Họ và tên : Trịnh Vinh Tuấn Đạt

Mã sinh viên : B21DCCN031

1. Trình bày sự hiểu biết về CNN, RNN, LSTM
   1. CNN

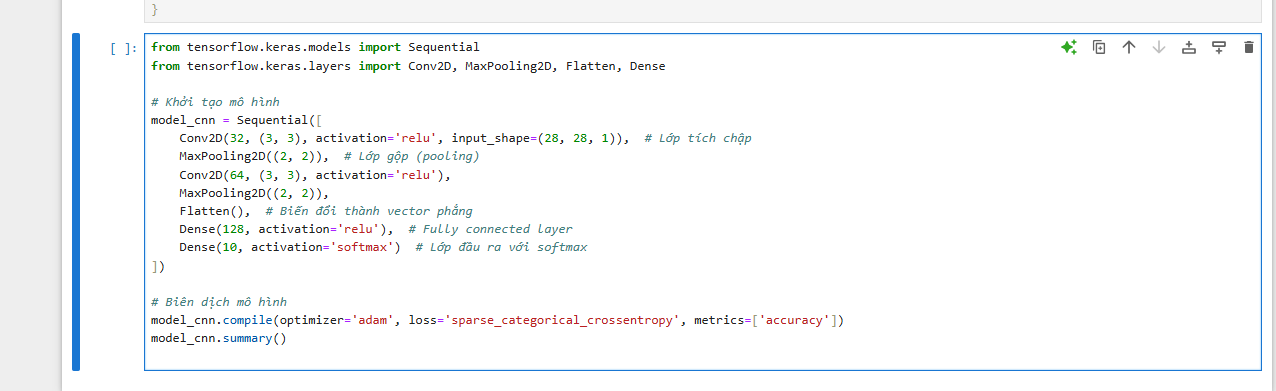
Bắt đầu từ những năm 1980, CNN được phát triển bởi Yann LeCun để nhận diện ký tự viết tay trong hệ thống đọc zip-code. Kiến trúc của CNN giúp xử lý tốt các dữ liệu hình ảnh, và đã được cải tiến thành nhiều loại CNN phức tạp như AlexNet (2012), VGGNet (2014) và ResNet (2015), đạt được thành công lớn trong các bài toán nhận diện hình ảnh.



Mạng CNN là một tập hợp các lớp Convolution chồng lên nhau và sử dụng các hàm nonlinear activation như ReLU và tanh để kích hoạt các trọng số trong các node. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo.

Các layer trong mô hình CNN liên kết được với nhau thông qua cơ chế convolution. Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Như vậy mỗi neuron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng ảnh cục bộ của neuron trước đó. Mỗi một lớp được sử dụng các filter khác nhau thông thường có hàng trăm hàng nghìn filter như vậy và kết hợp kết quả của chúng lại. Ngoài ra có một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu). Trong quá trình huấn luyện mạng (traning) CNN tự động học các giá trị qua các lớp filter dựa vào cách thức mà bạn thực hiện. Ví dụ trong tác vụ phân lớp ảnh, CNNs sẽ cố gắng tìm ra thông số tối ưu cho các filter tương ứng theo thứ tự raw pixel > edges > shapes > facial > high-level features. Layer cuối cùng được dùng để phân lớp ảnh.

Code mẫu :



Các lớp trong mô hình CNN:   
+ Conv2D

Mô tả: Đây là lớp tích chập (Convolutional Layer) 2D, áp dụng các filter (bộ lọc) lên các phần của ảnh đầu vào để trích xuất đặc trưng (feature). Conv2D là thành phần chính trong kiến trúc CNN, giúp mô hình học được các mẫu cục bộ trong hình ảnh.

Tham số chính: filters: Số lượng bộ lọc (số lượng đặc trưng cần trích xuất), trong đoạn code là 32 và 64. kernel\_size: Kích thước của filter (ở đây là 3x3).

Activation='relu': Sử dụng hàm kích hoạt ReLU để loại bỏ các giá trị âm, chỉ giữ lại giá trị dương.

+ MaxPooling2D

Mô tả: Lớp gộp (Pooling Layer) 2D này được dùng để giảm kích thước của đặc trưng trích xuất, giúp giảm thiểu số lượng tham số và ngăn chặn hiện tượng quá khớp (overfitting).

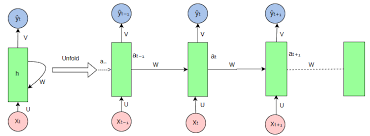
Tham số chính: pool\_size=(2, 2): Kích thước của filter dùng để lấy giá trị lớn nhất trong vùng này. Kích thước (2, 2) giúp giảm dữ liệu thành một nửa chiều dài và chiều rộng.

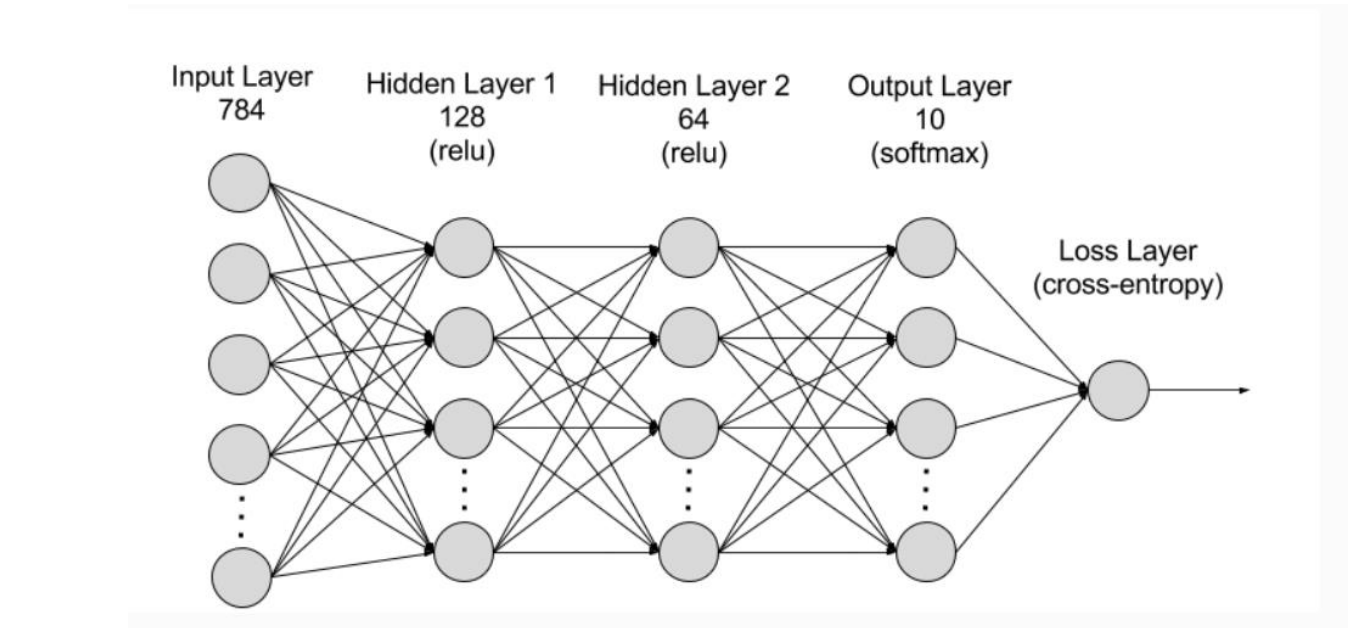
+ Flatten: Mô tả: Lớp này làm phẳng dữ liệu từ dạng ma trận 2D (hoặc 3D) thành vector 1D. Điều này giúp chuyển đổi dữ liệu từ các lớp tích chập thành dạng đầu vào phù hợp cho các lớp fully connected (dense layers).

+ Dense: Mô tả: Đây là lớp fully connected, kết nối tất cả các neuron từ lớp trước với mỗi neuron trong lớp này. Tham số chính: units: Số lượng neuron trong lớp. Ví dụ, lớp đầu tiên có 128 neuron và lớp cuối cùng có 10 neuron cho 10 nhãn phân loại. activation: Sử dụng relu trong lớp hidden và softmax trong lớp cuối để tính xác suất cho mỗi nhãn trong bài toán phân loại.

* 1. RNN

RNN được phát triển từ những năm 1980 và được ứng dụng nhiều vào các năm 1990.Các ý tưởng ban đầu về RNN đã được đề xuất bởi các nhà khoa học như David Rumelhart và Ronald J. Williams.RNN được phát triển để xử lý dữ liệu tuần tự (sequence data), đặc biệt hữu ích cho các ứng dụng ngôn ngữ tự nhiên (NLP). Tuy nhiên, RNN gặp khó khăn với các chuỗi dài do hiện tượng “vanishing gradient” (độ dốc mất dần) khiến mô hình khó học tốt các phụ thuộc xa. Ứng dụng phổ biến nhất hiện nay của RNN là xử lý ngôn ngữ tự nhiên, ví dụ như dịch máy, phân loại văn bản, và phân tích cảm xúc.



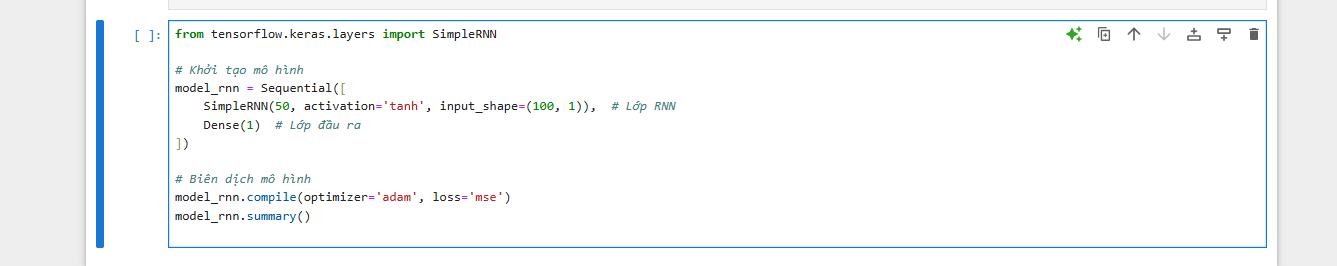


RNN được tạo thành từ các nơ-ron: các nút xử lý dữ liệu kết hợp cùng nhau để thực hiện các tác vụ phức tạp. Các nơ-ron được tổ chức dưới dạng lớp đầu vào, đầu ra và ẩn. Lớp đầu vào nhận thông tin để xử lý và lớp đầu ra cung cấp kết quả. Quá trình xử lý dữ liệu, phân tích và dự đoán diễn ra trong lớp ẩn.

Lớp ẩn RNN hoạt động bằng cách lần lượt truyền dữ liệu tuần tự nhận được đến các lớp ẩn. Tuy nhiên, RNN cũng có quy trình làm việc tự lặp lại hay hồi quy: lớp ẩn có thể ghi nhớ và sử dụng các đầu vào trước đó cho các dự đoán trong tương lai trong một thành phần bộ nhớ ngắn hạn. Quy trình này sử dụng đầu vào hiện tại và bộ nhớ đã lưu trữ để dự đoán chuỗi tiếp theo.

Ví dụ: hãy xem xét chuỗi: Apple is red (Táo màu đỏ). Bạn muốn RNN dự đoán red (màu đỏ) khi nhận được chuỗi đầu vào Apple is (Táo màu). Khi xử lý từ Apple (Táo), lớp ẩn sẽ lưu trữ một bản sao trong bộ nhớ. Tiếp theo, khi thấy từ is (màu), lớp ẩn gọi lại Apple (Táo) từ bộ nhớ của mình và hiểu toàn bộ chuỗi: Apple is (Táo màu) là ngữ cảnh. Sau đó, lớp ẩn có thể dự đoán red (màu đỏ) để cải thiện độ chính xác. Do đó, RNN trở nên hữu ích trong nhận dạng giọng nói, dịch máy và các tác vụ lập mô hình ngôn ngữ khác

Code mẫu



Mô hình RNN này bao gồm ba lớp, mỗi lớp đảm nhiệm một vai trò cụ thể trong quá trình xử lý và dự đoán dữ liệu:

layers.SimpleRNN(50, activation='relu', input\_shape=(X\_train.shape[1], 1))

Đây là lớp mạng nơ-ron hồi quy đơn giản (Simple Recurrent Neural Network - RNN) với 50 đơn vị ẩn (neurons).

input\_shape=(X\_train.shape[1], 1): Cấu trúc đầu vào là (số bước thời gian, số đặc trưng đầu vào). Ở đây, đầu vào được thiết lập với một đặc trưng trên mỗi bước thời gian.

activation='relu': Chức năng kích hoạt ReLU (Rectified Linear Unit) được áp dụng tại mỗi bước thời gian. ReLU thường giúp mô hình học nhanh hơn và hạn chế vấn đề vanishing gradient. Lớp này sẽ xử lý đầu vào tuần tự, giúp mô hình nắm bắt thông tin theo trình tự thời gian và học cách giữ thông tin từ các bước trước đó.

Lớp Dense 64 đơn vị ẩn: layers.Dense(64, activation='relu') Đây là lớp kết nối dày đặc (fully connected) với 64 neurons.

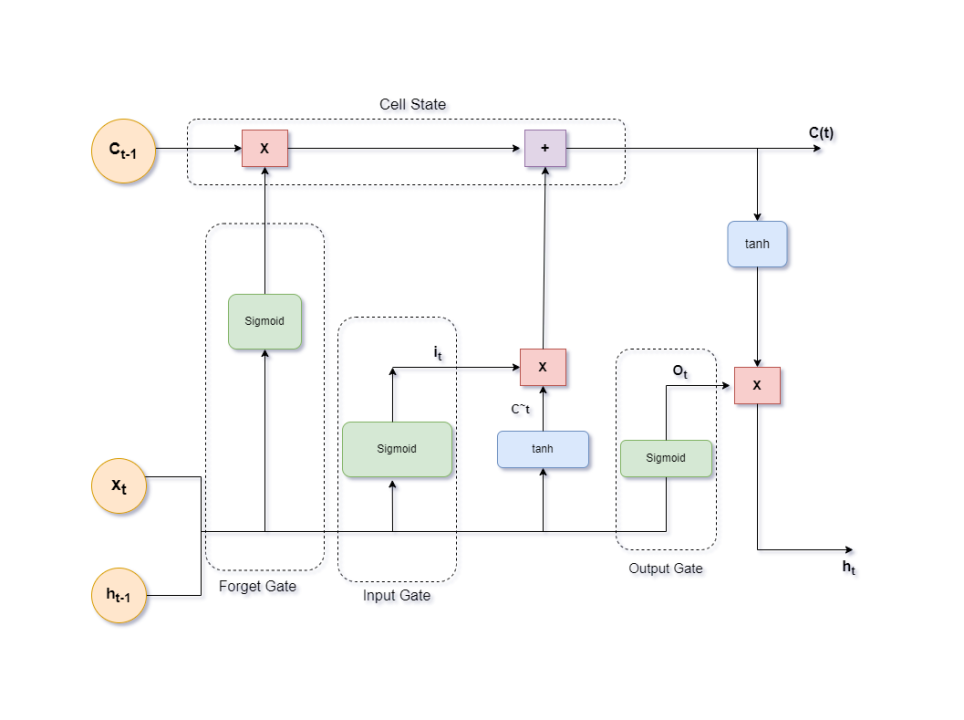
activation='relu': Kích hoạt ReLU được dùng để tạo phi tuyến cho mô hình, giúp học các mẫu phức tạp hơn trong dữ liệu. Lớp này hoạt động như một tầng ẩn thêm để tăng độ sâu của mô hình, học các đặc trưng từ đầu ra của lớp RNN phía trước.

Lớp Dense đầu ra: layers.Dense(1) Lớp cuối cùng này chứa một neuron, dùng để dự đoán một giá trị đầu ra duy nhất (giá trị hồi quy). Không sử dụng hàm kích hoạt ở lớp này vì giá trị đầu ra là một số thực liên tục, phù hợp cho các bài toán dự đoán giá trị số như hồi quy.

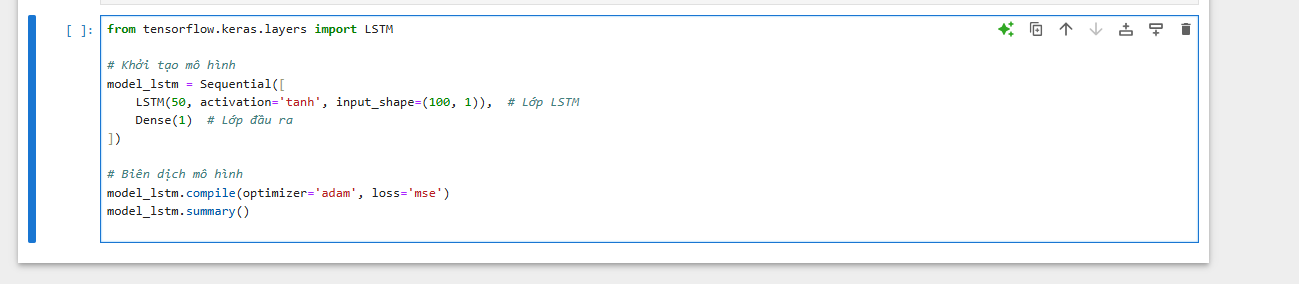
Mô hình này thực hiện tốt cho bài toán dự đoán chuỗi thời gian hoặc các bài toán hồi quy cần xử lý dữ liệu tuần tự và lấy thông tin từ những bước trước đó trong chuỗi.

* 1. LSTM

LSTM được giới thiệu vào năm 1997. Được phát triển bởi Sepp Hochreiter và Jürgen Schmidhuber, hai nhà nghiên cứu Đức nổi tiếng trong lĩnh vực học sâu. LSTM được thiết kế để khắc phục vấn đề vanishing gradient của RNN. Mô hình này sử dụng các “cổng” (input gate, forget gate, output gate) để kiểm soát cách thông tin được lưu trữ, ghi đè, và xuất ra. Nhờ vậy, LSTM có thể duy trì trạng thái ổn định và ghi nhớ thông tin quan trọng trong thời gian dài. LSTM rất phổ biến trong các bài toán yêu cầu ghi nhớ thông tin dài hạn như dịch máy, nhận diện giọng nói, và các hệ thống chatbot. Đặc biệt, LSTM đã cải thiện đáng kể độ chính xác của các mô hình xử lý ngôn ngữ tự nhiên.



Code mẫu



**Lớp LSTM layers.**

**LSTM(50, activation='relu', input\_shape=(X\_train.shape[1], 1)):**

LSTM (Long Short-Term Memory): Đây là lớp chính trong mô hình, chuyên xử lý dữ liệu tuần tự. LSTM giúp giải quyết vấn đề vanishing gradient mà các mô hình RNN truyền thống gặp phải, cho phép mô hình học từ các mối quan hệ dài hạn trong dữ liệu.

50: Số lượng đơn vị ẩn (neurons) trong lớp LSTM. Điều này quyết định kích thước của trạng thái ẩn và số lượng thông tin mà mô hình có thể lưu trữ và xử lý.

activation='relu': Hàm kích hoạt ReLU (Rectified Linear Unit) được sử dụng tại mỗi đơn vị. Hàm ReLU giúp tăng tốc độ huấn luyện và cung cấp tính phi tuyến cho mô hình.

input\_shape=(X\_train.shape[1], 1):

Định nghĩa kích thước đầu vào của mô hình. X\_train.shape[1] đại diện cho số bước thời gian (timesteps) trong mỗi chuỗi dữ liệu. 1 là số đặc trưng đầu vào tại mỗi bước thời gian (có thể có nhiều hơn nếu dữ liệu có nhiều đặc trưng).

**Lớp Dense (kết nối dày đặc)**

layers.Dense(64, activation='relu'): Đây là một lớp kết nối dày đặc với 64 neurons. Lớp này được sử dụng để tăng cường khả năng học của mô hình, cho phép nó học các đặc trưng phức tạp hơn từ đầu ra của lớp LSTM.

activation='relu': Hàm kích hoạt ReLU cũng được sử dụng ở lớp này để cung cấp tính phi tuyến và giúp mô hình học nhanh hơn.

**Lớp Dense đầu ra**

layers.Dense(1): Đây là lớp đầu ra của mô hình, chứa một neuron. Lớp này dùng để dự đoán giá trị đầu ra duy nhất (trong trường hợp này là giá trị hồi quy). Không sử dụng hàm kích hoạt tại lớp này vì đầu ra là một số thực liên tục, phù hợp với các bài toán hồi quy.

**Kết luận**

Mô hình LSTM này bao gồm một lớp LSTM để xử lý dữ liệu tuần tự và học từ các mối quan hệ dài hạn, sau đó là một lớp Dense với 64 neurons để tăng cường khả năng học, và cuối cùng là một lớp Dense đầu ra với một neuron để dự đoán giá trị hồi quy. Cấu trúc này giúp mô hình có khả năng nắm bắt các thông tin quan trọng trong dữ liệu chuỗi và tạo ra dự đoán chính xác hơn

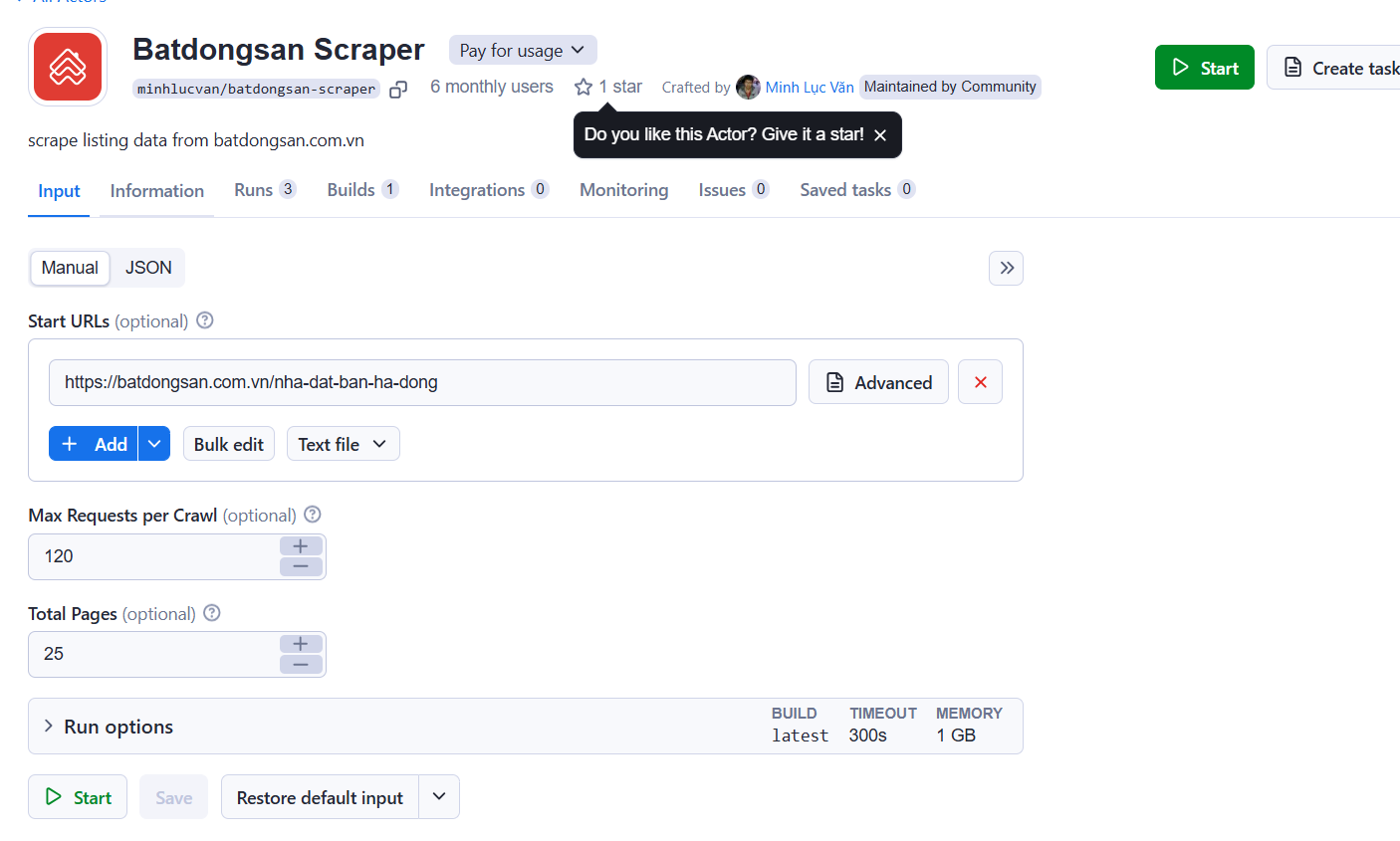
1. Xây dựng bộ data về nhà đất Hà Nội – Hà Đông. Sử dụng kĩ thuật CNN, RNN, LSTM để dự đoán nó

B1: Xây dựng bộ dữ liệu bằng cách sử dụng tool crawl data giá nhà đất Hà Đông

Sử dụng tool Apify để crawl dữ liệu

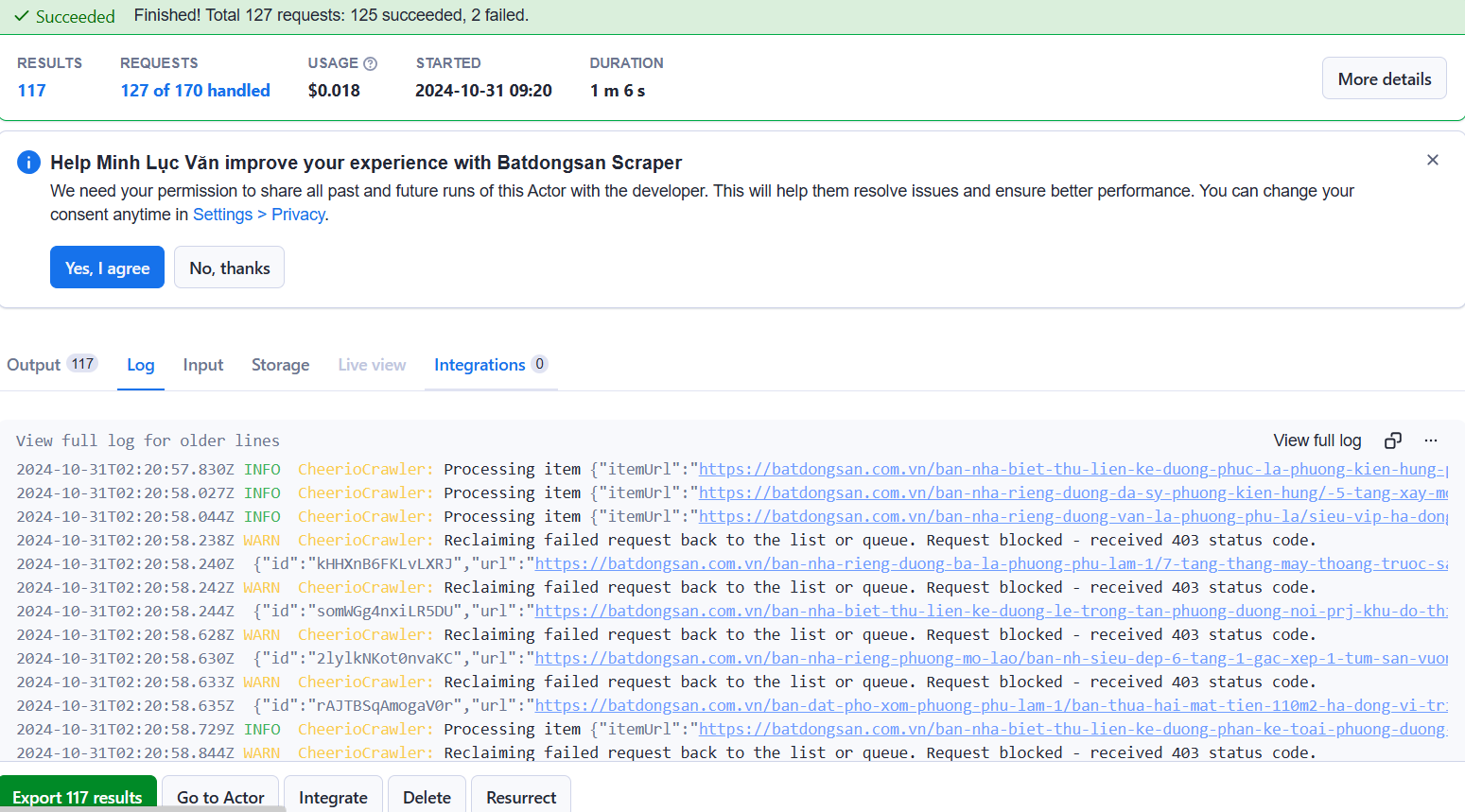
<https://apify.com/minhlucvan/batdongsan-scraper/api>

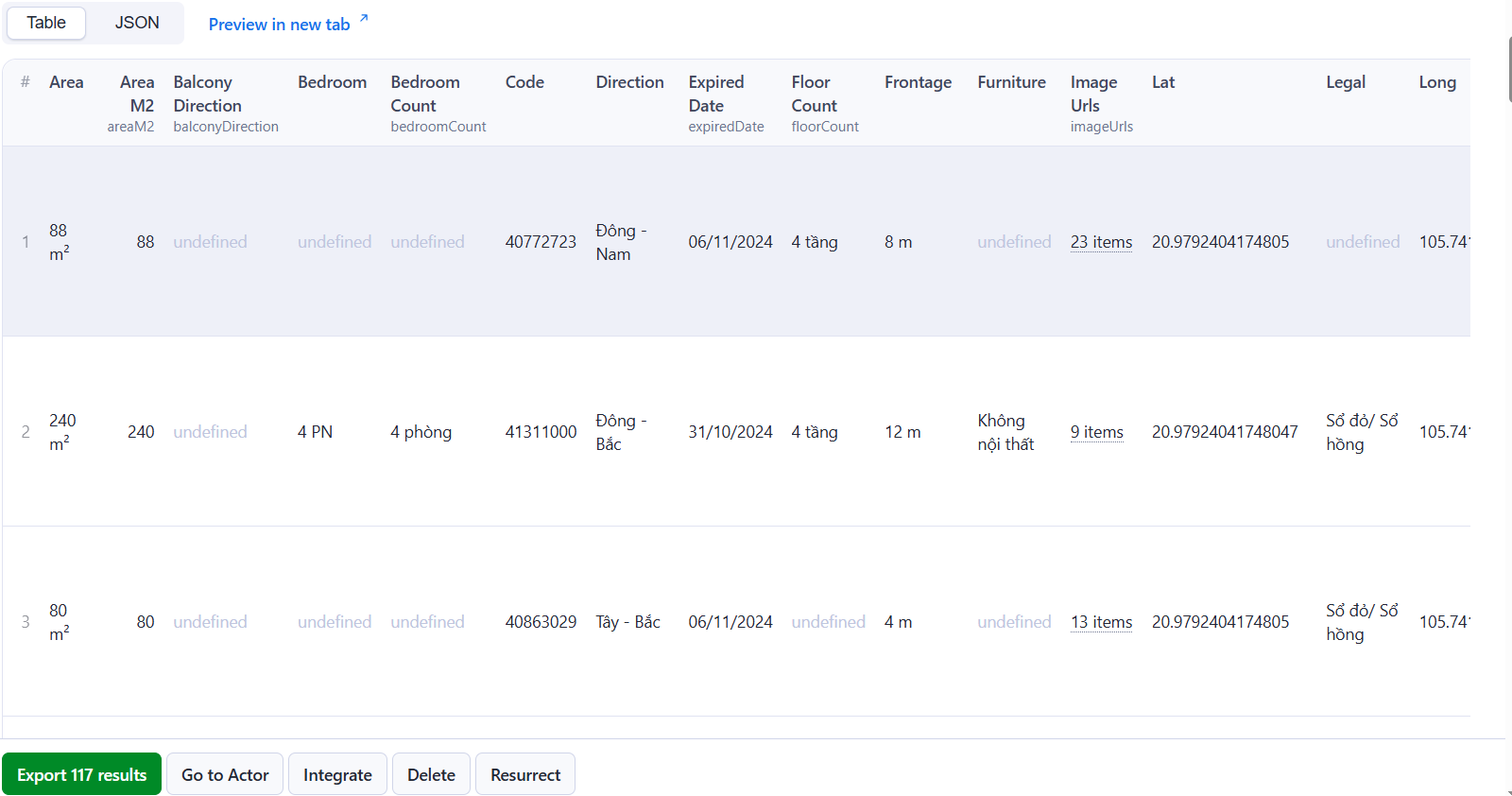
Cần tinh chỉnh một số tham số để thực hiện việc crawl dữ liệu



Với start Urls sẽ chứa đường dẫn trực tiếp đến web để có thể thực hiện crawl dữ liệu

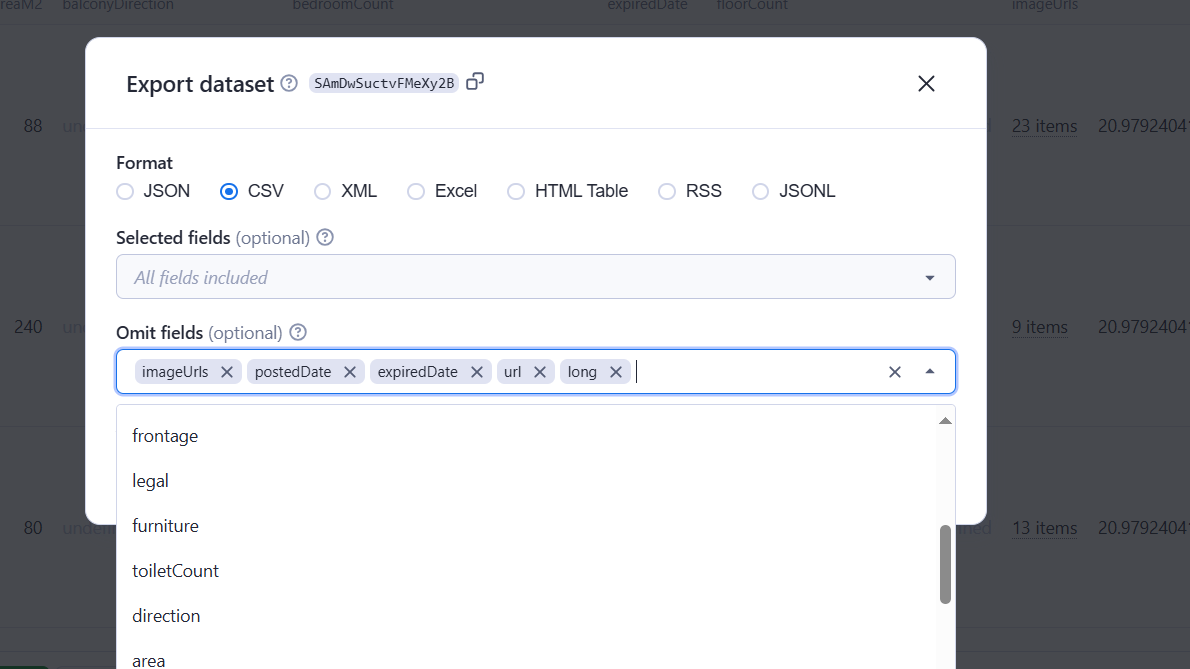
+ Để Maxrequest và Toltal page sau đó thực hiện quá trình crawl dữ liệu





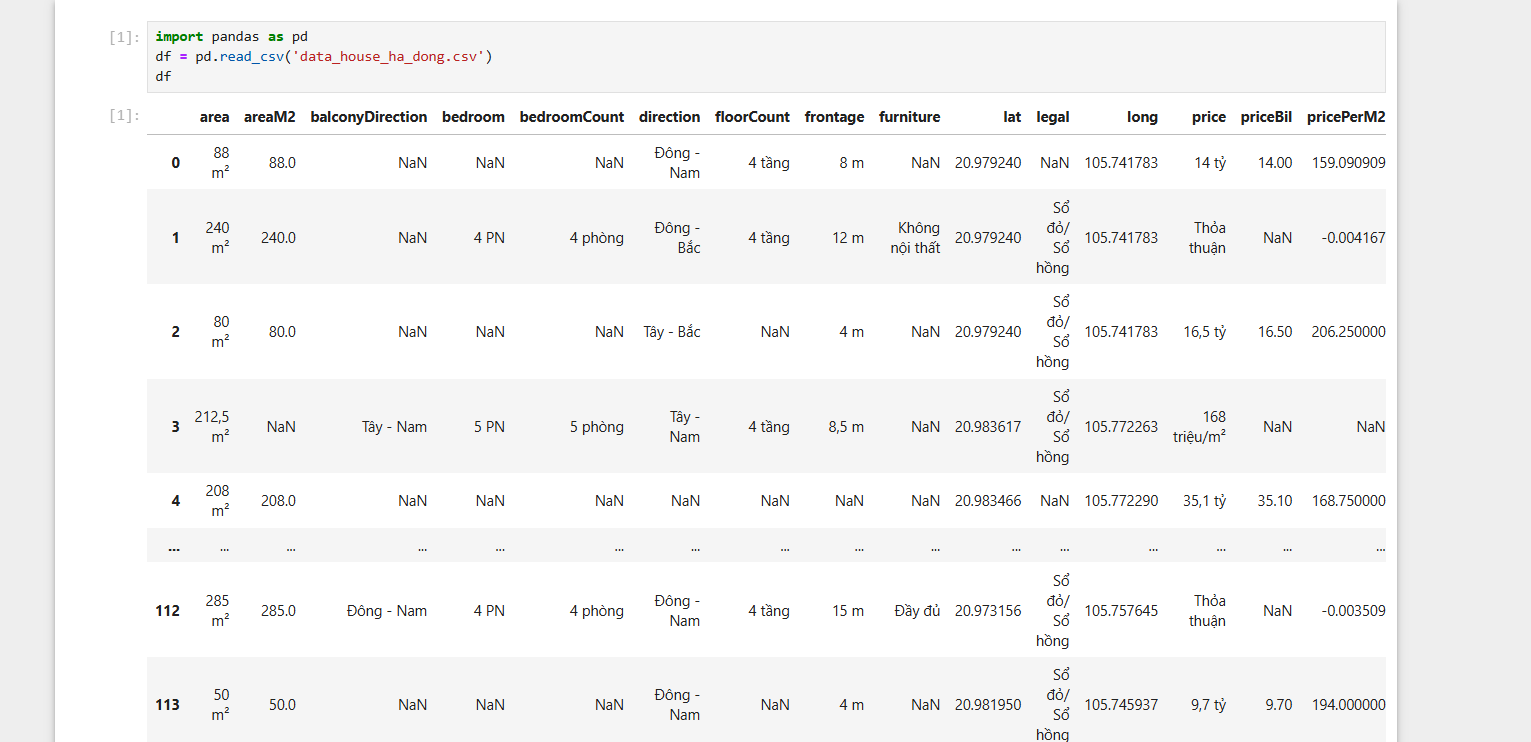
Ta sẽ nhận được 117 bản ghi về dữ liệu bao gồm các trường thuộc tính như arera, AreaeM2, bedroom, ….

Thực hiện export để tải dữ liệu về dưới dạng file csv

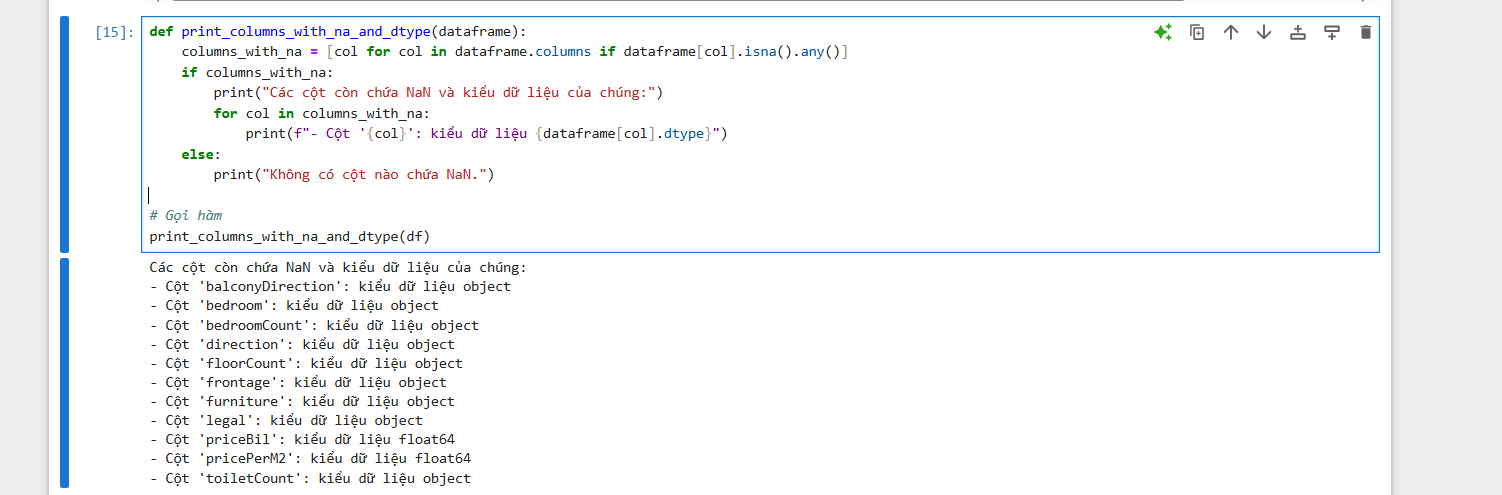


Tiến hành làm sạch dữ liệu thô:

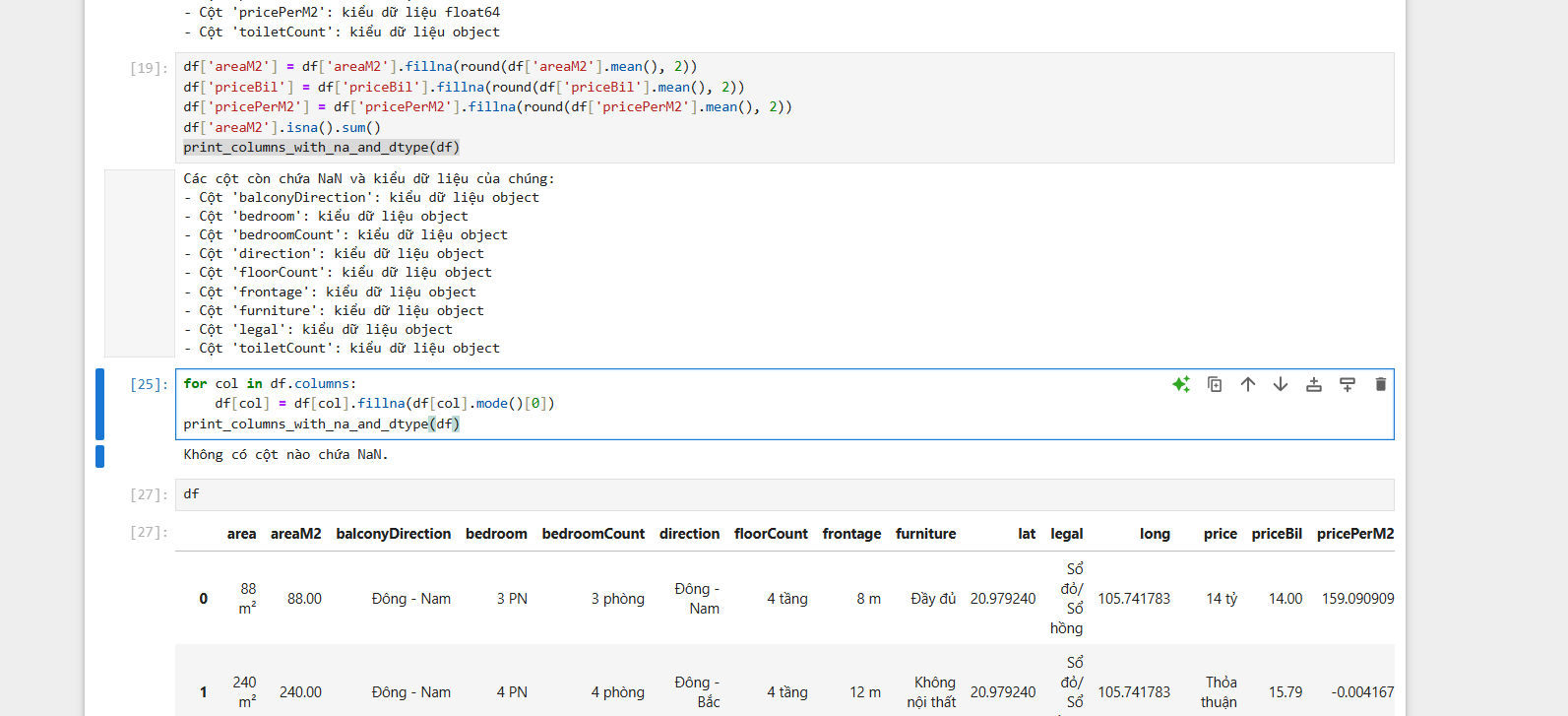
+ Hiển thị dữ liệu thô nhận được



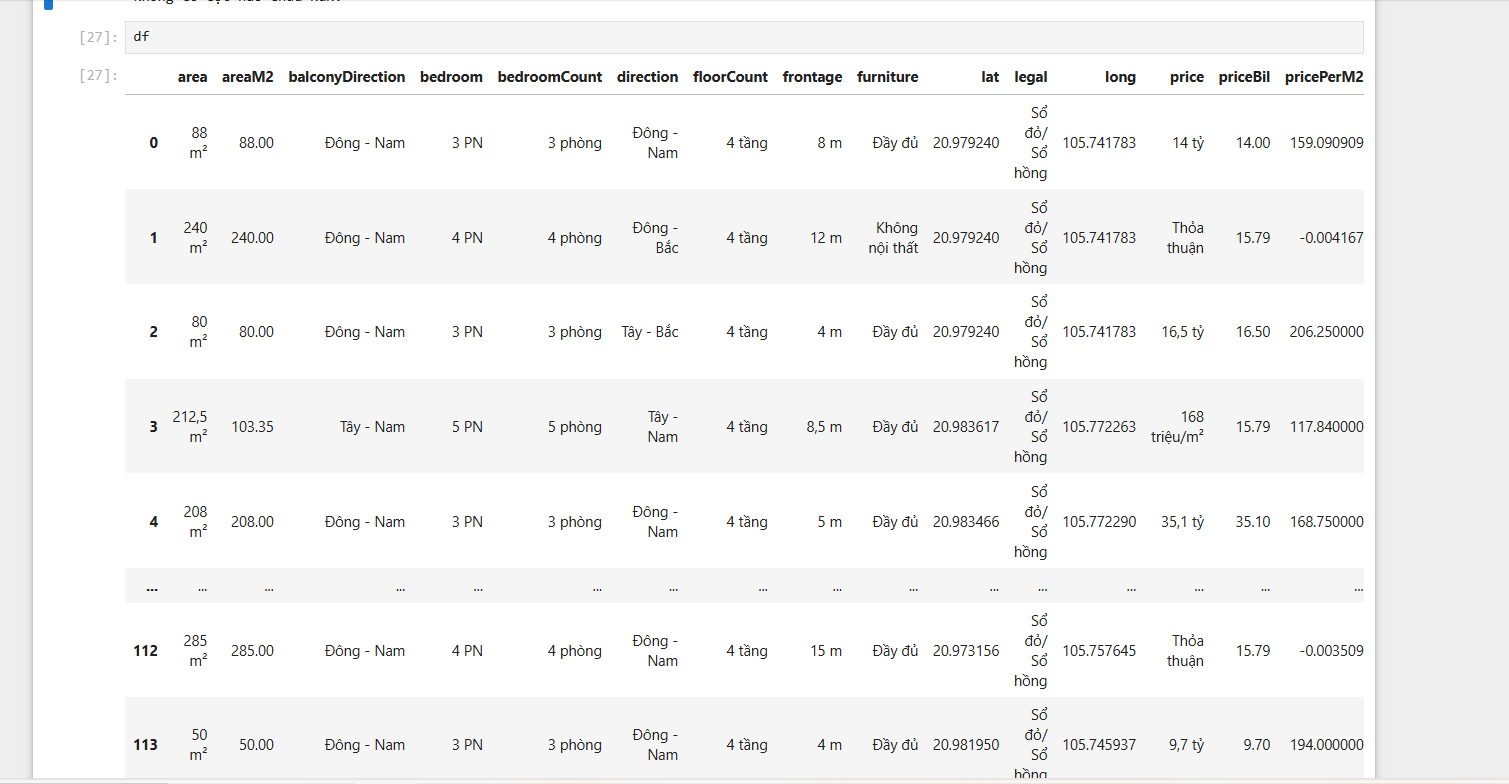
+ Tiến hành xác định các cột dữ liệu bị thiếu



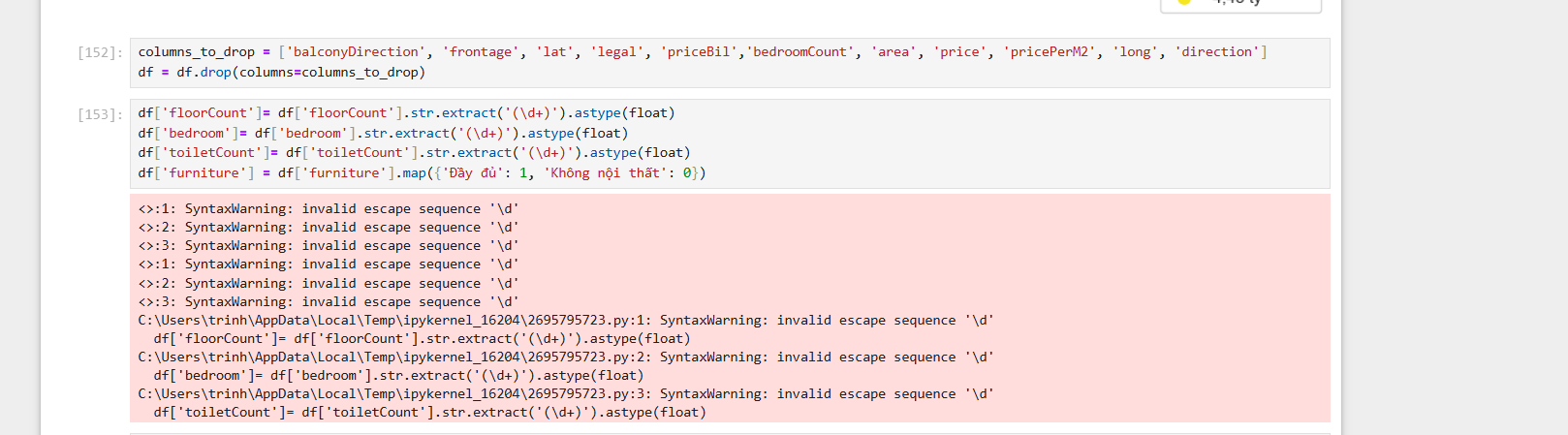
+Thực hiện chuẩn hóa và điền dữ liệu còn thiếu



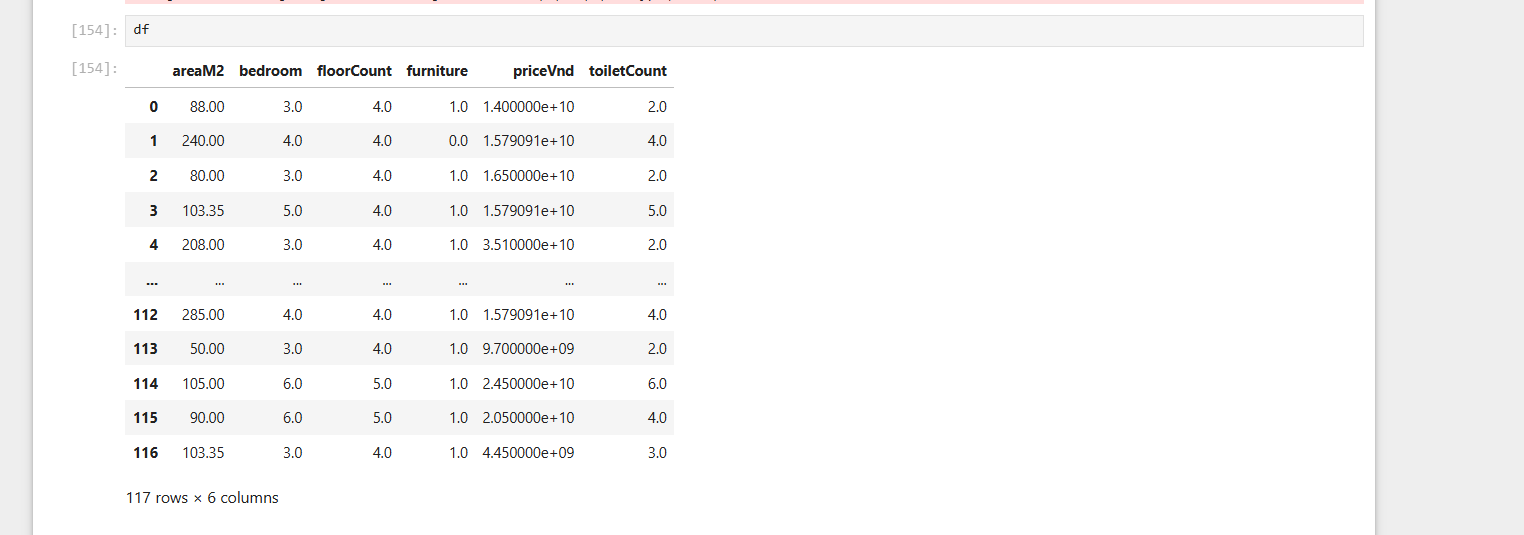
+ Dữ liệu sau khi được làm sạch



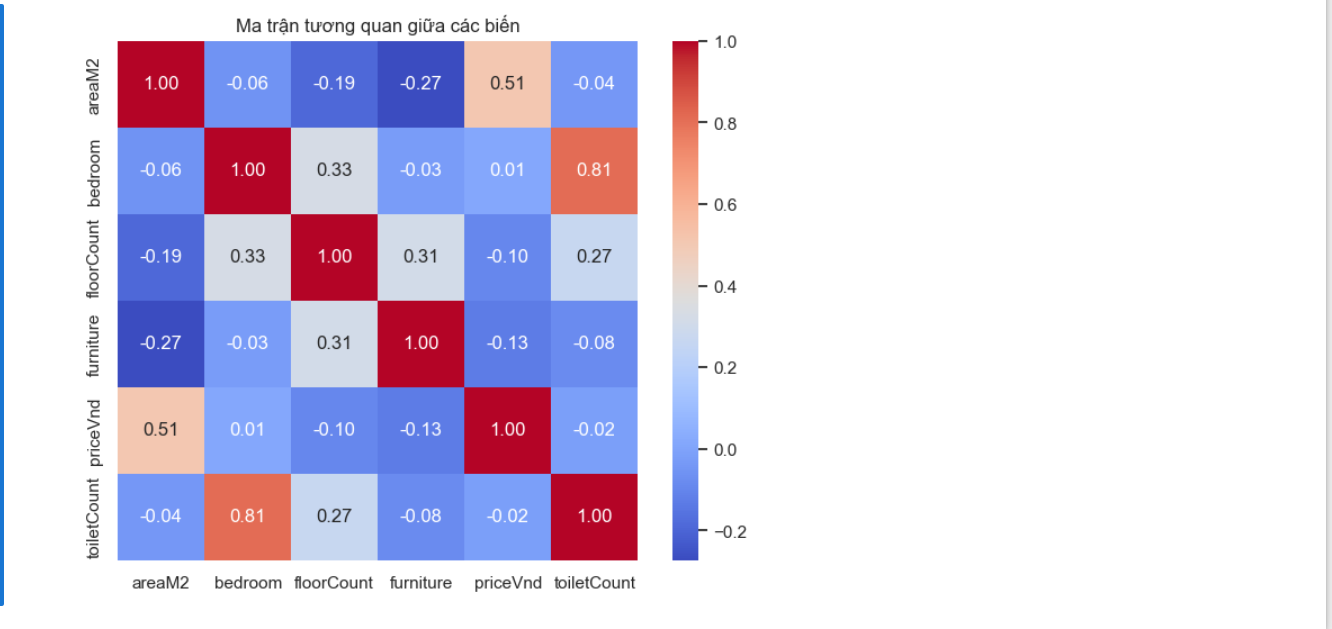
+ thực hiện lược bỏ thành phần không liên quan và chuyển sang dạng số để thực hiện huấn luyện dữ liệu



Dữ liệu được sử dụng để huấn luyện



+ biểu đồ thể hiện sự tương quan của dữ liệu



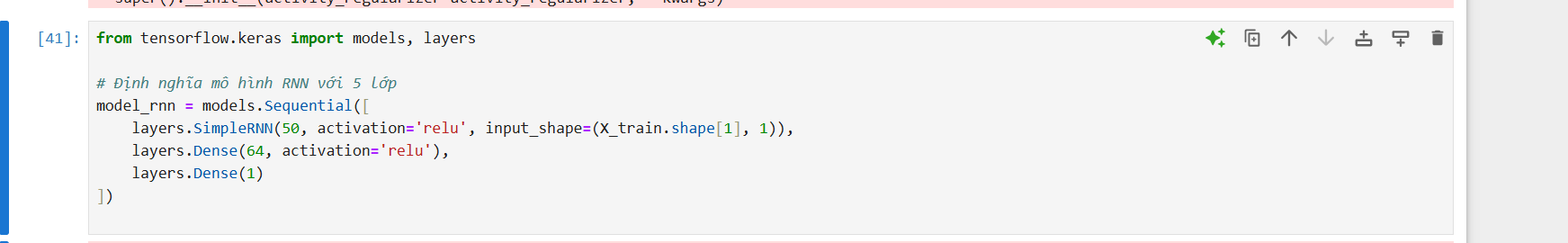
+ Thực hiện chia tập dữ liệu sử dụng skitlearn



B2: xây dựng mô hình CNN



B3: Xây dựng mô hình RNN



B4: Xây dựng mô hình LSTM

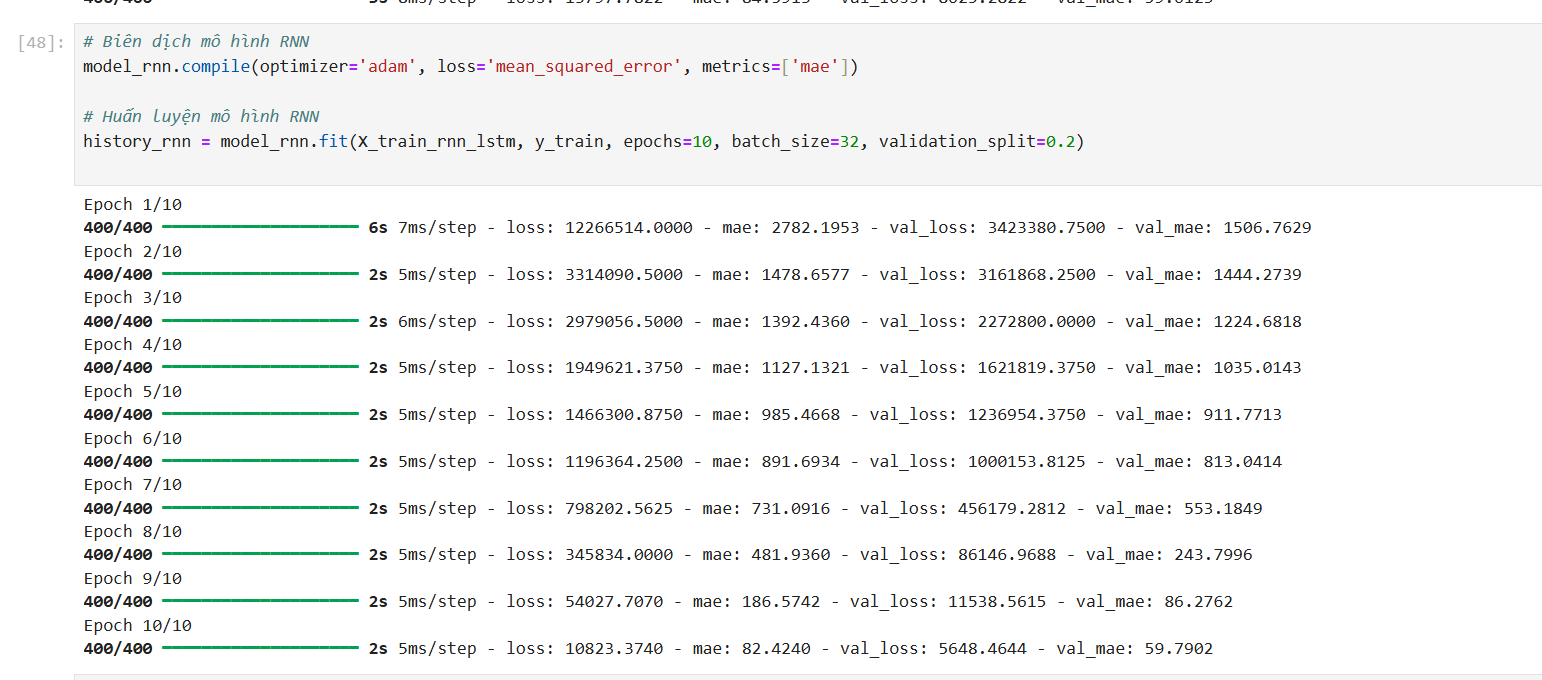


Huấn luyện mô hình

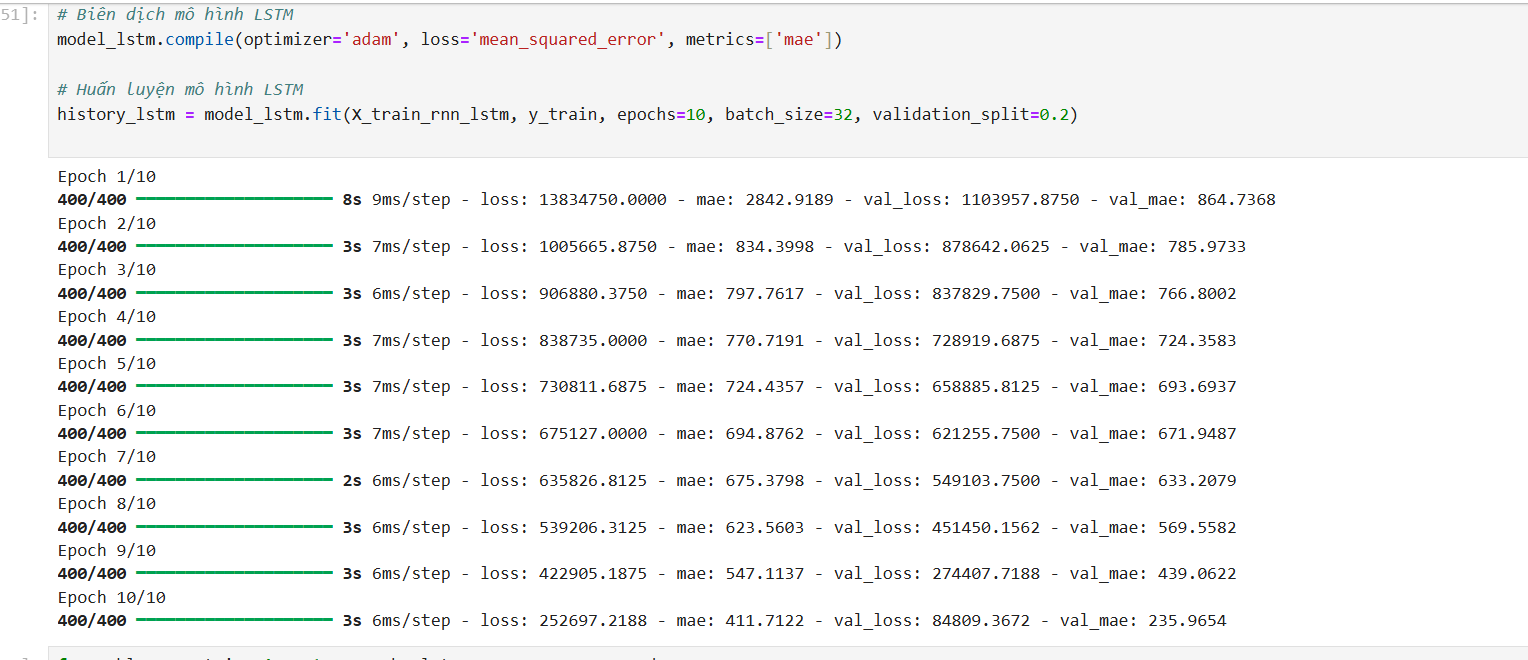
CNN



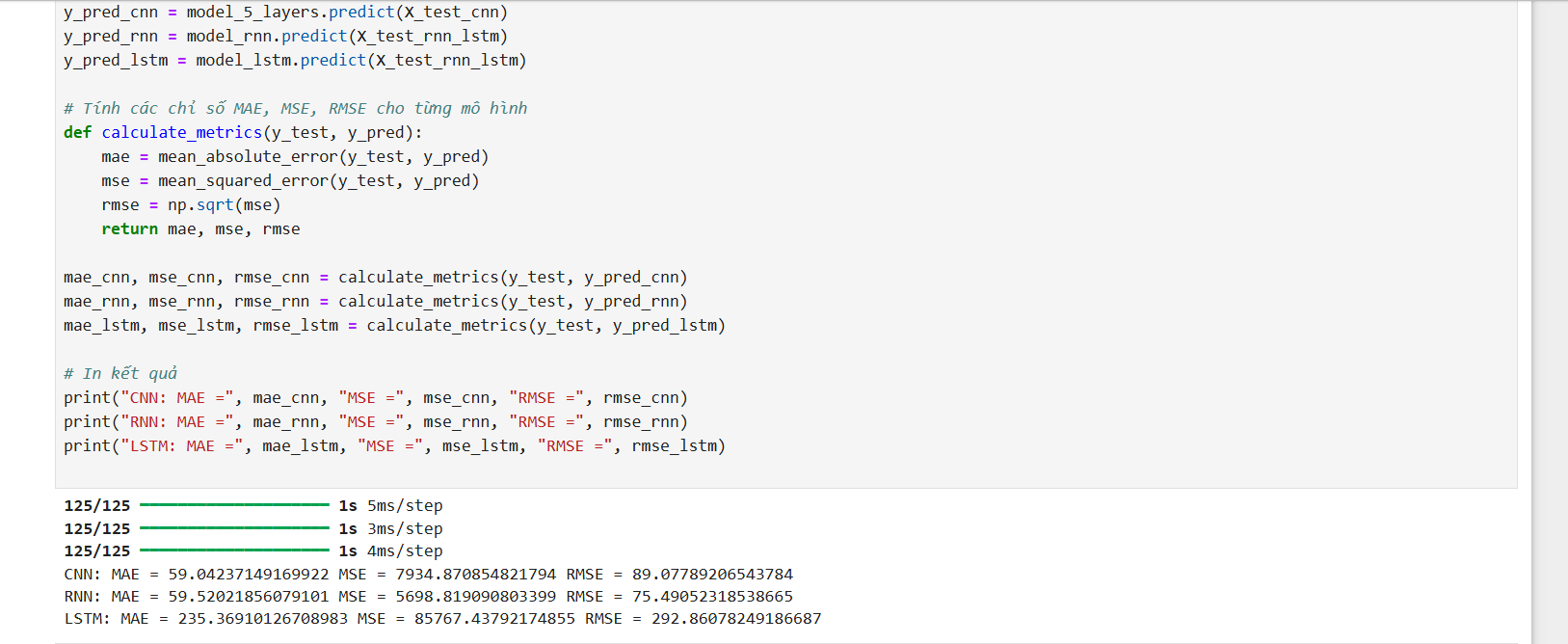
RNN



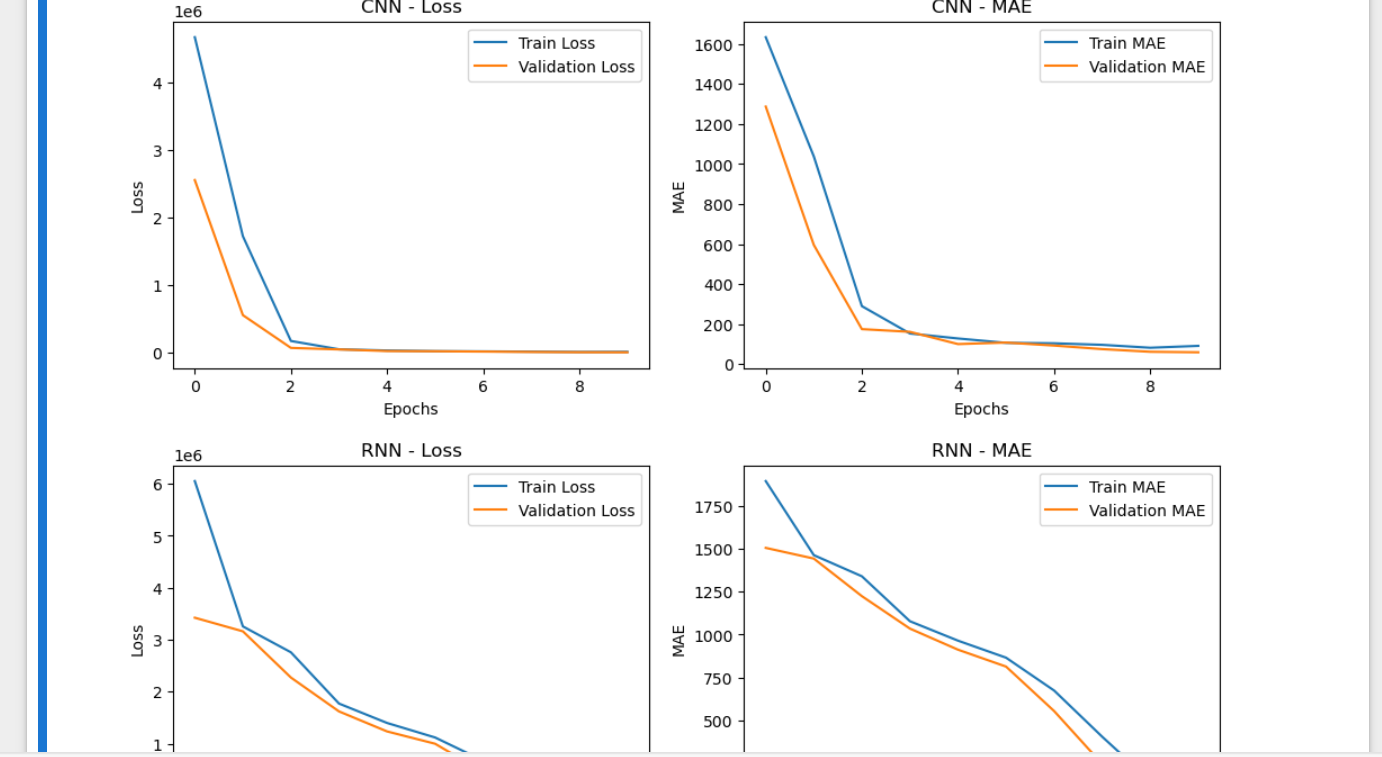
LSTM

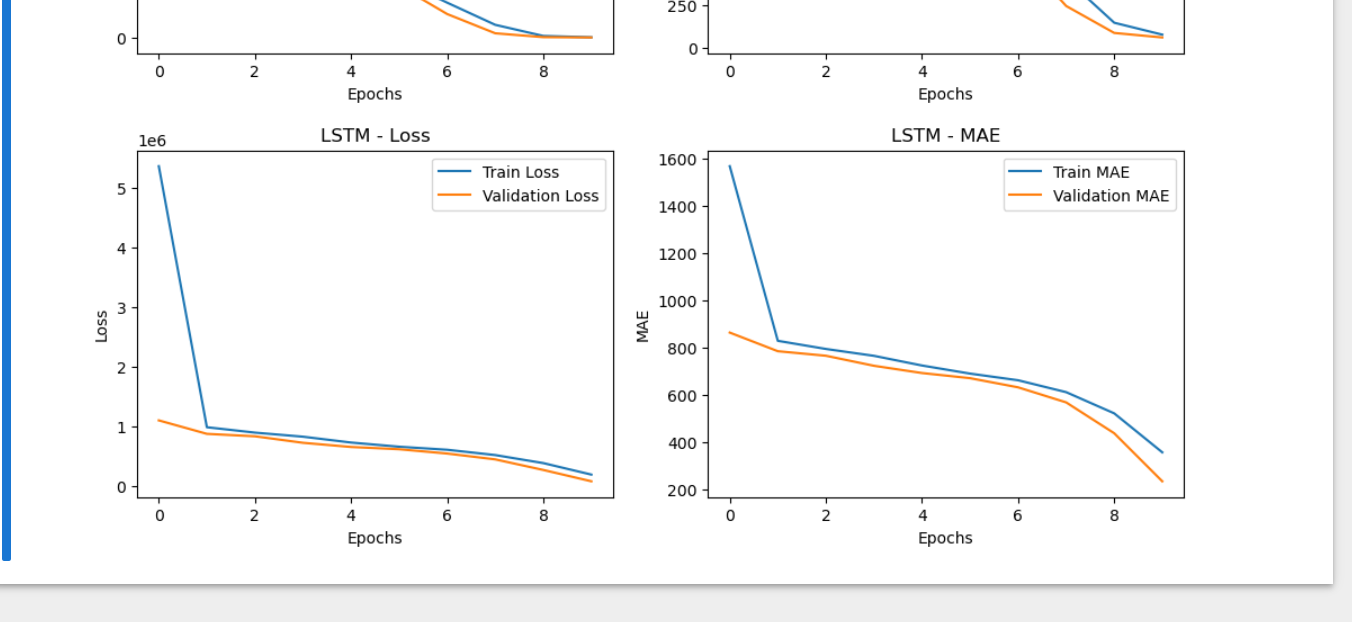


Tính toán các tham số mae, rmse, mse



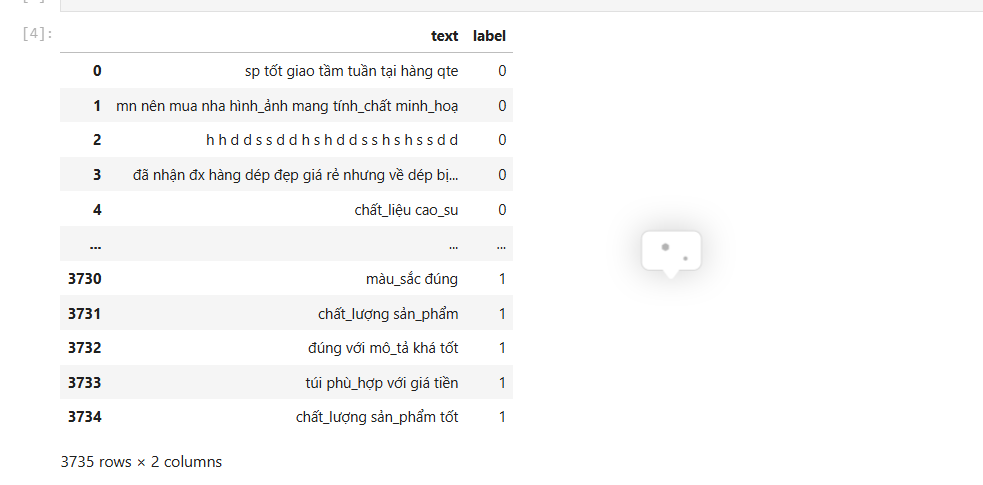
Vẽ biểu đồ so sánh



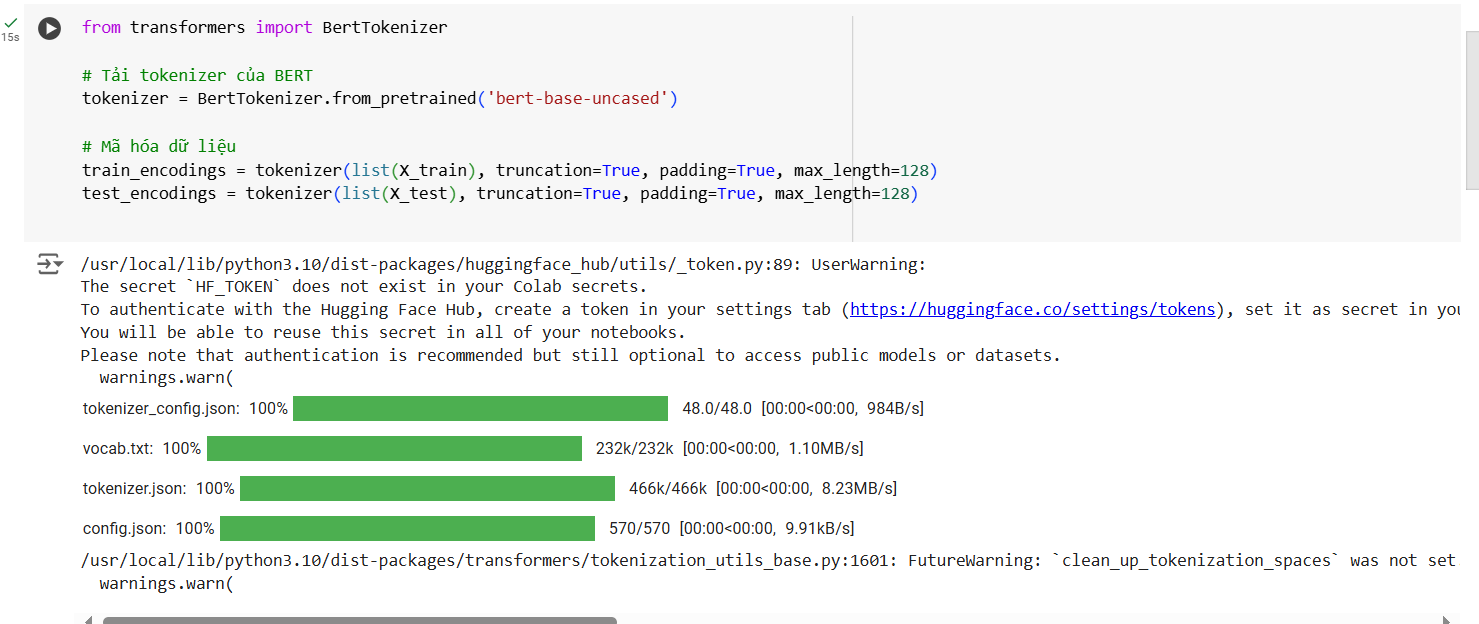


1. Tương tự câu 2 cho dữ liệu comment sản phẩm trên tiki, shoppe sử dụng bert

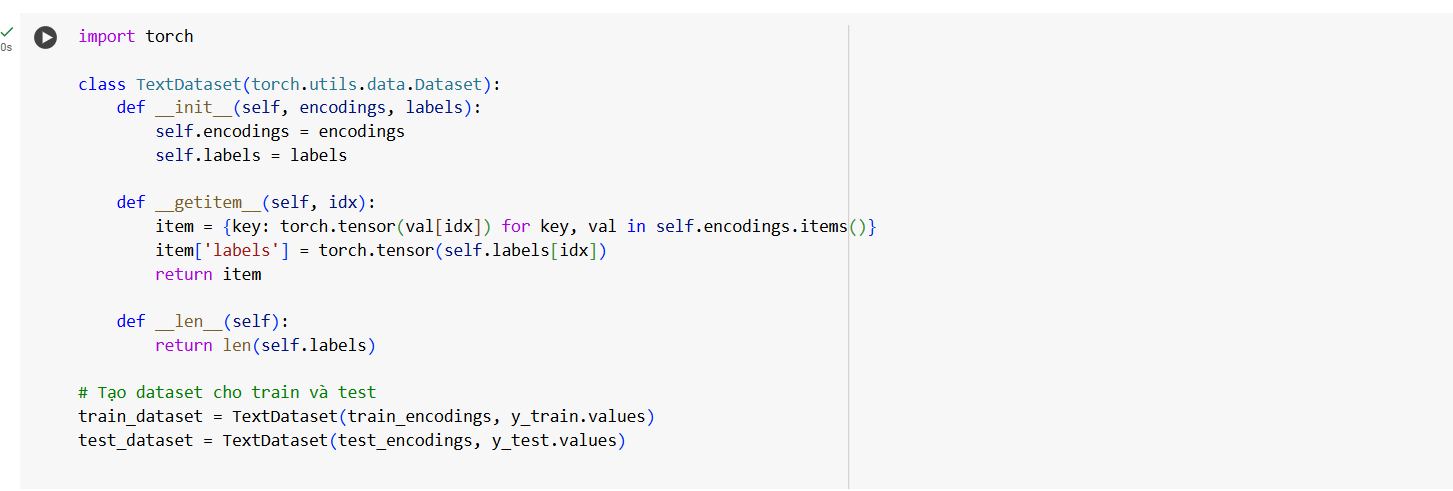
Hiển thị tập dữ liệu



Xây dựng mô hình



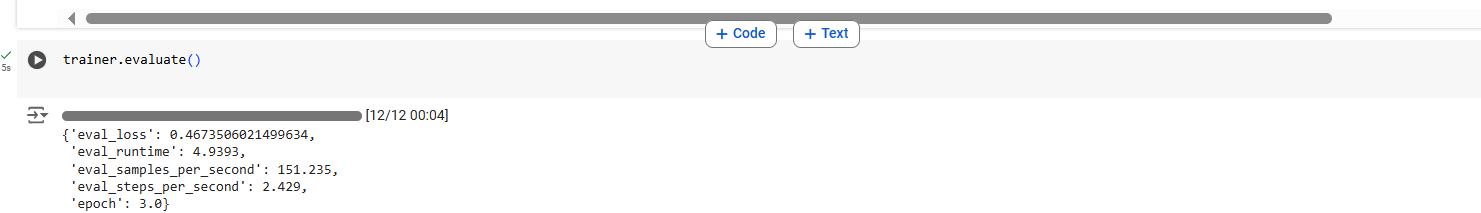
Chia tập dữ liệu



Huấn luyện mô hình



Đánh giá mô hình



Dự đoán kết quả

