phiếu từ các dự đoán riêng lẻ.

Trong hồi quy, KNN cũng dùng khái niệm "k láng giềng gần nhất" tương tự như trong phân loại, nhưng thay vì chọn nhãn dựa trên "đa số phiếu" (majority vote), nó lấy trung bình (average) của giá trị các láng giềng để dự đoán.

Sự khác biệt chính:

- Phân loại (classification): Dự đoán nhãn rời rạc (discrete), như "mèo", "chó", "chim".
- Hồi quy (regression): Dự đoán giá trị liên tục (continuous), như 25.5, 100.3, v.v.

KNN thuộc nhóm các mô hình "học lười" (lazy learning), tức là nó không thực sự học hay xây dựng một mô hình trong giai đoạn huấn luyện (training).

Thay vào đó, nó chỉ lưu trữ toàn bộ tập dữ liệu huấn luyện và chờ đến khi cần dự đoán (classification hoặc regression) mới bắt đầu tính toán.

KNN cần giữ toàn bộ tập dữ liệu huấn luyện trong bộ nhớ (memory) để so sánh với dữ liệu mới khi dự đoán.

Do đó, nó được gọi là:

- **Instance-based**: Dự đoán dựa trên các "thực thể" (instances) cụ thể trong dữ liệu huấn luyện.
- **Memory-based**: Phụ thuộc nhiều vào bộ nhớ để lưu trữ dữ liệu.

2. Cách hoạt động của KNN

2.1. Chuẩn bị dữ liệu

- Tập dữ liệu huấn luyện bao gồm các đặc trưng (features) và nhãn (labels) tương ứng.
- Các đặc trưng số (như chiều cao, cân nặng) thường được chuẩn hóa (ví dụ: bằng StandardScaler) để đảm bảo chúng đóng góp công bằng vào tính toán khoảng cách.

2.2. Chọn giá trị K

- K là số lượng điểm lân cận được xem xét, do người dùng chọn trước (siêu tham số - hyperparameter).
- Giá trị K ảnh hưởng trực tiếp đến độ chính xác của mô hình: K nhỏ có thể nhạy cảm với nhiễu, K lớn có thể làm mờ ranh giới giữa các lớp.

2.3. Tính toán khoảng cách

- Với mỗi điểm dữ liệu mới cần dự đoán, KNN tính khoảng cách từ điểm đó đến tất cả các điểm trong tập huấn luyện.
- Euclidean Distance (Khoảng cách Euclidean, p=2): Đây là thước đo khoảng cách phổ biến nhất, tính khoảng cách đường thẳng ngắn nhất giữa hai điểm trong không gian (giống như đo bằng thước kẻ)

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

- Manhattan Distance (Khoảng cách Manhattan, p=1): Còn gọi là "khoảng cách taxi" hoặc "khoảng cách khối thành phố", nó tính tổng giá trị tuyệt đối của chênh lệch giữa các tọa độ. Tên gọi xuất phát từ cách di chuyển trên lưới đường phố (chỉ đi ngang hoặc dọc, không đi chéo).

$$d(p,q) = \sum_{i=1}^{n} |p_i - q_i|$$

- Minkowski Distance (Khoảng cách Minkowski):Đây là dạng tổng quát hóa của cả Euclidean và Manhattan. Tham số ppp trong công thức cho phép điều chỉnh loại khoảng cách:
  - p=2: Trở thành Euclidean.
  - p=1: Trở thành Manhattan.

$$d(p,q) = \left(\sum_{i=1}^{n} |p_i - q_i|^p \right)^{\frac{1}{p}}$$

2.4Xác định K điểm gần nhất:

- Sắp xếp các khoảng cách từ nhỏ đến lớn và chọn K điểm có khoảng cách nhỏ nhất.

2.5Dự đoán:

- Trong bài toán phân loại: Dựa trên "đa số phiếu" (majority vote) của K điểm lân cận, nhãn nào chiếm ưu thế sẽ được chọn.
- Trong bài toán hồi quy: Lấy trung bình (hoặc trung bình có trọng số) của giá trị đầu ra của K điểm lân cận.

3. Ưu điểm và nhược điểm của thuật toán KNN

3.1. Ưu điểm của KNN

- Đơn giản và dễ hiểu: Thuật toán dễ triển khai và không yêu cầu giả định phức tạp về dữ liệu.
- Linh hoạt: Có thể áp dụng cho cả phân loại và hồi quy, đồng thời thích nghi với dữ liệu không tuân theo phân phối cụ thể.
- Không cần huấn luyện nặng: Vì không xây dựng mô hình trước, KNN tiết kiệm thời gian trong giai đoạn huấn luyện.

3.2. Nhược điểm của KNN

- Chi phí tính toán cao: Khi tập dữ liệu lớn, việc tính khoảng cách cho mỗi dự đoán trở nên tốn kém về thời gian và bộ nhớ.

- Nhạy cảm với nhiễu: Nếu dữ liệu có nhiều điểm nhiễu (outliers), kết quả dự đoán có thể bị sai lệch.
- Phụ thuộc vào chuẩn hóa: Nếu các đặc trưng không được chuẩn hóa, những đặc trưng có giá trị lớn sẽ lấn át các đặc trưng khác trong tính toán khoảng cách.
- Lựa chọn K khó khăn: Giá trị K không phù hợp có thể dẫn đến hiện tượng overfitting (K quá nhỏ) hoặc underfitting (K quá lớn).

II. Áp dụng KNN vào hệ thống gợi ý món ăn

1.Biểu Diễn Văn Bản(TfidfVectorizer)

Trong bài toán tìm kiếm món ăn tương tự dựa trên nguyên liệu, việc xử lý dữ liệu đầu vào đóng vai trò quan trọng trong việc đảm bảo hiệu quả của mô hình học máy. Danh sách nguyên liệu (ingredients_list) là dữ liệu văn bản tự do (free-text), không thể sử dụng trực tiếp trong các thuật toán học máy. Vì vậy, cần có một phương pháp để biến đổi văn bản thành các đặc trưng số mà mô hình có thể hiểu được.

Để thực hiện điều này, ta sử dụng TF-IDF Vectorizer, một kỹ thuật mạnh mẽ giúp chuyển đổi văn bản thành các vector số thông qua việc đánh trọng số các từ quan trọng. Điều này giúp giảm nhiễu từ những nguyên liệu phổ biến và nhấn mạnh những nguyên liệu mang tính đặc trưng cao.

1.1. Mục đích

Biến các chuỗi văn bản thành các vector số. So sánh mức độ tường đồng giữa các từ. Làm sạch dữ liệu và chuẩn hóa để chuẩn bị cho các thuật toán học máy

1.2. Cách hoạt động

- TF (Term Frequency): Đo lường số lần một từ xuất hiện trong tài liệu chia cho tổng số từ trong tài liệu.
- IDF (Inverse Document Frequency): Đo lường mức độ quan trọng của từ. Từ càng xuất hiện nhiều trong các tài liệu khác nhau, thì trọng số của nó càng thấp.
- TF-IDF: Kết hợp cả hai yếu tố trên để cho ra một trọng số đại diện cho tầm quan trọng của từ trong tài liệu cụ thể.

1.3. Lý do sử dụng TF-IDF Vectorizer

- Chuyển đổi văn bản thành vector số:
  - Máy tính không thể trực tiếp hiểu được văn bản, nhưng lại làm việc rất tốt với các con số.

- TF-IDF Vectorizer chuyển đổi các nguyên liệu món ăn (dạng văn bản) thành vector đặc trưng số.

- Giảm thiểu ảnh hưởng của từ thông dụng:
  - Trong các danh sách nguyên liệu, một số từ như "salt", "pepper", "water" xuất hiện ở hầu hết các công thức.
  - Nếu bạn sử dụng phương pháp Bag of Words (BoW) hoặc Count Vectorizer, những từ thông dụng này có thể khiến kết quả bị nhiễu.
  - TF-IDF giải quyết vấn đề này bằng cách tính toán trọng số thấp cho các từ phổ biến, vì chúng ít có ý nghĩa trong việc phân biệt giữa các công thức.
- Nhấn mạnh từ đặc trưng:
  - Những từ ít phổ biến nhưng có ý nghĩa phân loại cao như "pork belly", "smoked paprika" sẽ được TF-IDF gán trọng số cao.
  - Điều này giúp mô hình dễ dàng nhận diện đặc điểm nổi bật của mỗi công thức.

1.4 Áp dụng vào nguyên liệu(ingredient_list)

Bước 1: xây dựng từ vựng

- Quét toàn bộ văn bản trong cột ingredient_list
- Tạo từ vựng duy nhất: TfidfVectorizer thu thập tất cả các từ (nguyên liệu) duy nhất xuất hiện trong toàn bộ tập dữ liệu và gán mỗi từ một chỉ số (index). Ví dụ:
  - Từ vựng: {"flour": 0, "sugar": 1, "eggs": 2, "chicken": 3, "rice": 4, "soy sauce": 5}.
  - Mỗi từ (nguyên liệu) tương ứng với một cột trong ma trận kết quả.

Bước 2: Chuyển đổi thành ma trận

- Sau khi có từ vựng, TfidfVectorizer sẽ biểu diễn mỗi công thức dưới dạng một vector số:
  - Số cột của ma trận bằng số lượng từ trong từ vựng (số nguyên liệu duy nhất).
  - Số hàng của ma trận bằng số lượng công thức (số mẫu trong recipe_df).
- Với mỗi công thức:
  - Nếu một nguyên liệu trong từ vựng xuất hiện trong công thức, cột tương ứng sẽ có giá trị là điểm TF-IDF của nguyên liệu đó.
  - Nếu nguyên liệu không xuất hiện, giá trị ở cột đó sẽ là 0.

Bước 3: Tính điểm TF-IDF

- TF (Term Frequency): Tần suất xuất hiện của một nguyên liệu trong công thức (thường được chuẩn hóa theo độ dài của công thức).
- IDF (Inverse Document Frequency): Đo lường mức độ "hiếm" của nguyên liệu trong toàn bộ tập dữ liệu. Nguyên liệu xuất hiện trong nhiều công thức (như "salt") sẽ có IDF thấp, còn nguyên liệu hiếm (như "saffron") sẽ có IDF cao.
- Giá trị TF-IDF = TF × IDF.

Từ đó tạo ra ma trận X_ingredients, trong đó:

- Mỗi hàng là một công thức.
- Mỗi cột là một nguyên liệu trong từ vựng (tất cả nguyên liệu duy nhất trong tập dữ liệu).
- Giá trị là điểm TF-IDF của nguyên liệu đó trong công thức.

2. Chuẩn hóa dữ liệu(StandardScaler)

Bên cạnh đó, các thành phần khác của món ăn được biểu diễn dưới dạng số (ví dụ: lượng calo, chất đạm, chất béo...) có thể có quy mô (scale) khác nhau. Để chuẩn hóa dữ liệu và đảm bảo rằng mọi đặc trưng đều được xử lý công bằng bởi mô hình, ta áp dụng Standard Scaler.

2.1 Mục đích

dùng để chuẩn hóa dữ liệu số (numerical data) bằng cách đưa dữ liệu về cùng một thang đo chuẩn (standard scale)

2.2 Cách hoạt động

StandardScaler chuẩn hóa dữ liệu theo công thức:

$$X_{\text{scaled}} = \frac{X - \mu}{\sigma}$$

Trong đó:

- X là giá trị ban đầu của dữ liệu.
- μ là giá trị trung bình (mean) của đặc trưng đó trong tập dữ liệu.
- σ là độ lệch chuẩn (standard deviation) của đặc trưng đó trong tập dữ liệu.

Sau khi chuẩn hóa:

- Dữ liệu sẽ có giá trị trung bình = 0 và độ lệch chuẩn = 1.
- Tất cả các đặc trưng sẽ nằm trên cùng một thang đo, giúp cải thiện hiệu quả của các mô hình học máy.

2.3Tại sao cần StandardScaler?

2.3.1Đảm bảo công bằng giữa các đặc trưng:

- Các cột dữ liệu số của bạn như: calories, fat, carbohydrates, protein, cholesterol, sodium, fiber có quy mô khác nhau. Ví dụ:
  - calories có thể có giá trị hàng trăm hoặc hàng ngàn.
  - fiber có thể chỉ có giá trị từ 0 đến 10.
- Nếu không chuẩn hóa, các đặc trưng có giá trị lớn sẽ chi phối hoàn toàn mô hình, làm cho các đặc trưng nhỏ hơn trở nên không có ý nghĩa.

2.3.2Cải thiện hiệu quả của các thuật toán học máy:

- Các thuật toán như K-Nearest Neighbors (KNN) sử dụng khoảng cách (distance-based algorithms). Nếu không chuẩn hóa, các đặc trưng lớn sẽ gây sai lệch trong tính toán khoảng cách.
- Một số thuật toán khác như PCA (Principal Component Analysis), SVM (Support Vector Machine), hay Gradient Descent cũng yêu cầu dữ liệu được chuẩn hóa để đạt hiệu quả cao nhất.

2.4 Áp dụng vào các thành phần dinh dưỡng

Các thành phần dinh dưỡng có thang đo rất khác nhau:

- calories và sodium có giá trị lớn (hàng trăm đến hàng nghìn).
- fat, carbohydrates, protein, cholesterol có giá trị trung bình (hàng chục).
- fiber có giá trị nhỏ (dưới 10).
- Nếu không chuẩn hóa, các cột như calories hoặc sodium sẽ chi phối khoảng cách Euclidean trong mô hình KNN, làm giảm tầm quan trọng của các cột như fiber.

Yêu cầu của KNN: KNN dựa trên khoảng cách Euclidean:

$$\text{distance} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Nếu các đặc trưng không cùng thang đo, khoảng cách sẽ bị lệch về phía các đặc trưng có giá trị lớn. Chuẩn hóa đảm bảo rằng tất cả các đặc trưng đóng góp công bằng vào khoảng cách.

3. Huấn luyện mô hình

- Tiền xử lý nguyên liệu: Sử dụng TfidfVectorizer để chuyển recipe_df['ingredients_list'] thành ma trận X_ingredients (vector TF-IDF), với mỗi cột là một nguyên liệu duy nhất.
- Chuẩn hóa thành phần dinh dưỡng: Dùng StandardScaler để chuẩn hóa các cột số (calories, fat, carbohydrates, protein, cholesterol, sodium, fiber) thành X_numerical, đảm bảo trung bình = 0, độ lệch chuẩn = 1.
- Kết hợp đặc trưng: Ghép X_numerical và X_ingredients thành ma trận X_combined bằng np.hstack.
- Huấn luyện KNN: Khởi tạo NearestNeighbors với n_neighbors=3, metric='euclidean', và lưu trữ X_combined để tìm hàng xóm gần nhất.

4. Đưa dữ liệu đầu vào và cho ra kết quả

- Đầu vào: (ví dụ: input_features = [15, 36, 1, 42, 21, 81, 2, 'pork belly, smoked paprika, kosher salt']) (7 giá trị dinh dưỡng + danh sách nguyên liệu).
- Tiền xử lý:
    - Chuẩn hóa 7 giá trị dinh dưỡng bằng scaler.transform thành vector số.
    - Chuyển nguyên liệu thành vector TF-IDF bằng vectorizer.transform.
    - Ghép hai vector thành input_combined.
- Dự đoán: Dùng knn.kneighbors để tính khoảng cách Euclidean từ input_combined đến tất cả công thức trong X_combined, chọn 3 công thức có khoảng cách nhỏ nhất.
- Kết quả: Trả về DataFrame chứa recipe_name, ingredients_list, image_url của 3 công thức gần nhất (ví dụ: "Spicy Pork Stew", "Grilled Pork", "Beef Stir Fry").