Decision Tree:

Ưu điểm:

Dễ hiểu và dễ giải thích.

Không đòi hỏi nhiều dữ liệu để huấn luyện.

Có khả năng làm việc với dữ liệu phi cấu trúc và có thể xử lý cả dữ liệu dạng số và dạng không số.

Nhược điểm:

Dễ bị quá mức phức tạp và quá mức khớp (overfitting) với dữ liệu huấn luyện.

Khả năng tổng quát hóa kém trên dữ liệu mới.

Random Forest:

Ưu điểm:

Có khả năng giảm tác động của quá mức phức tạp và quá mức khớp thông qua việc kết hợp nhiều cây quyết định.

Hoạt động tốt trên dữ liệu có nhiều biến đặc trưng.

Nhược điểm:

Không dễ giải thích như một cây quyết định đơn lẻ.

Cần nhiều dữ liệu hơn so với Decision Tree để huấn luyện hiệu quả.

LSTM (Long Short-Term Memory):

Ưu điểm:

Phù hợp cho việc xử lý chuỗi thời gian và dữ liệu có sự phụ thuộc dài hạn.

Có khả năng học được các mối quan hệ phức tạp trong dữ liệu chuỗi thời gian.

Được xem là hiệu quả trong các bài toán dự đoán chuỗi thời gian.

Nhược điểm:

Đòi hỏi nhiều dữ liệu để huấn luyện hiệu quả.

Cần thời gian và tài nguyên tính toán lớn hơn so với Decision Tree và Random Forest.

Lựa chọn mô hình:

Nếu bạn muốn một mô hình dễ giải thích và không đòi hỏi nhiều dữ liệu, có thể sử dụng Decision Tree.

Nếu bạn muốn giảm tác động của quá mức phức tạp và quá mức khớp, cũng như làm việc tốt với dữ liệu có nhiều biến đặc trưng, có thể chọn Random Forest.

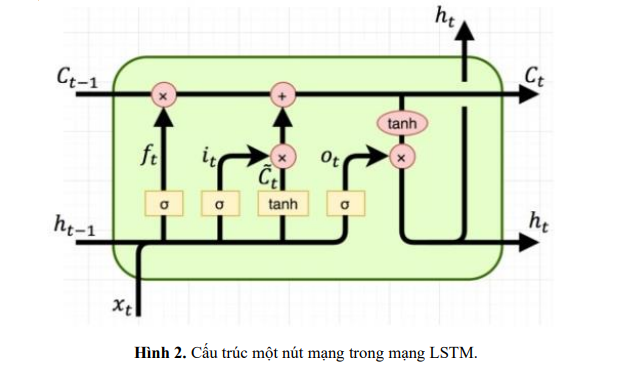
Nếu bạn đang làm việc với dữ liệu chuỗi thời gian và muốn mô hình có khả năng học được các mối quan hệ phức tạp, LSTM có thể là lựa chọn phù hợp.

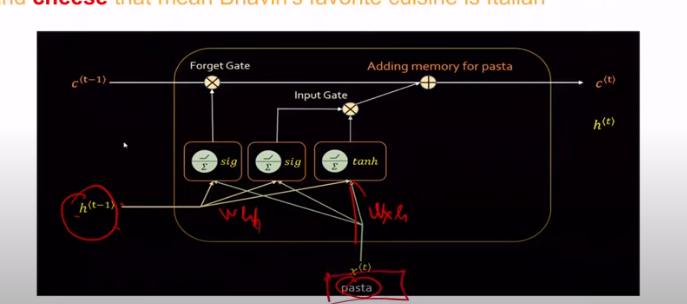
1. Tìm hiểu về LSTM
2. Các khái niệm

* LSTM: là một mạng cải tiến của mạng RNN nhằm giải quyết các vấn đề nhớ các bước dài của RNN
* Bài báo trình bày kết quả của nghiên cứu xây dựng một mô hình **mạng bộ nhớ dài–ngắn LSTM** (Long Short–Term Memory), **là một dạng đặc biệt của mạng nơ–ron hồi quy** (RNN–Recurrent Neural Network) để dự báo
* MSE: MSE = Σ (giá trị thực tế - giá trị dự đoán)² / số lượng quan sát
* Mạng LSTM được cải tiến từ mạng thần kinh hồi quy (RNN–Recurrent Neural Network) nhằm khắc phục những nhược điểm về phụ thuộc xa (Long–term Dependency) của mạng RNN truyền thống
* Về mặt lý thuyết, RNN có khả năng xử lý các phụ thuộc theo thời gian (temporal dependencies) bằng việc sử dụng bộ nhớ ngắn hạn và dựa trên việc xác định (luyện) các tham số một cách hiệu quả [12]. Tuy nhiên, đáng tiếc trong thực tế RNN không thể giải quyết các phụ thuộc theo thời gian khi chuỗi số liệu có các phụ thuộc xa (long–term dependencies)

1. Cấu trúc LSTM

* LSTM có cấu trúc dạng chuỗi các nút mạng như RNN, nhưng cấu trúc bên trong thì lại phức tạp hơn, bao gồm 4 tầng tương tác với nhau





LSTM: Mạng trí nhớ ngắn hạn định hướng dài hạn

Step1: Điểm đặc biệt của mạng LSTM nằm ở trạng thái **ô C (cell state),** **nơi lưu trữ các trọng số dài hạn của mô hình.**

Step2:Các thông số trạng thái **ô C**, trạng thái ẩn **h (hidden state),** đầu vào tại **thời điểm t xt** **được đưa vào nút mạng**.

Step3: Sau khi được xử lý qua các hàm kích hoạt sigmoid 𝜎, tanh và các phép toán véc–tơ, kết quả đầu ra là trạng thái ô C và trạng thái ẩn h tại thời điểm t sẽ được sử dụng cho nút mạng t+1 tiếp theo

**Hình ảnh cho thấy cấu trúc của một tế bào LSTM. Các thành phần chính của một tế bào LSTM là:**

* **Trạng thái ẩn (h): Đây là trạng thái của tế bào tại thời điểm hiện tại. Trạng thái ẩn được sử dụng để lưu trữ thông tin ngắn hạn và dài hạn.**
* **Trạng thái tế bào (c): Đây là trạng thái bên trong của tế bào. Trạng thái tế bào được sử dụng để lưu trữ thông tin dài hạn.**
* **Cổng quên (ft): Cổng quên điều chỉnh lượng thông tin từ trạng thái ẩn trước đó (ht-1) được truyền sang trạng thái ẩn hiện tại (ht).**
* **Cổng vào (it): Cổng vào điều chỉnh lượng thông tin mới (xt) được thêm vào trạng thái ẩn hiện tại (ht).**
* **Cổng ra (ot): Cổng ra điều chỉnh lượng thông tin từ trạng thái ẩn hiện tại (ht) được xuất ra làm đầu ra của tế bào (ot).**

**Các bước hoạt động của một tế bào LSTM như sau:**

1. **Đầu tiên, trạng thái ẩn trước đó (ht-1) và thông tin mới (xt) được kết hợp với nhau để tạo ra một vector kết hợp (u).**
2. **Vector kết hợp này sau đó được sử dụng để tính toán các giá trị của các cổng ft, it và ot.**
3. **Các giá trị của các cổng này được sử dụng để cập nhật trạng thái tế bào (c) và trạng thái ẩn (h).**

**Cụ thể, các bước này được thực hiện như sau:**

**Cập nhật trạng thái tế bào:**

**c = ft \* ct-1 + it \* tanh(W\_c \* xt + U\_c \* ht-1)**

**Trong đó:**

* **c là trạng thái tế bào mới**
* **ft là giá trị của cổng quên**
* **ct-1 là trạng thái tế bào trước đó**
* **it là giá trị của cổng vào**
* **xt là thông tin mới**
* **W\_c là trọng số của cổng quên**
* **U\_c là trọng số của cổng vào**

**Cập nhật trạng thái ẩn:**

**h = ot \* tanh(c)**

**Trong đó:**

* **h là trạng thái ẩn mới**
* **ot là giá trị của cổng ra**
* **c là trạng thái tế bào mới**

**Các cổng ft, it và ot được tính toán như sau:**

**Cổng quên:**

**ft = sigmoid(W\_f \* xt + U\_f \* ht-1)**

**Trong đó:**

* **ft là giá trị của cổng quên**
* **W\_f là trọng số của cổng quên**
* **U\_f là trọng số của cổng quên**

**Cổng vào:**

**it = sigmoid(W\_i \* xt + U\_i \* ht-1)**

**Trong đó:**

* **it là giá trị của cổng vào**
* **W\_i là trọng số của cổng vào**
* **U\_i là trọng số của cổng vào**

**Cổng ra:**

**ot = sigmoid(W\_o \* xt + U\_o \* ht-1)**

**Trong đó:**

* **ot là giá trị của cổng ra**
* **W\_o là trọng số của cổng ra**
* **U\_o là trọng số của cổng ra**

**LSTM là một loại mạng nơ-ron truy hồi (RNN) có thể lưu trữ và truy xuất thông tin trong thời gian dài. Điều này khiến LSTM trở thành một lựa chọn lý tưởng cho các nhiệm vụ đòi hỏi phải xử lý dữ liệu theo trình tự, chẳng hạn như dịch ngôn ngữ, tổng hợp văn bản và phân tích cảm xúc.**

**CÁC SIÊU THAM SỐ LSTM**

**Dưới đây là một số siêu tham số quan trọng mà bạn cần xem xét khi xây dựng mô hình LSTM:**

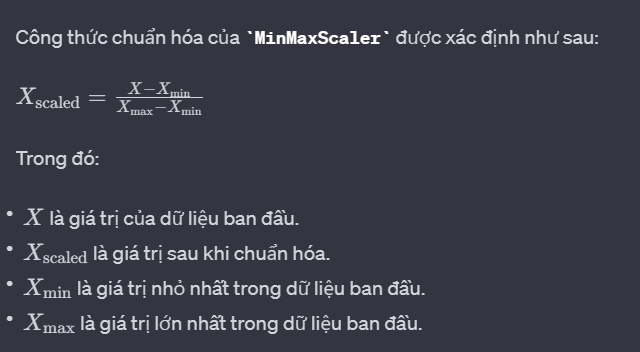
1. **Units (units):**
   * **Số lượng đơn vị LSTM (nút) trong lớp LSTM. Xác định số lượng bộ nhớ trong mô hình.**
2. **Activation Function (activation):**
   * **Hàm kích hoạt được sử dụng trong lớp LSTM. Các lựa chọn phổ biến bao gồm 'tanh', 'relu', 'sigmoid', và 'softmax'.**
3. **Input Shape (input\_shape):**
   * **Kích thước của dữ liệu đầu vào cho mỗi mẫu, xác định số lượng thời điểm và số lượng đặc trưng.**
4. **Batch Size (batch\_size):**
   * **Số lượng mẫu dữ liệu được sử dụng trong mỗi lần cập nhật trọng số. Ảnh hưởng đến quản lý tài nguyên tính toán.**
5. **Epochs (epochs):**
   * **Số lượng lần duyệt qua toàn bộ tập huấn luyện. Ảnh hưởng đến quá trình học và có thể dẫn đến overfitting nếu quá cao.**
6. **Learning Rate (learning\_rate):**
   * **Tốc độ học, quyết định bước chân (step size) trong quá trình tối ưu hóa.**
7. **Optimizer (optimizer):**
   * **Thuật toán tối ưu hóa, như 'adam', 'sgd' (Stochastic Gradient Descent), 'rmsprop',...**
8. **Loss Function (loss):**
   * **Hàm mất mát, thường là 'mse' (Mean Squared Error) cho bài toán hồi quy.**
9. **Dropout (dropout):**
   * **Tỷ lệ dropout, giúp giảm overfitting bằng cách loại bỏ ngẫu nhiên một số nơ-ron trong quá trình huấn luyện.**
10. **Recurrent Dropout (recurrent\_dropout):**
    * **Tỷ lệ dropout cho các kết nối tái lập (recurrent connections) trong lớp LSTM.**

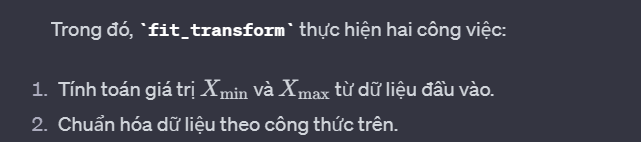
**Các siêu tham số trên đây là những siêu tham số cơ bản và quan trọng. Tùy thuộc vào bài toán cụ thể và tập dữ liệu, bạn có thể thêm vào danh sách các siêu tham số khác hoặc tinh chỉnh giá trị của chúng. Quá trình tinh chỉnh siêu tham số thường đòi hỏi thử nghiệm và đánh giá kỹ lưỡng để đạt được hiệu suất tốt nhất cho mô hình của bạn.**

**Câu 1: tại sao dùng hàm kích hoạt là rlu mà không phải là tanh hay sigmoid**

* Hàm kích hoạt "relu" (Rectified Linear Unit) thường được sử dụng trong các lớp ẩn của mạng nơ-ron, bao gồm cả LSTM, vì nó giúp mô hình học được các biểu diễn phi tuyến tính và có thể giảm vấn đề biến mất đạo hàm (vanishing gradient problem). "relu" đơn giản chỉ giữ giá trị dương và đặt giá trị âm về 0.
* Trong khi đó, hàm kích hoạt "sigmoid" thường được sử dụng trong các lớp cuối cùng của mô hình để chuyển đổi giá trị đầu ra thành một phạm vi giữa 0 và 1, phù hợp cho các bài toán như dự đoán xác suất (binary classification).
* Tuy nhiên, không có quy tắc cứng nhắc về việc sử dụng relu hoặc sigmoid. Thực tế, việc lựa chọn này có thể phụ thuộc vào đặc tính của bài toán và trải nghiệm thực tế. Thử nghiệm và điều chỉnh các tham số, bao gồm hàm kích hoạt, thường là một phần quan trọng của việc xây dựng mô hình học máy.

Câu 2: sử dụng hàm minmaxScaler() để chuẩn háo dữ liệu



****

**Chức năng chính của fit\_transform():**

1. **Fit**: Hàm này học các thông số từ dữ liệu huấn luyện. Trong quá trình học, nó tìm hiểu và áp dụng các tham số (như mean, variance, phạm vi, các thông số cần thiết cho việc chuẩn hóa) từ tập dữ liệu đầu vào.
2. **Transform**: Sau khi đã học các thông số từ dữ liệu huấn luyện, hàm này thực hiện việc biến đổi dữ liệu theo các thông số đã học để chuẩn bị dữ liệu để sử dụng cho mô hình hoặc phân tích.

**Câu 2: tại sao chọn hàm kích hoạt là “relu” mà không phải sigmoid hay tanh**

**Dưới đây là một số lý do mà hàm kích hoạt relu thường được sử dụng trong các tầng LSTM:**

1. **Tránh vấn đề biến mất đạo hàm (vanishing gradient): Hàm relu có đặc tính giữ lại giá trị dương và bỏ qua giá trị âm, điều này giúp giảm thiểu vấn đề biến mất đạo hàm, đặc biệt là khi lan truyền ngược trong quá trình huấn luyện mô hình.**
2. **Hạn chế giá trị đầu ra: Hàm relu không bao giờ đầu ra giá trị nhỏ hơn 0, điều này giúp mô hình học được các biểu diễn phi tuyến tính trong dữ liệu.**

**Tuy nhiên, trong một số trường hợp, bạn có thể muốn sử dụng các hàm kích hoạt khác như sigmoid hoặc tanh tùy thuộc vào yêu cầu cụ thể:**

* **Sigmoid: Thường được sử dụng trong tầng "output" của mô hình LSTM khi bạn đang giải quyết bài toán nhị phân hoặc bài toán phân loại có đầu ra giữa 0 và 1.**
* **Tanh: Cũng giữ lại giá trị dương và âm như relu, nhưng có phạm vi đầu ra là [-1, 1]. Các giá trị âm được giữ lại, giúp mô hình học được biểu diễn đối với các giá trị có thể là âm.**
* **Sigmoid:**
  + Hàm Sigmoid bão hào và triệt tiêu gradient.
  + Một nhược điểm dễ nhận thấy là khi đầu vào có trị tuyệt đối lớn (rất âm hoặc rất dương), gradient của hàm số này sẽ rất gần với 0. Điều này đồng nghĩa với việc các hệ số tương ứng với unit đang xét sẽ gần như không được cập nhật (còn được gọi là *vanishing gradient*).
* Tanh:
* Relu:
* **(+)** Tốc độ hội tụ nhanh hơn hẳn. ReLU có tốc độ hội tụ nhanh gấp 6 lần Tanh ([Krizhevsky et al.](http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf)). Điều này có thể do ReLU không bị bão hoà ở 2 đầu như Sigmoid và Tanh.
* **(+)** Tính toán nhanh hơn. Tanh và Sigmoid sử dụng hàm **exp** và công thức phức tạp hơn ReLU rất nhiều do vậy sẽ tốn nhiều chi phí hơn để tính toán.
* **(-)** Tuy nhiên ReLU cũng có một nhược điểm: Với các node có giá trị nhỏ hơn 0, qua ReLU activation sẽ thành 0, hiện tượng đấy gọi là “Dying ReLU“. Nếu các node bị chuyển thành 0 thì sẽ không có ý nghĩa với bước linear activation ở lớp tiếp theo và các hệ số tương ứng từ node đấy cũng không được cập nhật với gradient descent. => Leaky ReLU ra đời.
* **(-)** Khi learning rate lớn, các trọng số (weights) có thể thay đổi theo cách làm tất cả neuron dừng việc cập nhật.
* Relu(Rectified Linear Unit) : đơn vị tuyến tính đã được chỉnh sửa

**Câu 3: tại sao không dùng hàm relu cho RNN để giải quết vấn đề mất mát đạo hàm mà lại phải xây dựng nên LSTM**

* Một vấn đề chính của các mô hình **RNN truyền thống** là **vấn đề biến mất đạo hàm** (vanishing gradient problem) **khi xử lý các chuỗi dài**. Trong quá trình lan truyền ngược, **các gradient của hàm mất mát có thể trở nên rất nhỏ**, dẫn đến việc mô hình **không học được mối quan hệ giữa các bước thời gian xa nhau trong chuỗi**. Điều này làm giảm khả năng của RNN trong việc capture thông tin từ quá khứ trong chuỗi dài.

Câu 4: cách hoạt động của hàm LSTM

1. **Thách thức cơ bản: Quên quá nhanh**
   * Trong một số trường hợp, các mô hình thông thường có thể gặp vấn đề là quên thông tin quá nhanh khi xử lý dữ liệu chuỗi, như trong các tình huống dự đoán từ ngữ trong một câu.
2. **LSTM giải quyết vấn đề này bằng cách thêm một bộ nhớ dài hạn:**
   * LSTM giống như một người giữ ghi chú, giữ lại những điều quan trọng và quên đi những điều không quan trọng.
3. **Cấu trúc của LSTM:**
   * LSTM có một cấu trúc phức tạp, nhưng có thể tưởng tượng nó như một người có ba phần quan trọng:
     + **Cổng quên (Forget Gate):** Quyết định thông tin nào nên được giữ lại và thông tin nào nên bị quên.
     + **Cổng đầu vào (Input Gate):** Quyết định thông tin mới nào nên được thêm vào bộ nhớ.
     + **Cổng đầu ra (Output Gate):** Quyết định thông tin nào nên truyền ra ngoài từ bộ nhớ.
4. **Cách nó hoạt động:**
   * Khi một dữ liệu mới đến, LSTM quyết định xem nó có ý nghĩa gì và liệu có cần ghi nhớ không.
   * Cổng quên quyết định xem thông tin nào cần giữ lại từ bộ nhớ trước đó và thông tin nào cần quên đi.
   * Cổng đầu vào quyết định thông tin mới nào cần được thêm vào bộ nhớ.
   * Bộ nhớ lưu giữ thông tin từ quá khứ và thêm thông tin mới vào.
   * Cổng đầu ra quyết định thông tin nào sẽ truyền đi từ bộ nhớ.
5. **Ưu điểm:**
   * LSTM giúp mô hình học được và giữ lại thông tin quan trọng từ dữ liệu chuỗi dài mà không gặp vấn đề quên thông tin quá nhanh.
6. **Ứng dụng thực tế:**
   * LSTM thường được sử dụng trong nhiều ứng dụng, như dự đoán chuỗi thời gian, dịch máy, và xử lý ngôn ngữ tự nhiên.

Tóm lại, LSTM có thể được tưởng tượng như một trợ lý thông minh có khả năng giữ lại thông tin quan trọng và quên thông tin không quan trọng, giúp mô hình xử lý dữ liệu chuỗi một cách hiệu quả hơn.

Câu 5: sử dụng hàm kích hoạt trong LSTM

Trong mô hình LSTM, có một số hàm kích hoạt được sử dụng để định rõ các trạng thái của các cổng khác nhau. Dưới đây là các hàm kích hoạt phổ biến mà bạn có thể gặp khi làm việc với LSTM:

1. **Sigmoid (hàm kích hoạt sigmoid):**
   * Thường được sử dụng cho cổng quên (forget gate) và cổng đầu vào (input gate).
   * Được chọn vì đầu ra của sigmoid nằm trong khoảng (0, 1), giúp kiểm soát việc quên thông tin hoặc thêm thông tin mới vào bộ nhớ.
2. **Tanh (hàm kích hoạt tanh):**
   * Thường được sử dụng cho trạng thái bộ nhớ (cell state).
   * Tạo ra đầu ra nằm trong khoảng (-1, 1), giúp mô hình xử lý thông tin với biên độ lớn hơn.
3. **ReLU (Rectified Linear Unit):**
   * Thường được sử dụng cho đầu ra của các cổng đầu ra (output gate) hoặc đầu ra của bước thời gian hiện tại.
   * Giúp mô hình học được các đặc trưng phi tuyến tính.

* **Sử dụng hàm kích hoạt cho từng cổng**

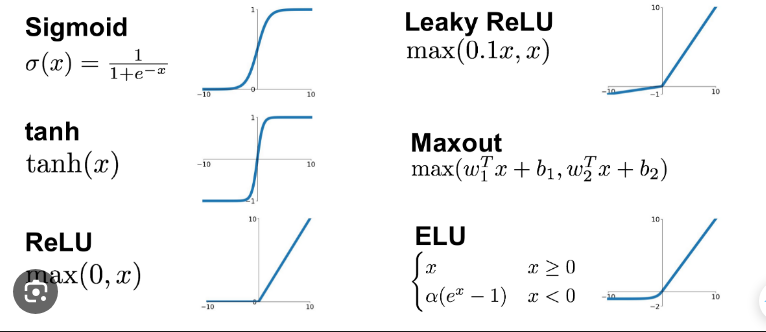
1. **Cổng Quên (Forget Gate):** Quyết định thông tin nào sẽ được giữ lại và thông tin nào sẽ bị quên từ trạng thái tế bào trước đó. Thông thường sử dụng hàm kích hoạt sigmoid để tạo ra giá trị từ 0 đến 1.
2. **Cổng Đầu Vào (Input Gate):** Quyết định thông tin mới nào sẽ được thêm vào trạng thái tế bào. Thông thường sử dụng hàm kích hoạt sigmoid để tạo ra giá trị từ 0 đến 1.
3. **Tanh Gate:** Tạo ra một vector của các giá trị ứng với các trạng thái ứng viên mới. Thông thường sử dụng hàm kích hoạt tanh để tạo ra giá trị từ -1 đến 1.
4. **Cổng Đầu Ra (Output Gate):** Quyết định giá trị thực tế của trạng thái tế bào. Thông thường sử dụng hàm kích hoạt sigmoid để tạo ra giá trị từ 0 đến 1.

**Câu: vấn đề của RNN**

* **Gradient quá lớn :** gradient có thể tăng lên quá lớn khi lan truyền qua nhiều tần trong mô hình, quá trình này gọi **là quá trình làm tăng gradient**
* **Gradient quá nhỏ :** gradient giảm dần đến mức quá nhỏ khi lan truyền qua nhiều tầng trong mô hình dàn trải của mạng no-ron hồi quy , quá trình này gọi **là quá trình triệt tiêu các gradient**

**ReLU có thể giúp giải quyết vấn đề mất mát đạo hàm và đạo hàm quá lớn, nhưng nó không phải lựa chọn tốt nhất cho mạng RNN (Recurrent Neural Network) và thường không được áp dụng trực tiếp vào RNN.**

**Điều này bởi vì RNN tồn tại một vấn đề khác gọi là "vanishing gradient" (đạo hàm biến mất).**

****

**Cú Pháp:**

**drop\_duplicates có thể được sử dụng với cú pháp sau:**

**python**

**Copy code**

**DataFrame.drop\_duplicates(subset=None, keep='first', inplace=False, ignore\_index=False)**

**subset: Các cột để kiểm tra sự trùng lặp. Nếu không được chỉ định, tất cả các cột sẽ được xem xét.**

**keep: Xác định xem phải giữ lại hàng đầu tiên hay cuối cùng khi có hàng trùng lặp. Giá trị mặc định là 'first'.**

**inplace: Nếu True, thay đổi trực tiếp trên DataFrame và không trả về DataFrame mới.**

**ignore\_index: Nếu True, tạo lại chỉ số (index) của DataFrame sau khi loại bỏ hàng trùng lặp.**