BÁO CÁO BÀI TẬP LỚN

MÔN: ĐỒ ÁN CHUYÊN NGÀNH

GVHD: Nguyễn Quốc Huy

# Linear Regression:

## 1. Giới thiệu về Linear Regression

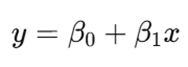
Linear Regression là một phương pháp trong thống kê và học máy được sử dụng để mô hình hóa mối quan hệ giữa một biến phụ thuộc và một hoặc nhiều biến độc lập.

Mô hình hồi quy tuyến tính đơn (Simple Linear Regression) chỉ có một biến độc lập, trong khi mô hình hồi quy tuyến tính đa biến (Multiple Linear Regression) có nhiều biến độc lập.

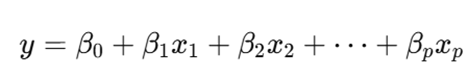
Linear Regression được sử dụng trong nhiều bài toán như dự đoán giá trị, phân tích xu hướng, và tìm hiểu mối quan hệ giữa các yếu tố.

## 2. Công thức của Linear Regression

Đối với Hồi quy tuyến tính đơn, công thức có dạng:



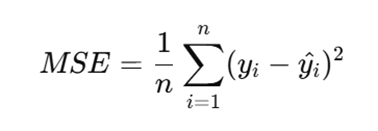
Đối với Hồi quy tuyến tính đa biến, công thức là:



## 3. Cách hoạt động của Linear Regression

Mô hình Linear Regression tìm các giá trị của các hệ số hệ số chặn, hệ số hồi quy sao cho tổng sai số giữa giá trị dự đoán và giá trị thực tế (thực hiện qua một hàm mất mát) là nhỏ nhất.

Hàm mất mát phổ biến trong hồi quy tuyến tính là Mean Squared Error (MSE), được tính theo công thức:



## 4. Các giả định trong Linear Regression

Quan hệ tuyến tính: Mối quan hệ giữa biến phụ thuộc và các biến độc lập là tuyến tính.

Độc lập của sai số: Các sai số phải độc lập với nhau.

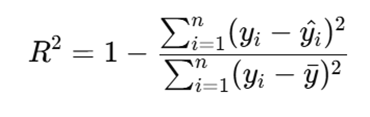
Phân phối chuẩn của sai số: Sai số phải có phân phối chuẩn (hoặc gần chuẩn) với trung bình bằng 0.

Đồng nhất phương sai (Homoscedasticity): Phương sai của sai số là không đổi trên mọi mức độ của biến độc lập.

## 5. Đánh giá hiệu suất của mô hình Linear Regression

Các chỉ số đánh giá mô hình Linear Regression bao gồm:

R-squared (R²): Đại diện cho tỷ lệ phương sai của biến phụ thuộc mà mô hình giải thích. R² có giá trị từ 0 đến 1, với giá trị càng gần 1 cho thấy mô hình càng tốt.



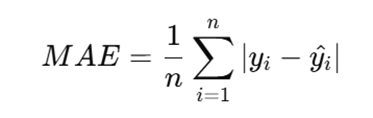
Trong đó:

 là giá trị trung bình của biến phụ thuộc.

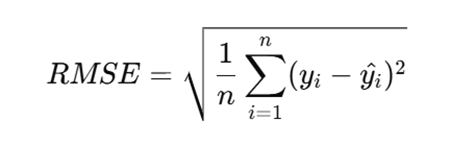
Adjusted R-squared: Điều chỉnh R² khi thêm biến độc lập vào mô hình, giúp đánh giá mô hình chính xác hơn, đặc biệt khi có nhiều biến độc lập.

• p-value: Dùng để kiểm tra ý nghĩa thống kê của các hệ số hồi quy. Nếu p-value nhỏ hơn mức ý nghĩa (thường là 0.05), hệ số hồi quy có ý nghĩa thống kê.

• Mean Absolute Error (MAE): Trung bình giá trị tuyệt đối của sai số.



Root Mean Squared Error (RMSE): Căn bậc hai của MSE.



## 6. Các bước thực hiện:

### 6.1. Chuẩn bị Dữ liệu:

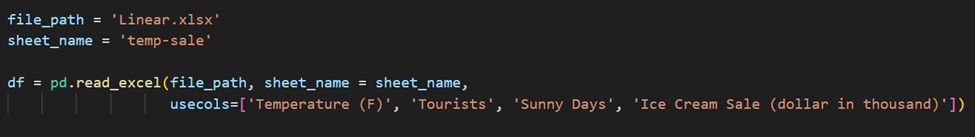
Dữ liệu được lấy từ một file Excel với tên là Linear.xlsx và chứa thông tin trên sheet temp-sale. Các cột dữ liệu được sử dụng trong mô hình bao gồm:

Temperature (F): Nhiệt độ (Fahrenheit)

Tourists: Số lượng khách du lịch

Sunny Days: Số ngày nắng

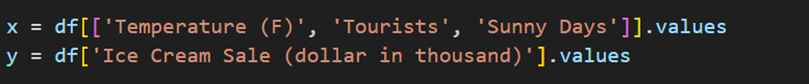
Ice Cream Sale (dollar in thousand): Doanh thu bán kem (tính bằng nghìn đô la)



### 6.2. Tạo Ma Trận X và Y

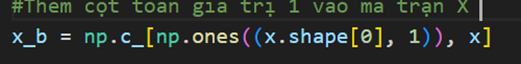
X (Đặc trưng): Chứa ba biến đầu vào: Nhiệt độ, Số lượng khách du lịch và Số ngày nắng.

Y (Biến mục tiêu): Chứa giá trị doanh thu bán kem.



### 6.3. Thêm Cột 1 vào Ma Trận X (Để tính toán hệ số chặn)

Để thực hiện hồi quy tuyến tính, cần phải thêm một cột toàn giá trị 1 vào ma trận X, giúp tính toán hệ số chặn (intercept).



### 6.4. Tính Toán Các Hệ Số Hồi Quy

Các hệ số hồi quy được tính bằng công thức:

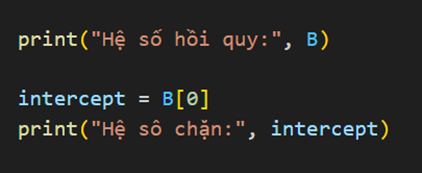


Trong đó, B là vector chứa các hệ số hồi quy, bao gồm cả hệ số chặn.

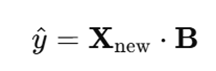


### 6.5. In Kết Quả

Sau khi tính toán các hệ số hồi quy, kết quả được in ra:



### 6.6. Dự Đoán Doanh Thu Bán Kem

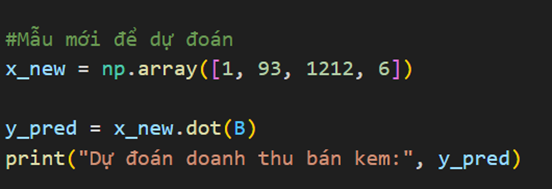


Sử dụng mô hình hồi quy để dự đoán doanh thu bán kem với các giá trị đầu vào mới:

Nhiệt độ: 93°F

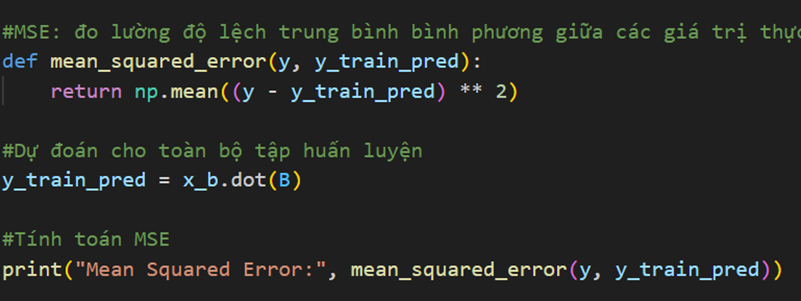
Số lượng khách du lịch: 1212

Số ngày nắng: 6



### 6.7. Tính Toán Sai Số Trung Bình Bình Phương (MSE)

Để đánh giá chất lượng của mô hình, ta tính toán Mean Squared Error (MSE), một chỉ số đo độ lệch trung bình bình phương giữa các giá trị thực tế và các giá trị dự đoán.



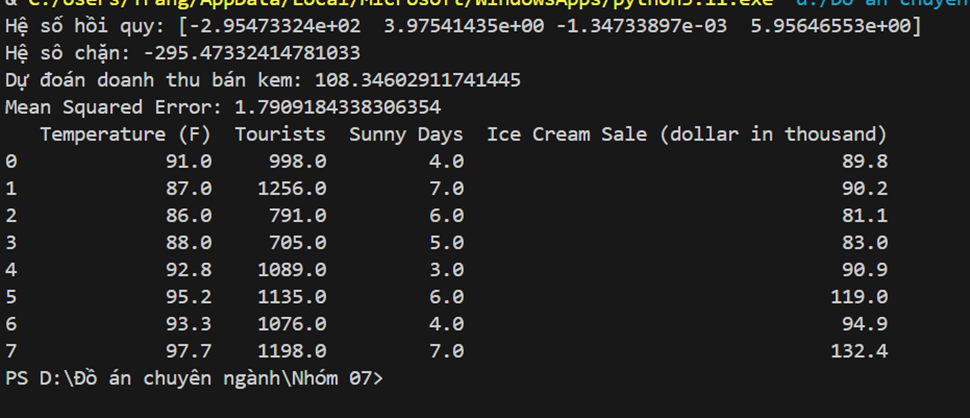
### 6.8. Kết Quả Cuối Cùng

Sau khi tính toán, mã in ra các kết quả sau:

Hệ số hồi quy và hệ số chặn

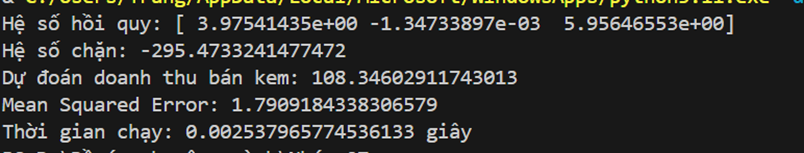
Dự đoán doanh thu bán kem cho bộ dữ liệu mới

MSE (Sai số trung bình bình phương)

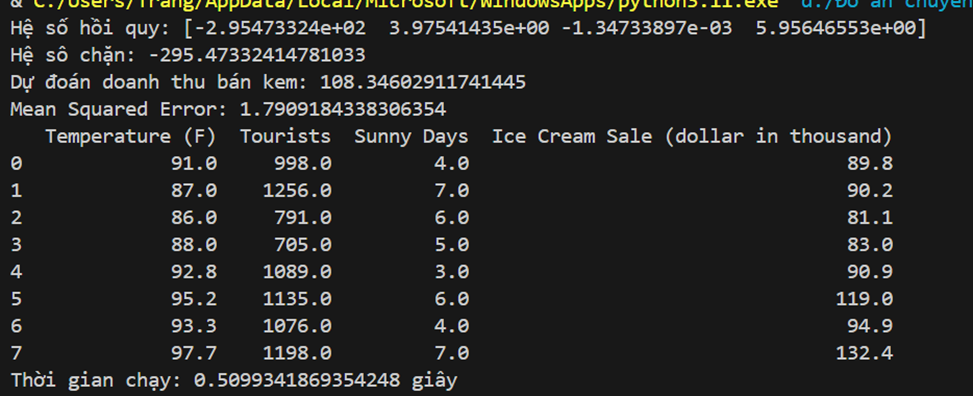


### 6.9. So sánh với Linear Regression khi sử dụng thư viện:

a) Khi sử dụng thư viện sklearn (hàm LinearRegression):



b) Khi tính toán thủ công:



So sánh Độ chính xác và Thời gian chạy:

Độ chính xác (Mean Squared Error):

Cả hai đoạn mã đều cho kết quả MSE gần như giống nhau (1.79), cho thấy độ chính xác của mô hình tương đương nhau.

Điều này có nghĩa là mô hình của bạn có thể dự đoán khá chính xác đối với dữ liệu huấn luyện, bất kể bạn sử dụng phương pháp tính toán thủ công hay thư viện sklearn.

Thời gian chạy:

Thời gian chạy của đoạn mã sử dụng thư viện sklearn (0.0049 giây) nhanh hơn rất nhiều so với phương pháp thủ công (0.5099 giây).

Điều này là do thư viện sklearn tối ưu hóa thuật toán và sử dụng các phương pháp tính toán hiệu quả hơn so với việc tính toán bằng tay với các phép toán ma trận phức tạp.

Kết luận:

Độ chính xác: Các mô hình của cả hai phương pháp đều cho kết quả tương tự, không có sự chênh lệch đáng kể về độ chính xác.

Thời gian chạy: Sử dụng thư viện sklearn có thời gian chạy nhanh hơn rất nhiều so với việc tính toán thủ công. Điều này làm cho phương pháp sử dụng thư viện trở nên hiệu quả hơn, đặc biệt khi làm việc với các tập dữ liệu lớn hoặc khi yêu cầu tốc độ tính toán cao.

## 7. Ứng dụng của Linear Regression

Dự đoán giá trị: Dự đoán các giá trị liên quan như giá nhà, doanh thu, lượng tiêu thụ sản phẩm dựa trên các yếu tố độc lập.

Phân tích mối quan hệ: Tìm hiểu và phân tích mối quan hệ giữa các biến (ví dụ: sự ảnh hưởng của quảng cáo đến doanh thu).

Tối ưu hóa: Đưa ra các quyết định dựa trên mô hình hồi quy để tối ưu hóa các yếu tố trong kinh doanh, sản xuất.

## 8. Các vấn đề và hạn chế của Linear Regression

Không phù hợp với dữ liệu phi tuyến tính: Linear Regression yêu cầu mối quan hệ tuyến tính giữa các biến, do đó không thể áp dụng cho các dữ liệu có quan hệ phi tuyến.

Nhạy cảm với nhiễu: Linear Regression dễ bị ảnh hưởng bởi các giá trị ngoại lệ hoặc nhiễu (outliers).

Multicollinearity: Khi có mối quan hệ mạnh giữa các biến độc lập, mô hình có thể trở nên không ổn định và không đáng tin cậy.

# Logistic Regression:

## 1. Giới thiệu về Logistic Regression

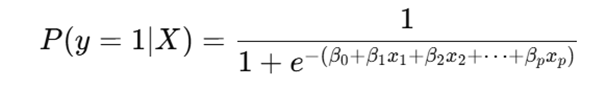
Logistic Regression là một phương pháp học máy được sử dụng để mô hình hóa xác suất của một sự kiện nào đó xảy ra, dựa trên các đặc trưng đầu vào. Đây là một kỹ thuật phân loại, thường được sử dụng cho các bài toán phân loại nhị phân (binary classification), chẳng hạn như dự đoán spam email (có hay không), dự đoán bệnh (có bệnh hay không),...

Mặc dù tên gọi có từ "hồi quy", Logistic Regression thực sự là một mô hình phân loại chứ không phải mô hình hồi quy như hồi quy tuyến tính.

## 2. Công thức của Logistic Regression

Mô hình Logistic Regression sử dụng hàm sigmoid để dự đoán xác suất.

Công thức mô hình có dạng:



Hàm sigmoid (logistic function):



Hàm này đưa ra giá trị trong khoảng (0,1), được dùng để biểu diễn xác suất.

## 3. Các bước thực hiện:

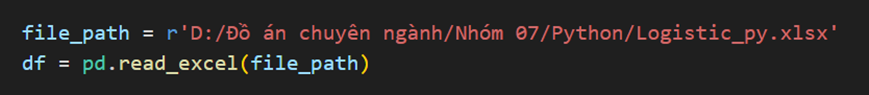
Mục tiêu:

Đoạn mã này thực hiện việc tính toán các chỉ số liên quan đến mô hình hồi quy logistic cho một tập dữ liệu chứa thông tin về các gene của bệnh nhân và kết quả thực tế của bệnh nhân. Cụ thể, mô hình này tính toán giá trị z, xác suất P(1), dự đoán kết quả, độ lệch giữa kết quả thực tế và dự đoán, xác suất likelihood, và cuối cùng là log loss (một chỉ số đánh giá độ chính xác của mô hình).

### 3.1. Đọc dữ liệu từ file Excel:

Dữ liệu được đọc từ file Excel có đường dẫn 'D:/Đồ án chuyên ngành/Nhóm 07/Python/Logistic\_py.xlsx' vào một DataFrame bằng thư viện pandas.

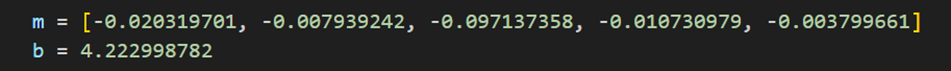
Các cột trong dữ liệu bao gồm thông tin về các gene (gene1 đến gene5) và kết quả thực tế của bệnh nhân (actual\_outcome).



### 3.2. Khởi tạo các tham số mô hình:

m: Là danh sách các trọng số của từng gene (5 gene). Đây là các hệ số của mô hình logistic đã được huấn luyện từ trước.

b: Là hệ số chặn (intercept), cũng được huấn luyện từ trước.

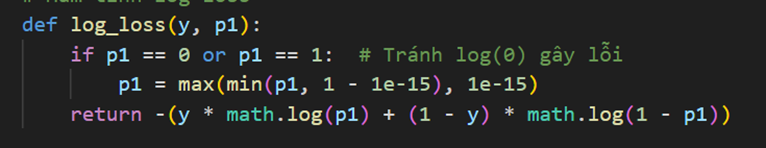


### 3.3. Hàm log\_loss:

Hàm log\_loss tính độ mất mát logarit (log loss), một chỉ số quan trọng trong mô hình phân loại nhị phân. Log loss đo lường sự khác biệt giữa xác suất dự đoán của mô hình và giá trị thực tế.

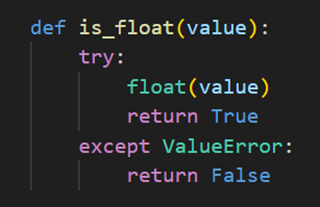
Công thức tính log loss:





### 3.4. Kiểm tra giá trị dữ liệu:

Hàm is\_float được sử dụng để kiểm tra xem một giá trị có phải là số thực hay không. Nếu không phải, giá trị sẽ được gán bằng 0.0 để tránh lỗi khi xử lý.



### 3.5. Duyệt qua các bệnh nhân và tính toán:

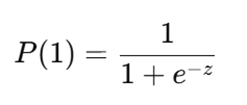
Đoạn mã duyệt qua từng hàng trong DataFrame, lấy thông tin gene của bệnh nhân và kết quả thực tế (actual\_outcome).

Các giá trị gene (gene1 đến gene5) được chuyển đổi sang kiểu số thực nếu có thể. Nếu không, giá trị sẽ được gán là 0.

Giá trị z được tính bằng tổng có trọng số của các gene cộng với hệ số chặn:



Xác suất P(1) được tính bằng công thức sigmoid:



Kết quả dự đoán sẽ là 1 nếu P(1)≥0.5 và là 0 nếu P(1)<0.5.

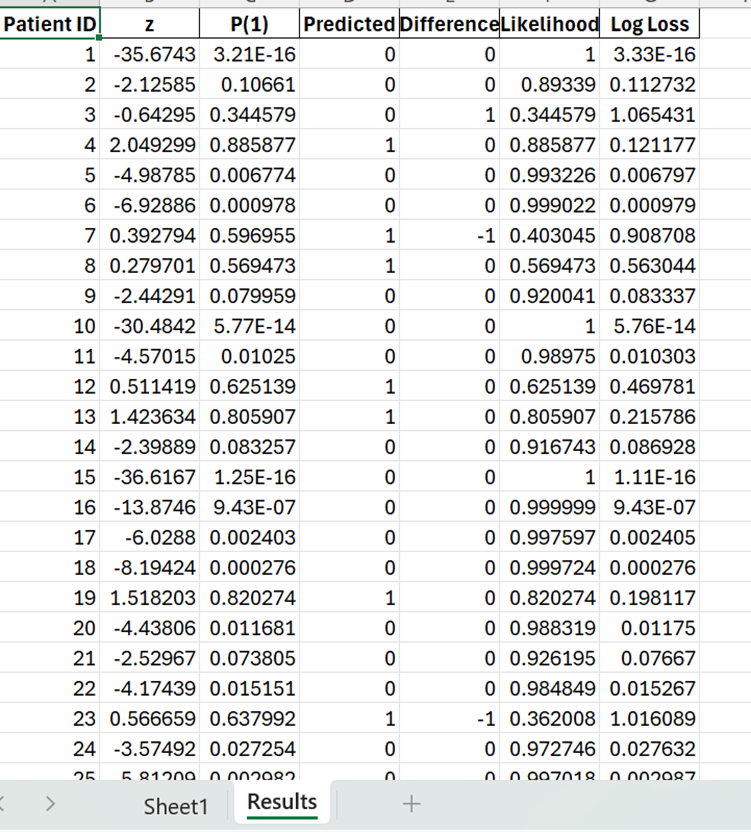
Sau đó, tính sự khác biệt giữa kết quả thực tế và dự đoán, xác suất likelihood, và cuối cùng là log loss.



### 3.6. Lưu kết quả vào Excel:

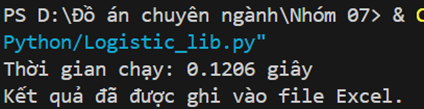
Các kết quả tính toán (bao gồm Patient ID, giá trị z, xác suất P(1), dự đoán, sự khác biệt, likelihood, và log loss) được lưu vào một danh sách results.

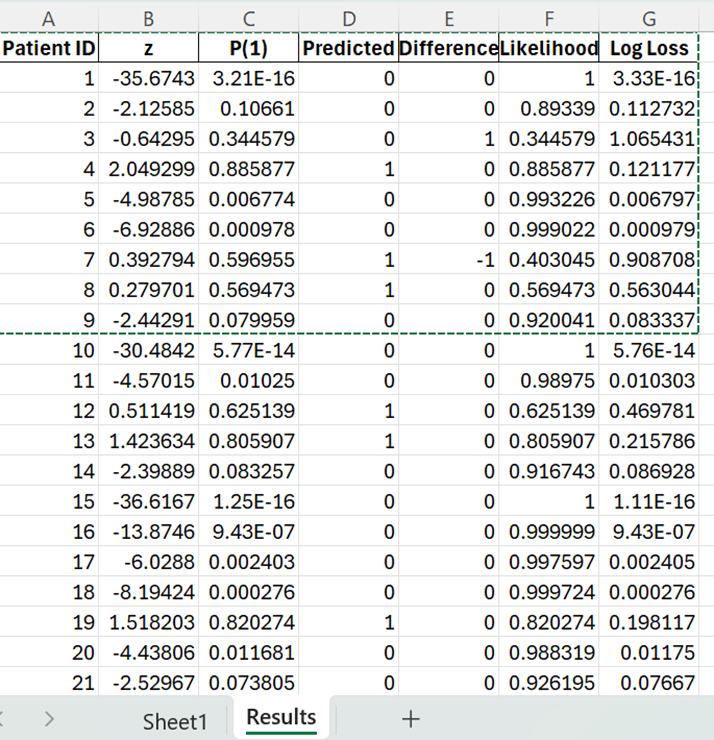
Danh sách này được chuyển thành một DataFrame và ghi vào một sheet mới trong file Excel.



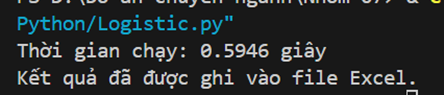
## 4. So sánh Độ chính xác và Thời gian chạy:

### a) Khi dùng thư viện:





### b) Khi chạy bằng thủ công:



Nhận xét:

1. So sánh Thời gian Chạy

Thời gian khi sử dụng thư viện: 0.1206 giây

Thời gian khi code thủ công: 0.5946 giây

Rõ ràng là sử dụng thư viện nhanh hơn so với code thủ công, với thời gian chạy gần như ngắn hơn khoảng 5 lần.

2. So sánh Độ chính xác

Cả hai phương pháp đều cho ra cùng một kết quả trong các cột P(1), Predicted, Likelihood, và Log Loss. Điều này có nghĩa là độ chính xác của cả hai phương pháp không có sự khác biệt.

Kết luận

Thời gian chạy: Sử dụng thư viện có lợi thế rõ ràng về tốc độ, giúp tiết kiệm thời gian khi xử lý một lượng lớn dữ liệu.

Độ chính xác: Cả hai phương pháp đều chính xác như nhau, vì vậy không cần lo lắng về sự khác biệt kết quả.

3. Các giả định trong Logistic Regression:

Quan hệ tuyến tính giữa biến độc lập và logit của xác suất: Trong Logistic Regression, mối quan hệ giữa các biến độc lập và logit của xác suất phải là tuyến tính, chứ không phải mối quan hệ tuyến tính với xác suất trực tiếp.

Không có đa cộng tuyến (Multicollinearity): Các biến độc lập không được phép có mối quan hệ tuyến tính mạnh mẽ với nhau.

4. Ứng dụng của Logistic Regression

Phân loại nhị phân: Logistic Regression được sử dụng cho các bài toán phân loại nhị phân như phân loại email spam, phân loại bệnh, dự đoán khả năng mua hàng, v.v.

Phân tích nguy cơ: Dự đoán xác suất một sự kiện xảy ra, ví dụ như khả năng một khách hàng sẽ rời bỏ dịch vụ hoặc một bệnh nhân có nguy cơ mắc bệnh.

Tối ưu hóa chiến lược marketing: Dự đoán sự thành công của chiến dịch quảng cáo, xác định các nhóm khách hàng mục tiêu.

5. Các vấn đề và hạn chế của Logistic Regression

Chỉ áp dụng cho phân loại nhị phân: Logistic Regression không thể trực tiếp áp dụng cho các bài toán phân loại đa lớp. Tuy nhiên, có thể mở rộng thành Multinomial Logistic Regression cho các bài toán phân loại nhiều lớp.

Giả định tuyến tính: Mô hình giả định mối quan hệ tuyến tính giữa các biến độc lập và logit của xác suất, điều này có thể không đúng trong nhiều trường hợp.

Dễ bị ảnh hưởng bởi outliers: Logistic Regression có thể bị ảnh hưởng mạnh bởi các điểm dữ liệu ngoại lệ.

# Naïve-Bayes:

## Cơ sở lý thuyết của thuật toán Naive Bayes

### 1.1 Nguyên lý Naive Bayes

Nguyên lý Naive Bayes là cơ sở của thuật toán Naive Bayes, một phương pháp phân loại dựa trên Định lý Bayes. Nguyên lý này giúp dự đoán xác suất của một đối tượng thuộc về một lớp nào đó, dựa trên các đặc trưng của đối tượng. Tên "Naive" (ngây thơ) xuất phát từ giả định rằng các đặc trưng là độc lập có điều kiện với nhau khi biết lớp của đối tượng. Dù giả định này không luôn chính xác trong thực tế, nó giúp đơn giản hóa tính toán và tạo ra một mô hình phân loại mạnh mẽ, dễ hiểu.

### 1.2 Giả định độc lập

Giả định độc lập trong Naive Bayes là một giả định quan trọng giúp đơn giản hóa quá trình tính toán xác suất trong thuật toán Naive Bayes. Giả định này cho rằng các đặc trưng (feature) của đối tượng là độc lập với nhau khi biết lớp (class) của đối tượng đó. Điều này có nghĩa là, nếu biết đối tượng thuộc về một lớp cụ thể, thì mỗi đặc trưng của nó sẽ không ảnh hưởng đến các đặc trưng khác.

Giả định độc lập trong Naive Bayes được gọi là giả định độc lập có điều kiện. Giả định này nói rằng, với mỗi cặp đặc trưng xi và xj, xác suất P (xi∣C) sẽ không bị ảnh hưởng bởi xj khi biết lớp C. Hay nói cách khác, xác suất có điều kiện của tập hợp đặc trưng X=(x1, x2, …, xn) có thể được biểu diễn như tích của từng xác suất riêng rẽ:

P(X∣C) = P (x1∣C) . P(x2∣C) ... P(xn∣C)

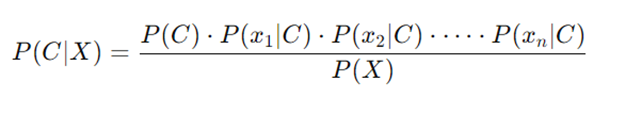
Điều này giúp đơn giản hóa đáng kể công thức Bayes, vốn tính toán xác suất cho một lớp với một tập hợp đặc trưng phức tạp. Nhờ giả định độc lập, việc tính toán này chỉ còn là tính xác suất của từng đặc trưng riêng rẽ cho mỗi lớp, rồi nhân các xác suất đó lại với nhau.

Ý nghĩa: Giả định này giúp Naive Bayes trở thành một thuật toán đơn giản, nhanh chóng, có thể hoạt động tốt trên các tập dữ liệu có nhiều đặc trưng mà không cần phải tính toán mối quan hệ phức tạp giữa các đặc trưng.

Hạn chế: Trong thực tế, các đặc trưng không hoàn toàn độc lập với nhau. Ví dụ, trong dữ liệu y khoa, các triệu chứng có thể có mối liên hệ với nhau, và trong ngôn ngữ tự nhiên, các từ có mối quan hệ ngữ cảnh. Vì vậy, giả định này có thể dẫn đến mô hình không chính xác trong một số trường hợp.

### 1.3 Phương trình Naive Bayes

Phương trình Naive Bayes được sử dụng để tính xác suất có điều kiện của một lớp (class) C cho trước một tập các đặc trưng (features) X=(x1, x2, …, xn). Cụ thể, phương trình này tính xác suất mà một đối tượng thuộc về lớp C khi biết các đặc trưng của nó, dựa trên Định lý Bayes. Công thức Naive Bayes được biểu diễn như sau:



Trong đó:

P(C∣X): Xác suất có điều kiện mà đối tượng thuộc về lớp C khi đã biết các đặc trưng X.

P(C): Xác suất tiên nghiệm của lớp C (tức là xác suất mà một đối tượng thuộc lớp C trước khi xem xét các đặc trưng).

P(xi∣C): Xác suất có điều kiện của từng đặc trưng xi khi biết đối tượng thuộc lớp C. Theo giả định độc lập, các xác suất này có thể tính riêng lẻ và nhân lại với nhau.

P(X): Xác suất của tập hợp đặc trưng X, là một hằng số cho tất cả các lớp và không nhất thiết phải tính trực tiếp vì nó chỉ là một hệ số chuẩn hóa.

Phương trình Naive Bayes không cần P(X) trong phân loại vì P(X) là hằng số cho mọi lớp, nên trong phân loại Naive Bayes, chúng ta thường chỉ cần tính tử số:

P(C∣X) ∝ P(C) . P(x∣C) . P(x2∣C)….P(xn∣C).

Khi thực hiện phân loại, ta chỉ cần tính P(C) . P(x1∣C) . P(x2∣C)……P(xn∣C) cho từng lớp C và chọn lớp có giá trị cao nhất làm kết quả phân loại.

Ví dụ: Trong file excel Naïve-Bayes-9-a, nói về Lung Cancer, ở đây Lung Cancer là thuộc tình chứa các class C, những thuộc tình Size, Thickness và Number chứa các đặc trưng X.

## Các bước thực hiện thuật toán Naive Bayes

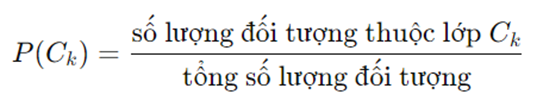
### 2.1 Ước lượng xác suất tiên nghiệm (prior probability)

Mục tiêu: Tính toán xác suất tiên nghiệm (prior probability) cho mỗi lớp Ck trong tập dữ liệu.

Cách thực hiện:

Đếm số lượng đối tượng thuộc từng lớp trong tập huấn luyện.

Tính xác suất tiên nghiệm của mỗi lớp Ck bằng công thức:



Ý nghĩa: Xác suất tiên nghiệm cho biết tần suất xảy ra của mỗi lớp trước khi xem xét đến các đặc trưng.

### 2.2 Ước lượng xác suất có điều kiện

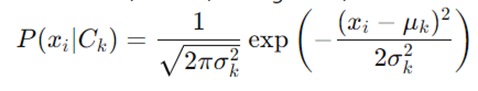
Mục tiêu: Tính xác suất có điều kiện P(xi∣Ck) cho từng đặc trưng xi của đối tượng, khi đối tượng thuộc lớp Ck.

Cách thực hiện:

Với từng đặc trưng xi và từng lớp Ck, tính xác suất P(xi∣Ck) dựa trên tần suất xuất hiện của xi trong các đối tượng thuộc lớp Ck.

Với các đặc trưng liên tục, thường sử dụng phân phối Gaussian (chuẩn) để ước lượng xác suất có điều kiện.

Công thức cho xác suất có điều kiện khi đặc trưng liên tục:



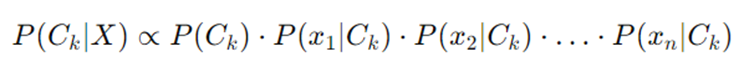
Trong đó, μk là trung bình và σk là độ lệch chuẩn của đặc trưng xi trong lớp Ck.

### 2.3 Phân loại đối tượng mới

Mục tiêu: Sử dụng các xác suất đã ước lượng để dự đoán lớp của một đối tượng mới với các đặc trưng X={x1, x2, ...,xn}.

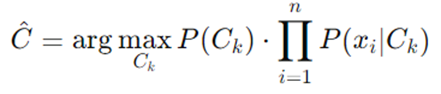
Cách thực hiện:

Tính xác suất có điều kiện của từng lớp Ck dựa trên các đặc trưng của đối tượng mới:



So sánh giá trị P(Ck∣X) cho các lớp Ck.

Lớp nào có xác suất cao nhất thì đối tượng mới sẽ được phân loại vào lớp đó:



## Thực nghiệm và đánh giá hiệu suất

### 3.1 Chuẩn bị dữ liệu

Bước đầu tiên trong quy trình thực hiện là chuẩn bị dữ liệu từ file Excel chứa thông tin của 1.978 mẫu về các yếu tố liên quan đến ung thư phổi. Bộ dữ liệu bao gồm các biến sau:

Hotspot-size: kích thước của điểm nóng (gồm các giá trị ex-large, large, medium, small, none).

Spot-number: số lượng điểm nóng (với các mức scale-1, scale-2, scale-3, scale-4, scale-5).

Thickness: độ dày của vết phát hiện (với các giá trị light, grey, dark).

Lung Cancer: mức độ ung thư phổi (negative, stage-1, stage-2, stage-3).

Để đảm bảo dữ liệu sẵn sàng cho việc triển khai mô hình, các bước tiền xử lý được thực hiện như sau:

Xử lý dữ liệu bị thiếu: Kiểm tra và loại bỏ hoặc thay thế các giá trị thiếu để đảm bảo độ chính xác của mô hình.

Chia tập dữ liệu: Bộ dữ liệu được chia thành hai tập:

Tập huấn luyện (chiếm 70-80%) tổng số mẫu để huấn luyện mô hình.

Tập kiểm tra (20-30% còn lại) để đánh giá mô hình khi huấn luyện.

### 3.2 Triển khai Naive Bayes

Sau khi hoàn thành việc chuẩn bị dữ liệu, mô hình Naive Bayes trên Excel sẽ được triển khai dựa trên các bước sau:

1. **Xác suất tiên nghiệm của từng lớp (Lung Cancer):**

- Tính xác suất tiên nghiệm cho mỗi lớp Lung Cancer (negative, stage-1, stage-2, stage-3) dựa vào số lượng mẫu thuộc lớp đó chia cho tổng số mẫu.

- Công thức excel: =COUNTIF(range\_class, class\_value) / COUNTA(range\_class)

1. **Xác suất có điều kiện cho từng đặc trưng (Hotspot-size, Spot-number, Thickness)**

- Với mỗi đặc trưng, tính xác suất có điều kiện dựa vào số lần xuất hiện của một giá trị nhất định khi đã biết đối tượng thuộc một lớp cụ thể. Ví dụ: Size = large có Lung Cancer= negative

- Công thức excel: =COUNTIFS(range\_class, class\_value, range\_feature, feature\_value) / COUNTIF(range\_class, class\_value)

1. **Tính toán xác suất cuối cùng cho từng lớp**

- Sử dụng công thức Naive Bayes để tính xác suất đối với mỗi lớp. Nếu các đặc trưng đã cho là Hotspot-size, Spot-number, và Thickness, thì xác suất để thuộc về mỗi lớp sẽ là tích của các xác suất có điều kiện của từng đặc trưng t với xác suất tiên nghiệm P’(t|C). Xác suất có điều kiện của từng lớp C đối với sự kiện t đã xảy ra P(C|t) sẽ là tích của các xác suất có điều kiện của từng đặc trưng với xác suất tiên nghiệm chia cho xác suất của tập hợp đặc trưng t là tổng của các xác suất có điều kiện P’(t|C).

- Công thức Excel:

= prior\_prob \* conditional\_prob1 \* conditional\_prob2 \* conditional\_prob3

= P’(t|C) / P(t)

1. **Phân loại đối tượng mới**

- Sau khi tính các xác suất cho từng lớp, chọn lớp có xác suất cao nhất làm kết quả phân loại.

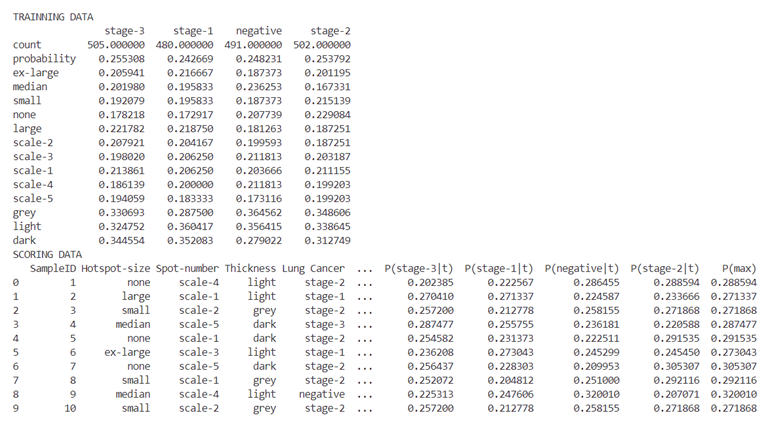
- Trong Excel, có thể dùng công thức MAX để tìm xác suất lớn nhất và hàm IF hoặc INDEX/MATCH để xác định lớp tương ứng.

### 3.3 Đánh giá mô hình

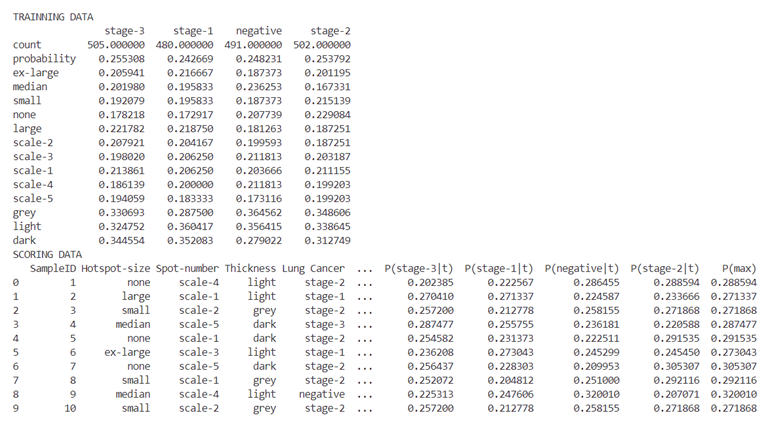
Để đánh giá độ hiệu quả của mô hình, chúng tôi sử dụng các chỉ số đánh giá tiêu chuẩn trong phân loại như sau:

1. **Tính các chỉ số đánh giá:**

- Độ chính xác (Accuracy): Tính phần trăm dự đoán đúng trên tổng số mẫu trong tập kiểm tra. Kết quả từ code Python và Excel đưa ra gần như là giống nhau.



Hình 1.3.3.1. Kết quả in ra của Python



Hình 1.3.3.2. Kết quả của Excel

1. **Phân tích kết quả:**

- Sử dụng biểu đồ hoặc ma trận nhầm lẫn (Confusion Matrix) để trực quan hóa kết quả phân loại chi tiết, xác định những trường hợp mà mô hình dự đoán sai.

- Phân tích các lỗi phân loại để hiểu rõ hơn về các hạn chế của mô hình và có thể đưa ra phương án cải tiến trong tương lai.

### 3.4 So sánh mô hình Naive Bayes sử dụng thư viện với mô hình Naive Bayes không sử dụng thư viện

|  | Mô hình Naive Bayes sử dụng thư viện | Mô hình Naive Bayes không sử dụng thư viện |
| --- | --- | --- |
| Thời gian thực hiện phép tính |  | 0.03283548355102539 |
| Độ chính xác |  |  |
|  |  |  |

# Pre-processing Data:

## 1. Giới thiệu về Tiền xử lý Dữ liệu (Pre-processing Data)

Tiền xử lý dữ liệu là một bước quan trọng trong quy trình phân tích dữ liệu, giúp cải thiện chất lượng của dữ liệu và tối ưu hóa hiệu suất của các mô hình học máy. Mục tiêu chính của tiền xử lý là biến đổi dữ liệu thô thành dạng dễ sử dụng và phù hợp cho các thuật toán học máy. Việc làm sạch và chuyển đổi dữ liệu đúng cách có thể cải thiện độ chính xác của mô hình phân tích và giảm thiểu các lỗi.

## 2. Các bước thực hiện:

Mục tiêu

Đoạn mã Python này thực hiện các bước tiền xử lý dữ liệu từ một tệp Excel. Mục tiêu chính bao gồm:

+ Đọc và làm sạch dữ liệu

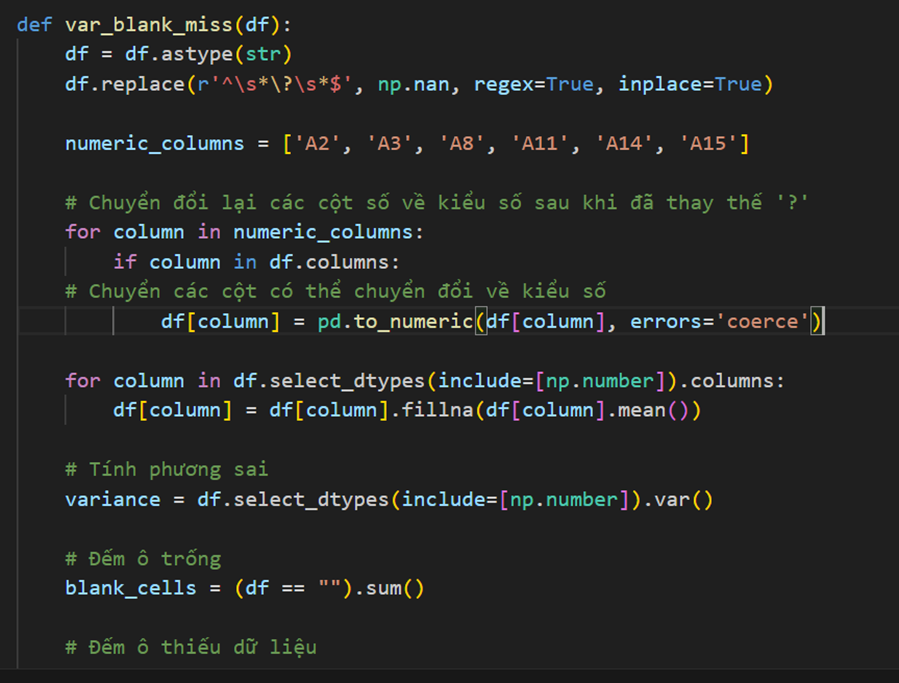
+ Xử lý các giá trị thiếu

+ Chuẩn hóa và tích hợp dữ liệu

+ Chuẩn hóa dữ liệu để đảm bảo dữ liệu nằm trong các khoảng quy chuẩn

+ Lưu dữ liệu sau khi xử lý vào một tệp Excel mới

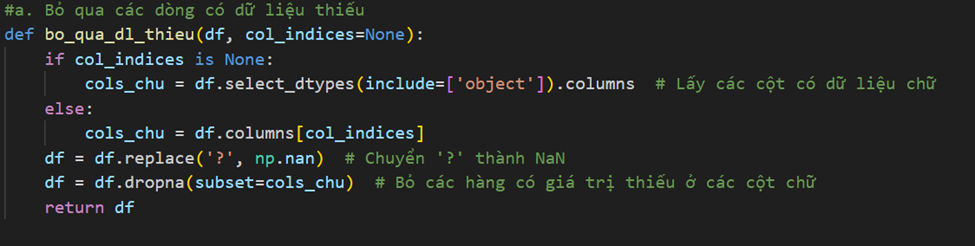
### 2.1 Tính toán và liệt kê Phương sai, Ô trống và Ô bị thiếu dữ liệu trong bảng:



### 2.2. bo\_qua\_dl\_thieu(df, col\_indices=None)

Mục đích: Bỏ qua các dòng có dữ liệu thiếu trong các cột chỉ định.

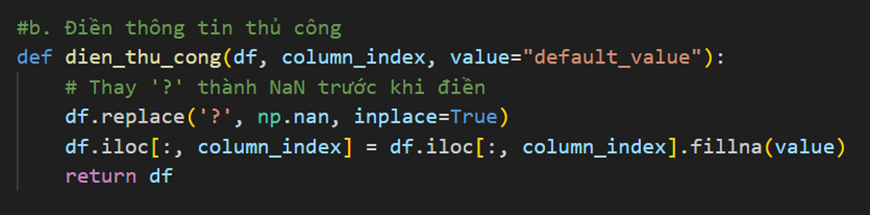
Chi tiết: Chuyển "?" thành NaN và loại bỏ các dòng có dữ liệu thiếu.



### 2.3. dien\_thu\_cong(df, column\_index, value="default\_value")

Mục đích: Điền giá trị thủ công cho các ô dữ liệu thiếu.

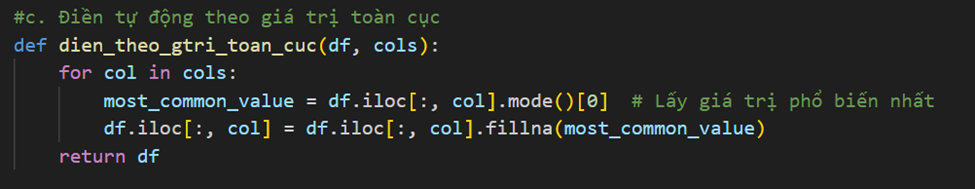
Chi tiết: Thay thế "?" bằng NaN và điền giá trị mặc định vào các ô trống trong cột chỉ định.



### 2.4. dien\_theo\_gtri\_toan\_cuc(df, cols)

Mục đích: Điền giá trị thiếu bằng giá trị phổ biến nhất.

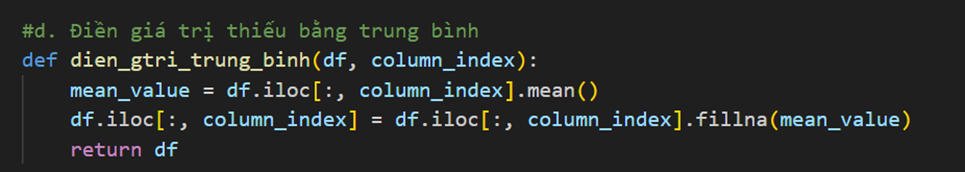
Chi tiết: Tìm giá trị xuất hiện nhiều nhất trong cột chỉ định và điền vào các ô trống.



### 2.5. dien\_gtri\_trung\_binh(df, column\_index)

Mục đích: Điền giá trị thiếu bằng giá trị trung bình của cột.

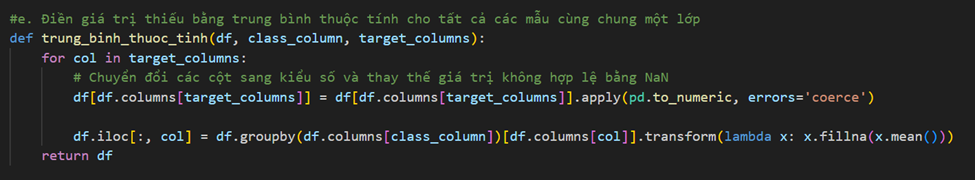
Chi tiết: Tính giá trị trung bình và điền vào các ô trống trong cột chỉ định.



### 2.6. trung\_binh\_thuoc\_tinh(df, class\_column, target\_columns)

Mục đích: Điền giá trị thiếu theo giá trị trung bình của các mẫu thuộc cùng một lớp.

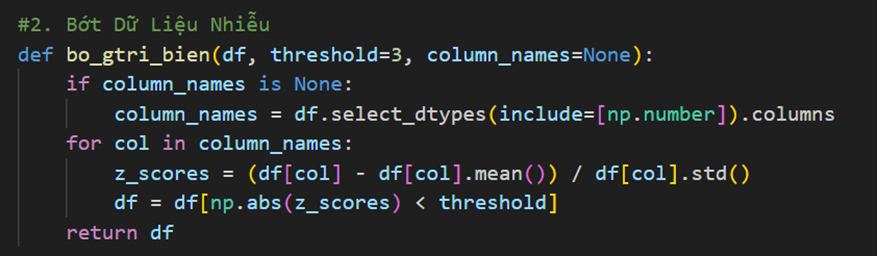
Chi tiết: Dựa vào cột lớp, tính trung bình cho từng lớp và điền vào các ô trống trong cột mục tiêu.



### 2.7. bo\_gtri\_bien(df, threshold=3, column\_names=None)

Mục đích: Bỏ các giá trị ngoại biên trong dữ liệu.

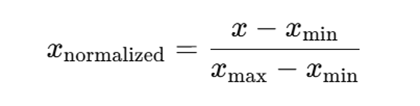
Chi tiết: Tính điểm Z của các giá trị, sau đó loại bỏ các giá trị có điểm Z vượt ngưỡng cho phép.

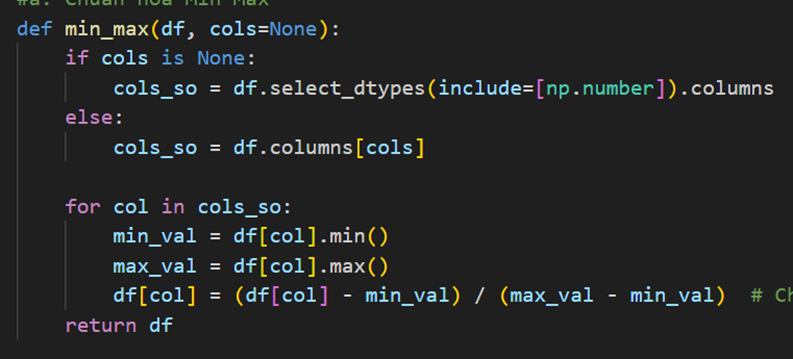


## 3. Chuẩn Hóa Dữ Liệu

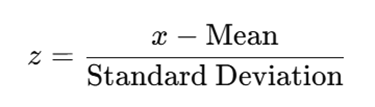
### 3.1. Min-Max Normalization (min\_max(df, cols=None)):

- Chuẩn hóa dữ liệu vào khoảng [0, 1] theo công thức:

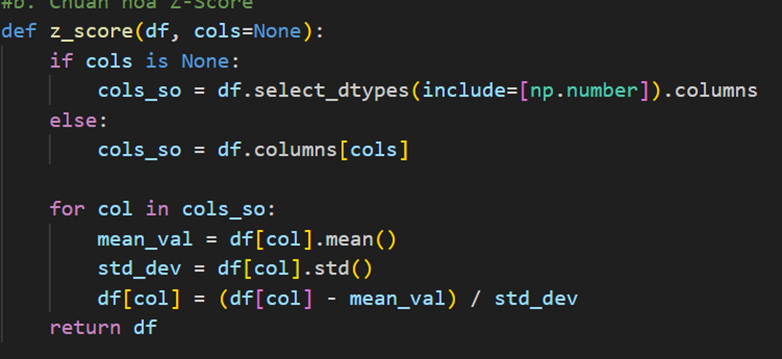




### 3.2. Z-Score Normalization (Chuẩn hóa Z-Score):

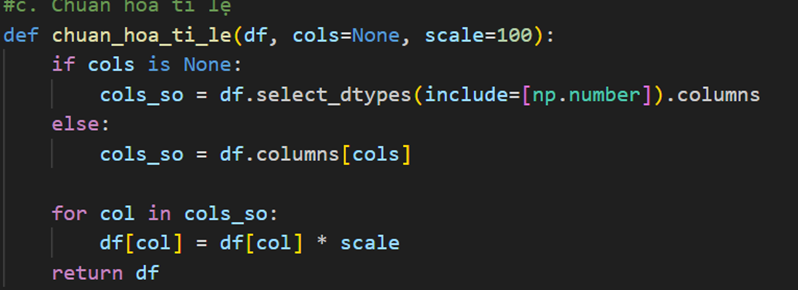


Đưa dữ liệu về phân phối chuẩn với trung bình bằng 0 và độ lệch chuẩn bằng 1.



### 3.3. Scaling (chuan\_hoa\_ti\_le(df, cols=None, scale=100)):

Nhân dữ liệu với một hệ số nhất định.



### 3.4. So sánh Độ chính xác và Thời gian chạy:

**a) Khi dùng thư viện:**



**b) Khi làm thủ công:**



**Nhận xét:**

**1. Thời gian chạy:**

- Thời gian chạy khi dùng thư viện: 1.094 giây.

- Thời gian chạy khi làm thủ công: 1.106 giây.

Chênh lệch thời gian chạy giữa hai phương pháp là rất nhỏ, chỉ khoảng 0.012 giây. Điều này cho thấy trong trường hợp này, cả hai phương pháp đều có hiệu suất rất gần nhau và không có sự khác biệt rõ ràng về mặt thời gian.

**2. Độ chính xác:**

- Số dòng dữ liệu ban đầu: 691.

- Số dòng dữ liệu còn lại sau khi làm sạch:

- Dùng thư viện: 634 dòng.

- Làm thủ công: 609 dòng.

**Phân tích:**

- Dùng thư viện: Sau khi làm sạch dữ liệu, chỉ còn lại 634 dòng dữ liệu. Điều này cho thấy rằng thư viện đã giữ lại nhiều dữ liệu hơn sau khi xử lý các giá trị thiếu hoặc không hợp lệ. Điều này có thể chỉ ra rằng thư viện có thể tự động xử lý tốt hơn với các giá trị bị thiếu mà không làm mất quá nhiều dữ liệu.

- Làm thủ công: Sau khi làm sạch dữ liệu, chỉ còn lại 609 dòng dữ liệu, cho thấy phương pháp thủ công đã loại bỏ nhiều dòng hơn. Điều này có thể là kết quả của việc xử lý các giá trị thiếu hoặc không hợp lệ bằng cách thủ công, trong đó có thể có nhiều quy trình loại bỏ dòng hơn so với thư viện.

**3. Đánh giá độ chính xác:**

Thư viện: Có thể xử lý dữ liệu nhanh chóng và hiệu quả hơn, không loại bỏ quá nhiều dòng, đảm bảo giữ lại phần lớn dữ liệu (634 dòng).

Thủ công: Có thể loại bỏ dữ liệu không hợp lệ chặt chẽ hơn, nhưng đồng thời có thể làm mất dữ liệu quan trọng, dẫn đến chỉ còn 609 dòng.

**4. Kết luận:**

Về thời gian chạy: Không có sự khác biệt đáng kể giữa hai phương pháp, với chỉ một chênh lệch nhỏ là 0.012 giây. Điều này có thể không quá quan trọng trong trường hợp dữ liệu nhỏ.

Về độ chính xác: Phương pháp dùng thư viện có vẻ duy trì độ chính xác cao hơn (634 dòng) so với phương pháp thủ công (609 dòng), vì thư viện có thể tự động xử lý các giá trị thiếu hoặc không hợp lệ mà không loại bỏ quá nhiều dữ liệu.

**5. Tầm quan trọng của Tiền Xử Lý Dữ Liệu**

Tăng độ chính xác của mô hình: Tiền xử lý giúp cải thiện chất lượng dữ liệu đầu vào, từ đó tăng độ chính xác của mô hình học máy.

Giảm thiểu overfitting: Bằng cách chuẩn hóa và chọn lọc đặc trưng, ta có thể giúp mô hình không bị overfit và dễ dàng tổng quát hóa với dữ liệu mới.

Tiết kiệm thời gian và tài nguyên: Xử lý dữ liệu đúng cách giúp tối ưu hóa thời gian huấn luyện và sử dụng tài nguyên tính toán, nhất là khi làm việc với các bộ dữ liệu lớn.

**6. Công Cụ và Thư Viện Tiền Xử Lý Dữ Liệu**

Pandas: Một thư viện mạnh mẽ trong Python cho việc xử lý và phân tích dữ liệu.

NumPy: Thư viện hỗ trợ các thao tác với mảng và toán học số học.

Scikit-learn: Cung cấp các công cụ tiền xử lý dữ liệu như chuẩn hóa, mã hóa, chọn lọc đặc trưng.

Imbalanced-learn: Thư viện hỗ trợ các phương pháp xử lý dữ liệu không đồng nhất.

**7. Ví Dụ và Ứng Dụng Tiền Xử Lý**

Dự đoán bệnh ung thư: Tiền xử lý dữ liệu có thể bao gồm loại bỏ các giá trị thiếu, mã hóa các đặc trưng phân loại (như giới tính, phương pháp điều trị), chuẩn hóa các giá trị đầu vào, và giảm chiều dữ liệu để cải thiện hiệu quả mô hình phân loại.

Phân tích khách hàng: Dữ liệu khách hàng có thể có giá trị thiếu, ngoại lai, và các đặc trưng phân loại (như ngành nghề, vị trí) cần được mã hóa và chuẩn hóa trước khi đưa vào mô hình phân loại hành vi khách hàng.

# Neural Network:

## 1. Giới thiệu

- Là một mạng neural nhân tạo, mà các kết nối truyền thẳng từ đầu vào đến đầu ra. Đây là mô hình machine learning khá tốt được sử dụng trong rất nhiều lĩnh vực, SLFN có khả năng xấp xỉ một tập dữ liệu phức tạp trực tiếp từ dữ liệu đầu vào.

- Cấu trúc của mạng neural truyền thẳng một lớp ẩn bao gồm: 1 lớp đầu vào, 1 lớp ẩn và một lớp đầu ra.

## 2. Thuật toán

- Khởi tạo: (w\_1, w\_1,…, w\_n t) ∈ [-0,5,0,5]

- Kích hoạt: Tính kết xuất thực sự tại lần lặp p=1

Y\_((p))=step[∑\_(i=0)^n▒〖x\_i (p) w\_i (p)-0〗]

- Cập nhật trọng số: w\_i(p+1)= w\_i(p)+∆ w\_1(p)

Trong đó: ∆ w\_1(p) = α x x\_1 (p) x e(p) (luật học delta)

- Lặp: tăng p lên 1 và lặp lại bước 2.

## 3. Kiến trúc mạng neural

Mỗi một mạng lưới neural nhân tạo là một Perceptron đa tầng, một Neural Network thường bao gồm 3 kiểu tầng cụ thể như sau:

- Input Layer (tầng đầu vào): Nằm bên trái của hệ thống, bao gồm dữ liệu thông tin đầu vào.

- Output Layer (tầng đầu ra): Nằm bên phải của hệ thống, bao gồm dữ liệu thông tin đầu ra.

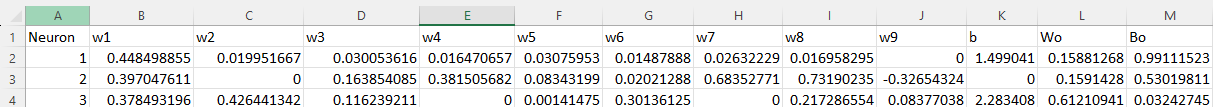
- Hidden Layer (tầng ẩn): Nằm ở giữa tầng đầu vào và đầu ra, thể hiện quá trình suy luận và xử lý thông tin của hệ thống.

Mỗi nút mạng trong Neural Network là một Sigmoid Neural. Thường các nút mạng này sẽ có hàm kích hoạt khác nhưng hiện tại đang áp dụng thuật toán đồng nhất để dễ dàng hoạt động hơn. Ở mỗi tầng, số lượng sigmoid neural khác nhau tùy thuộc vào cách thức xử lý dữ liệu. Trong quá trình hoạt động, các chuyên gia sẽ để các tầng ẩn – hidden layer với số lượng neural khác nhau.

## 4. Khởi chạy

Excel

Cho dữ liệu sau:



- B2-J2 là trọng số đầu vào với neural thứ nhất ở lớp ẩn, K2 là bias của neural thứ nhất ở lớp ẩn

- B3-J3 là trọng số đầu vào với neural thứ nhất ở lớp ẩn, K3 là bias của neural thứ nhất ở lớp ẩn

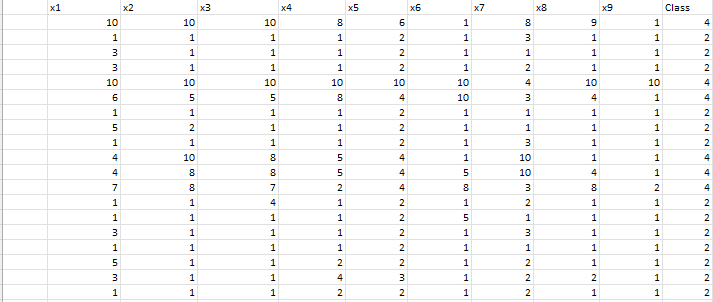
- B4-J4 là trọng số đầu vào với neural thứ nhất ở lớp ẩn, K4 là bias của neural thứ nhất ở lớp ẩn

- L2 là trọng số neural thứ nhất với lớp đầu ra

- L3 là trọng số neural thứ nhất với lớp đầu ra

- L4 là trọng số neural thứ nhất với lớp đầu ra

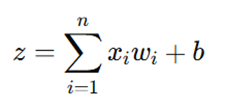
- M2 là bias của neural lớp đầu ra



- Tập kiểm tra có 9 đặc trưng (x1-x9) và lớp nhãn có 2 giá trị (2 và 4)

- Số dữ liệu: 683 mẫu

Áp dụng công thức sau:



Trong đó

- z: Tổng trọng số, là giá trị mà neuron lớp tiếp theo nhận được sau khi qua hàm kích hoạt

- n: Số lượng đầu vào

- x1,x2,...xn: các đầu vào của neural

- w1,w2,...,wn: các trọng số tương ứng với các đầu vào

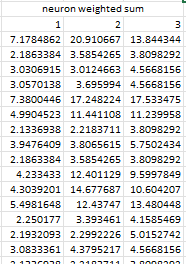
Áp dụng tính tổng trọng số trong excel:

- Neural 1 (ô L11):=SUMPRODUCT($B11:$J11, INDEX($B$2:$J$4, L$10, 0))+@INDEX($K$2:$K$4, L$10, 1) = 7.17848616

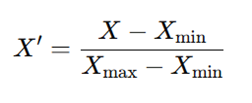
- Neural 2(ô M11):=SUMPRODUCT($B11:$J11, INDEX($B$2:$J$4, M$10, 0))+@INDEX($K$2:$K$4, M$10, 1) = 20.9106668

- Neural 3(ô N11):=SUMPRODUCT($B11:$J11, INDEX($B$2:$J$4, N$10, 0))+@INDEX($K$2:$K$4, N$10, 1) = 13.8443444

- Thực hiện tương tự cho các mẫu còn lại, ta thu được:



Tiếp theo, sử dụng công thức



để chuẩn hoá tổng trọng số theo min, max [0,1]

Với:

- X: giá trị gốc của dữ liệu cần chuẩn hoá

- Xmin: giá trị nhỏ nhất trong tập dữ liệu

- Xmax: giá trị lớn nhất trong tập dữ liệu

- X’ là giá trị sau khi chuẩn hoá

ô L6: =MAX(L11:L693)

ô M6: =MAX(M11:M693)

ô N6: =MAX(N11:N693)

ô L7: =MAX(L11:L693)

ô M7: =MAX(N11:N693)

ô N7: =MAX(N11:N693)

Ta thu được:



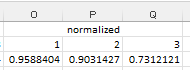
Từ kết quả đầu ra tổng trọng số (L11,M11,N11) và giá trị Max, Min, áp dụng công thức toán học:

ô O11: =(L11-L$7)/(L$6-L$7)

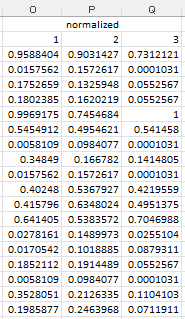
ô P11: =(M11-M$7)/(M$6-M$7)

ô Q11: =(N11-N$7)/(N$6-N$7)

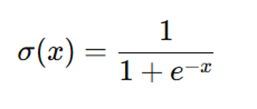
Ta thu được:



Thực hiện cho các mẫu còn lại:



Tiếp theo, áp dụng hàm kích hoạt sigmoid cho giá trị đầu ra của chuẩn hoá tổng trọng số:



Với:

- e: cơ số logarid tự nhiên (~2.718)

- x: đầu vào

áp dụng vào dữ liệu excel với giá trị chuẩn hoá dữ liệu đầu tiên (O11,P11,Q11)

neuron 1(R11): =1/(1+EXP(-O11))

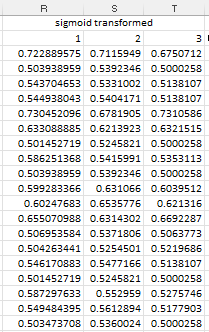
neuron 2(S11): =1/(1+EXP(-P11))

neuron 3(T11): =1/(1+EXP(-Q11))

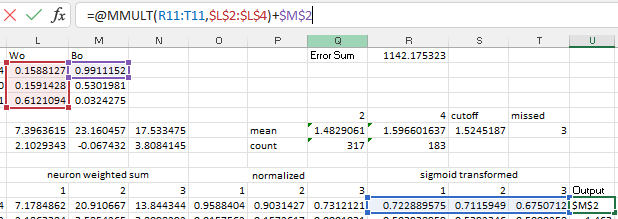
Ta thu được:



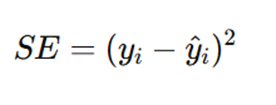
áp dụng cho các mẫu còn lại:



Tính output bằng công thức =@MMULT(R11:T11,$L$2:$L$4)+$M$2 tương tự cho các ô còn lại

ta có

Tính lỗi:



yi: giá trị thực tế của mẫu thứ i

y^i: giá trị dự đoán của mô hình cho mẫu thứ i

áp dụng vào dữ liệu excel ta có:

V11=(U11-K11)^2

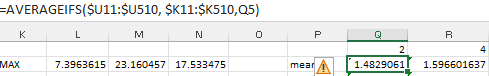
với U là output và K là giá trị thực tế của mẫu dữ liệu



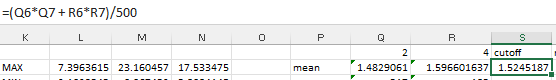
Dự đoán:

- Tính trung bình cộng output với các mẫu có class là 2: =AVERAGEIFS($U11:$U510, $K11:$K510,Q5)

- Tính trung bình cộng output với các mẫu có class là 4: =AVERAGEIFS($U11:$U510, $K11:$K510,R5)



ô cutoff:



output < hoặc = cutoff => mẫu thuộc class 2, nếu > cutoff => mẫu thuộc class 4



## 5. Ứng dụng thực tế

- Thực hiện chẩn đoán y tế bằng các hình ảnh y khoa.

- Dự đoán tài chính thông qua việc xử lý các dữ liệu của công cụ tài chính đó trước đây.

- Giám sát chất lượng và cách thức vận hành.

- Phân biệt các hợp chất hóa học.

- Phân tích dữ liệu hành vi và lọc mạng xã hội nhằm thực hiện quá trình tiếp thị đúng với mục tiêu và đối tượng.

- Mạng neural còn có công dụng dự đoán năng lượng và phụ tải điện năng.

- Ngoài ra còn có các ứng dụng vô cùng quan trọng như: Thị giác máy tính, Nhận dạng giọng nói, Khả năng xử lý ngôn ngữ tự nhiên, Đóng vai trò là công cụ đề xuất

# Philippe Fournier-Viger

Toán:

Tìm hiểu các sản phẩm mà hay xuất hiện cùng nhau.

Ví dụ: các sản phẩm hay xuất hiện cùng nhau trong hóa đơn của 1 cửa hàng, khám phá các thông tin ẩn sau các dữ liệu

Bài toán: cho các hóa đơn:

Hóa đơn 1: A, B, C, D, E

Hóa đơn 2: A, B

Hóa đơn 3: C, D, E

Hóa đơn 4: A, C, E

Tìm các sản phẩm có số lần xuất hiện trên hóa đơn lớn hơn 1

Cách giải:

Naïve Appriach

Vì các sản phẩm có số lần xuất hiện lớn hơn 1 nên min sup = 2

Đếm số lượng sản phẩm đơn loại:

A: 3, B: 2, C: 3, D: 2, E: 3

Bắt cặp 2 sản phẩm để đếm số lượng của chúng:

A, B: 2 A, C: 2 A, D: 1 A, E: 2

B, C: 1 B, D: 1 B, E: 1 C, D: 2

C, E: 3 D, E: 2

Bắt cặp 3 sản phẩm để đếm số lượng của chúng:

A, B, C: 1 A, B, D: 1 A, B, E: 1 A, C, D: 1

A, C, E: 2 A, D, E: 1 B, C, D: 1 B, C, E: 1

B, D, E: 1 C, D, E: 2

Bắt cặp 4 sản phẩm để đếm số lượng của chúng:

A, B, C, D: 1 A, B, C, E: 1 A, B, D, E: 1

A, C, D, E: 1 B, C, D, E: 1

Bắt cặp 5 sản phẩm để đếm số lượng của chúng:

A, B, C, D, E: 1

Kết quả:

A: 3, B: 2, C: 3, D: 2, E: 3

A, B: 2 A, C: 2 A, E: 2 C, D: 2 C, E: 3 D, E: 2

A, C, E: 2 C, D, E: 2

Apriory

Tương tự Naïve Appriach nhưng them bước bỏ các tập hợp nhỏ hơn min sub sau khi đếm xong các tập hơp để tiết kiệm chi phí

Ví dụ: ở bước bắt cặp 2 sản phẩm:

A, B: 2 A, C: 2 A, D: 1 A, E: 2

B, C: 1 B, D: 1 B, E: 1 C, D: 2

C, E: 3 D, E: 2

ta có thể bỏ AD, BC, BD, BE

Ta còn:

A, B: 2 A, C: 2 A, E: 2 C, D: 2

C, E: 3 D, E: 2

Ta bắt cặp tiếp từ những cặp trên ta có:  
A, B, C: 1 A, B, E: 1 A, C, E: 2 A, C, D: 1

A, D, E: 1 C, D, E: 2

ta có thể bỏ ABC, ABE, ACD, ADE

Ta còn:

A, C, E: 2 C, D, E: 2

Ta bắt cặp tiếp từ 2 cặp trên ta có:  
A, C, D, E: 1 (bỏ)

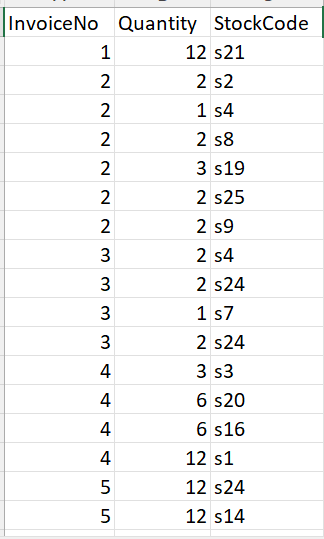
Vậy kết quả:

A: 3, B: 2, C: 3, D: 2, E: 3

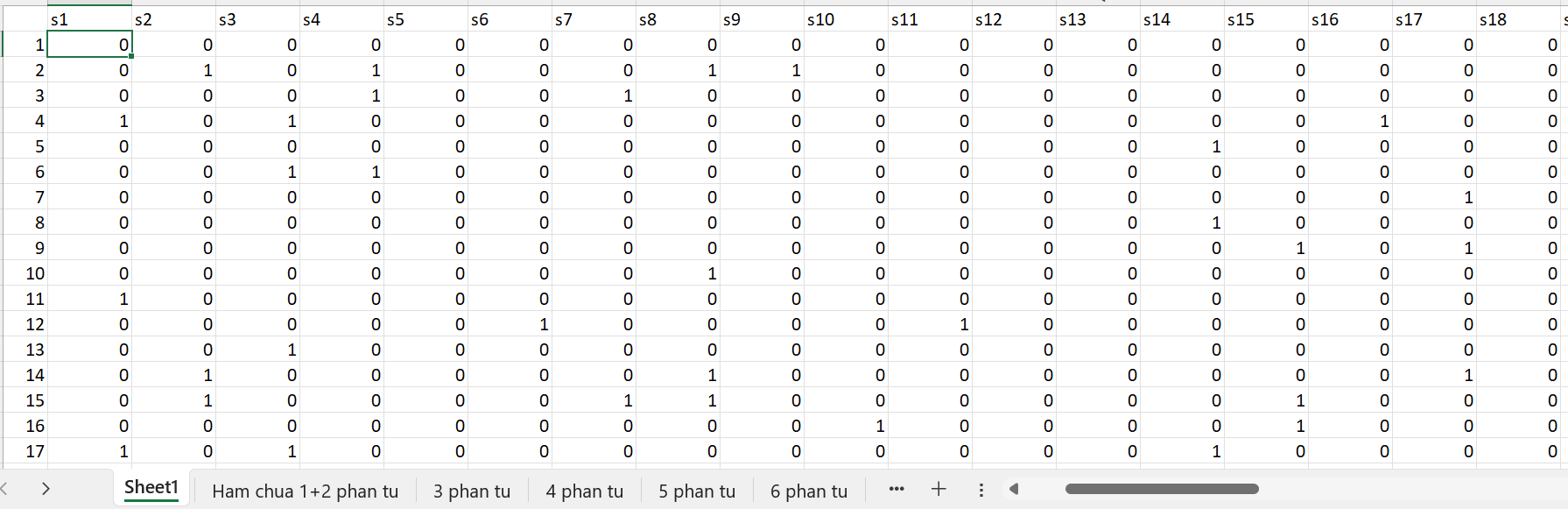
A, B: 2 A, C: 2 A, E: 2 C, D: 2 C, E: 3 D, E: 2

A, C, E: 2 C, D, E: 2

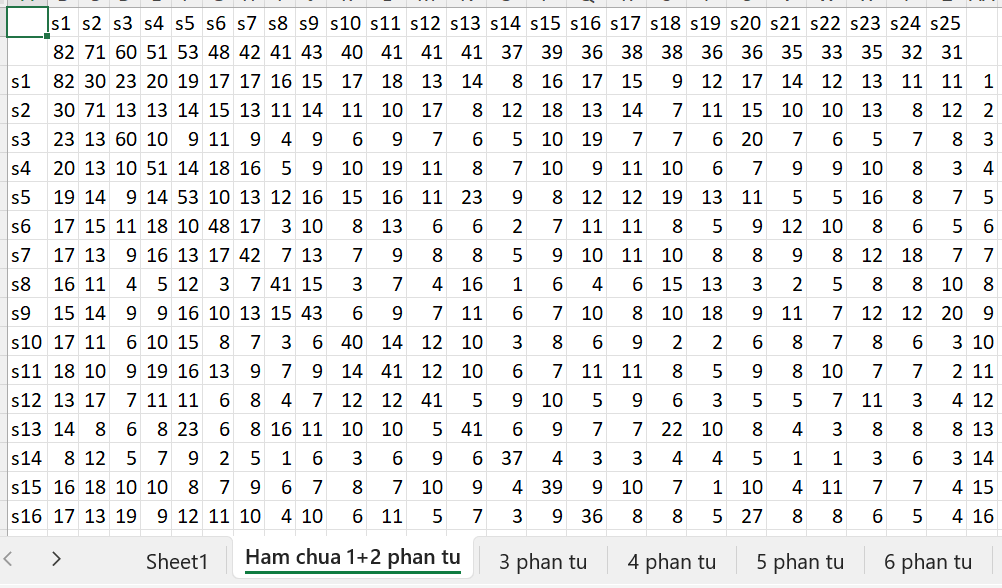
Excel:

Ta có file excel chứa thông tin như sau  


Phân tích:



Sử dụng hàm =COUNTIFS($A$2:$A$1114,$D2,$C$2:$C$1114,E$1) để duyệt file trên để dễ dàng xem các sản phẩm có trong hóa đơn



Hàng thứ 2 là hàm đếm số lượng sản phẩm tương ứng xuất hiện

=COUNTIF(Sheet1!E$2:E$1114,1)

Từ hàng thứ 3 là hàm đếm số lượng sản phẩm gom nhóm 2:

Ví dụ:

s1, s1 là sản phẩm chứa s1

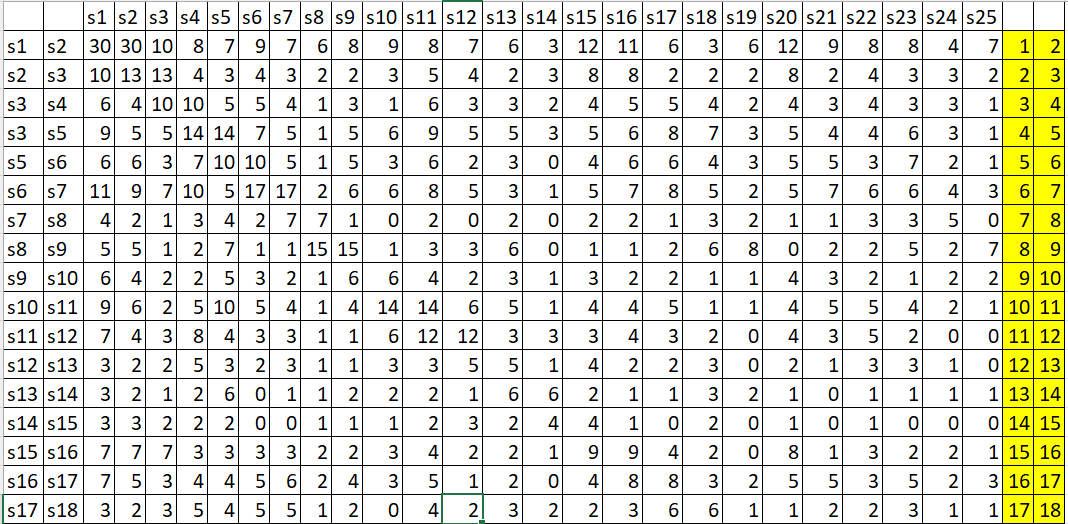
s1, s2 là sản phẩm chứa s1, s2

Tương tự

=COUNTIFS(Sheet1!E$2:E$1114,1,INDEX(Sheet1!$E$2:$BB$1114,0,$AA3),1)

Với hàm index là hàm đếm số lượng cột tương ứng với cột bên phải

Làm tương tự cho tới khi cả bảng có count = 0:



Nếu có min sub, dừng lại khi tất cả các phần tử trong bảng nhỏ hơn min sub

Python

Naïve Appriach

# tạo tập hợp con

def generate\_candidates(sublist, length, prefix=()):

    if len(prefix) == length:

        return [prefix]

    candidates = []

    for i, item in enumerate(sublist):

        new\_prefix = prefix + (item,)

        remaining\_items = sublist[i+1:]

        candidates.extend(generate\_candidates(remaining\_items, length, new\_prefix))

    return candidates

# thêm các items vào candidate\_counts và đếm số lần xuất hiện

def add\_candidates(candidate\_counts, candidates):

    for candidate in candidates:

        candidate = tuple(sorted(candidate))

        if candidate in candidate\_counts:

            candidate\_counts[candidate] += 1

        else:

            candidate\_counts[candidate] = 1

dataset = load\_data()

candidate\_counts = {}

# đếm lại số lần xuất hiện của mỗi candidate

for sublist in dataset:

    for length in range(1, len(sublist) + 1):

        candidates = generate\_candidates(sublist, length)

        add\_candidates(candidate\_counts, candidates)

# In ra các ứng viên và số lần xuất hiện của chúng

print("Các ứng viên và số lần xuất hiện:")

for candidate, count in candidate\_counts.items():

    if count>=min\_sup:  
 print(f"{candidate}: {count}")

Apriory

#tạo tập hợp con

def generate\_candidates(sublist, length, prefix=()):

    if len(prefix) == length:

        return [prefix]

    candidates = []

    for i, item in enumerate(sublist):

        new\_prefix = prefix + (item,)

        remaining\_items = sublist[i+1:]

        candidates.extend(generate\_candidates(remaining\_items, length, new\_prefix))

    return candidates

#kiểm tra tập con thỏa min\_sup

def calculate\_support(dataset, candidates):

    candidate\_counts = {}

    for transaction in dataset:

        for candidate in candidates:

            if set(candidate).issubset(set(transaction)):

                candidate\_tuple = tuple(sorted(candidate))

                if candidate\_tuple in candidate\_counts:

                    candidate\_counts[candidate\_tuple] += 1

                else:

                    candidate\_counts[candidate\_tuple] = 1

    return {k: v for k, v in candidate\_counts.items() if v >= min\_sup}

dataset = load\_data()

candidate\_counts = {}

# đếm lại số lần xuất hiện của mỗi candidate

for sublist in dataset:

    for length in range(1, len(sublist) + 1):

        candidates = generate\_candidates(sublist, length)

        for candidate in candidates:

            candidate\_counts[tuple(sorted(candidate))] = candidate\_counts.get(tuple(sorted(candidate)), 0) + 1

filtered\_candidates = calculate\_support(dataset, candidate\_counts.keys())

print("Các ứng viên và số lần xuất hiện:")

for candidate, count in filtered\_candidates.items():

    print(f"{list(candidate)}: {count}")

So sánh

DataSet

Run time khi sử dụng dữ liệu khoảng 1000 dòng gồm khoảng 350 hóa đơn

Run time khi sử dụng dữ liệu khoảng 800 hóa đơn

Run time khi sử dụng dữ liệu khoảng 10000 hóa đơn

Độ chính xác

Để so sánh chính xác giữa thư viện và code thì em thấy không khả thi vì code thư viện sử dụng đơn vị phần tram (tần suất xuất hiệnĐ còn code của em sử dụng đơn vị lần (số lần xuất hiện) nên khó phân tích nhưng nếu so sánh với excel thì độ chính xác là 1.0

Running time

Run time khi sử dụng dữ liệu khoảng 1000 dòng gồm khoảng 350 hóa đơn

Thời gian Naïve Appriach: 2.1023201942443848

Thời gian Apriory: 50.85669684410095

Thư viện mlxtend: 0.005984783172607422

Run time khi sử dụng dữ liệu khoảng 10000 hóa đơn

# KNN

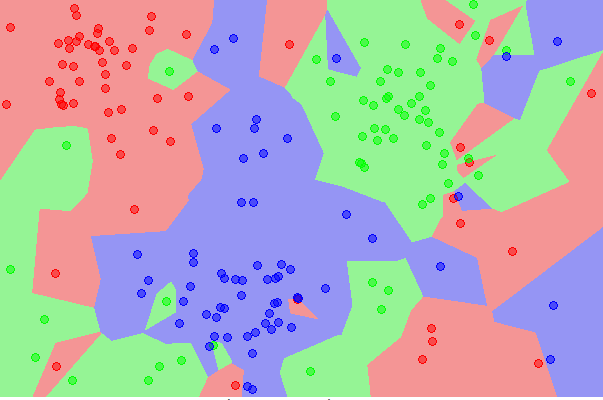
## 1. Giới thiệu

K-nearest neighbor là một trong những thuật toán supervised-learning đơn giản nhất (mà hiệu quả trong một vài trường hợp) trong Machine Learning. Khi training, thuật toán này *không học* một điều gì từ dữ liệu training (đây cũng là lý do thuật toán này được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới. K-nearest neighbor có thể áp dụng được vào cả hai loại của bài toán Supervised learning là Classification và Regression. KNN còn được gọi là một thuật toán Instance-based hay Memory-based learning.

Với KNN, trong bài toán Classification, label của một điểm dữ liệu mới (hay kết quả của câu hỏi trong bài thi) được suy ra trực tiếp từ K điểm dữ liệu gần nhất trong training set. Label của một test data có thể được quyết định bằng major voting (bầu chọn theo số phiếu) giữa các điểm gần nhất, hoặc nó có thể được suy ra bằng cách đánh trọng số khác nhau cho mỗi trong các điểm gần nhất đó rồi suy ra label. Chi tiết sẽ được nêu trong phần tiếp theo.

Trong bài toán Regresssion, đầu ra của một điểm dữ liệu sẽ bằng chính đầu ra của điểm dữ liệu đã biết gần nhất (trong trường hợp K=1), hoặc là trung bình có trọng số của đầu ra của những điểm gần nhất, hoặc bằng một mối quan hệ dựa trên khoảng cách tới các điểm gần nhất đó.

Một cách ngắn gọn, KNN là thuật toán đi tìm đầu ra của một điểm dữ liệu mới bằng cách *chỉ* dựa trên thông tin của K điểm dữ liệu trong training set gần nó nhất (K-lân cận), *không quan tâm đến việc có một vài điểm dữ liệu trong những điểm gần nhất này là nhiễu*. Hình dưới đây là một ví dụ về KNN trong classification với K = 1.

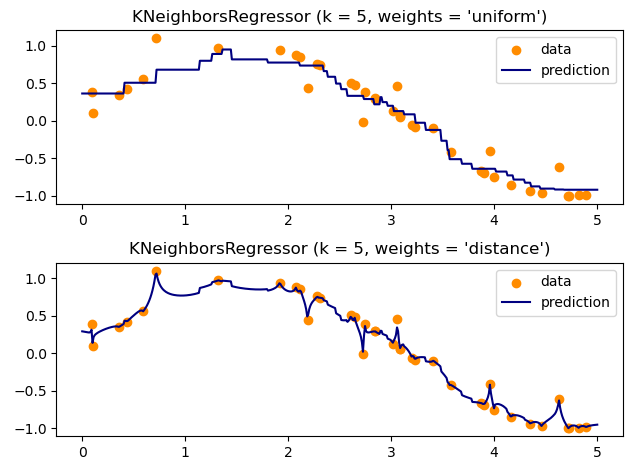


**Khoảng cách trong không gian vector**

Trong không gian một chiều, khoảng cách giữa hai điểm là trị tuyệt đối giữa hiệu giá trị của hai điểm đó. Trong không gian nhiều chiều, khoảng cách giữa hai điểm có thể được định nghĩa bằng nhiều hàm số khác nhau, trong đó độ dài đường thằng nổi hai điểm chỉ là một trường hợp đặc biệt trong đó.

**KNN cho Regression**

Với bài toán Regression, chúng ta cũng hoàn toàn có thể sử dụng phương pháp tương tự: ước lượng đầu ra dựa trên đầu ra và khoảng cách của các điểm trong K-lân cận. Việc ước lượng như thế nào các bạn có thể tự định nghĩa tùy vào từng bài toán.



**Chuẩn hoá dữ liệu**

Khi có một thuộc tính trong dữ liệu (hay phần tử trong vector) lớn hơn các thuộc tính khác rất nhiều (ví dụ thay vì đo bằng cm thì một kết quả lại tính bằng mm), khoảng cách giữa các điểm sẽ phụ thuộc vào thuộc tính này rất nhiều. Để có được kết quả chính xác hơn, một kỹ thuật thường được dùng là *Data Normalization* (chuẩn hóa dữ liệu) để đưa các thuộc tính có đơn vị đo khác nhau về cùng một khoảng giá trị, thường là từ 0 đến 1, trước khi thực hiện KNN. Các kỹ thuật chuẩn hóa được áp dụng với không chỉ KNN mà còn với hầu hết các thuật toán khác.

## 2. Ưu và nhược điểm của KNN

* Ưu điểm
* Độ phức tạp tính toán của quá trình training là bằng 0.
* Việc dự đoán kết quả của dữ liệu mới rất đơn giản.
* Không cần giả sử gì về phân phối của các class.
* Nhược điểm
* KNN rất nhạy cảm với nhiễu khi K nhỏ.
* KNN là một thuật toán mà mọi tính toán đều nằm ở khâu test. Trong đó việc tính khoảng cách tới *từng* điểm dữ liệu trong training set sẽ tốn rất nhiều thời gian, đặc biệt là với các cơ sở dữ liệu có số chiều lớn và có nhiều điểm dữ liệu. Với K càng lớn thì độ phức tạp cũng sẽ tăng lên. Ngoài ra, việc lưu toàn bộ dữ liệu trong bộ nhớ cũng ảnh hưởng tới hiệu năng của KNN.

## 3. Thuật toán

Toán:

Cho 1 điểm và 1 file dữ liệu được chia thành các cụm

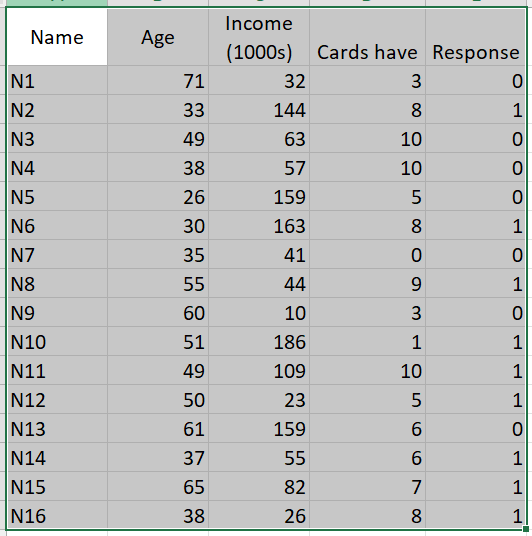
Xác định xem điểm dó thuộc cụm nào dựa theo số điển gần điểm đó nhất

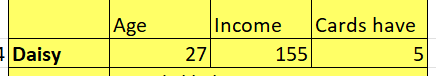
Cách giải:

Tính khoảng cách các k điểm gần nhất, điểm cần được phân loại sẽ thuộc nhóm có số lượng nhiều nhất

Excel:

Cho dữ liệu và 1 điểm

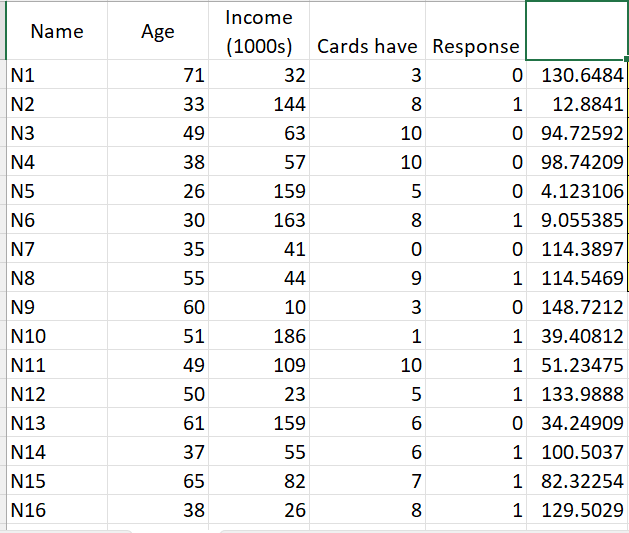




Xác định response của điểm Dasy khi K = 3, 5, 7, 9, 11

Các làm tính khoảng các gữi điểm Dasy với các điểm trên file bằng công thức:

=SQRT((B2-$H$2)^2+(C2-$I$2)^2+(D2-$J$2)^2)



dung hàm small để tìm số nhỏ theo yêu cầu của đề:

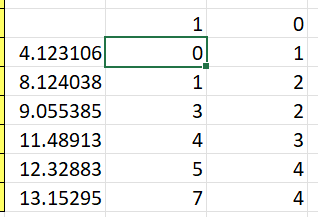
=SMALL($F$2:$F$201,G4)

Sau đó cùng count if để đếm số lượng Response nhỏ hơn hoặc bằng small

=COUNTIFS($E$2:$E$201,L$3,$F$2:$F$201,"<="&$K4)

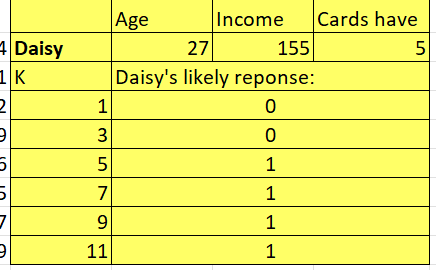
Có thểm làm tắt 2 hàm này:

=COUNTIFS($E$2:$E$201,L$3,$F$2:$F$201,"<="&$ SMALL($F$2:$F$201,$G4))



So sánh 2 giá trị response để xem giá trị nào lớn hơn thì trả về response đó

=IF(L4>M4,$L$3,$M$3)



Python

# Định nghĩa hàm tính khoảng cách Euclidean giữa hai điểm

def euclidean\_distance(point1, point2):

    squared\_distance = sum((x - y) \*\* 2 for x, y in zip(point1, point2))

    return squared\_distance \*\* 0.5

# Hàm KNN

def knn(X\_train, y\_train, X\_test, k):

    predictions = []

    for test\_point in X\_test:

        # Tính toán khoảng cách giữa điểm test và tất cả điểm trong tập train

        distances = []

        for i, train\_point in enumerate(X\_train):

            distance = euclidean\_distance(train\_point, test\_point)

            distances.append((distance, y\_train[i]))

        # Sắp xếp khoảng cách và chọn k điểm gần nhất

        distances.sort(key=lambda x: x[0])

        k\_nearest = distances[:k]

        # Dự đoán nhãn cho điểm test dựa trên đa số trong k điểm gần nhất

        counts = {}

        for \_, label in k\_nearest:

            counts[label] = counts.get(label, 0) + 1

        prediction = max(counts, key=counts.get)

        predictions.append(prediction)

    return predictions

# Dự đoán nhãn cho các điểm test sử dụng KNN

predictions = knn(X\_train, y\_train, X\_test, k)

print()

print("Predictions:", predictions)

So sánh

DataSet

200 dòng

Khoảng 650 dòng

1000 dòng

Độ chính xác

Dung hàm accuracy\_score để đo độ chính xác

Độ chính xác giữa 2 thuật toán luôn luôn là: 1.0

Running time

Trong dữ liệu 200 dòng,

Thời gian chạy KNN tự viết: 0.002023935317993164

Thời gian chạy KNN từ scikit-learn: 0.001996755599975586

Trong dữ liệu 650 dòng:

Thời gian chạy KNN tự viết: 0.008977651596069336

Thời gian chạy KNN từ scikit-learn: 0.0019953250885009766

Trong dữ liệu 1000 dòng:

Thời gian chạy KNN tự viết: 0.016963481903076172

Thời gian chạy KNN từ scikit-learn: 0.002980470657348633

# K-means Clustering:

## 1. Giới thiệu

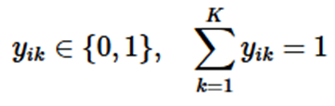
- Trong thuật toán K-means clustering, chúng ta không biết nhãn (label) của từng điểm dữ liệu. Mục đích là làm thể nào để phân dữ liệu thành các cụm (cluster) khác nhau sao cho dữ liệu trong cùng một cụm có tính chất giống nhau.

- Ý tưởng đơn giản nhất về cluster (cụm) là tập hợp các điểm ở gần nhau trong một không gian nào đó (không gian này có thể có rất nhiều chiều trong trường hợp thông tin về một điểm dữ liệu là rất lớn).

- Trong không gian ba chiều, lấy ví dụ là các hành tinh, thì (tạm gọi là) lãnh không của mỗi hành tinh sẽ là một đa diện. Trong không gian nhiều chiều hơn, chúng ta sẽ có những thứ (mà tôi gọi là) siêu đa diện (hyperpolygon).

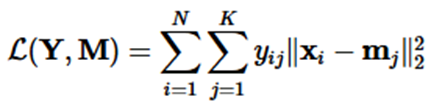
## 2. Công thức

Ràng buộc của y\_i có thể viết dưới dạng toán học như sau:

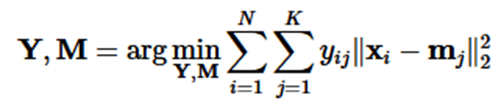


Hàm mất mát và bài toán tối ưu

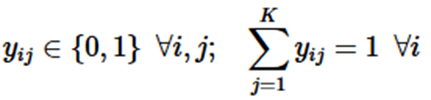
- Sai số cho toàn bộ dữ liệu:



- Chúng ta cần tối ưu bài toán sau:



Thoả mãn:



## 3. Thuật toán & khởi chạy

Cho 1 file dữ liệu chia các điểm trong file đó thành k nhóm

Cách giải

Chọn ngẫu nhiên k điểm để gom nhóm

Tính khoảng cách giữ các điểm đó và các điểm trong file

Gom nhóm các điểm có khoảng cách nhỏ nhất

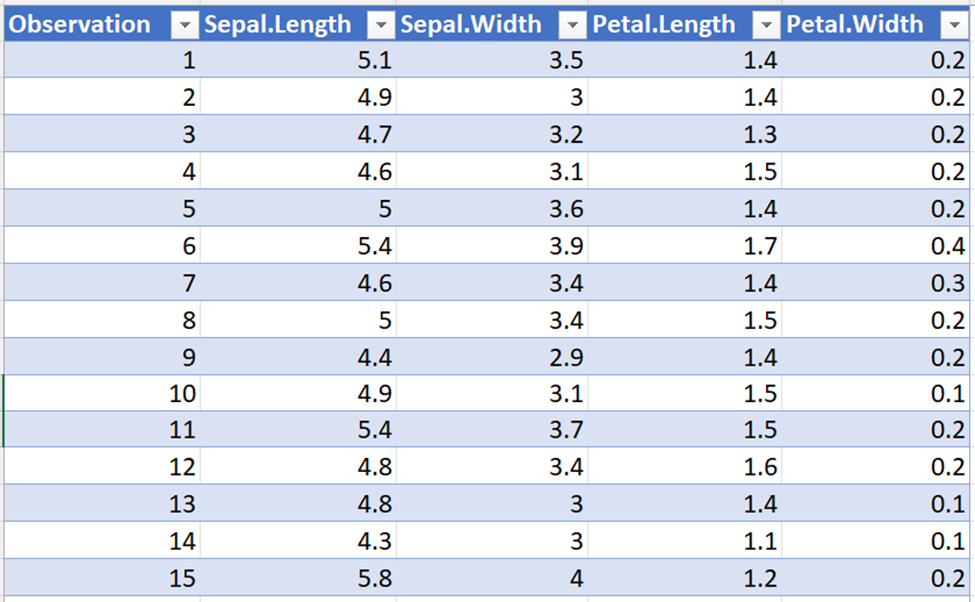
Tính lại các điểm mới vừa dựa vào trung bình các thuộc tính của các điểm trong nhóm vừa được tạo

Lặp bước tính khoảng cách giữa các điểm mới và các điểm trong file

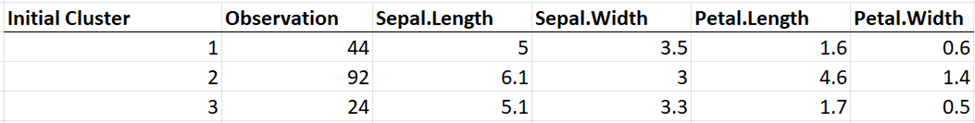
Lặp cho đến khi các điểm mới được tạo giống với điểm vừa được tạo

Excel

Cho dữ liệu:



=VLOOKUP(K4,$A$2:$E$151,2) để lấy dữ liệu từ file đã cho



Tính khoảng cách giữa các điểm trong file và các điểm đã chọn:

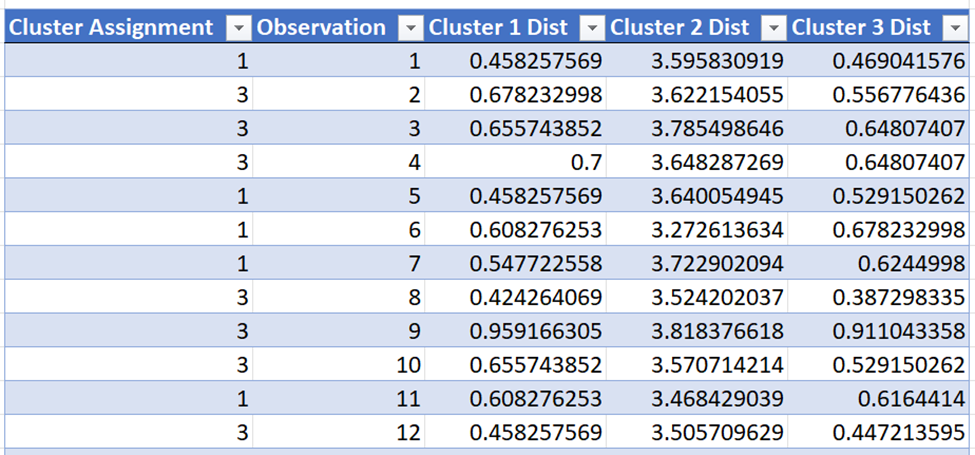
=SQRT(($B2-$L$4)^2+($C2-$M$4)^2+($D2-$N$4)^2+($E2-$O$4)^2)

=SQRT(($B2-$L$5)^2+($C2-$M$5)^2+($D2-$N$5)^2+($E2-$O$5)^2)

=SQRT(($B2-$L$6)^2+($C2-$M$6)^2+($D2-$N$6)^2+($E2-$O$6)^2)

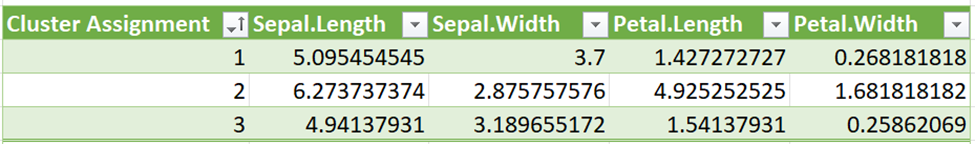
Chi nhóm theo khoảng cách của chúng

=MATCH(MIN(L15:N15),L15:N15,0)



Tính trung bình các thuộc tính trong nhóm được chia vào các điểm

=AVERAGEIFS(B$2:B$152,$J$14:$J$164,$J4)



Lặp lại 2 bước này cho đến khi thấy điểm mới và điểm liền trước của nó bằng nhau

Python:

import random

# Hàm tính khoảng cách Euclidean giữa hai điểm

def euclidean\_distance(point1, point2):

    return sum((x - y) \*\* 2 for x, y in zip(point1, point2)) \*\* 0.5

# Hàm tìm centroids ban đầu bằng cách chọn ngẫu nhiên trong dữ liệu

def initialize\_centroids(X, k):

    centroids = random.sample(X, k)

    return centroids

# Hàm gán các điểm vào các cụm dựa vào centroids

def assign\_points\_to\_clusters(X, centroids):

    clusters = {i: [] for i in range(len(centroids))}

    for point in X:

        closest\_centroid\_idx = min(range(len(centroids)), key=lambda i: sum((x - y) \*\* 2 for x, y in zip(point, centroids[i])) \*\* 0.5)

        clusters[closest\_centroid\_idx].append(point)

    return clusters

# Hàm cập nhật centroids dựa vào điểm trung bình của các điểm trong cụm

def update\_centroids(clusters):

    new\_centroids = []

    for cluster\_points in clusters.values():

        if cluster\_points:

            new\_centroid = tuple(sum(vals) / len(vals) for vals in zip(\*cluster\_points))

            new\_centroids.append(new\_centroid)

        else:

            new\_centroids.append(random.choice(list(clusters.values())))

    return new\_centroids

# Hàm K-Means

def kmeans(X, k, max\_iters=100):

    centroids = initialize\_centroids(X, k)

    print("Centroids ban đầu:")

    for i, centroid in enumerate(centroids):

        print(f"Centroid {i + 1}: {centroid}")

    for \_ in range(max\_iters):

        clusters = assign\_points\_to\_clusters(X, centroids)

        new\_centroids = update\_centroids(clusters)

        # Tính và hiển thị khoảng cách từ mỗi điểm đến centroids

        total\_distance = 0

        for i, centroid in enumerate(new\_centroids):

            for point in clusters[i]:

                distance = euclidean\_distance(point, centroid)

                print(f"Khoảng cách từ điểm {point} đến centroid {i + 1}: {distance}")

                total\_distance += distance

        print(f"Tổng khoảng cách: {total\_distance}")

        # Kiểm tra điều kiện dừng

        if new\_centroids == centroids:

            break

        centroids = new\_centroids

    return clusters

So sánh

DataSet

Độ chính xác

Không thể so sánh vì khi random thì hàm sẽ ngẫu nhiên chọn 3 điểm nên đôi khi các nhóm sẽ khác nhau

Running time

Thời gian chạy K-Means tự viết: 0.000997781753540039

Thời gian chạy K-Means từ scikit-learn: 0.23682737350463867

## 4. Ưu điểm và hạn chế

\* Ưu điểm:

- Đơn giản

- Hiệu quả trong thực tế

- Đảm bảo hội tụ trong thời gian đa thức

- Linh hoạt trong việc lựa chọn phương pháp đo khoảng cách

\* Hạn chế:

- Việc lựa chọn các tính khoảng cách cho bài toán cụ thể khó

- Nhạy cảm với các điểm dữ liệu outlier

## 5. Ứng dụng thực tế

- Phân loại khách hàng: Dựa vào dữ liệu về hành vi mua sắm, thói quen sử dụng dịch vụ, các công ty có thể phân loại khách hàng thành các nhóm để đưa ra chính sách marketing, chăm sóc khách hàng hiệu quả hơn.

- Phân loại văn bản: Dựa vào các từ khóa xuất hiện trong văn bản, có thể phân loại các văn bản vào các chủ đề khác nhau như tin tức, kinh tế, thể thao, giải trí, giáo dục, v.v.

- Xử lý ảnh: K-means Clustering có thể giúp giảm số lượng màu trong hình ảnh, loại bỏ nhiễu và làm rõ đường biên giữa các đối tượng trong ảnh.

- Giảm chiều dữ liệu: K-means Clustering có thể giúp giảm số lượng biến trong dữ liệu, giảm không gian chiều để các mô hình máy học dễ dàng hơn trong việc phân tích dữ liệu và tránh hiện tượng overfitting.

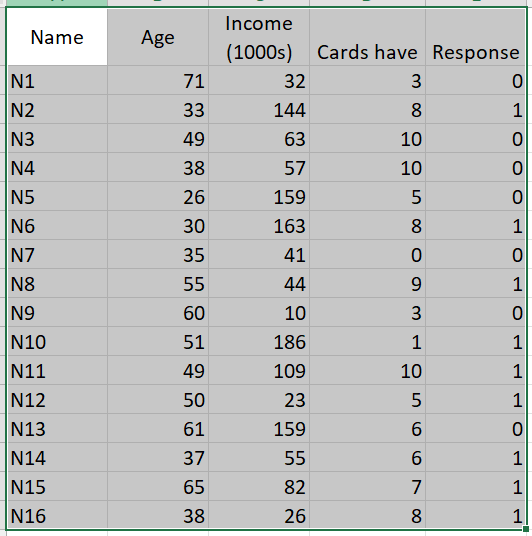
- Phân tích dữ liệu gen: Trong lĩnh vực sinh học, K-means Clustering được áp dụng để phân tích và quản lý dữ liệu gen, từ đó phát hiện sự giống nhau giữa các mẫu gen hoặc tìm ra mối quan hệ giữa các loài rộng lớn trong y học và nghiên cứu sinh học.

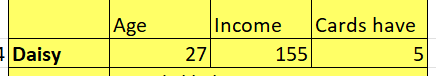
- An ninh mạng: Trong lĩnh vực an ninh mạng, K-means Clustering được áp dụng để phát hiện giao dịch bất thường hoặc đột nhập vào hệ thống mạng. Khi phân tích dữ liệu về các giao dịch hay các yêu cầu truy cập, những giao dịch bất thường sẽ nằm ngoài các cụm thông thường được tạo ra, từ đó giúp phát hiện ra các giao dịch gian lận hay đột nhập.

…

Excel:

Cho dữ liệu và 1 điểm

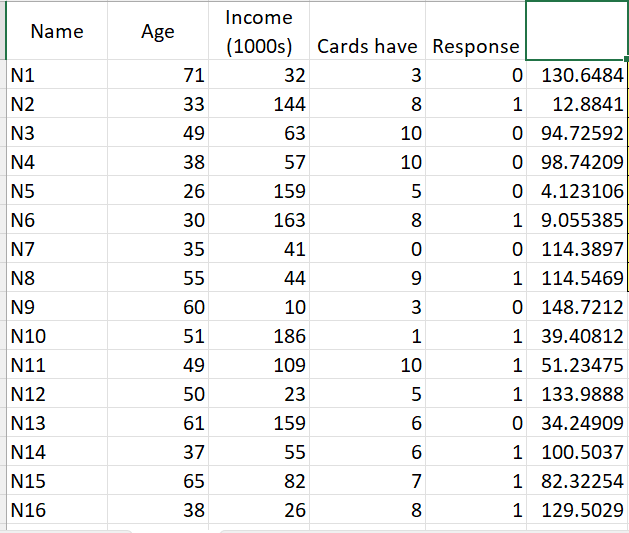




Xác định response của điểm Dasy khi K = 3, 5, 7, 9, 11

Các làm tính khoảng các gữi điểm Dasy với các điểm trên file bằng công thức:

=SQRT((B2-$H$2)^2+(C2-$I$2)^2+(D2-$J$2)^2)



dung hàm small để tìm số nhỏ theo yêu cầu của đề:

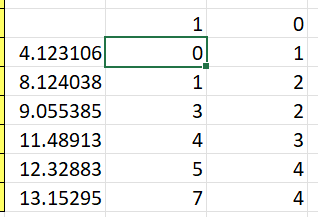
=SMALL($F$2:$F$201,G4)

Sau đó cùng count if để đếm số lượng Response nhỏ hơn hoặc bằng small

=COUNTIFS($E$2:$E$201,L$3,$F$2:$F$201,"<="&$K4)

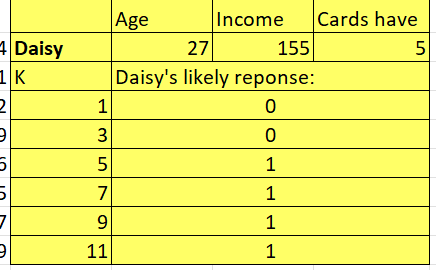
Có thểm làm tắt 2 hàm này:

=COUNTIFS($E$2:$E$201,L$3,$F$2:$F$201,"<="&$ SMALL($F$2:$F$201,$G4))



So sánh 2 giá trị response để xem giá trị nào lớn hơn thì trả về response đó

=IF(L4>M4,$L$3,$M$3)



Python

# Định nghĩa hàm tính khoảng cách Euclidean giữa hai điểm

def euclidean\_distance(point1, point2):

    squared\_distance = sum((x - y) \*\* 2 for x, y in zip(point1, point2))

    return squared\_distance \*\* 0.5

# Hàm KNN

def knn(X\_train, y\_train, X\_test, k):

    predictions = []

    for test\_point in X\_test:

        # Tính toán khoảng cách giữa điểm test và tất cả điểm trong tập train

        distances = []

        for i, train\_point in enumerate(X\_train):

            distance = euclidean\_distance(train\_point, test\_point)

            distances.append((distance, y\_train[i]))

        # Sắp xếp khoảng cách và chọn k điểm gần nhất

        distances.sort(key=lambda x: x[0])

        k\_nearest = distances[:k]

        # Dự đoán nhãn cho điểm test dựa trên đa số trong k điểm gần nhất

        counts = {}

        for \_, label in k\_nearest:

            counts[label] = counts.get(label, 0) + 1

        prediction = max(counts, key=counts.get)

        predictions.append(prediction)

    return predictions

# Dự đoán nhãn cho các điểm test sử dụng KNN

predictions = knn(X\_train, y\_train, X\_test, k)

print()

print("Predictions:", predictions)

So sánh

So sánh

DataSet

200 dòng

Khoảng 650 dòng

1000 dòng

Độ chính xác

Dung hàm accuracy\_score để đo độ chính xác

Độ chính xác giữa 2 thuật toán luôn luôn là: 1.0

Running time

Trong dữ liệu 200 dòng,

Thời gian chạy KNN tự viết: 0.002023935317993164

Thời gian chạy KNN từ scikit-learn: 0.001996755599975586

Trong dữ liệu 650 dòng:

Thời gian chạy KNN tự viết: 0.008977651596069336

Thời gian chạy KNN từ scikit-learn: 0.0019953250885009766

Trong dữ liệu 1000 dòng:

Thời gian chạy KNN tự viết: 0.016963481903076172

Thời gian chạy KNN từ scikit-learn: 0.002980470657348633