**MỤC ĐÍCH ĐỀ TÀI**

**Phần 1: Tổng quan**

* 1. **Mục tiêu**

Tìm hiểu mô hình LSTM và bài toán dự báo trên chuỗi thời gian.

## 1.2 Bài toán chuỗi thời gian – time series

* Dữ liệu chuỗi thời gian là tập hợp các quan sát thu được thông qua việc đo lường lặp đi lặp lại theo thời gian. Khi vẽ các điểm này trên biểu đồ, một trong các trục của luôn là thời gian.
* Các chỉ số chuỗi thời gian đề cập đến dữ liệu được theo dõi theo từng khoảng thời gian cụ thể. Ví dụ như số lượng hàng tồn kho được bán trong cửa hàng từ ngày này sang ngày khác.
* Dữ liệu chuỗi thời gian xuất hiện ở khắp mọi nơi, vì thời gian là thành phần của mọi thứ có thể quan sát được. Khi thế giới của chúng ta ngày càng được trang bị nhiều thiết bị hơn, các cảm biến và hệ thống liên tục phát ra dòng dữ liệu chuỗi thời gian không ngừng. Dữ liệu như vậy có nhiều ứng dụng trong các ngành công nghiệp khác nhau như: Hoạt động điện trong não, số liệu lượng mưa, giá cổ phiếu, số lượng người đăng ký hàng tháng.

### 1.2.1 Phân tích trên chuỗi thời gian

Phân tích chuỗi thời gian là quá trình ghi lại dữ liệu tại các thời điểm cụ thể trong một chu kỳ để phát hiện mô hình, tính chất theo mùa và yếu tố bất thường, nhằm dự báo sự kiện tương lai. Quá trình này bao gồm đánh giá các biến đổi trong dữ liệu quá khứ và dự đoán giá trị tương lai. Phân tích này trích xuất thông tin thống kê và đặc tính dữ liệu từ các quan sát liên tục theo thời gian.

Ví dụ, ghi lại lưu lượng giao thông tại một ngã tư theo từng giờ trong ngày giúp bạn có cái nhìn tổng quan về mẫu lưu lượng và dự đoán chính xác hơn trong tương lai nếu các điều kiện tương tự. Phân tích chuỗi thời gian nhận dạng mô hình từ dữ liệu quan sát để dự báo, điều này rất quan trọng trong máy học – lĩnh vực cho phép chương trình máy tính tự học và trở nên thông minh hơn theo thời gian mà không cần hướng dẫn cụ thể.

### 1.2.2 Cấu trúc và các thành phần của bài toán chuỗi thời gian

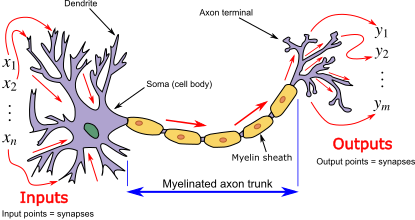
* *Xu hướng dài hạn (Long-term trend):* Xu hướng dài hạn đại diện cho hướng chung tổng quan của dữ liệu, bỏ qua bất kỳ tác động ngắn hạn nào như biến động mùa vụ hoặc nhiễu.
* *Mùa vụ (Seasonality):* Mùa vụ liên quan đến các biến động có chu kỳ được lặp lại trong suốt toàn bộ thời kỳ của chuỗi thời gian.
* *Tính không đổi (Stationarity):* Tính không đổi là một đặc tính quan trọng của chuỗi thời gian. Một chuỗi thời gian được coi là không đổi nếu trung bình, phương sai và hiệp biến của nó không thay đổi đáng kể theo thời gian. Có nhiều biến đổi mà có thể trích xuất phần không đổi từ một quá trình không đổi.
* *Nhiễu (Noise):* Mọi tập dữ liệu đều có nhiễu, đây là sự biến động hoặc biến thiên ngẫu nhiên do các yếu tố không kiểm soát.
* *Tự tương quan (Autocorrelation):* Tự tương quan là mối tương quan giữa chuỗi thời gian và phiên bản bị trễ của nó. Nó được sử dụng để xác định mùa vụ và xu hướng trong dữ liệu chuỗi thời gian.

## Hướng giải quyết

Các phương pháp dự báo chuỗi thời gian là các kỹ thuật được sử dụng để dự đoán hoạt động tương lai dựa trên dữ liệu lịch sử và mô hình hành vi đã quan sát. Một số phương pháp phổ biến bao gồm:

* Các mô hình cổ điển / thống kê: Bao gồm các phương pháp như Trung bình Di động, Bộ lọc mô hình suy giảm (Exponential Smoothing), Mô hình Tự hồi quy Tích hợp Trung bình Di động (ARIMA), Mô hình SARIMA, TBATS, vv.
* Máy học (Machine Learning): Bao gồm các phương pháp như Hồi quy Tuyến tính (Linear Regression), XGBoost, Random Forest, hoặc bất kỳ mô hình máy học nào kết hợp với các phương pháp giảm chiều dữ liệu.
* Học sâu (Deep Learning): Sử dụng mạng thần kinh sâu như Mạng Nơ-ron Tái lặp (RNN), Mạng Nơ-ron Bộ Nhớ Ngắn Dài Hạn (LSTM),...

Ở đây chúng em sử dụng mô hình LSTM, RNN, BiLSTM, GRU để giải quyết bài toán



**Phần 2: EDA – PHÂN TÍCH KHÁM PHÁ DỮ LIỆU**

Sử dụng tập dữ liệu chứa thông tin về giá chứng khoán của Yahoo. Dữ liệu được thu thập từ ngày 23 tháng 11 năm 2015 đến ngày 20 tháng 11 năm 2020.

Đường dẫn : [yahoo-stock-price.csv](https://www.kaggle.com/datasets/arashnic/time-series-forecasting-with-yahoo-stock-price)

Tổng quan

Tập dữ liệu có 1.825 dòng và 7 cột:

+ Date: ngày giao dịch cổ phiếu.

+ High: giá cao nhất của cổ phiếu vào ngày giao dịch.

+ Low: giá thấp nhất của cổ phiếu vào ngày giao dịch. Phù hợp cho các nhà đầu tư tìm kiếm các điểm vào lệnh tốt nhất trong ngày.

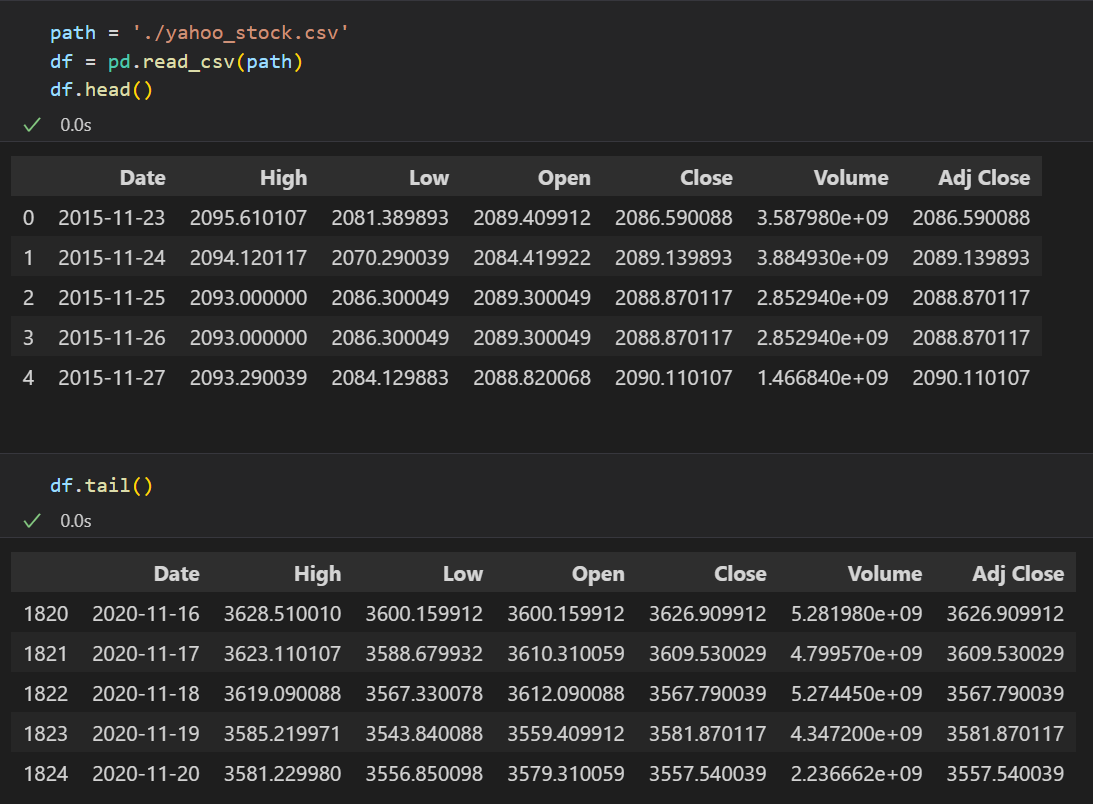
+ Open: giá mở cửa của cổ phiếu vào ngày giao dịch.

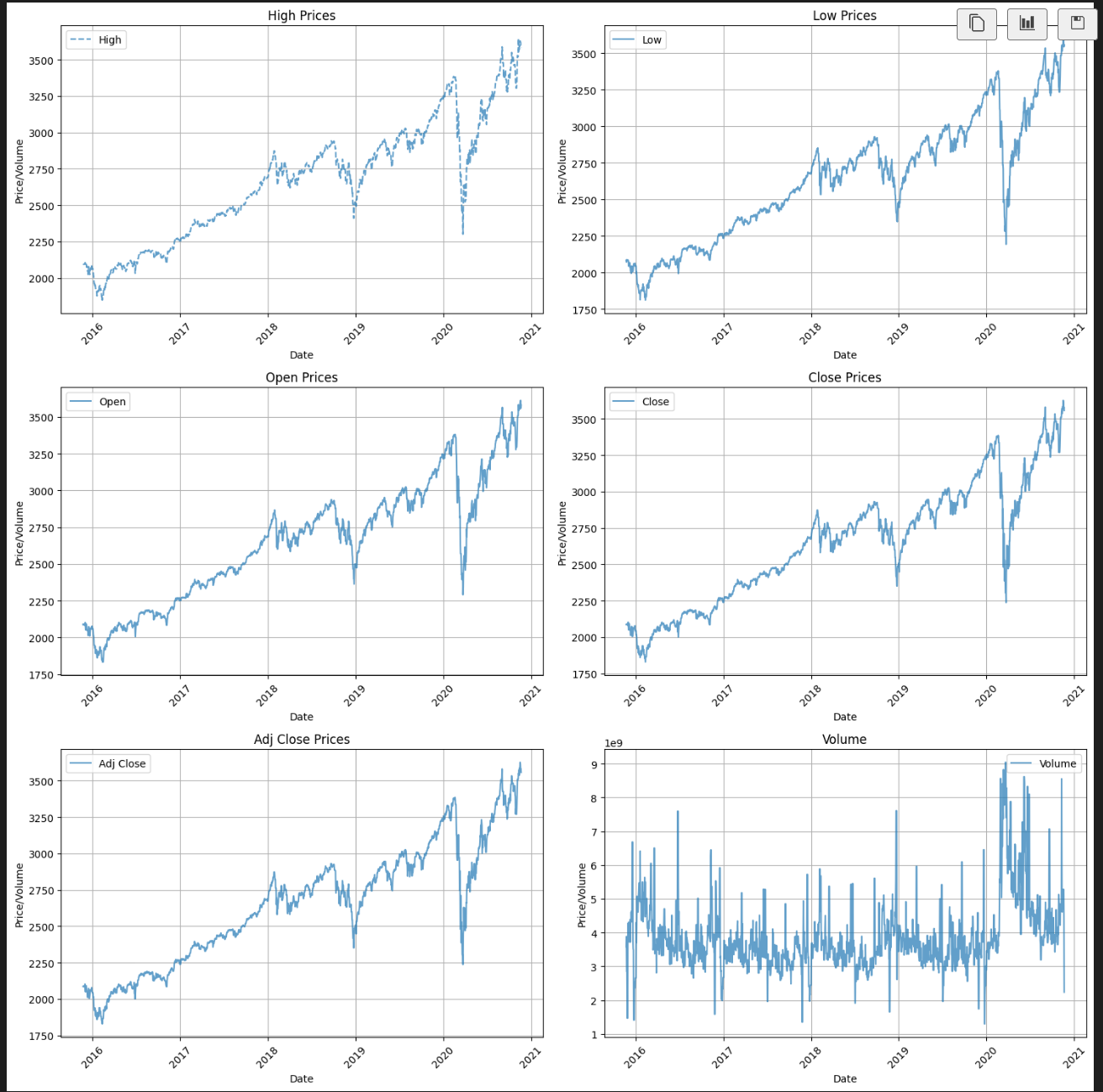
+ Close: giá đóng cửa của cổ phiếu vào ngày giao dịch. Phù hợp cho các nhà đầu tư và nhà phân tích muốn đưa ra quyết định giao dịch hoặc đầu tư dựa trên giá cuối ngày.

+ Volume: khối lượng giao dịch của cổ phiếu trong ngày giao dịch.

+ Adj Close: giá đóng cửa điều chỉnh của cổ phiếu vào ngày giao dịch

Mục tiêu là dự đoán giá cao nhất (High) của cổ phiếu trong ngày giao dịch . Điều này rất hữu ích cho các chiến lược giao dịch ngắn hạn hoặc các nhà đầu tư muốn tối ưu hóa lợi nhuận từ biến động giá. Phù hợp cho các nhà giao dịch theo chiến lược mua thấp bán cao trong cùng một ngày giao dịch (day trading).





**Phần 3: Chuẩn hóa dữ liệu**

Sử dụng min-max scaler để chuẩn hóa dữ liệu



**Phần 4: Chia dữ liệu**

Vì đây là bài toán time series với dữ liệu được sắp xếp theo thời gian nên ta cần chia các tập dữ liệu train ,validation, test theo thứ tự thời gian với các phần dữ liệu độc lập nhau.

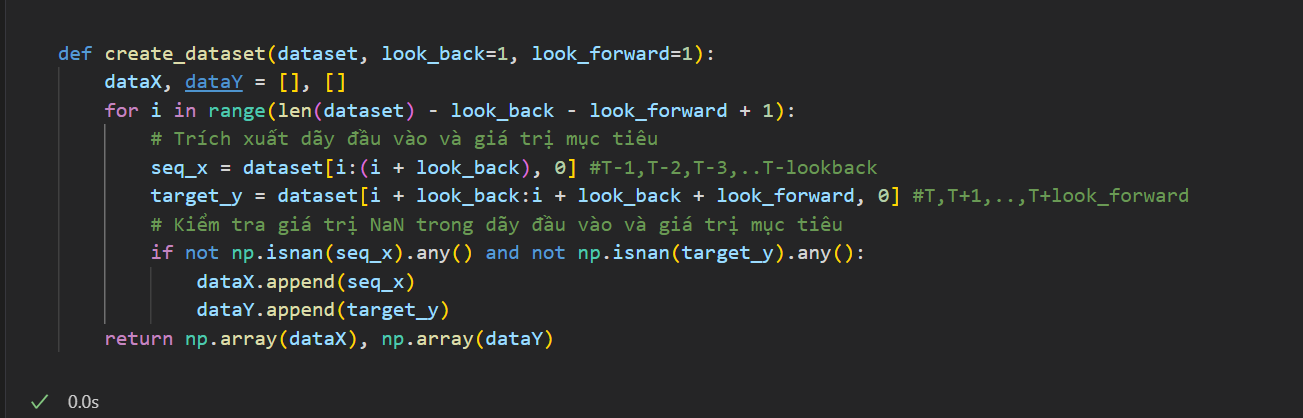


**Phần 5: Tạo ma trận dữ liệu đầu vào và đầu ra**

Trước khi xây dựng mô hình LSTM để dự báo chuỗi thời gian, một bước quan trọng là chuẩn bị dữ liệu đầu vào một cách phù hợp. Hàm create\_dataset được sử dụng để chuẩn bị dữ liệu cho mô hình LSTM trong bài toán dự đoán chuỗi thời gian. Nó tạo ra các mẫu dữ liệu từ chuỗi thời gian, sử dụng một cửa sổ trượt qua dữ liệu để xây dựng các đầu vào và đầu ra cho mô hình học sâu.

Với

* x là các giá trị đầu vào tức T-1 , T-2 ,..T-lookback
* y là các giá trị đầu ra tức T, T+1 , T+ look\_forward



**Phần 6: Tạo các mô hình**

## Vấn đề của các mạng truyền thống

Con người không bắt đầu tư duy lại từ đầu mỗi giây mà duy trì thông tin liên tục**. Các mạng nơ-ron truyền thống, như CNN, không hiệu quả trong việc xử lý dữ liệu có thứ tự thời gian hoặc chuỗi vì không thể duy trì thông tin từ các bước trước**. Ví dụ, mạng nơ-ron truyền thống khó có thể sử dụng thông tin từ các sự kiện trước trong một bộ phim để dự đoán các sự kiện sau.

## Simple RNN - Recurrent Neural Networks

Mạng nơ-ron hồi quy(RNNs) giải quyết vấn đề này bằng cách có các vòng lặp cho phép thông tin duy trì trong quá trình tính toán. Mỗi bước của mạng RNN có thể chuyển thông tin từ bước trước sang bước tiếp theo, giúp xử lý dữ liệu chuỗi hiệu quả hơn.



Xây dựng mô hình RNN thông qua hàm create\_rnn\_model với các lớp

* RNN Layer 1: Nhận đầu vào (chuỗi thời gian với số đơn vị ẩn = units) và trả về toàn bộ chuỗi đầu ra (tất cả các trạng thái ẩn cho mỗi thời điểm trong chuỗi đầu vào) nhờ ‘return\_sequences=True’
* Dropout Layer 1: Sử dụng kỹ thuật regularization hạn chế overfitting bằng cách tắt ngẫu nhiên một phần đơn vị (nút,neuron) trong quá trình huấn luyện.
* RNN Layer 2: Nhận chuỗi đầu ra từ lớp SimpleRNN đầu tiên và trả về trạng thái ẩn cuối cùng thay vì toàn bộ chuỗi đầu ra với ‘return\_sequences=False’
* Dropout Layer 2: Ngăn chặn overfitting tương tự như Dropout Layer 1.
* Dense Layer: Chuyển đổi trạng thái ẩn cuối cùng của SimpleRNN thành đầu ra cuối cùng.

(fully connected) với số đơn vị đầu ra ứng với output\_units.Lớp này kết nối toàn bộ đầu ra từ lớp SimpleRNN cuối cùng đến đầu ra cuối cùng của mô hình.

. Hàm trả về một mô hình đã được biên dịch sẵn sàng cho quá trình training, với hàm mất mát là ‘mean\_squared\_error’ và trình tối ưu hóa ‘adam’.



### Khả năng đặc biệt của RNN:

RNN có khả năng kết nối thông tin từ quá khứ với các tác vụ hiện tại, giúp cải thiện hiểu biết hoặc dự đoán tình huống hiện tại. Ví dụ, trong xử lý video, các khung hình trước cung cấp ngữ cảnh để hiểu khung hình hiện tại. Tuy nhiên, hiệu quả của RNN phụ thuộc vào bối cảnh tác vụ cụ thể.

Ví dụ :

* Dự đoán thời tiết: Dựa trên dữ liệu vài ngày gần đây, RNN có thể phân tích xu hướng thời tiết để dự đoán ngày tiếp theo. Thông tin cần thiết không phải từ quá xa trong quá khứ.
* Dự đoán thị trường chứng khoán: Để dự đoán xu hướng trong một quý hoặc năm, cần xem xét dữ liệu lịch sử dài hơn. Thị trường bị ảnh hưởng bởi các sự kiện kinh tế, chính trị từ nhiều năm trước.

### Hạn chế của RNN truyền thống:

RNN truyền thống gặp khó khăn trong việc **xử lý thông tin dài hạn** do vấn đề **biến mất gradient (vanishing gradient** - Khi gradient được tính toán và truyền ngược (Qúa trình Backpropagation) qua nhiều lớp trong quá trình train, chúng có thể bị giảm dần đến mức rất nhỏ do các phép nhân liên tiếp với các giá trị nhỏ hơn 1 (trong các hàm kích hoạt như sigmoid hoặc tanh,relu). Làm cho việc cập nhật trọng số của các lớp ban đầu rất chậm, dẫn đến mô hình học chậm hoặc thậm chí không học được gì từ dữ liệu), ***làm giảm khả năng kết nối các sự kiện quan trọng từ quá khứ xa (mất mát thông tin dài hạn). Vấn đề này có thể được giảm thiểu bằng cách sử dụng các hàm kích hoạt và phương pháp khởi tạo trọng số phù hợp.***

Kết quả là mạng không học được các đặc trưng quan trọng từ dữ liệu.

### Giải pháp với LSTM và GRU:

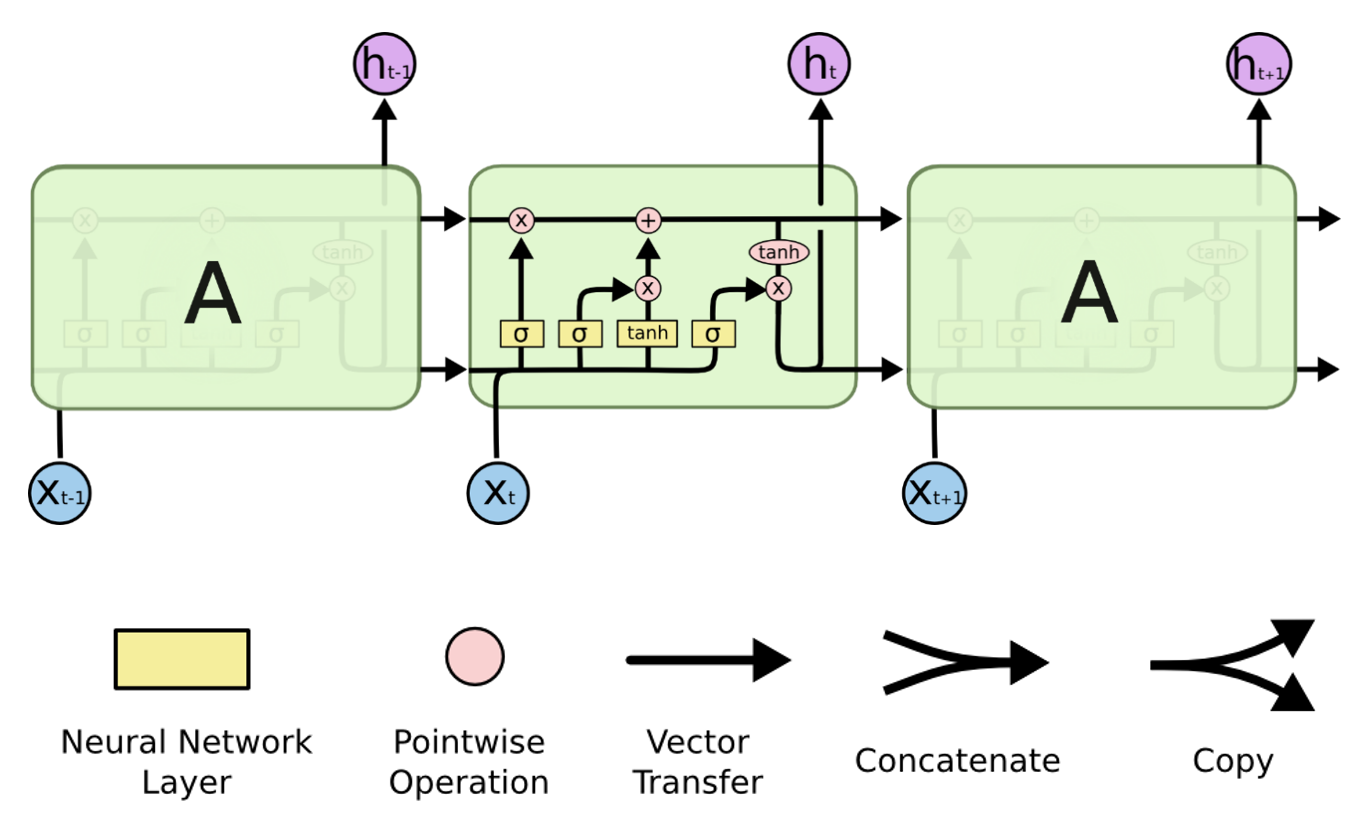
LSTM và GRU, với cơ chế cổng, cho phép duy trì thông tin quan trọng qua thời gian dài, hiệu quả hơn trong việc xử lý thông tin dài hạn so với RNN truyền thống giúp giải quyết vấn đề ***mất mát thông tin dài hạn***

## LSTM

Mạng Long Short Term Memory – Bộ Nhớ Ngắn Hạn Dài-"LSTMs" -đặc biệt của RNNs, có khả năng học các “long-term dependencies”. Được giới thiệu bởi Hochreiter & Schmidhuber (1997), và đã được tinh chỉnh và phổ biến hóa bởi nhiều người trong các nghiên cứu tiếp theo.

LSTM nổi bật với cấu trúc được gọi là "cell state," giống như một dải băng chuyền xuyên suốt mô hình. Điều này giúp LSTMs duy trì và truyền thông tin qua chuỗi dữ liệu dài mà không gặp phải sự suy giảm hoặc thay đổi đáng kể.

Trạng thái của ô trong LSTM hoạt động như một con đường thông tin, nơi mà dữ liệu có thể di chuyển qua toàn bộ chuỗi dữ liệu. Điều này được thực hiện với các tương tác tuyến tính đơn giản, giúp duy trì độ ổn định của thông tin. Khả năng này làm cho LSTM trở nên hiệu quả trong việc giữ thông tin quan trọng qua thời gian dài mà không bị mất mát hoặc biến dạng.

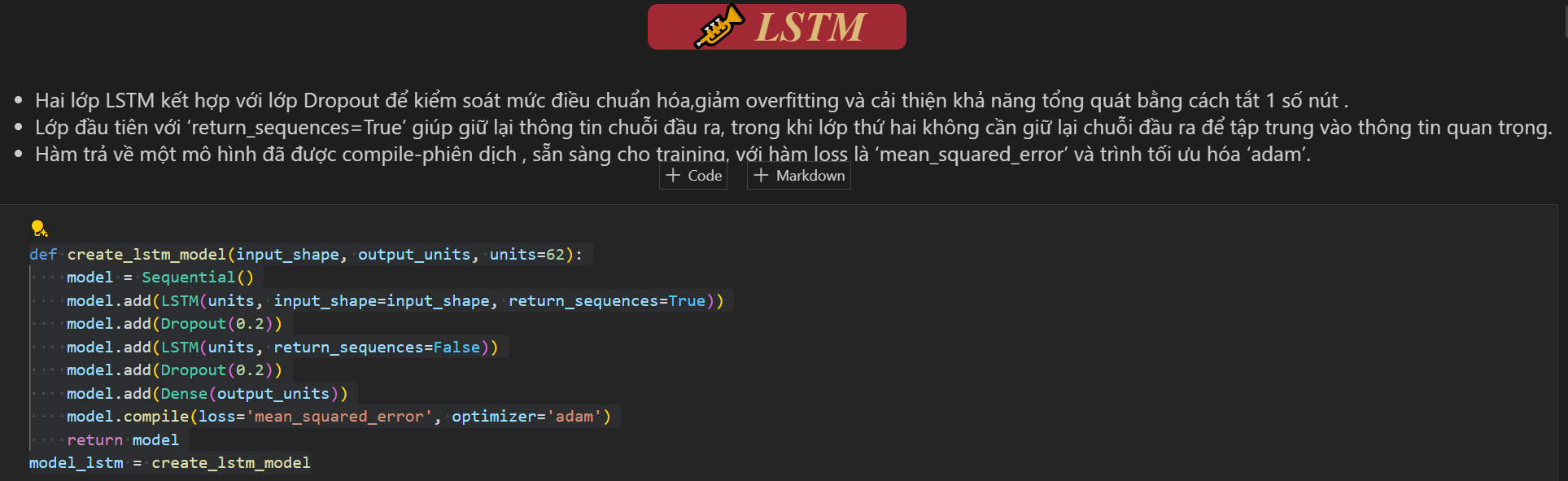


Tương tự Xây dựng mô hình LSTM thông qua hàm create\_lstm\_model với các lớp

* LSTM Layer 1: Nhận đầu vào (chuỗi thời gian với số đơn vị ẩn = units) và trả về toàn bộ chuỗi đầu ra (tất cả các trạng thái ẩn cho mỗi thời điểm trong chuỗi đầu vào) nhờ ‘return\_sequences=True’
* Dropout Layer 1: Sử dụng kỹ thuật regularization hạn chế overfitting bằng cách tắt ngẫu nhiên một phần đơn vị (nút) trong quá trình huấn luyện.
* LSTM Layer 2: Nhận chuỗi đầu ra từ lớp LSTM đầu tiên và trả về trạng thái ẩn cuối cùng thay vì toàn bộ chuỗi đầu ra với ‘return\_sequences=False’
* Dropout Layer 2: Ngăn chặn overfitting tương tự như Dropout Layer 1.
* Dense Layer: Chuyển đổi trạng thái ẩn cuối cùng của LSTM thành đầu ra cuối cùng.

(fully connected) với số đơn vị đầu ra ứng với output\_units.Lớp này kết nối toàn bộ đầu ra từ lớp LSTM cuối cùng đến đầu ra cuối cùng của mô hình.

. Hàm trả về một mô hình đã được biên dịch sẵn sàng cho quá trình training, với hàm mất mát là **‘mean\_squared\_error’** và trình tối ưu hóa ‘**adam’**.



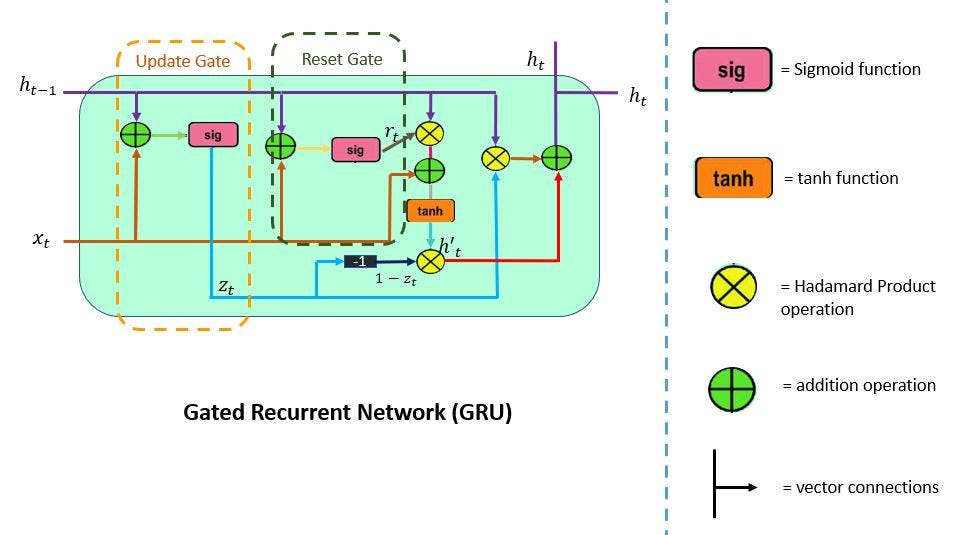
## GRU

Mô hình Gated Recurrent Unit (GRU) là một kiến trúc mạng nơ-ron hồi quy nâng cao, tương tự như LSTM nhưng với một cấu trúc đơn giản hơn . GRU được Kyunghyun Cho và cộng sự giới thiệu vào năm 2014 như một biến thể của LSTM với mục đích làm cho các mô hình RNN có thể hiệu quả hơn trong việc học các phụ thuộc dài hạn và giảm bớt vấn đề của **gradient vanishing** mà không cần đến bộ nhớ LSTM đầy đủ

GRU duy trì trạng thái ẩn giống như LSTM để chứa thông tin qua các bước thời gian, nhưng nó chỉ sử dụng hai cổng để kiểm soát quá trình học:

* Cổng Cập Nhật (Update Gate): Quyết định mức độ thông tin từ trạng thái ẩn trước đó sẽ được giữ lại và mức độ thông tin mới sẽ được thêm vào.
* Cổng Đặt Lại (Reset Gate): Quyết định mức độ thông tin trạng thái ẩn trước đó sẽ được bỏ qua, điều này cho phép mô hình bỏ qua thông tin không liên quan khi học các phụ thuộc.

Cả hai cổng này giúp GRU quản lý thông tin qua thời gian và giải quyết vấn đề của gradient biến mất bằng cách điều chỉnh luồng thông tin trong quá trình lan truyền ngược (backpropagation).



Tương tự Xây dựng mô hình GRU thông qua hàm create\_gru\_model bằng cách tạo 1 mô hình tuần tự ‘model = **Sequential()**’ với các lớp

* GRU Layer 1: Nhận đầu vào (chuỗi thời gian với số đơn vị ẩn = units) và trả về toàn bộ chuỗi đầu ra (tất cả các trạng thái ẩn cho mỗi thời điểm trong chuỗi đầu vào) nhờ ‘return\_sequences=True’
* Dropout Layer 1: Sử dụng kỹ thuật regularization hạn chế overfitting bằng cách tắt ngẫu nhiên một phần đơn vị (nút) trong quá trình huấn luyện.
* GRU Layer 2: Nhận chuỗi đầu ra từ lớp GRU đầu tiên và trả về trạng thái ẩn cuối cùng thay vì toàn bộ chuỗi đầu ra với ‘return\_sequences=False’
* Dropout Layer 2: Ngăn chặn overfitting tương tự như Dropout Layer 1.
* Dense Layer: Chuyển đổi trạng thái ẩn cuối cùng của GRU thành đầu ra cuối cùng.

(fully connected) với số đơn vị đầu ra ứng với output\_units.Lớp này kết nối toàn bộ đầu ra từ lớp GRU cuối cùng đến đầu ra cuối cùng của mô hình.

. Hàm trả về một mô hình đã được biên dịch sẵn sàng cho quá trình training, với hàm mất mát là ‘mean\_squared\_error’ và trình tối ưu hóa ‘adam’.

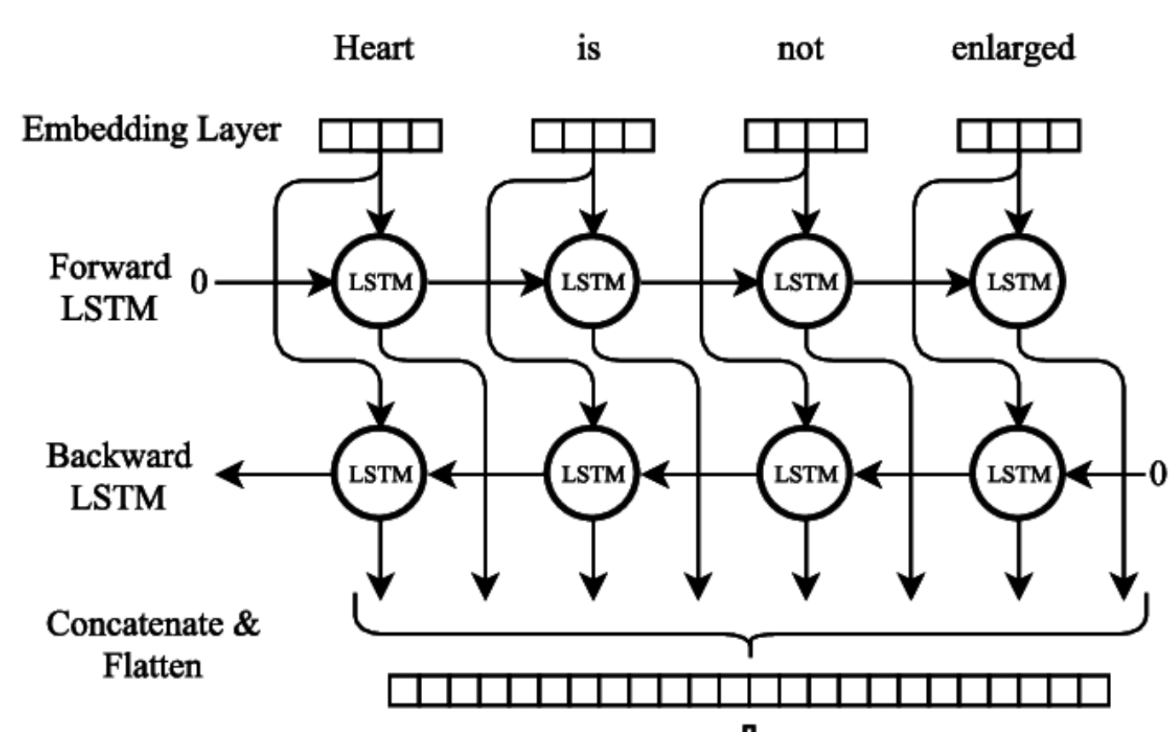


## BiLSTM

LSTM có xu hướng tập trung quá nhiều vào thông tin từ quá khứ và có thể bỏ sót thông tin quan trọng từ tương lai gần. BiLSTM giải quyết vấn đề này bằng cách sử dụng hai chiều, tức là nó xem xét thông tin từ cả quá khứ và tương lai của một điểm dữ liệu trong chuỗi, cải thiện khả năng hiểu và dự đoán dữ liệu chuỗi thời gian.

* Mỗi đơn vị LSTM có ba cổng: cổng quên (forget gate), cổng nhập (input gate) và cổng đầu ra (output gate). Những cổng này giúp mô hình quyết định thông tin nào sẽ được lưu giữ, cập nhật hoặc loại bỏ.
* Tính Chất Đa Hướng của BiLSTM: Điểm đặc biệt của BiLSTM - bao gồm hai lớp LSTM, mỗi lớp học một hướng của dữ liệu: một hướng tiến (forward direction) và một hướng lùi (backward direction). Điều này cho phép mô hình học được cả thông tin từ quá khứ (qua hướng tiến) và từ tương lai (qua hướng lùi) đối với mỗi điểm dữ liệu trong chuỗi.
* Xử lý Dữ liệu Chuỗi: Trong BiLSTM, dữ liệu được đưa qua cả hai lớp LSTM. Trong lớp hướng tiến, dữ liệu được xử lý từ đầu chuỗi đến cuối chuỗi, trong khi lớp hướng lùi xử lý dữ liệu theo hướng ngược lại, từ cuối chuỗi về đầu chuỗi.
* Kết hợp kết quả: Sau khi xử lý qua cả hai lớp LSTM, kết quả từ mỗi lớp được kết hợp lại với nhau. Điều này có thể được thực hiện bằng cách nối (concatenation) hoặc tổng hợp (summation) thông tin từ cả hai hướng.

BiLSTM phổ biến trong các tác vụ liên quan đến xử lý ngôn ngữ tự nhiên (NLP) như dịch máy, nhận dạng giọng nói, và phân tích cảm xúc. Nó giúp mô hình có khả năng hiểu được ngữ cảnh hai chiều của từng từ trong một câu.





Tương tự Xây dựng mô hình BiLSTM thông qua hàm create\_bi\_lstm\_model bằng cách tạo 1 mô hình tuần tự ‘model = **Sequential()**’ với các lớp

* Layer 1- Lớp LSTM song song, cho phép mạng nơ-ron học từ cả hai chiều (từ trước ra sau và từ sau ra trước).Nhận đầu vào (chuỗi thời gian với số đơn vị ẩn = units) và để trả về toàn bộ chuỗi đầu ra, giúp duy trì thông tin thứ tự thời gian. (tất cả các trạng thái ẩn cho mỗi thời điểm trong chuỗi đầu vào) nhờ ‘return\_sequences=True’
* Dropout Layer 1: Sử dụng kỹ thuật regularization hạn chế overfitting bằng cách tắt ngẫu nhiên một phần đơn vị (nút) trong quá trình huấn luyện.
* BiDirectional LSTM Layer 2: Nhận chuỗi đầu ra từ lớp BiLSTM đầu tiên và trả về trạng thái ẩn cuối cùng (đầu ra cuối cùng) thay vì toàn bộ chuỗi đầu ra với ‘return\_sequences=False’
* Dropout Layer 2: Ngăn chặn overfitting tương tự như Dropout Layer 1.
* Dense Layer: Chuyển đổi trạng thái ẩn cuối cùng của GRU thành đầu ra cuối cùng.

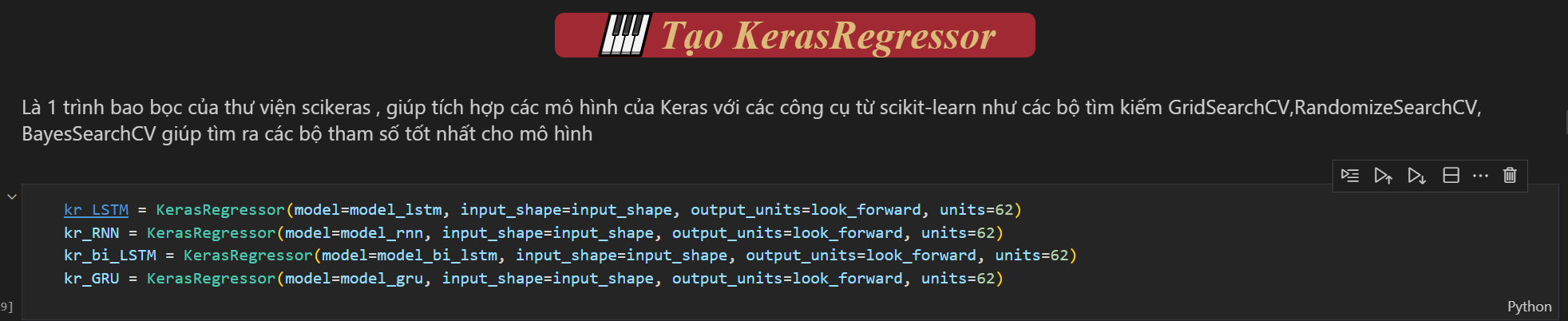
(fully connected) với số đơn vị đầu ra ứng với output\_units.Lớp này kết nối toàn bộ đầu ra từ lớp GRU cuối cùng đến đầu ra cuối cùng của mô hình.

Hàm trả về một mô hình đã được biên dịch sẵn sàng cho quá trình training, với hàm mất mát là ‘mean\_squared\_error’ và trình tối ưu hóa ‘adam’.

**Phần 7: Huấn luyện mô hình**

## Tạo KerasRegressor

Là 1 trình bao bọc của thư viện scikeras , giúp tích hợp các mô hình của Keras với các công cụ từ scikit-learn như các bộ tìm kiếm GridSearch,RandomizedSearchCV, BayesSearchCV giúp tìm ra các bộ tham số tốt nhất cho mô hình



## Tìm các bộ tham số tối ưu cho các mô hình

Để đạt được hiệu suất tối ưu cho mô hình học sâu, chúng ta có thể sử dụng các phương pháp tối ưu hóa như: Grid Search, Randomized Search, và BayesSearchCV

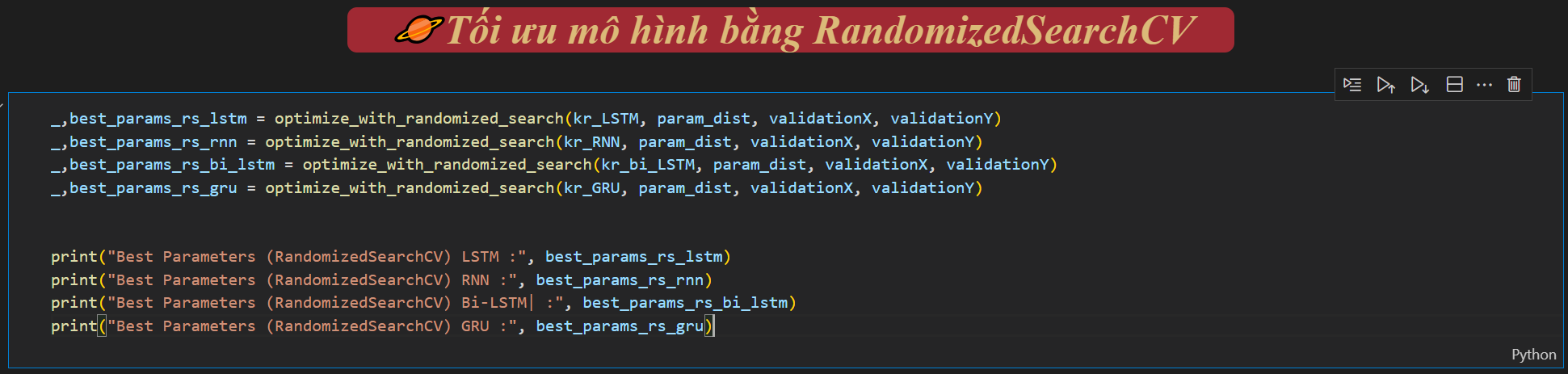
optimize\_with\_randomized\_search thực hiện tìm kiếm trong một không gian tham số được xác định trước bằng cách chọn ngẫu nhiên một số lượng cấu hình tham số nhất định và đánh giá hiệu suất của mô hình với mỗi cấu hình đó.

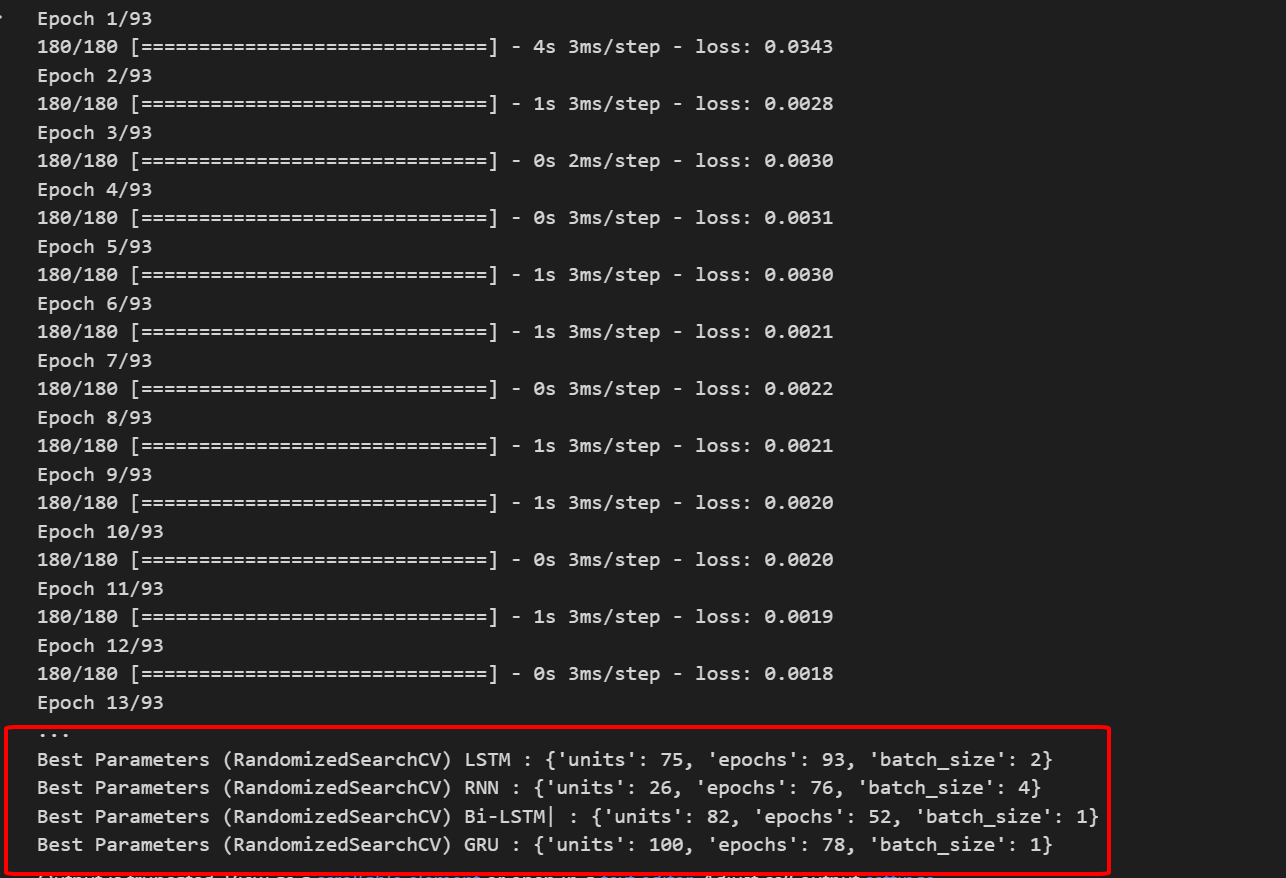
optimize\_with\_bayesian\_optimization sử dụng phương pháp tìm kiếm Bayesian bằng cách dùng mô hình xác suất, một phương pháp linh hoạt và hiệu quả trong việc tìm ra tham số tối ưu trên không gian tham số.

Bayes thường hiệu quả hơn vì nó sử dụng thông tin từ các lần thử trước để định hướng quá trình tìm kiếm.

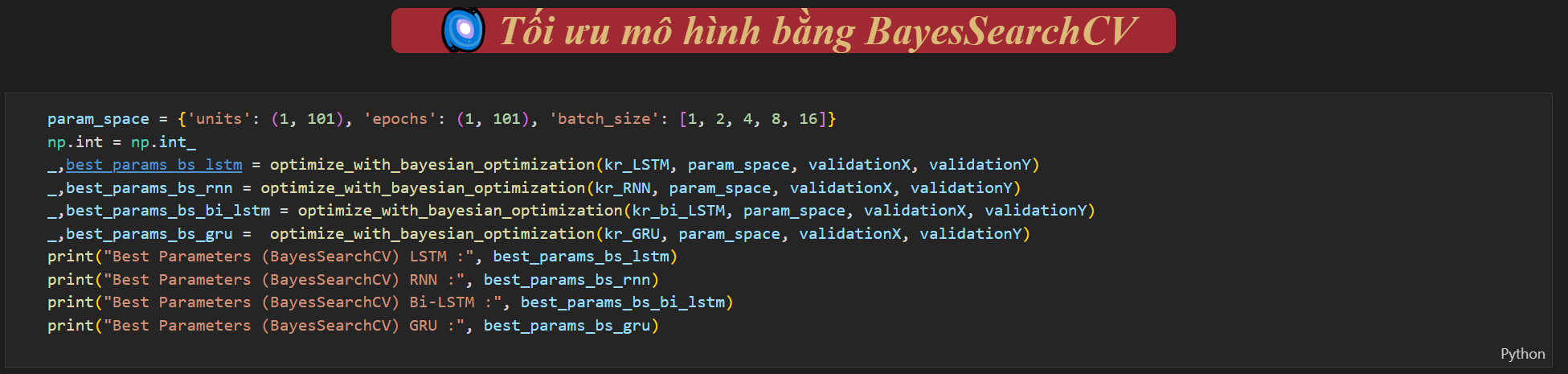


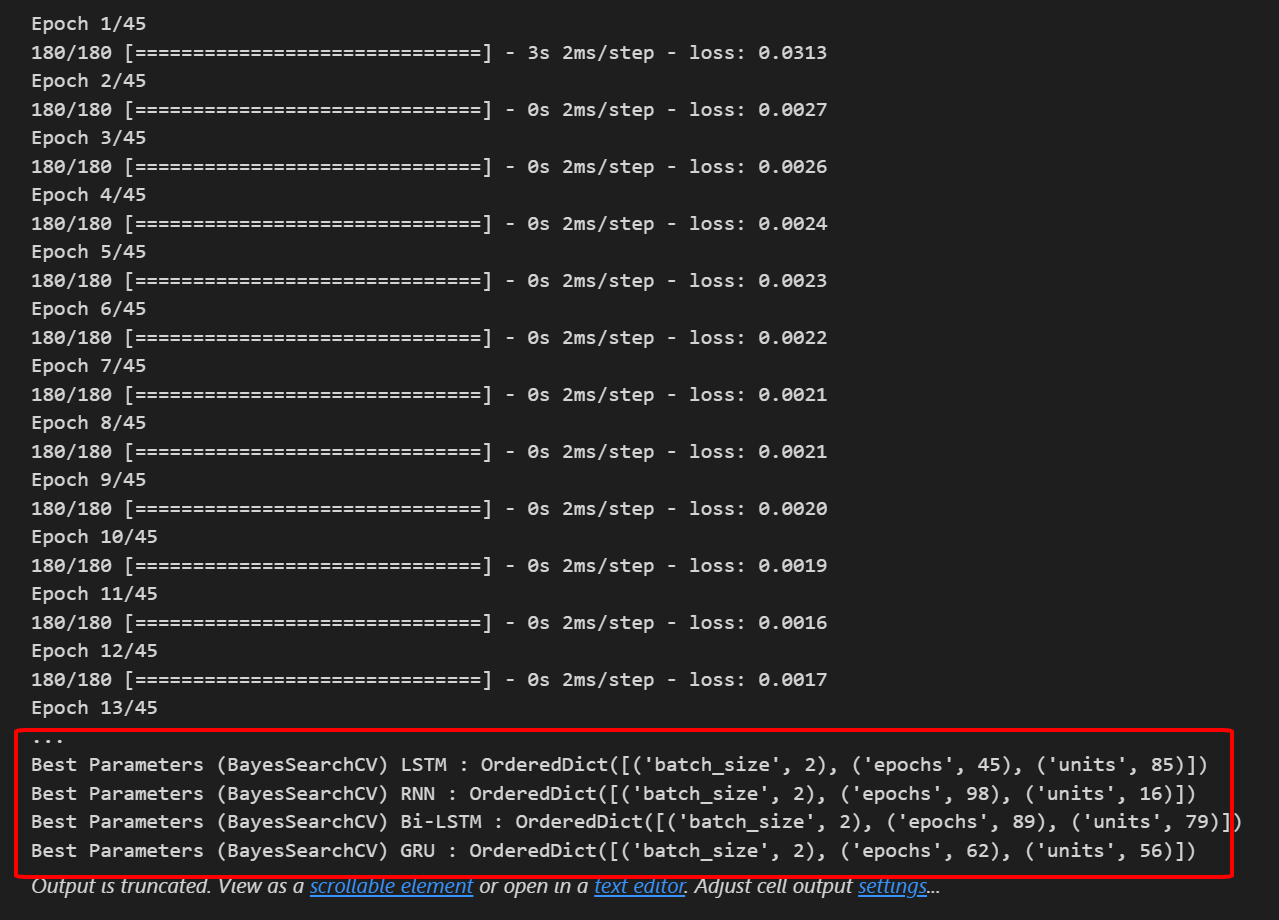
### RandomizedSearchCV





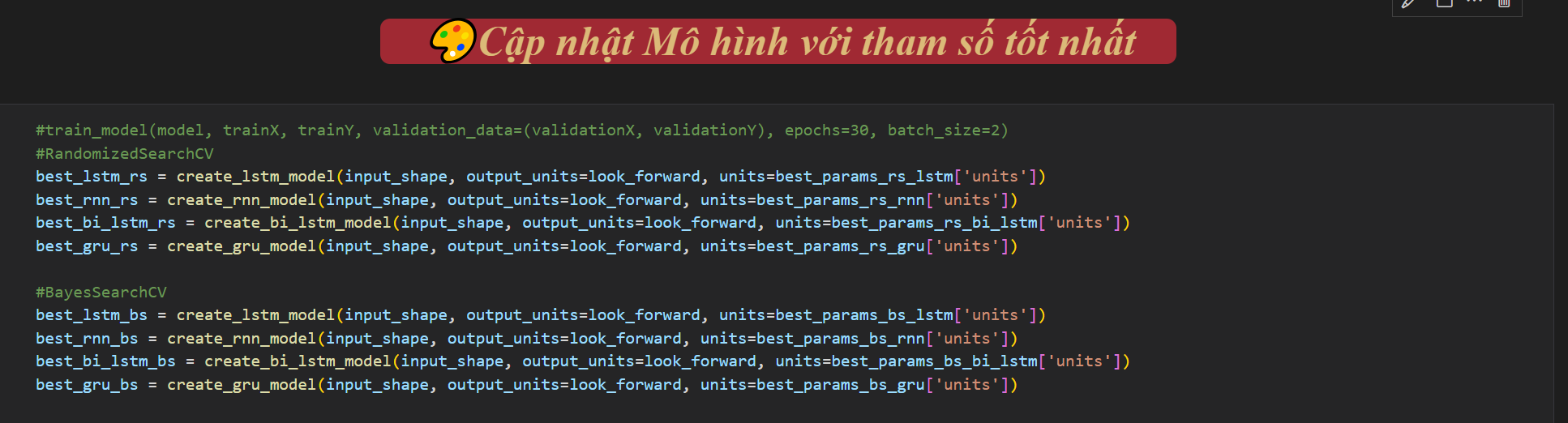
### BayesSearchCV





## Cập nhật các Mô hình với bộ tham số tốt nhất (units)

Cập nhật tham số units (số đơn vị ẩn – nút) cho các mô hình



## Huấn luyện các mô hình với bộ tham số tốt nhất (epochs,batch\_size)

***Batch Size (Kích thước Batch):*** Số lượng mẫu dữ liệu được sử dụng trong một lần cập nhật trọng số.

***Epoch (Kỳ):*** Một vòng lặp qua toàn bộ tập dữ liệu huấn luyện

Sử dụng hàm train\_model sử dụng dữ liệu train , số lượng epochs, batch\_size được lấy từ bộ tham số tốt nhất (best param) và dữ liệu validation, hàm này tự động huấn luyện mô hình và theo dõi hiệu suất trên tập validation trong suốt quá trình đào tạo. Điều này giúp kiểm soát quá mức hóa và tăng cường khả năng tổng quát hóa của mô hình, cung cấp một cách thuận tiện để huấn luyện và đánh giá mô hình.

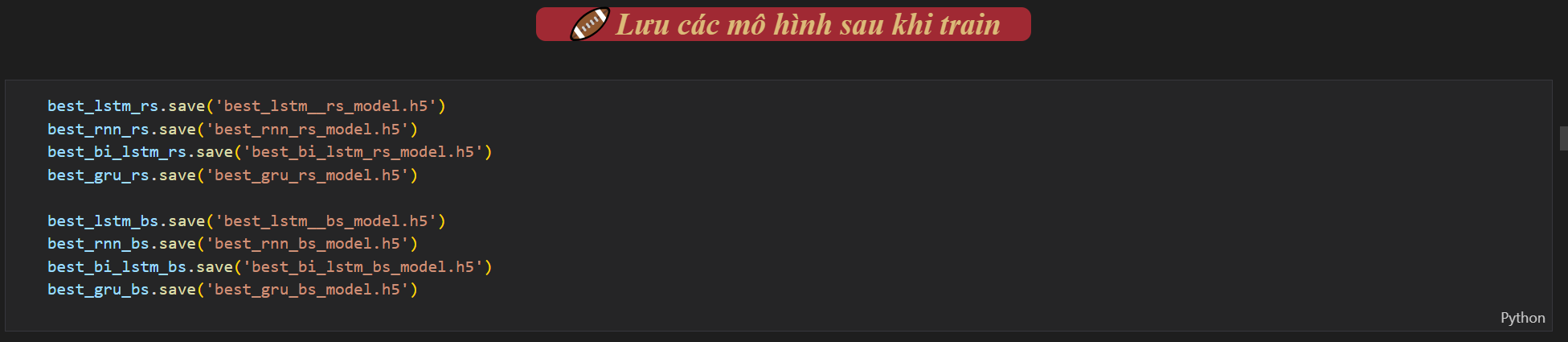
### RandomizedSearchCV



### BayesSearchCV



### Lưu các mô hình sau khi train để thuận tiện tái sử dụng



**Phần 8: Dự đoán và đánh giá**

## Hàm dự đoán và đánh giá



Sau khi đã huấn luyện mô hình trên tập train và validation , ta sẽ tiến hành dự đoán giá trị đầu ra trên dữ liệu mới ( tập test )

## RandomizedSearchCV

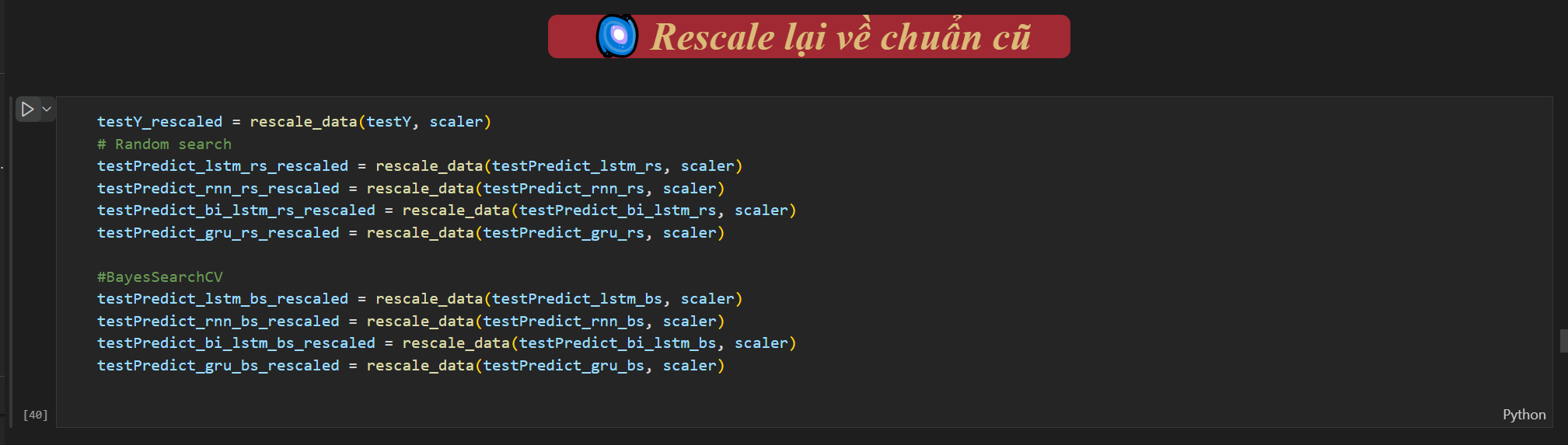


## BayesSearchCV



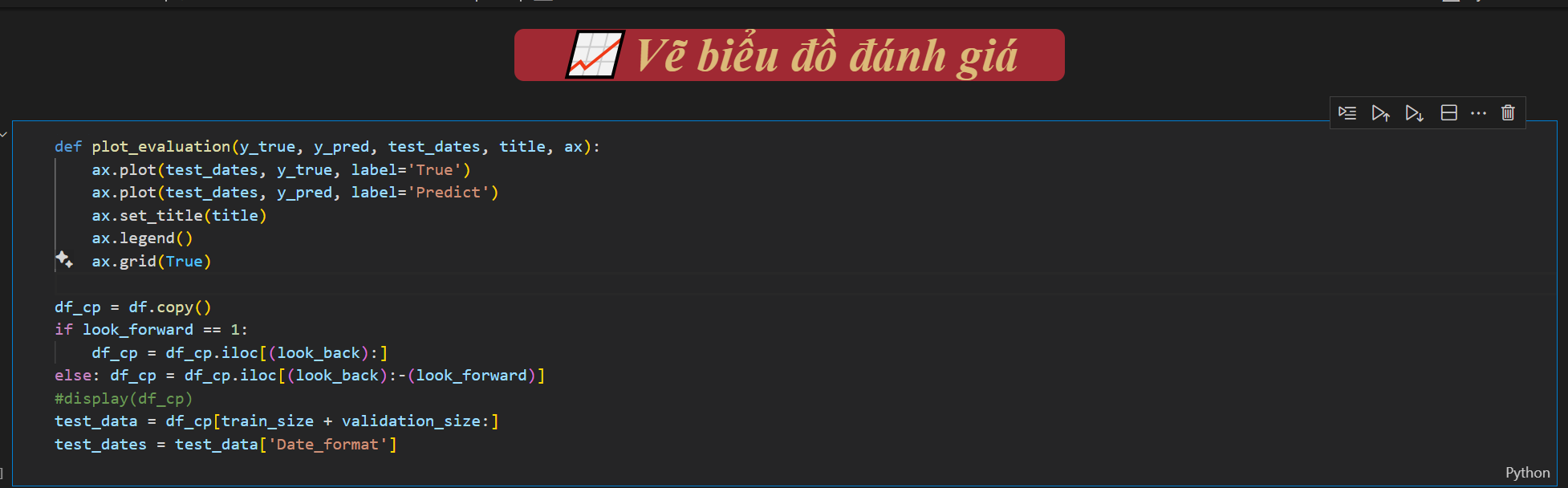
Sau khi dự đoán , ta đưa giá trị dự đoán về chuẩn cũ để thực hiện trực quan dữ liệu

## Đưa dữ liệu về chuẩn cũ

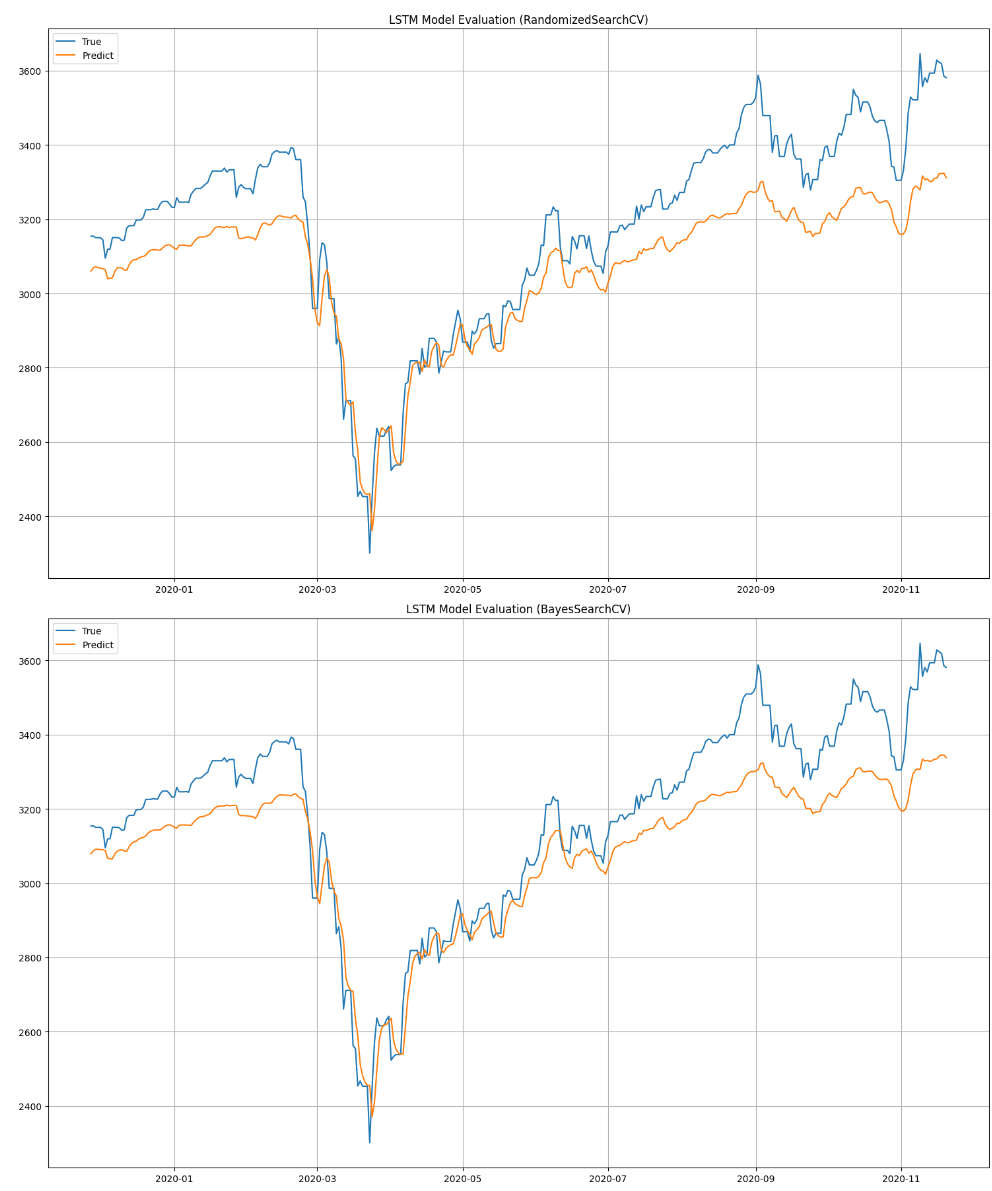


## Vẽ biểu đồ

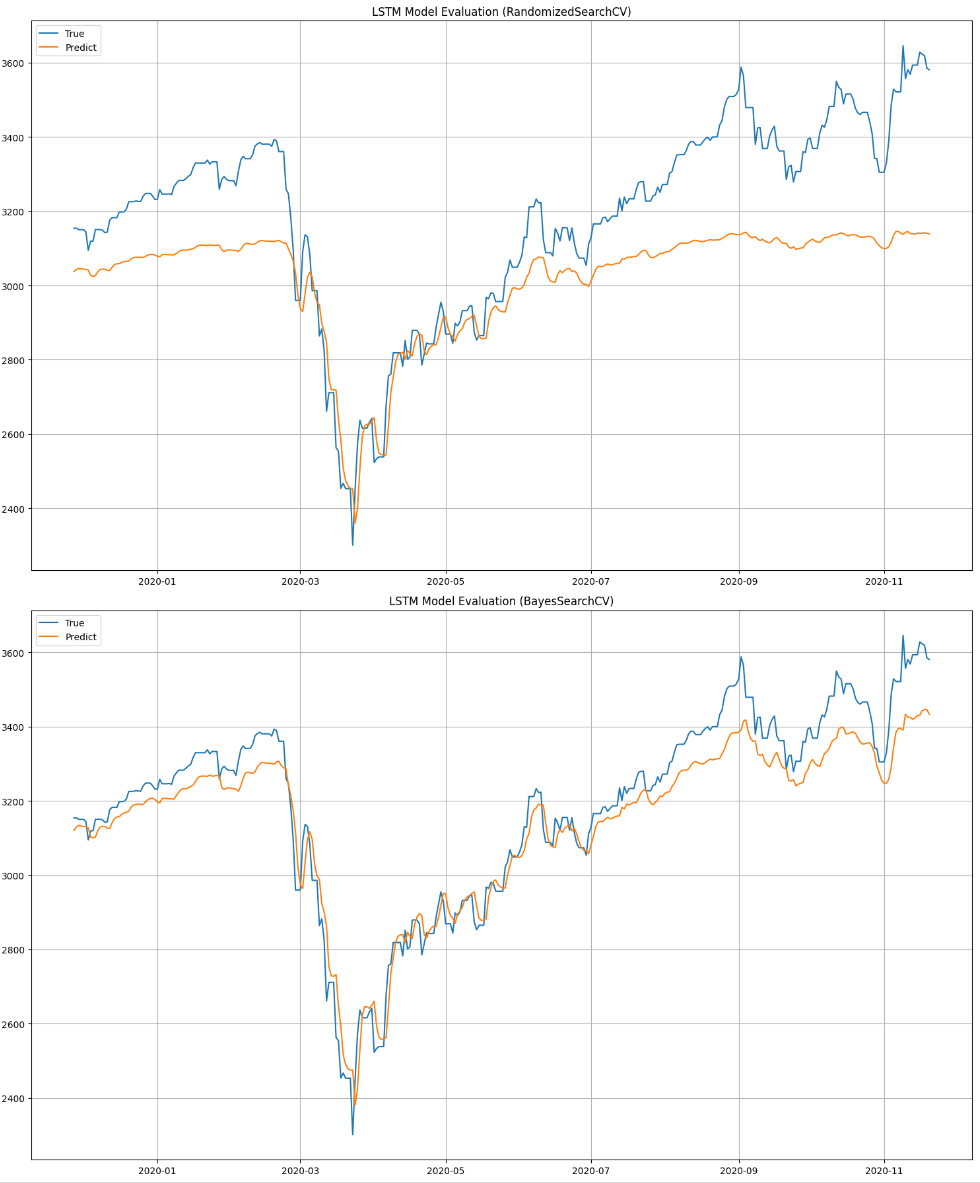
Hàm vẽ biểu đồ và tạo test\_dates làm dữ liệu cho trục x



### LSTM



Lần 2



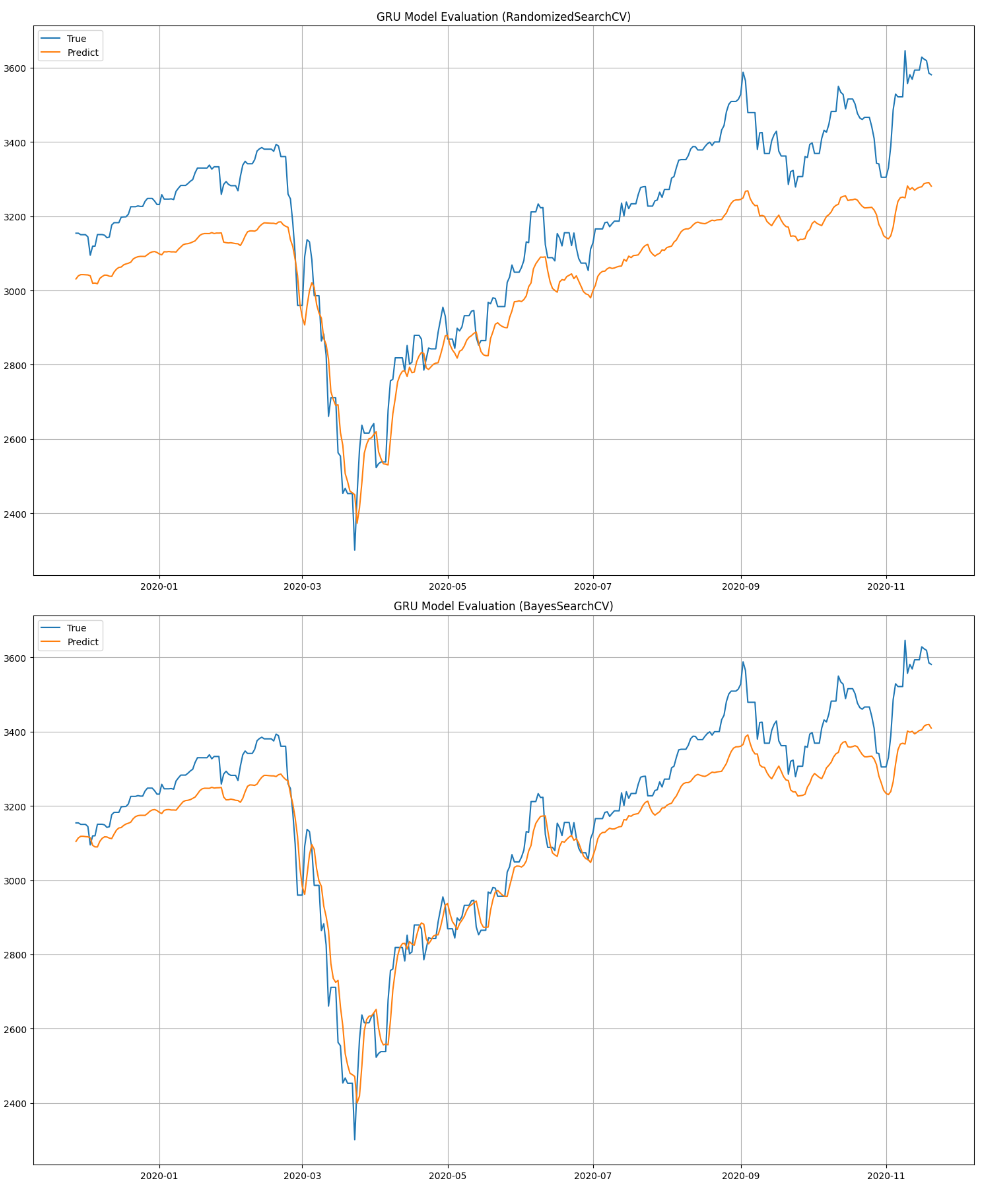
### RNN



### BiLSTM



### GRU



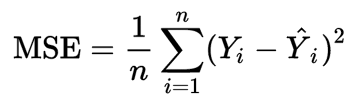
## Bảng đánh giá

Tiến hành đánh giá hiệu suất của mô hình thông qua các độ đo như:

* MAE
* MAE là một độ đo lỗi thường được sử dụng để đo lường sự chênh lệch trung bình giữa giá trị dự đoán và giá trị thực tế.
* Càng thấp, mô hình càng chính xác. MAE tính toán trung bình giá trị tuyệt đối của các chênh lệch.



* MSE
* MSE là một độ đo lỗi khác, là trung bình của bình phương của các chênh lệch giữa giá trị dự đoán và giá trị thực tế.
* MSE càng thấp, mô hình càng chính xác. Tuy nhiên, MSE có thể nhạy cảm với giá trị ngoại lệ.

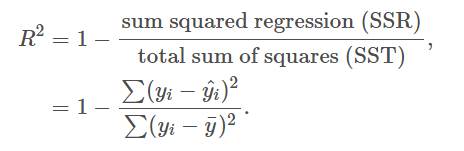


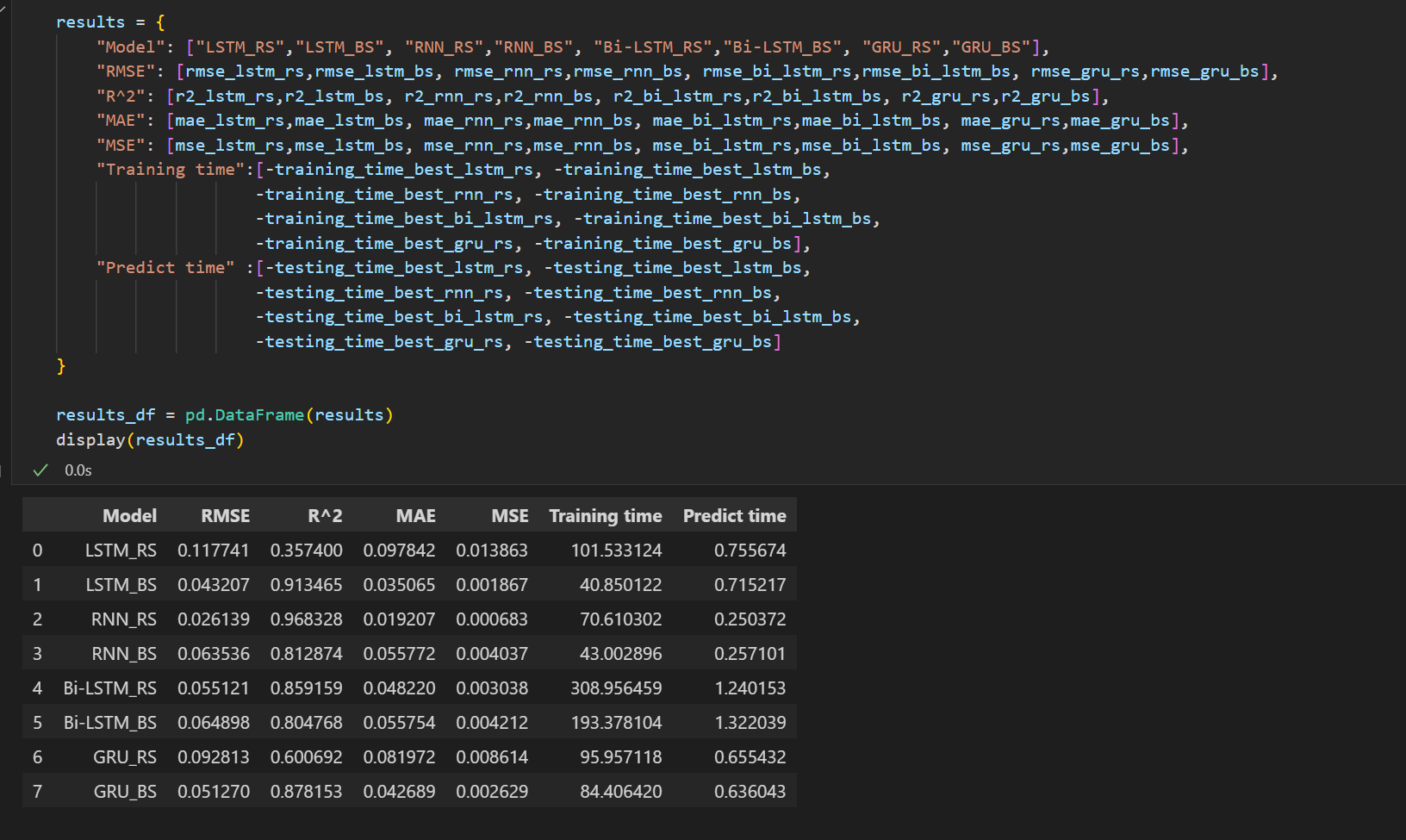
* **Root mean squared error (RMSE)**:

RMSE = MSE

* RMSE là căn bậc hai của MSE và thường được sử dụng để đo lường độ lớn của lỗi mô hình.
* RMSE càng thấp, mô hình càng chính xác. Nó giữ lại đơn vị của giá trị dự đoán và giá trị thực tế.
* **R-squared (R2 Score)** hay R^2
* Là một độ đo đánh giá khả năng giải thích của mô hình, biểu thị phần trăm

biến thiên trong dữ liệu mà mô hình có thể giải thích.  
Giá trị R2 dao động từ 0 đến 1, với 1 là hoàn hảo. Giá trị càng gần 1, mô hình càng tốt.





Theo lý thuyết thì Bayesian Optimization có hiệu suất tốt hơn trong việc tối ưu hóa tham số mô hình so với các phương pháp như Grid Search và Randomized Search. Tuy nhiên, trong thực tế, hiệu suất của các phương pháp tối ưu hóa có thể phụ thuộc vào đặc tính cụ thể của mô hình và bài toán.

Nhìn chung trên kết quả ta thấy với LSTM , GRU thì Bayesian Optimization Optimization vẫn là phương pháp tối ưu hóa mạnh mẽ nhất, dẫn đến kết quả RMSE thấp hơn **nhiều** so với RandomizeSearch , tuy nhiên ta thấy ở BiLSTM và RNN thì dường như ngược lại , RandomizedSearch lại cho kết quả tốt hơn . Điều này có thể được giải thích bằng tính linh hoạt và khả năng tìm kiếm hiệu quả của Randomized Search trong không gian tham số lớn và phức tạp của BiLSTM.

### Về thời gian chạy

Nhìn chung, Bayesian Optimization có thời gian huấn luyện ngắn hơn so với Randomized Search. Tuy nhiên, trong trường hợp của Bi-LSTM, Randomized Search lại mất ít thời gian hơn để huấn luyện mô hình.

Sự khác biệt lớn nhất về thời gian huấn luyện giữa hai phương pháp được thấy ở Bi-LSTM và LSTM, cho thấy Bayesian Optimization thường nhanh hơn trong việc tìm kiếm tham số tối ưu cho các mô hình này.

* Bayesian Optimization thường hiệu quả hơn với LSTM và GRU, cả về hiệu suất và thời gian huấn luyện.
* Randomized Search lại hiệu quả hơn với RNN và Bi-LSTM, đặc biệt trong không gian tham số phức tạp.

Sự khác biệt này có thể xuất phát từ khả năng khám phá không gian tham số của từng phương pháp. Bayesian Optimization thường mạnh trong không gian tham số nhỏ và có cấu trúc rõ ràng, trong khi Randomized Search lại có lợi thế trong không gian tham số lớn và phức tạp. Tuy nhiên, thời gian chạy của Bayesian Optimization thường ngắn hơn, làm cho nó trở thành lựa chọn tốt hơn trong nhiều trường hợp.