HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BỘ MÔN THỰC TẬP CỞ SỞ

BÀI TẬP LỚN CUỐI KÌ

THỰC TẬP CỞ SỞ

Tên sinh viên	Mã sinh viên
1. Nguyễn Trung Kiên	B20DCCN357
2. Trần Thị Thắm	B20DCCN657
3. Vũ Thị Yến	B20DCCN754

Giảng viên hướng dẫn: Ths.Đinh Xuân Trường

HÀ NỘI,Tháng 6/2023

Lời cảm ơn

Lời đầu tiên chúng em xin chân thành cảm ơn Thầy vì Thầy đã tạo điều kiện thuận lợi nhất cho chúng em hoàn thành bài tập lớn của môn Thực tập cơ sở này. Chúng em với kiến thức hạn chế nên sẽ không thể tránh khỏi nhiều sai sót và với mong muốn được học hỏi, chúng em rất mong nhận được sự góp ý và hướng dẫn thêm của thầy để em rút kinh nghiệm cho những bài tập tiếp theo được tốt hơn.

Em xin chân thành cảm ơn!

THÀNH VIÊN THAM GIA VÀ ĐÓNG GÓP:

Tên sinh viên	**Công Việc**
Trần Thị Thắm	- Làm slide - Chuẩn bị nội dung phần tổng quan - Chuẩn bị nội dung phần thu thập dữ liệu - Thuyết trình
Nguyễn Trung Kiên	- Làm báo cáo - Chuẩn bị nội dung phần đánh giá mô hình - Tìm hiểu thuật toán LSTM
Vũ Thị Yến	- Chuẩn bị nội dung phần thực nghiệm - Demo code

Mục lục

Danh sách hình vẽ

I. Nội dung

1. Tổng quan

Xu hướng chơi cổ phiếu ngày càng được phát triển rộng rãi, đông người tìm hiểu và đầu tư. Vì vậy mà việc con người tìm hiểu cho mình những trang web dự đoán giá, dự đoán xu hướng phát triển ngày càng nhiều.

Ở đây, chúng em xây dựng một trang web dự đoán giá cổ phiếu dựa vào thuật toán LSTM (Long - Short Term Memory) - bằng cách xử lý thông tin chuỗi và tận dụng các cơ chế cập nhật thông tin, LSTM có khả năng nhận biết và học được các mẫu phức tạp trong dữ liệu thời gian, giúp đưa ra dự đoán chính xác về giá cổ phiếu trong tương lai. Với sự kết hợp giữa công nghệ tiên tiến và phân tích dữ liệu mạnh mẽ, nền tảng này sử dụng các mô hình và thuật toán chính xác để dự đoán giá cổ phiếu trong các ngày tiếp theo. Dựa trên dữ liệu lịch sử và yếu tố tài chính hiện tại, trang web sẽ cung cấp cho bạn các dự đoán rõ ràng và chi tiết về xu hướng giá cổ phiếu. Bạn sẽ nhận được thông tin về các mức hỗ trợ và kháng cự quan trọng, các tín hiệu giao dịch tiềm năng và các yếu tố có thể ảnh hưởng đến giá cổ phiếu trong thời gian tới. Điểm đáng chú ý là trang web dự đoán giá cổ phiếu tập trung duy nhất vào chức năng dự đoán giá trong các ngày, cho phép bạn tìm hiểu và đưa ra quyết định đầu tư dựa trên thông tin cụ thể về tương lai gần. Bạn sẽ tiết kiệm được thời gian và công sức trong việc phân tích và theo dõi thị trường, và thay vào đó tập trung vào những thông tin quan trọng để đưa ra quyết định đúng đắn.

Với trang web dự đoán giá cổ phiếu, bạn sẽ có được một công cụ mạnh mẽ để đánh giá rủi ro và tiềm năng sinh lợi của các cổ phiếu trong tương lai gần. Bạn có thể tận dụng thông tin chính xác và chi tiết để đưa ra quyết định đầu tư thông minh và tối ưu hóa tỷ lệ thành công của mình trên thị trường cổ phiếu.

2. Thu thập dữ liệu

Dữ liệu được tải về từ trang https://finance.yahoo.com

Sử dụng thư viện yfinance trong Python được để lấy dữ liệu tài chính từ Yahoo Finance. Với yfinance, ta có thể truy cập và tải về dữ liệu cổ phiếu, chỉ số thị trường và nhiều loại dữ liệu tài chính khác từ Yahoo Finance thông qua API của nó. Thư viện yfinance cung cấp một cách dễ dàng để lấy dữ liệu tài chính trong Python. Ta có thể lấy thông tin như giá cổ phiếu, khối lượng giao dịch, giá mở cửa, giá cao nhất, giá thấp nhất và nhiều thông tin khác. Thư viện này cũng hỗ trợ việc tải dữ liệu lịch sử, dữ liệu theo khoảng thời gian cụ thể và thậm chí cả dữ liệu phân cấp (hierarchical data) như thông tin về tài sản cơ bản (fundamental data) và dữ liệu kỹ thuật (technical data).

Dữ liệu lấy từ Yahoo Finance bao gồm 7 trường: Date, Open, High, Low, Close, Adj Close, Volume :

1. Date: Đây là trường thể hiện ngày tháng của dữ liệu tài chính tương ứng. Nó cho biết ngày giao dịch hoặc thời điểm xác định của mỗi dòng dữ liệu.

2. Open: Trường Open thể hiện giá mở cửa (opening price) của cổ phiếu hoặc tài sản tài chính tại thời điểm bắt đầu phiên giao dịch.

3. High: Trường High thể hiện giá cao nhất (highest price) đạt được trong phiên giao dịch.

4. Low: Trường Low thể hiện giá thấp nhất (lowest price) đạt được trong phiên giao dịch.

5. Close: Trường Close thể hiện giá đóng cửa (closing price) của cổ phiếu hoặc tài sản tài chính tại thời điểm kết thúc phiên giao dịch.

6. Adj Close (Adjusted Close): Trường Adj Close thể hiện giá đóng cửa đã điều chỉnh (adjusted closing price) của cổ phiếu hoặc tài sản tài chính. Điều chỉnh giá đóng cửa thường liên quan đến việc điều chỉnh cho các yếu tố như chia cổ tức (stock splits), cổ phiếu thưởng (stock dividends) và các yếu tố khác có thể ảnh hưởng đến giá cổ phiếu.

7. Volume: Trường Volume thể hiện số lượng cổ phiếu hoặc tài sản tài chính đã được giao dịch trong phiên tương ứng. Đơn vị thông thường của Volume là số lượng cổ phiếu.

Bộ dữ liệu được lấy từ ngày 08/03/2020 đến 28/12/2021 gồm 2795 dòng và 7 trường:

Date	Open	High	Low	Close*	Adj Close**	Volume
Dec 28, 2021	180.16	181.33	178.53	179.29	177.74	79,144,300
Dec 27, 2021	177.09	180.42	177.07	180.33	178.77	74,919,600
Dec 23, 2021	175.85	176.85	175.27	176.28	174.75	68,356,600
Dec 22, 2021	173.04	175.86	172.15	175.64	174.12	92,135,300
Dec 21, 2021	171.56	173.20	169.12	172.99	171.49	91,185,900
Dec 20, 2021	168.28	170.58	167.46	169.75	168.28	107,499,100
Dec 17, 2021	169.93	173.47	169.69	171.14	169.66	195,432,700
Dec 16, 2021	179.28	181.14	170.75	172.26	170.77	150,185,800
Dec 15, 2021	175.11	179.50	172.31	179.30	177.75	131,063,300
Dec 14, 2021	175.25	177.74	172.21	174.33	172.82	139,380,400
Dec 13, 2021	181.12	182.13	175.53	175.74	174.22	153,237,000
Dec 10, 2021	175.21	179.63	174.69	179.45	177.90	115,402,700

Hình 1: Bộ dữ liệu được tải về từ trang Yahoo Finance

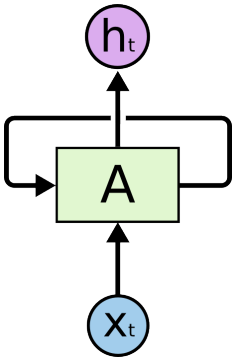
3. Đánh giá mô hình

3.1. Thuật toán LSTM

LSTM là một mạng cải tiến của RNN nhằm giải quyết vấn đề nhớ các bước dài của RNN.

Mạng nơ-ron truyền thống không thể lưu lại dữ liệu đã thực hiện trước đó, đây có thể coi là một khuyết điểm chính của mạng nơ-ron truyền thống. Ví dụ muốn phân loại các bối cảnh xảy ra trong một bộ phim thì cần phải nhớ các tình huống xảy ra trước đó thì mới hiểu được tình huống hiện tại, cái này không thể làm ở mạng nơ-ron truyền thống.

Mạng nơ-ron hồi quy (Recurrent Neutron Network) sinh ra để giải quyết vấn đề này. Mạng chứa các vòng lặp bên trong cho phép thông tin lưu lại được.



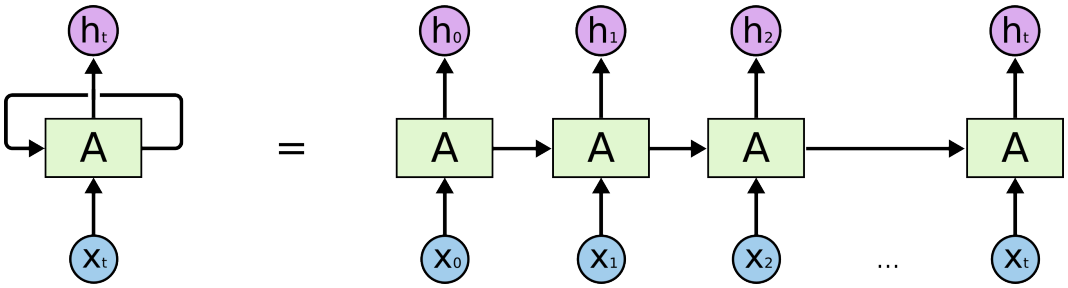
Hình 2: Mạng nơ ron truy hồi với vòng lặp

Hình vẽ trên mô tả một đoạn của mạng nơ-ron hồi quy A với đầu vào là xt và đầu ra là ht. LSTM là một dạng đặc biệt của mạng nơ-ron hồi quy, với nhiều bài toán thì nó tốt hơn mạng hồi quy thuần.

Một điểm nổi bật của RNN chính là ý tưởng kết nối các thông tin phía trước để dự đoán cho hiện tại.

Việc này tương tự như ta sử dụng các cảnh trước của bộ phim để hiểu được cảnh hiện thời. nhưng RNN có làm được điều này hay không thì còn tùy.

Đôi lúc ta chỉ cần xem lại thông tin vừa có thôi là đủ để biết được tình huống hiện tại. Ví dụ, ta có câu: "các đám may trên bầu trời" thì ta chỉ cần đọc tới "các đám may trên bầu" là đủ biết được chữ tiếp theo là "trời" rồi. Trong tình huống này, khoảng cách tới thông tin có được cần để dự đoán là nhỏ, nên RNN hoàn toàn có thể học được.



Hình 3: Cấu trúc trải phẳng của mạng nơ ron truy hồi

Nhưng trong nhiều tình huống ta buộc phải sử dụng nhiều ngữ cảnh hơn để suy luận. Ví dụ, dự đoán chữ cuối cùng trong đoạn: "I grew up in France... I speak fluent French.". Rõ ràng là các thông tin gần ("I speak fluent") chỉ có phép ta biết được đằng sau nó sẽ là tên của một ngôn ngữ nào đó, còn không thể nào biết được đó là tiếng gì. Muốn biết là tiếng gì, thì ta cần phải có thêm ngữ cảnh "I grew up in France" nữa mới có thể suy luận được. Rõ ràng là khoảng cách thông tin lúc này có thể đã khá xa rồi.

Thật không may là với khoảng cách càng lớn dần thì RNN bắt đầu không thể nhớ và học được nữa.

Về mặt lý thuyết, rõ ràng là RNN có khả năng xử lý các phụ thuộc xa (long-term dependencies). Chúng ta có thể xem xét và cài đặt các tham số sao cho khéo là có thể giải quyết được vấn đề này. Tuy nhiên, đáng tiếc trong thực tế RNN có vẻ không thể học được các tham số đó. Vấn đề này đã được khám phá khá bởi

Hochreiter và Bengio, et al. (1994), trong các bài báo của mình, họ đã tìm được những lý do căn bản để giải thích tại sao RNN không thể học được.

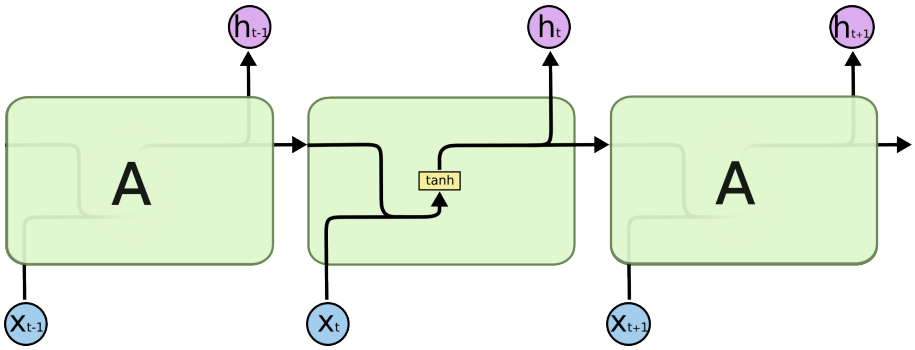
Tuy nhiên, LSTM không vấp phải vấn đề đó.

MẠNG LSTM

Mạng bộ nhớ dài-ngắn (Long Short Term Memory networks), thường được gọi là LSTM - là một dạng đặc biệt của RNN, nó có khả năng học được các phụ thuộc xa. LSTM được giới thiệu bởi Hochreiter Schmidhuber (1997), và sau đó đã được cải tiến và phổ biến bởi rất nhiều người trong ngành. Chúng hoạt động cực kì hiệu quả trên nhiều bài toán khác nhau nên dần đã trở nên phổ biến như hiện nay.

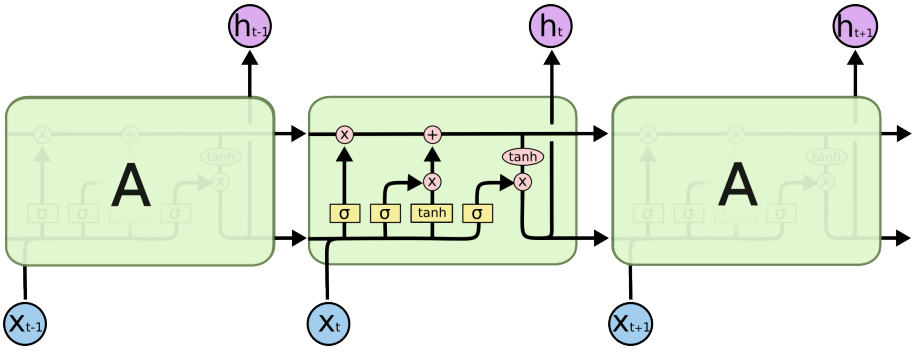
LSTM được thiết kế để tránh được vấn đề phụ thuộc xa (long-term dependency). Việc nhớ thông tin trong suốt thời gian dài là đặc tính mặc định của chúng, chứ ta không cần phải huấn luyện nó để có thể nhớ được. Tức là ngay nội tại của nó đã có thể ghi nhớ được mà không cần bất kì can thiệp nào.

Mọi mạng hồi quy đều có dạng là một chuỗi các mô-đun lặp đi lặp lại của mạng nơ-ron. Với mạng RNN chuẩn, các mô-dun này có cấu trúc rất đơn giản, thường là một tầng tanh.



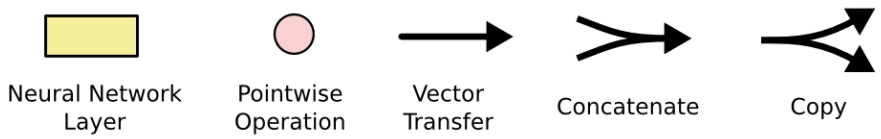
Hình 4: Sự lặp lại kiến trúc module trong mạng RNN chứa một tầng ẩn

LSTM cũng có kiến trúc dạng chuỗi như vậy, nhưng các mô-đun trong nó có cấu trúc khác với mạng RNN chuẩn. Thay vì chỉ có một tầng mạng nơ-ron, chúng có tới 4 tầng tương tác với nhau một cách rất đặc biệt.



Hình 5: Sự lặp lại kiến trúc module trong mạng LSTM chứa 4 tầng ẩn (3 sigmoid và 1 tanh) tương tác

Điều cần làm bây giờ là làm hãy làm quen với các kí hiệu mà ta sẽ sử dụng ở dưới đây:



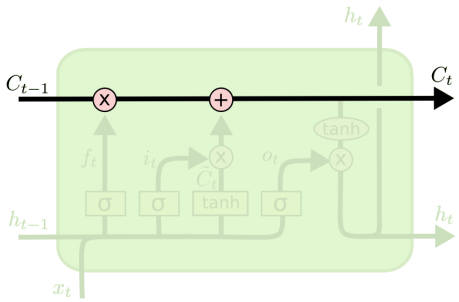
Hình 6: Diễn giải các kí hiệu trong đồ thị mạng nơ ron (áp dụng chung cho toàn bộ bài)

Ở sơ đồ trên, mỗi một đường mang một véc-tơ từ đầu ra của một nút tới đầu vào của một nút khác. Các hình trong màu hồng biểu diễn các phép toán như phép cộng véc-tơ chẳng hạn, còn các ô màu vàng được sử dụng để học trong các từng mạng nơ-ron. Các đường hợp nhau kí hiệu việc kết hợp, còn các đường rẽ nhánh ám chỉ nội dung của nó được sao chép và chuyển tới các nơi khác nhau.

Ý tưởng LSTM

Chìa khóa của LSTM là trạng thái tế bào (cell state) - chính đường chạy thông ngang phía trên của sơ đồ hình vẽ.

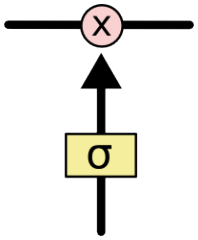
Trạng thái tế bào là một dạng giống như băng truyền. Nó chạy xuyên suốt tất cả các mắt xích (các nút mạng) và chỉ tương tác tuyến tính đôi chút. Vì vậy mà các thông tin có thể dễ dàng truyền đi thông suốt mà không sợ bị thay đổi.



Hình 7: Đường đi của ô trạng thái (cell state) trong mạng LSTM

LSTM có khả năng bỏ đi hoặc thêm vào các thông tin cần thiết cho trạng thái tế báo, chúng được điều chỉnh cẩn thận bởi các nhóm được gọi là cổng (gate).

Các cổng là nơi sàng lọc thông tin đi qua nó, chúng được kết hợp bởi một tầng mạng sigmoid và một phép nhân.



Hình 8: Một cổng của hàm sigmoid trong LSTM

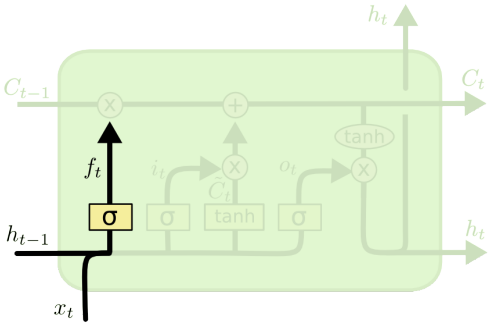
Tầng sigmoid sẽ cho đầu ra là một số trong khoản [0, 1], mô tả có bao nhiêu thông tin có thể được thông qua. Khi đầu ra là 0 thì có nghĩa là không cho thông tin nào qua cả, còn khi là 1 thì có nghĩa là cho tất cả các thông tin đi qua nó.

Một LSTM gồm có 3 cổng như vậy để duy trì và điều hành trạng thái của tế bào.

Bên trong LSTM

Bước đầu tiên của LSTM là quyết định xem thông tin nào cần bỏ đi từ trạng thái tế bào. Quyết định này được đưa ra bởi tầng sigmoid - gọi là "tầng cổng quên" (forget gate layer). Nó sẽ lấy đầu vào là ht1 và xt rồi đưa ra kết quả là một số trong khoảng [0, 1] cho mỗi số trong trạng thái tế bào Ct1. Đầu ra là 1 thể hiện rằng nó giữ toàn bộ thông tin lại, còn 0 chỉ rằng toàn bộ thông tin sẽ bị bỏ đi.

Quay trở lại với ví dụ mô hình ngôn ngữ dự đoán từ tiếp theo dựa trên tất cả các từ trước đó, với những bài toán như vậy, thì trạng thái tế bào có thể sẽ mang thông tin về giới tính của một nhân vật nào đó giúp ta sử dụng được đại từ nhân xưng chuẩn xác. Tuy nhiên, khi đề cập tới một người khác thì ta sẽ không muốn nhớ tới giới tính của nhân vật nữa, vì nó không còn tác dụng gì với chủ thế mới này.

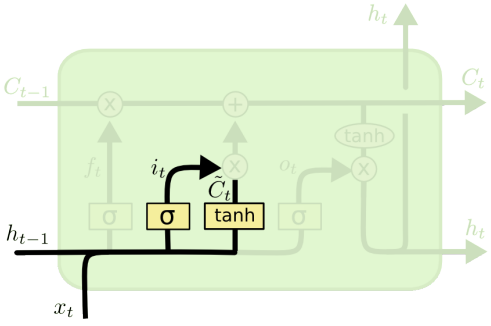


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Hình 9: Tầng cổng quên (forget gate layer)

Bước tiếp theo là quyết định xem thông tin mới nào ta sẽ lưu vào trạng thái tế bào. Việc này gồm 2 phần. Đầu tiên là sử dụng một tầng sigmoid được gọi là "tầng cổng vào" (input gate layer) để quyết định giá trị nào ta sẽ cập nhập. Tiếp theo là một tầng tanh tạo ra một véc-tơ cho giá trị mới Ct nhằm thêm vào cho trạng thái. Trong bước tiếp theo, ta sẽ kết hợp 2 giá trị đó lại để tạo ra một cập nhập cho trạng thái.

Chẳng hạn với ví dụ mô hình của chúng ta, ta sẽ muốn thêm giới tính của nhân vật mới này vào trạng thái tế bào và thay thế giới tính của nhân vật trước đó.



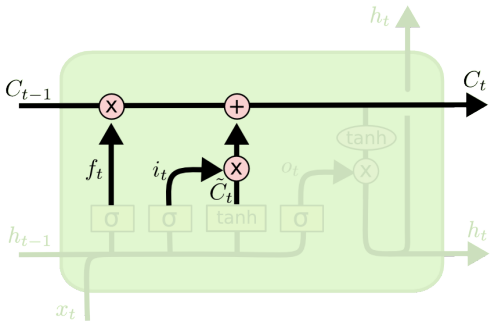
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Hình 10: Cập nhật giá trị cho ô trạng thái bằng cách kết hợp 2 kết quả từ tầng cổng vào và tầng ẩn hàm tanh

Giờ là lúc cập nhập trạng thái tế bào cũ Ct 1 thành trạng thái mới Ct. Ở các bước trước đó đã quyết định những việc cần làm, nên giờ ta chỉ cần thực hiện là xong.

Ta sẽ nhân trạng thái cũ với ft để bỏ đi những thông tin ta quyết định quên lúc trước. Sau đó cộng thêm it * Ct . Trạng thái mới thu được này phụ thuộc vào việc ta quyết định cập nhập mỗi giá trị trạng thái ra sao.

Với bài toán mô hình như thế này, chính là việc ta bỏ đi thông tin về giới tính của nhân vật cũ, và thêm thông tin về giới tính của nhân vật mới như ta đã quyết định ở các bước trước đó.

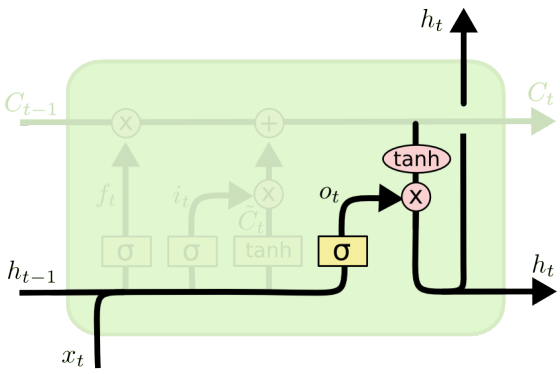


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Hình 11: Ô trạng thái mới

Cuối cùng, ta cần quyết định xem ta muốn đầu ra là gì. Giá trị đầu ra sẽ dựa vào trạng thái tế bào, nhưng sẽ được tiếp tục sàng lọc. Đầu tiên, ta chạy một tầng sigmoid để quyết định phần nào của trạng thái tế bào ta muốn xuất ra. Sau đó, ta đưa nó trạng thái tế bảo qua một hàm tanh để co giá trị nó về khoảng [-1, 1], và nhân nó với đầu ra của cổng sigmoid để được giá trị đầu ra ta mong muốn.

Với ví dụ về mô hình ngôn ngữ, chỉ cần xem chủ thể mà ta có thể đưa ra thông tin về một trạng từ đi sau đó. Ví dụ, nếu đầu ra của chủ thể là số ít hoặc số nhiều thì ta có thể biết được dạng của trạng từ đi theo sau nó phải như thế nào.



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Hình 12: Điều chỉnh thông tin ở đầu ra thông qua hàm tanh

3.2. Mô hình ứng dụng thuật toán LSTM trong dự đoán giá cổ phiếu

Khai báo thư viện :

```
import numpy as np
# hỗ trợ cho việc tính toán các mảng nhiều chiều
import pandas as pd
# thao tác và phân tích dữ liệu
import yfinance as yf
import matplotlib.pyplot as plt
# vẽ biểu đồ, đồ thị
%matplotlib inline
from sklearn.preprocessing import MinMaxScaler
# hàm đưa dữ liệu về giá trị (0,1)
from keras.preprocessing.sequence import TimeseriesGenerator
# Gộp dữ liệu vào chuyển thành dữ liệu time series
from keras.models import Sequential
# khởi tạo mạng neurol
from keras.layers import Dense
# một lớp để chuyển dữ liệu từ lớp input vào model ????
from keras.layers import LSTM
# mô hình LSTM
from keras.layers import Dropout
# giúp bỏ bớt các node, giúp lọc lại những node có thông tin cần thiết
from tensorflow import keras
# from datetime import datatime
plt.style.use("fivethirtyeight")
# style của thư viện mathplotlib
import warnings
warnings.filterwarnings("ignore")
# Những thư viện cần dùng
```

Hình 13: Đoạn code khai báo thư viện

numpy: Thư viện numpy cung cấp hỗ trợ cho việc tính toán các mảng nhiều chiều. Nó cung cấp các hàm và phương pháp để thực hiện các phép toán số học, ma trận, vectơ hóa và xử lý dữ liệu.

pandas: Thư viện pandas được sử dụng để thao tác và phân tích dữ liệu. Nó cung cấp các cấu trúc dữ liệu linh hoạt như DataFrame và Series để lưu trữ và xử lý dữ liệu dạng bảng. Pandas cung cấp các phương pháp để truy vấn, lọc, sắp xếp và xử lý dữ liệu.

yfinance: Thư viện yfinance được sử dụng để truy xuất dữ liệu tài chính từ Yahoo Finance. Nó cung cấp các công cụ để lấy thông tin về giá cả, khối lượng giao dịch và các chỉ số tài chính của các công ty và các thị trường tài chính khác.

matplotlib.pyplot: Thư viện matplotlib.pyplot là một công cụ mạnh mẽ để vẽ biểu đồ và đồ thị. Nó cung cấp các hàm để tạo các biểu đồ đường, cột, scatter và các loại biểu đồ khác để trực quan hóa dữ liệu.

sklearn.preprocessing.MinMaxScaler: MinMaxScaler là một class trong thư viện scikit-learn (sklearn) được sử dụng để chuẩn hóa dữ liệu về khoảng giá trị (0, 1). Nó thực hiện việc chia tỷ lệ các giá trị trong dữ liệu thành một phạm vi mong muốn, giúp đưa dữ liệu về cùng một phạm vi và tránh ảnh hưởng của giá trị cực đại và cực tiểu.

keras.preprocessing.sequence.TimeseriesGenerator: TimeseriesGenerator là một class trong thư viện keras được sử dụng để gộp dữ liệu thành dữ liệu chuỗi thời gian. Nó chuyển đổi một chuỗi dữ liệu thành các cặp dữ liệu huấn luyện và nhãn tương ứng để sử dụng trong mô hình dự báo chuỗi thời gian.

keras.models.Sequential: Sequential là một class trong thư viện keras, được sử dụng để khởi tạo một mạng nơ-ron tuần tự. Nó cho phép chúng ta xây dựng các

mô hình nơ-ron bằng cách thêm các lớp một cách tuần tự.

keras.layers.Dense: Dense là một lớp trong thư viện keras, được sử dụng để thêm một lớp đầy đủ kết nối (fully connected layer) vào mô hình nơ-ron. Lớp Dense chuyển dữ liệu từ lớp đầu vào vào mô hình và thực hiện các phép tính tuyến tính và phi tuyến để tạo ra đầu ra.

keras.layers.LSTM: LSTM là một lớp trong thư viện keras, được sử dụng để thêm một lớp mô hình LSTM vào mạng nơ-ron. Lớp LSTM thực hiện các phép tính đặc biệt để xử lý và nhớ thông tin từ các chuỗi dữ liệu.

keras.layers.Dropout: Dropout là một lớp trong thư viện keras, được sử dụng để giảm overfitting trong mô hình nơ-ron. Lớp Dropout tạm thời loại bỏ ngẫu nhiên một số lượng node trong mạng nơ-ron, giúp lọc lại những node có thông tin cần thiết và làm giảm khả năng quá khớp.

tensorflow.keras: Thư viện tensorflow.keras là một phần của thư viện tensorflow và được sử dụng để xây dựng và huấn luyện mô hình học sâu. Nó cung cấp các công cụ và hàm để xây dựng và tối ưu hóa các mạng nơ-ron.

matplotlib.style.use("fivethirtyeight"): Phương thức style.use() trong thư viện matplotlib được sử dụng để thiết lập kiểu biểu đồ. Trong trường hợp này, "fivethirtyeight" là một kiểu biểu đồ được thiết kế để tạo ra các biểu đồ với phong cách của trang web FiveThirtyEight.

warnings.filterwarnings("ignore"): Hàm filterwarnings() trong thư viện warnings được sử dụng để bỏ qua hoặc ẩn các cảnh báo không cần thiết hoặc phiền nhiễu. Trong trường hợp này, "ignore" được sử dụng để ẩn các cảnh báo.

Thiết lập dữ liệu vào và in ra dữ liệu:

```
#@title Chương trình dự đoán cổ phiếu bằng thuật toán LSTM
Macophieu = "AAPL" #@param ["GOOG", "AMZN", "FB","AAPL","TSLA"]

prop = 'Close'  #@param ["Open", "High", "Low","Close","Volume"]

companyName = 'Apple' #@param["Google", "Amazon", "Facebook", "Apple", "Tesla"]

# Chọn cổ phiếu cần dự đoán
tickerData = yf.Ticker(Macophieu)
#@title Date fields
start ='2010-11-10'#@param {type:"date"}
end ='2023-05-26'#@param {type:"date"}
tickerDf = tickerData.history(period='1d', start=start, end=end)
#lấy giá trị từ ngày đến những ngày trước đó
data = tickerDf
data.reset_index(inplace=True)
# đánh số thứ tự lại cho data thay cho cột ngày
print(data)
```

Hình 14: Đoạn code thiết lập dữ liệu vào và in ra dữ liệu

Macophieu: Biến này cho phép người dùng chọn mã cổ phiếu cần dự đoán. Chương trình cho phép lựa chọn giữa các mã cổ phiếu như GOOG, AMZN, META, AAPL, và TSLA.

prop: Biến này cho phép người dùng chọn thuộc tính của cổ phiếu cần dự đoán. Các thuộc tính có thể lựa chọn bao gồm "Open" (giá mở cửa), "High" (giá cao nhất), "Low" (giá thấp nhất), "Close" (giá đóng cửa), và "Volume" (khối lượng giao dịch).

companyName: Biến này cho phép người dùng chọn tên công ty tương ứng với mã cổ phiếu đã chọn. Các lựa chọn bao gồm "Google", "Amazon", "Meta", "Apple", và "Tesla".

tickerData: Biến này tạo đối tượng Ticker từ thư viện yfinance để truy cập thông tin của cổ phiếu dựa trên mã cổ phiếu đã chọn.

start và end: Biến này cho phép người dùng chọn khoảng thời gian muốn dự đoán giá cổ phiếu, được chọn thông qua các lựa chọn ngày. tickerDf: Biến này lấy dữ liệu lịch sử của cổ phiếu trong khoảng thời gian đã chọn và lưu trữ trong DataFrame.

data: Biến này là một bản sao của tickerDf, được sử dụng để thực hiện các xử lý và phân tích dữ liệu tiếp theo.

data.reset_index(inplace=True): Dòng này đặt lại chỉ số cho DataFrame data, thay thế cột ngày bằng một chỉ số số nguyên tăng dần.

print(data): Dòng này in ra DataFrame data, hiển thị thông tin về giá cổ phiếu và các thuộc tính trong khoảng thời gian đã chọn.

Kết Quả :

```
                       Date        Open        High         Low  \
0    2010-11-10 00:00:00-05:00    9.611635    9.676292    9.517838
1    2010-11-11 00:00:00-05:00    9.561858    9.665067    9.539092
2    2010-11-12 00:00:00-05:00    9.592210    9.607387    9.216718
3    2010-11-15 00:00:00-05:00    9.363335    9.426473    9.296857
4    2010-11-16 00:00:00-05:00    9.280162    9.337230    9.085889
...                        ...         ...         ...         ...
3151 2023-05-19 00:00:00-04:00  176.389999  176.389999  174.940002
3152 2023-05-22 00:00:00-04:00  173.979996  174.710007  173.449997
3153 2023-05-23 00:00:00-04:00  173.130005  173.380005  171.279999
3154 2023-05-24 00:00:00-04:00  171.089996  172.419998  170.520004
3155 2023-05-25 00:00:00-04:00  172.410004  173.899994  171.690002

           Close     Volume  Dividends  Stock Splits
0       9.653830  384227200        0.0           0.0
1       9.611945  361284000        0.0           0.0
2       9.350280  795846800        0.0           0.0
3       9.320230  403606000        0.0           0.0
4       9.154796  657650000        0.0           0.0
...          ...        ...        ...           ...
3151  175.160004   55772400        0.0           0.0
3152  174.199997   43570900        0.0           0.0
3153  171.559998   50747300        0.0           0.0
3154  171.839996   45143500        0.0           0.0
3155  172.990005   56058300        0.0           0.0

[3156 rows x 8 columns]
```

Hình 15: Kết quả sau khi chạy đoạn code thiết lập dữ liệu vào và in ra dữ liệu

Vẽ biểu đồ với các dữ liệu đã chọn :

```
plt.figure(figsize = (16,6))
plt.plot(data['Date'],data[prop])
plt.title( prop + " history ")
plt.xlabel('Date', fontsize = 18)
plt.ylabel(f'{prop}', fontsize = 18)
plt.show()
```

Hình 16: Đoạn code dùng để vẽ biểu đồ

plt.figure(figsize=(16,6)): Dòng này tạo một hình ảnh mới để vẽ biểu đồ với kích thước (chiều rộng, chiều cao) là (16,6) inch.

plt.plot(data['Date'], data[prop]): Dòng này vẽ biểu đồ đường bằng hàm plot của thư viện matplotlib. Trục x của biểu đồ được lấy từ cột 'Date' trong DataFrame data, và trục y được lấy từ cột prop đã chọn.

plt.title(prop + " history"): Dòng này đặt tiêu đề cho biểu đồ, sử dụng thuộc tính đã chọn (prop) cùng với từ "history".

plt.xlabel('Date', fontsize=18): Dòng này đặt nhãn cho trục x của biểu đồ là "Date", với kích thước chữ là 18.

plt.ylabel(f'prop', fontsize=18): Dòng này đặt nhãn cho trục y của biểu đồ với tên thuộc tính đã chọn (prop), với kích thước chữ là 18.

plt.show(): Dòng này hiển thị biểu đồ đã vẽ trên màn hình.

Kết quả chạy ra :

Hình 17: Kết qủa vẽ biểu đồ với các dữ liệu đã chọn

Chia dữ liệu để tập huấn và kiểm tra :

```
data_end = int(np.floor(0.8*(data.shape[0])))
# lấy cái mốc là data_end theo tử lệ 2:8
train = data[0:data_end][prop]
# lấy 80% là giá đóng cho tập train
test = data[data_end:][prop]
# lấy 20% data là giá đóng cho tập test
date_test = data[data_end:]['Date']
# lấy 20% là ngày trùng vs tập test
```

Hình 18: Đoạn code chia dữ liệu để tập huấn và kiểm tra

Phần code trên được sử dụng để chia dữ liệu thành tập huấn luyện và tập kiểm tra :

- data_end = int(np.floor(0.8*(data.shape[0]))): Dòng này tính toán chỉ số data_end dựa trên tỷ lệ 80% của số lượng dữ liệu trong DataFrame data. data_end là chỉ số tương ứng với điểm chia giữa tập huấn luyện và tập kiểm tra.
- train = data[0:data_end][prop]: Dòng này lấy 80% đầu tiên của dữ liệu theo chỉ số data_end và chỉ chọn thuộc tính đã chọn (prop). Đây là tập dữ liệu huấn luyện, chứa giá trị thuộc tính đã chọn của cổ phiếu trong khoảng thời gian huấn luyện.
- test = data[data_end:][prop]: Dòng này lấy 20% cuối cùng của dữ liệu từ chỉ số data_end đến hết và chỉ chọn thuộc tính đã chọn (prop). Đây là tập dữ liệu kiểm tra, chứa giá trị thuộc tính đã chọn của cổ phiếu trong khoảng thời gian kiểm tra.
- date_test = data[data_end:]['Date']: Dòng này lấy cột 'Date' tương ứng với tập dữ liệu kiểm tra. Điều này cho phép lưu trữ ngày tương ứng với giá trị kiểm tra của cổ phiếu.

Tổng quát, phần code này chia dữ liệu thành tập huấn luyện và tập kiểm tra, sử dụng tỷ lệ 80:20. Tập huấn luyện chứa 80% dữ liệu đầu tiên, trong khi tập kiểm tra chứa 20% dữ liệu cuối cùng.

Chuyển đổi dữ liệu thành dạng phù hợp :

```
train = train.values.reshape(-1)
test = test.values.reshape(-1)
date_test = date_test.values.reshape(-1)
# chuyển ma trận sao cho phù hợp với đầu vào của keras
```

Hình 19: Đoạn code chuyển đổi dữ liệu

Trước khi đưa dữ liệu vào mô hình LSTM của Keras, ta cần chuyển đổi dữ liệu thành dạng phù hợp. Phần code trên thực hiện các bước chuyển đổi dữ liệu như sau:

- train = train.values.reshape(-1): Dòng này chuyển đổi tập dữ liệu huấn luyện (train) từ dạng DataFrame thành một mảng 1D bằng cách sử dụng phương thức values. Sau đó, hàm reshape(-1) được sử dụng để chuyển đổi mảng thành một vectơ 1D, phù hợp với đầu vào của mô hình LSTM.

- test = test.values.reshape(-1): Tương tự như bước trên, dòng này chuyển đổi tập dữ liệu kiểm tra (test) từ dạng DataFrame thành một mảng 1D, sử dụng values và reshape(-1).

- date_test = date_test.values.reshape(-1): Tương tự như bước trên, dòng này chuyển đổi tập ngày (date_test) từ dạng DataFrame thành một mảng 1D, sử dụng values và reshape(-1).

Việc chuyển đổi này cần thiết để đảm bảo rằng dữ liệu đầu vào của mô hình LSTM có định dạng phù hợp, trong trường hợp này là một mảng 1D.

Chia dữ liệu thành các đầu vào và đầu ra phù hợp cho mô hình LSTM:

```
def get_data(train,test,time_step,num_predict,date):
  x_train= list()
  y_train = list()
  x_test = list()
  y_test = list()
  date_test= list()
# khởi tọa các list trống để lưu data test, train
# time_step = 30 số lượng ngày đưa vào, num_predict = 1 số lượng đầu ra
  for i in range(0,len(train) - time_step - num_predict):
    x_train.append(train[i:i+time_step])
    y_train.append(train[i+time_step:i+time_step+num_predict])
# y(dự đoán của 30 đầu vào) = f(x(giá close 30 ngày)w(giá trị cần cải thiện trong model) + bias(giá trị cần cải thiện trong model))
# đưa vào giá của 30 ngày tương ứng với 1 state và đầu ra y mũ cho 1 node tương tự n data còn lại
# đưa giá trị vào mảng x_train(giá đóng   của 80% tập data ngày trước để đưa vào cho máy học), y_train(giá đóng  của ngày thứ 31)
  for i in range(0, len(test) - time_step - num_predict):
    x_test.append(test[i:i+time_step])
    y_test.append(test[i+time_step:i+time_step+num_predict])
    date_test.append(date[i+time_step:i+time_step+num_predict])
# Xử lý đưa data vào list tương tự như tập train
  return np.asarray(x_train), np.asarray(y_train), np.asarray(x_test), np.asarray(y_test), np.asarray(date_test)
  # chuyển dữ liệu thành dạng mảng
```

Hình 20: Đoạn code chia dữ liệu

Phần mã trên định nghĩa một hàm có tên get_data để chia dữ liệu thành các đầu vào và đầu ra phù hợp cho mô hình LSTM. Dưới đây là giải thích từng bước trong hàm :

Khởi tạo các danh sách trống: x, y_train, x_test, y_test, date_test là các danh sách rỗng để lưu trữ dữ liệu huấn luyện và kiểm tra.

Vòng lặp for đầu tiên (cho tập huấn luyện) .Với mỗi giá trị i từ 0 đến len(train) - time_step - num_predict, thực hiện các bước sau:

- x_train.append(train[i:i+time_step]): Thêm một phần tử vào danh sách x_train. Phần tử này là một mảng con của train từ chỉ số i đến i + time_step, đại diện cho một chuỗi dữ liệu đầu vào có độ dài time_step.

- y_train.append(train[i+time_step:i+time_step+num_predict]): Thêm một phần tử vào danh sách y_train. Phần tử này là một mảng con của train từ chỉ số i + time_step đến i + time_step + num_predict, đại diện cho một chuỗi dữ liệu đầu ra có độ dài num_predict.

Vòng lặp for thứ hai (cho tập kiểm tra): Tương tự như vòng lặp trước, với mỗi giá trị i từ 0 đến len(test) - time_step - num_predict, thực hiện các bước sau:

- x_test.append(test[i:i+time_step]): Thêm một phần tử vào danh sách x_test. Phần tử này là một mảng con của test từ chỉ số i đến i + time_step, đại diện cho một chuỗi dữ liệu đầu vào có độ dài time_step.
- y_test.append(test[i+time_step:i+time_step+num_predict]): Thêm một phần tử vào danh sách y_test. Phần tử này là một mảng con của test từ chỉ số i + time_step đến i + time_step + num_predict, đại diện cho một chuỗi dữ liệu đầu ra có độ dài num_predict.
- date_test.append(date[i+time_step:i+time_step+num_predict]): Thêm một phần tử vào danh sách date_test. Phần tử này là một mảng con của date từ chỉ số i + time_step đến i + time_step + num_predict, đại diện cho các ngày tương ứng với dữ liệu kiểm tra.

Cuối cùng, hàm trả về các mảng đã được chuyển đổi thành numpy arrays: x_train, y_train, x_test, y_test, date_test.

Hàm này giúp chia dữ liệu thành các mẫu đầu vào và đầu ra có đúng kích thước và định dạng cho mô hình LSTM.

Chuẩn hóa dữ liệu trước khi đưa vào mô hình LSTM giúp đảm bảo hiệu suất tốt :

```
# gọi hàm dựa data vào mảng train test time_step num_predict, date_test
x_train, y_train, x_test, y_test, date_test = get_data(train,test,30,1, date_test)

# chuyển về dạng ma trận đưa vào minmaxscaler()
x_train = x_train.reshape(-1,30)
x_test = x_test.reshape(-1,30)

# dua ve 0->1 cho tap train
scaler = MinMaxScaler()

# gọi hàm scaler để nén hoặc giải nén data về khoảng (0,1) để máy hiểu góp phần tăng tốc độ máy học
# fit_transform nén data lại cho model cho 4 ma trận x, y_train x,y test
x_train = scaler.fit_transform(x_train)
y_train = scaler.fit_transform(y_train)

x_test = scaler.fit_transform(x_test)
y_test = scaler.fit_transform(y_test)
```

Hình 21: Đoạn code chuẩn hóa dữ liệu

Phần mã trên thực hiện các bước chuẩn hóa dữ liệu trước khi đưa vào mô hình LSTM. Dưới đây là giải thích từng bước: Gọi hàm get_data để chia dữ liệu thành các mẫu đầu vào và đầu ra phù hợp:

- train và test là các tập dữ liệu đã được chia thành dữ liệu huấn luyện và kiểm tra.
- 30 là tham số time_step, số lượng ngày được đưa vào để dự đoán ngày tiếp theo.
- 1 là tham số num_predict, số lượng giá trị dự đoán.
- date_test là danh sách các ngày tương ứng với dữ liệu kiểm tra.

Chuyển đổi dữ liệu về dạng ma trận:

- x_train = x_train.reshape(-1,30): Chuyển đổi mảng x_train thành một ma trận có kích thước (-1, 30), trong đó -1 đại diện cho số mẫu dữ liệu và 30 đại diện cho time_step.
- x_train = x_train.reshape(-1,30): Chuyển đổi mảng x_train thành một ma trận có kích thước (-1, 30), trong đó -1 đại diện cho số mẫu dữ liệu và 30 đại diện cho time_step.

Chuẩn hóa dữ liệu về khoảng (0,1) sử dụng MinMaxScaler:

- scaler = MinMaxScaler(): Khởi tạo một đối tượng MinMaxScaler.
- x_train = scaler.fi_transform(x_train): Áp dụng phương thức fit_transform của scaler để chuẩn hóa mảng x_train về khoảng (0,1).
- y_train = scaler.fit_transform(y_train): Tương tự như trên, chuẩn hóa mảng y_train.
- x_test = scaler.fit_transform(x_test): Chuẩn hóa mảng x_test.
- y_test = scaler.fit_transform(y_test): Chuẩn hóa mảng y_test.

Các bước này giúp đưa dữ liệu về định dạng và khoảng giá trị phù hợp để đảm bảo hiệu suất tốt cho mô hình LSTM.

Chuyển đổi dữ liệu thành ma trận 3 chiều để phù hợp với đầu vào của mô hình LSTM :

```
# chuyển về dạng ma trận đưa vào keras() thêm một chiều thứ 3 để có bias => để thành ma trận 3D cho phù hợp với bài toán

# Reshape lai cho x_train
x_train = x_train.reshape(-1,30,1)
y_train = y_train.reshape(-1,1)

#reshape lai cho test
x_test = x_test.reshape(-1,30,1)
y_test = y_test.reshape(-1,1)
date_test = date_test.reshape(-1,1)
```

Hình 22: Đoạn code chuyển đổi dữ liệu thành ma trận 3 chiều

Phần mã trên thực hiện việc chuyển đổi dữ liệu thành ma trận 3D để phù hợp với đầu vào của mô hình LSTM. Dưới đây là giải thích từng bước:

Reshape lại x_train và y_train:

- x_train = x_train.reshape(-1, 30, 1): Chuyển đổi mảng x_train thành một ma trận có kích thước (-1, 30, 1), trong đó -1 đại diện cho số mẫu dữ liệu, 30 đại diện cho time_step, và 1 đại diện cho chiều thứ ba, cần thiết cho mô hình LSTM.
- y_train = y_train.reshape(-1, 1): Chuyển đổi mảng y_train thành một ma trận có kích thước (-1, 1), với -1 đại diện cho số mẫu dữ liệu và 1 đại diện cho chiều thứ hai.

Reshape lại x_test, y_test, và date_test:

- Tương tự như trên, chuyển đổi các mảng x_test, y_test, và date_test thành các ma trận có kích thước phù hợp để đưa vào mô hình LSTM.

Việc reshape thành ma trận 3D là cần thiết để đảm bảo rằng dữ liệu được truyền vào mô hình LSTM có kích thước đúng và phù hợp với cấu trúc của mô hình.

Xác định cấu trúc của mô hình LSTM và cài đặt các tham số quan trọng như số lớp LSTM, tỷ lệ dropout và hàm tối ưu để huấn luyện mô hình :

```
# chuyển về dạng ma trận đưa vào keras() thêm một chiều thứ 3 để có bias => để thành ma trận 3D cho phù hợp với bài toán

# Reshape lai cho x_train
x_train = x_train.reshape(-1,30,1)
y_train = y_train.reshape(-1,1)

#reshape lai cho test
x_test = x_test.reshape(-1,30,1)
y_test = y_test.reshape(-1,1)
date_test = date_test.reshape(-1,1)
```

Hình 23: Đoạn code Xác định cấu trúc của mô hình LSTM và cài đặt các tham số quan trọng

Phần mã trên xác định và xây dựng mô hình nơ ron LSTM bằng cách sử dụng thư viện Keras. Dưới đây là giải thích từng bước:

Khởi tạo các tham số cho mô hình:

- n_input = 30: Số lượng ngày được đưa vào mô hình để dự đoán giá trị tiếp theo.
- n_features = 1: Số lượng đặc trưng (cột) của dữ liệu đầu vào. Trong trường hợp này, chỉ có một cột là giá mở.

Khởi tạo mô hình:

- model = Sequential(): Khởi tạo một mô hình tuần tự.

Thêm các lớp LSTM vào mô hình:

- model.add(LSTM(units=50, input_shape=(n_input, n_features) ,return_sequences=True)): Thêm một lớp LSTM với 50 đơn vị (units), input_shape được đặt là (n_input, n_features) để phù hợp với kích thước dữ liệu đầu vào. Tham số return_sequences=True được đặt để trả về chuỗi các hidden state từ lớp LSTM này.
- model.add(Dropout(0.3)): Thêm một lớp Dropout với tỷ lệ dropout là 0.3. Lớp Dropout giúp giảm overfitting bằng cách loại bỏ một phần các nút trong mạng

Thêm các lớp LSTM và Dropout tiếp theo:

- model.add(LSTM(units=50, return_sequences=True)): Thêm một lớp LSTM tiếp theo với 50 đơn vị và return_sequences=True.

- model.add(Dropout(0.3)): Thêm một lớp Dropout với tỷ lệ dropout là 0.3.

Thêm lớp LSTM và Dropout cuối cùng:

- model.add(LSTM(units=50)): Thêm một lớp LSTM cuối cùng với 50 đơn vị và return_sequences=False, tức là không trả về chuỗi các hidden state.
- model.add(Dropout(0.3)): Thêm một lớp Dropout với tỷ lệ dropout là 0.3.

Thêm lớp Dense cuối cùng:

- model.add(Dense(1)): Thêm một lớp Dense với một đơn vị. Đây là lớp đầu ra của mô hình, có tác dụng dự đoán giá trị tiếp theo.

Compile mô hình:

- model.compile(optimizer='adam', loss='mse'): Đặt hàm tối ưu là 'adam' và hàm lỗi là 'mse' (Mean Squared Error). Adam là một thuật toán tối ưu hóa phổ biến và MSE là hàm lỗi được sử dụng trong bài toán dự đoán giá trị liên tục.

Các bước trên xác định cấu trúc của mô hình LSTM và cài đặt các tham số quan trọng như số lớp LSTM, tỷ lệ dropout và hàm tối ưu để huấn luyện mô hình.

Thực hiện huấn luyện mô hình LSTM trên dữ liệu và lưu trữ mô hình đã được huấn luyện để sử dụng cho việc dự đoán sau này :

```
%cd /content/drive/MyDrive/co phieu
# fit đưa tất cả vào model
model.fit(x_train, y_train, epochs=200, validation_split=0.2, verbose=1, batch_size=30)
model.save(f'{Macophieu}_{prop}.h5')
```

Hình 24: Đoạn code Thực hiện huấn luyện và lưu trữ mô hình

Công việc của đoạn code này là :

%cd /content/drive/MyDrive/co phieu: Di chuyển đến thư mục "/content/drive /MyDrive/co phieu". Đây là thư mục nơi lưu trữ tệp tin với tên và đường dẫn đã được chỉ định.

model.fit(x_train, y_train, epochs = 200, validation_split = 0.2, verbose=1, batch_size=30): Bắt đầu quá trình huấn luyện mô hình. Hàm fit() được sử dụng để huấn luyện mô hình trên dữ liệu đào tạo. Các đối số được truyền vào bao gồm:

- x_train: Dữ liệu đầu vào của tập huấn luyện.
- y_train: Đầu ra mong muốn của tập huấn luyện.
- epochs=200: Số lượng epochs, tức là số lần mô hình sẽ được huấn luyện trên toàn bộ tập huấn luyện.
- validation_split=0.2: Tỷ lệ dữ liệu được sử dụng cho việc đánh giá hiệu suất của mô hình trên tập validation.

- verbose=1: Kiểm soát đầu ra khi huấn luyện. Giá trị 1 cho phép hiển thị tiến trình huấn luyện.
- batch_size=30: Kích thước của các batch, tức là số lượng mẫu được sử dụng để tính gradient trong mỗi lần cập nhật tham số.

model.save(f'{Macophieu}_{prop}.h5'): Lưu mô hình đã được huấn luyện thành một tệp tin có định dạng h5. Tên của tệp tin được đặt dựa trên tên cổ phiếu và thuộc tính đã chọn.

Những công việc trên thực hiện việc huấn luyện mô hình LSTM trên dữ liệu và lưu trữ mô hình đã được huấn luyện để sử dụng cho việc dự đoán sau này.

Thực hiện tải mô hình đã được huấn luyện, dự đoán giá trị đầu ra và giải nén dữ liệu để so sánh với giá trị thực tế trong quá trình kiểm tra mô hình :

```
%cd /content/drive/MyDrive/co phieu
# Test model vừa train
model = keras.models.load_model(f'{Macophieu}_{prop}.h5')
# load model
test_output = model.predict(x_test)
# lấy tập x_test cho vào model ra test_output là data dự đoán

# giải nén dữ liệu từ (0,1) ra giá thực tế
test_1 = scaler.inverse_transform(test_output)
test_2 =scaler.inverse_transform(y_test)
```

Hình 25: Đoạn code Thực hiện tải mô hình đã được huấn luyện

Công việc của đoạn code này là :

%cd /content/drive/MyDrive/co phieu: Di chuyển đến thư mục "/content/drive /MyDrive/co phieu". Đây là thư mục chứa tệp tin mô hình đã được lưu trữ.

model = keras.models.load_model(f'{Macophieu}_{prop}.h5'): Tải lại mô hình đã được huấn luyện từ tệp tin h5 đã được lưu trữ trước đó. Mô hình sẽ được gán cho biến model để sử dụng trong việc dự đoán.

test_output = model.predict(x_test): Dùng mô hình để dự đoán giá trị đầu ra cho tập dữ liệu kiểm tra x_test. Kết quả được lưu trong biến test_output, chứa các dự đoán của mô hình.

test_1 = scaler.inverse_transform(test_output): Giải nén dữ liệu đã được dự đoán từ khoảng (0,1) về giá trị thực tế ban đầu bằng cách sử dụng phép nghịch đảo của scaler. Kết quả được lưu trong biến test_1.

test_2 = scaler.inverse_transform(y_test): Giải nén dữ liệu thực tế từ khoảng (0,1) về giá trị thực tế ban đầu bằng cách sử dụng phép nghịch đảo của scaler. Kết quả được lưu trong biến test_2.

Các bước trên thực hiện việc tải mô hình đã được huấn luyện, dự đoán giá trị đầu ra và giải nén dữ liệu để so sánh với giá trị thực tế trong quá trình kiểm tra mô hình.

Vẽ biểu đồ để so sánh dữ liệu dự đoán (test_1) và dữ liệu thực tế (test_2) trên trục thời gian (date_test[30:]) và hiển thị chú thích cho biểu đồ để phân biệt dữ liệu dự đoán và dữ liệu thực tế :

```
plt.figure(figsize=(16,6))
# đưa lên biểu đồ
plt.plot(date_test[30:], test_1[30:], color='r')
plt.plot(date_test[30:], test_2[30:], color='b')
plt.title(f"STOCK {companyName} COMPANY")
plt.xlabel("Date")
plt.ylabel("Price")
plt.legend(('prediction', 'reality'),loc='upper right')
plt.show()
```

Hình 26: Đoạn code vẽ biểu đồ so sánh dữ liệu dự đoán

Công việc của đoạn code này là :

plt.figure(figsize=(16,6)): Tạo một figure mới với kích thước 16x6 inches, để chuẩn bị vẽ biểu đồ.

plt.plot(date_test[30:], test_1[30:], color='r'): Vẽ đường biểu đồ cho dữ liệu dự đoán test_1 trên trục x là date_test[30:]. Màu đường biểu đồ được đặt là màu đỏ (color='r').

plt.plot(date_test[30:], test_2[30:], color='b'): Vẽ đường biểu đồ cho dữ liệu thực tế test_2 trên trục x là date_test[30:]. Màu đường biểu đồ được đặt là màu xanh (color='b').

plt.title(f"STOCK companyName COMPANY"): Đặt tiêu đề cho biểu đồ, trong đó companyName sẽ được thay thế bằng tên công ty tương ứng.

plt.xlabel("Date"): Đặt nhãn cho trục x là "Date".

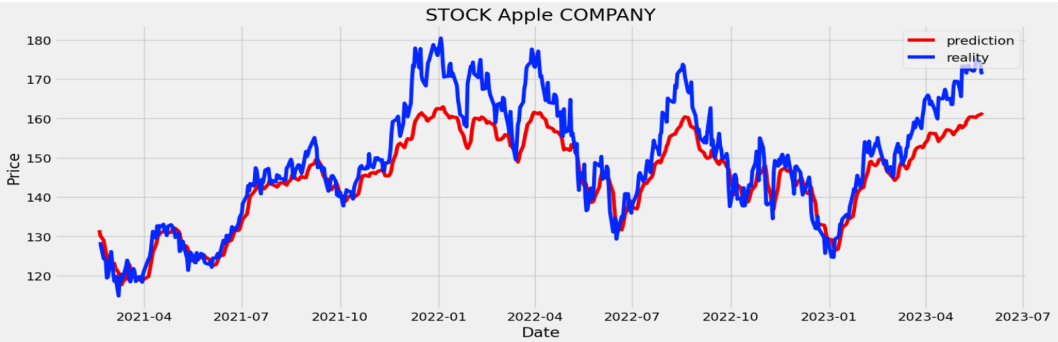
plt.ylabel("Price"): Đặt nhãn cho trục y là "Price".

plt.legend(('prediction', 'reality'),loc='upper right'): Hiển thị chú thích (legend) cho biểu đồ, với hai mục là "prediction" và "reality" tương ứng với dự đoán và thực tế. Chú thích được đặt ở vị trí góc trên bên phải (loc='upper right').

plt.show(): Hiển thị biểu đồ đã vẽ.

Các bước trên thực hiện việc vẽ biểu đồ để so sánh dữ liệu dự đoán (test_1) và dữ liệu thực tế (test_2) trên trục thời gian (date_test[30:]), và hiển thị chú thích cho biểu đồ để phân biệt dữ liệu dự đoán và dữ liệu thực tế.

Kết Quả:



Hình 27: Biểu đồ để so sánh dữ liệu dự đoán

Đánh giá hiệu suất của mô hình trên dữ liệu kiểm tra và hiển thị kết quả đánh giá cũng như giá dự đoán của ngày hôm qua :

Đánh giá mô hình sử dụng Accuracy: Accuracy là độ đo của bài toán phân loại mà đơn giản nhất, tính toán bằng cách lấy số dự đoán đúng chia cho toàn bộ các dự đoán:

$$\text{Acc} = \frac{\text{Số dự đoán đúng}}{\text{Tổng số dự đoán}} \tag{1}$$

Với bài đánh giá này, ta chỉ cần quan tâm đến bao nhiêu phần trăm lượng dữ liệu được phân loại đúng mà không cần chỉ ra được cụ thể mỗi loại được phân loại như thế nào, lớp nào được phân loại đúng nhiều nhất hay dữ liệu của lớp nào thường bị phân loại nhầm nhất vào các lớp khác. Vì vậy mà ta sẽ chọn chỉ số đánh giá mô hình Accuracy cho bài toán đánh giá mô hình này.

Hàm mất mát sử dụng là MSE: tính giá trị trung bình của bình phương sai số giữa các dự đoán và thực tế.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y}_i)^2$$

trong đó: n là số lượng các mẫu trong tập dữ liệu, y là giá trị thực tế, $\hat{y}_i$ là giá trị dự đoán.

MSE lấy bình phương sai số để tăng trọng số của các sai số lớn hơn so với MAE. Giá trị MSE càng nhỏ, tức là sai số trung bình bình phương càng thấp và mô hình dự đoán càng chính xác. MSE thường được sử dụng nhiều hơn MAE trong các bài toán dự đoán, đặc biệt là khi có sự chênh lệch lớn giữa các sai số.

Như vậy, giá trị hàm loss thể hiện mức độ sai lệch giữa dự đoán của mô hình và giá trị thực tế trên tập dữ liệu. Vì chỉ số Accuracy là tỷ lệ các dự đoán đúng trên tổng số mẫu, nên giá trị của Acc được tính bằng cách lấy 1 trừ đi giá trị hàm loss.

```
test_output = model.predict(x_test)
print("Evaluate on test data")
results = model.evaluate(x_test, y_test, batch_size=128)
print('\n')
print("test loss, test acc:", results,1-results)
print('Giá dự đoán của ngày hôm qua: '+str(test_1[-1:]))
```

Hình 28: Đoạn code Đánh giá hiệu suất của mô hình

Đoạn code trên được sử dụng để đánh giá hiệu suất của mô hình trên dữ liệu kiểm tra và hiển thị kết quả đánh giá cũng như giá dự đoán của ngày hôm qua :

test_output = model.predict(x_test): Sử dụng mô hình đã được huấn luyện để dự đoán giá trị đầu ra (test_output) dựa trên dữ liệu đầu vào kiểm tra (x_test).

print("Evaluate on test data"): In ra thông báo đánh giá trên dữ liệu kiểm tra.

results = model.evaluate(x_test, y_test, batch_size=128): Đánh giá hiệu suất của mô hình trên dữ liệu kiểm tra (x_test, y_test) và lưu kết quả vào biến results. Kết quả đánh giá thường bao gồm giá trị hàm mất mát (loss) và các chỉ số đánh giá khác.

print("test loss, test acc:", results,1-results): In ra kết quả đánh giá, bao gồm hàm mất mát (results) và độ chính xác (1-results).

print('Giá dự doán của ngày hôm qua: '+str(test_1[-1:])): In ra giá trị dự đoán của ngày hôm qua, được lấy từ biến test_1. Đây là giá trị cuối cùng của test_1, được chuyển đổi thành chuỗi để hiển thị thông qua hàm str().

Kết quả :

```
19/19 [==============================] - 0s 9ms/step
Evaluate on test data
5/5 [==============================] - 1s 17ms/step - loss: 0.0124


test loss, test acc: 0.012441701255738735 0.9875582987442613
Giá dự doán của ngày hôm qua: [[158.44844]]
```

Hình 29: Kết Quả chạy đánh giá hiệu suất

NHẬN XÉT :

Như kết quả tính toán được bằng hàm tính toán , ta có:

- Test loss(Đây là giá trị đo lường mức độ sai khác giữa giá trị dự đoán và giá trị thực tế trên tập dữ liệu kiểm tra) chỉ có 0.012441701255738735
- Test acc(Đây là tỷ lệ phần trăm của các dự đoán chính xác trên tập dữ liệu kiểm tra) đạt tận 0.9875582987442613
- Với chỉ số Acc là 99% tức giá trị dự đoán của mô hình rất sát với giá trị thực tế, gần như là không lệch. Từ đó ta có thể khẳng định là hiếm có trường hợp hoặc không có hiện tượng overfiting xảy ra.

Kết Luận: mô hình có thể dự đoán rất sát xu hướng của giá cổ phiếu thực tế. Độ chính xác của mô hình có thể được nâng cao hơn bằng cách đào tạo với nhiều dữ liệu hơn và tăng các lớp LSTM.

II. Thực nghiệm

Xây dựng giao diện hiển thị kết quả sau khi thiết lập và đánh giá mô hình bằng mô hình LSTM

Tạo các function như:

- Hàm xuly(data, SYMB, fd): thực hiện các bước chuẩn bị dữ liệu, tải mô hình đã được lưu trữ và sử dụng mô hình để dự đoán giá trị dựa trên dữ liệu đầu vào. Sau đó, nó áp dụng quá trình giải chuẩn để đưa kết quả về dạng gốc.

```
def xuly(data,SYMB,fd):
    input = got_data(data)
    scaler = MinMaxScaler()
    scaler.fit_transform(input)
    input.reshape(-1,30,1)
    #Lấy dữ liệu
    model = keras.models.load_model('Model\\' + SYMB+'_'+fd+'.h5')
    testk = model.predict(input)
    m = float(testk)
    mn = min(scaler.data_min_)
    e = (max(scaler.data_max_)-min(scaler.data_min_))
    return m*e+mn
```

- Hàm plot_raw_data(date1, ketqua, SYMB): tạo một biểu đồ đường Scatter và hiển thị nó trong ứng dụng Streamlit.

```
def plot_raw_data(date1,ketqua,SYMB):
        fig = go.Figure()
        fig.add_trace(go.Scatter(x=date1, y=ketqua, name="stock_open"))
        fig.layout.update(title_text='Predict Chart of '+ str(SYMB), xaxis_rangeslider_visible=True)
        st.plotly_chart(fig)
```

- Hàm got_day(): tạo giao diện người dùng cho phép người dùng chọn một ngày cần dự đoán. Ngày này được điều chỉnh để đảm bảo rằng nó là ngày sau cùng của thời điểm được chọn và giờ là 23:00. Hàm trả về ngày đã được điều chỉnh để xử dụng trong các quá trình dự đoán.

```
def got_day():
    st.sidebar.markdown("## Predict")
    train_test_forecast_c = st.sidebar.container()
    st.title('Predict stock price')

    day = train_test_forecast_c.date_input("Day need to predict")

    day = day-BDay(0) + datetime.timedelta(days=1) - datetime.timedelta(hours=1)
    return day
```

- Hàm predict(data, fd, STMB, day): thực hiện dự đoán giá cổ phiếu cho từng ngày từ today đến day. Nó lấy dữ liệu gần nhất từ cột fd của data, áp dụng xử lí dữ liệu thông qua hàm xuly và vẽ biểu đồ dự đoán. Kết quả dự đoán và thông tin tương ứng với mỗi ngày được trình ày dưới dạng DataFrame.

```
def predict(data,fd,SYMB,day):
    today = datetime.datetime.today()
    today = today - BDay(0)
    data = data[:][fd]
    data = data[-30:]
    L = list()
    for i in data:
        L.append(i)
    date1 = list()
    ketqua = list()
    mix = {}
    while today <= day:
        date1.append(today)
        x = xuly(data,SYMB,fd)
        L = L[1:] + [x]
        data = np.asarray(L)
        ketqua.append(x)
        mix[today]=x
        today = today - BDay(-1)

    plot_raw_data(date1,ketqua,SYMB)
    data_item = mix.items()
    data_list = list(data_item)
    df = pd.DataFrame(data_list,columns=['Day','Value'])
    st.write(df)
```

- Hàm fig_1(stock, fd, SYMB): vẽ biểu đồ biểu diễn sự biến đổi tỷ lệ lợi nhuận hàng ngày của cổ phiếu, kèm theo giá trị trung bình và chú thích tương ứng.

```
def fig_1(stock,fd,SYMB):
    fig_1,ax = plt.subplots()
    fig_1.set_figheight(5)
    fig_1.set_figwidth(12)
    r_t = np.log((stock[fd]/stock[fd].shift(1)))
    mean = np.mean(r_t)
    r_t[0] = mean
    ax.plot(r_t, linestyle='--', marker='o')
    ax.axhline(y=mean, label='mean return', c='red')
    ax.legend()
    st.write('\n')
    st.title('Prive movement chart '+str(fd)+' of '+SYMB)
    st.pyplot(fig_1)
    return r_t,mean
```

- Hàm fig_3(stock, fd, SYMB, r_t, mean): vẽ biểu đồ phân phối của tỷ lệ lợi nhuận hàng ngày và đánh dấu giá trị trung bình trên biểu đồ.

```
def fig_3(stock,fd,SYMB,r_t,mean):
    st.title('Overall average daily profit of '+SYMB)
    fig_3,ax = plt.subplots()
    fig_3.set_figheight(5)
    fig_3.set_figwidth(12)
    sns.distplot(r_t, bins = 20)
    plt.axvline(x=mean, label='mean return', c='red')
    plt.legend()
    plt.xlabel('return rate')
    plt.ylabel('frequency')
    st.pyplot(fig_3)
```

Cuối cùng hiển thị lên giao diện web thông qua thư viện Streamlit:

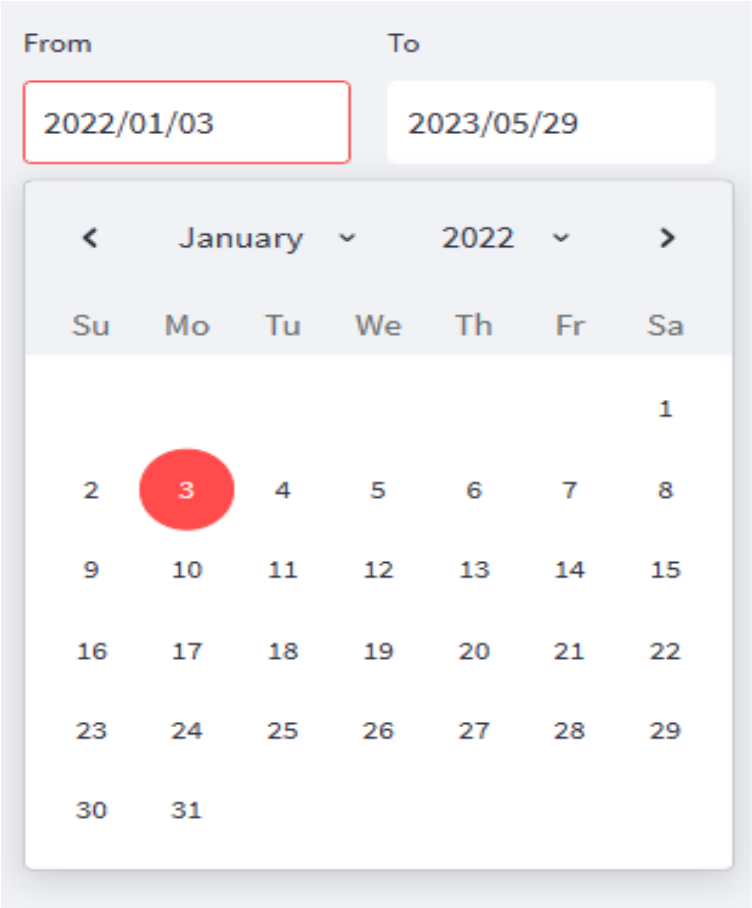
- Tạo và cấu hình giao diện

  Kết quả trên giao diện web :

Nhóm 5: dự đoán giá cổ phiếu

Hình 30: Logo web và tên nhóm

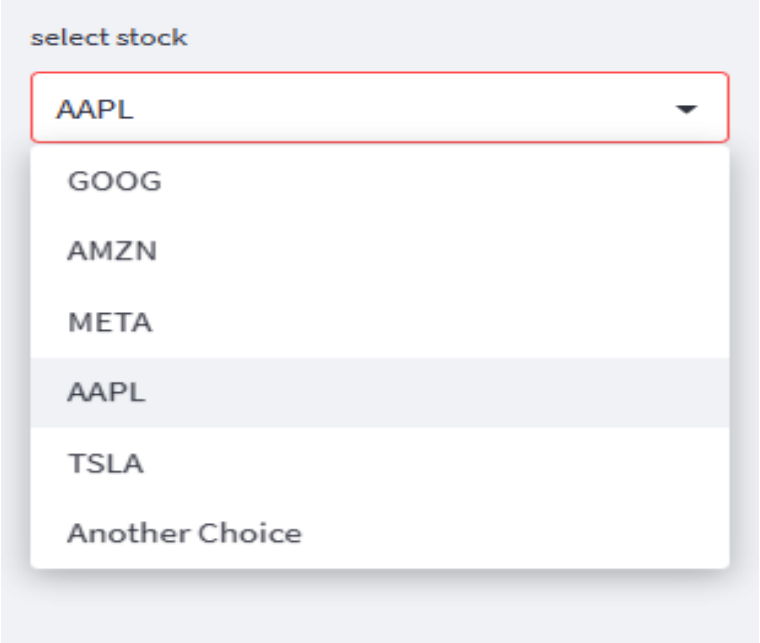
- Giao diện phần chọn ngày from – to
  Kết quả trên giao diện web :



Hình 31: Giao diện chọn khoảng thời gian lấy dữ liệu

- Chọn mã cổ phiếu cần dự đoán

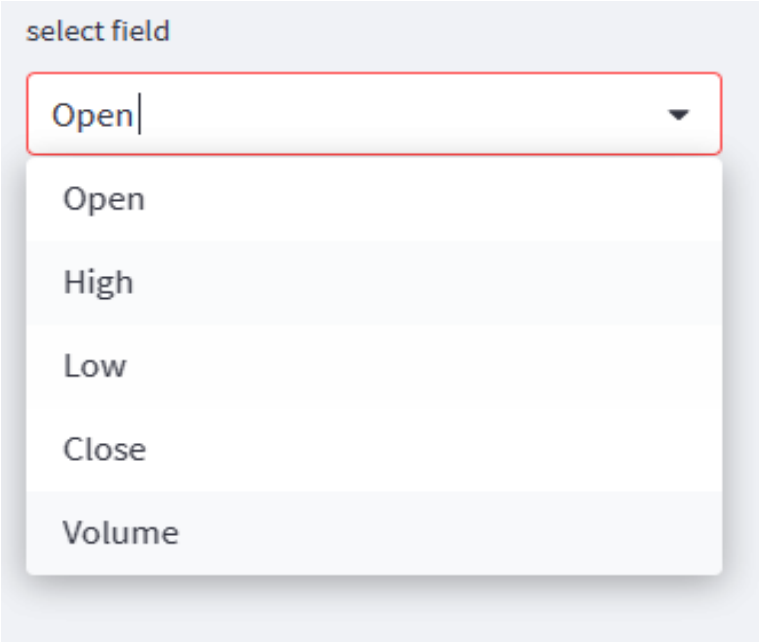
  Kết quả trên giao diện web :



Hình 32: Giao diện chọn mã cổ phiếu

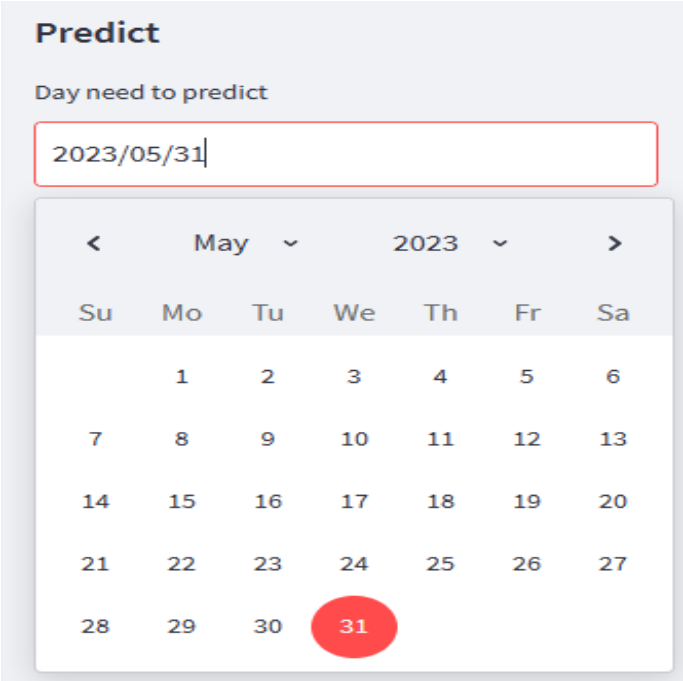
- Chọn các field như OPEN, HIGH, LOW, CLOSE, VOLUME

  Kết quả trên giao diện web :



Hình 33: Giao diện chọn field

- Hàm got_day() cho người dùng chọn ngày dự đoán
  Kết quả trên giao diện web :



Hình 34: Giao diện chọn ngày dự đoán

- Khi người dùng chọn các mã khác "Another Choice"

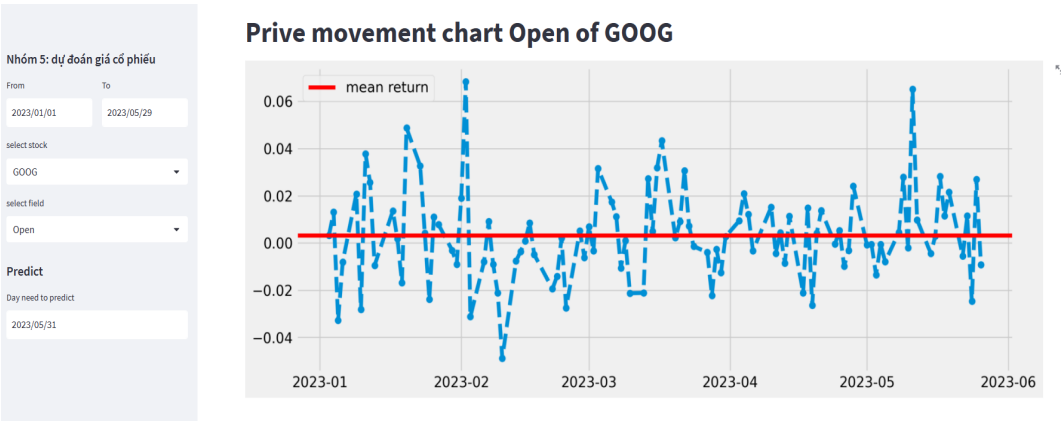
```
if SYMB != "Another Choice":
# # # # ------------------------Plot stock linecharts--------------------
    st.title('Price data of '+SYMB+' stock')
    tickerData = yf.Ticker(SYMB)
    stock = tickerData.history(period='1d', start=START, end=END)
    field = np.array([ "Open", "High", "Low","Close","Volume"])  # TODO : include all stocks
    fd = window_selection_c.selectbox("select field", field)
    xuatdothi_1(stock[fd])
    st.title('Price data of '+SYMB+' stock')
    stock = stock.drop('Stock Splits', axis=1)
    stock = stock.drop('Dividends', axis=1)
    #-------------------------------------------------------------------------------------------------
    r_t,mean = fig_1(stock,fd,SYMB)
    #-------------------------------------------------------------------------------------------------
    fig_3(stock,fd,SYMB,r_t,mean)
    # #----part-1---------------------------------Session state intializations-------------------------
    if "TEST_INTERVAL_LENGTH" not in st.session_state:
        # set the initial default value of test interval
        st.session_state.TEST_INTERVAL_LENGTH = 60
    if "TRAIN_INTERVAL_LENGTH" not in st.session_state:
        # set the initial default value of the training length widget
        st.session_state.TRAIN_INTERVAL_LENGTH = 500
    if "HORIZON" not in st.session_state:
        # set the initial default value of horizon length widget
        st.session_state.HORIZON = 60
    if 'TRAINED' not in st.session_state:
        st.session_state.TRAINED=False
    # #-----------------------------------------------------------Train_test_forecast_splits------------
    today = today - BDay(0)
    d30bf = today - BDay(100)
    data = tickerData.history(period='1d', start=d30bf, end=today)
    data.reset_index(inplace=True)
    day = got_day()
    predict(data,fd,SYMB,day)
```

Ví dụ, chọn dự đoán mã cổ phiếu SYMB = GOOD, từ ngày 01/01/2023 đến 29/05/2023, stock = OPEN, thì thu được:

- Price data of GOOG stock

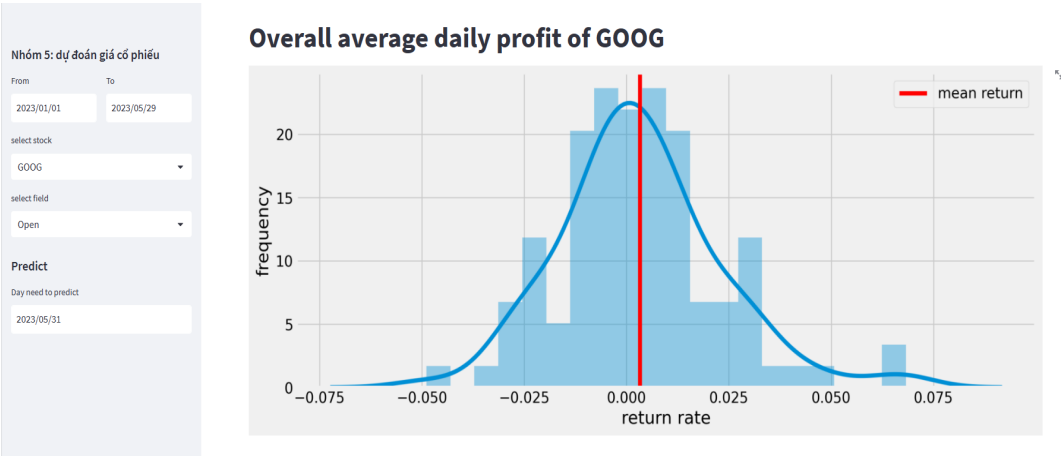
Hình 35: Dữ liệu mã cổ phiếu GOOG

- Price movement chart Open of GOOG



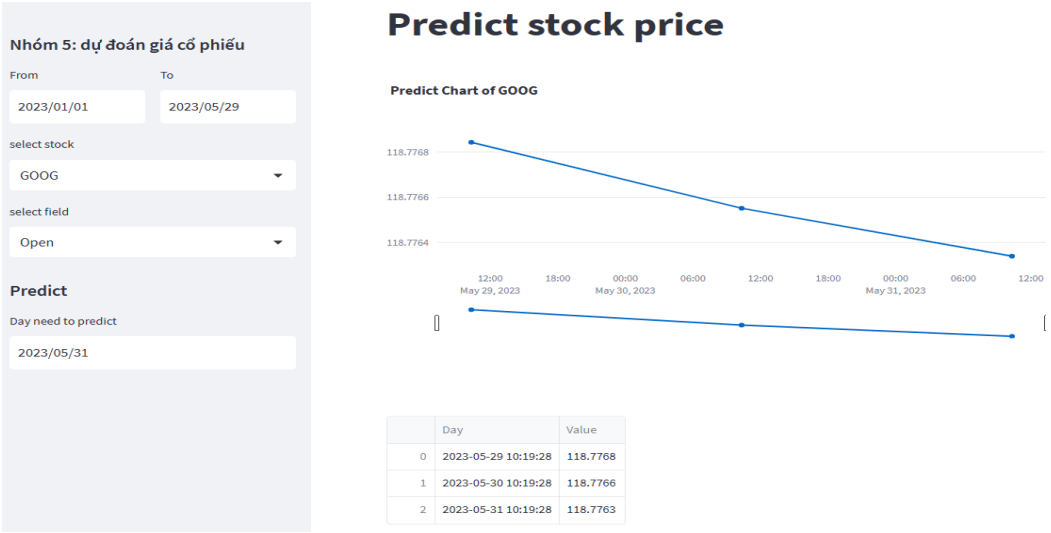
Hình 36: Biểu đồ chuyển động giá Open của GOOG

- Overall average daily profit of GOOG



Hình 37: Tổng lợi nhuận trung bình hàng ngày của GOOG

- Predict stock price



Hình 38: Dự đoán giá cổ phiếu

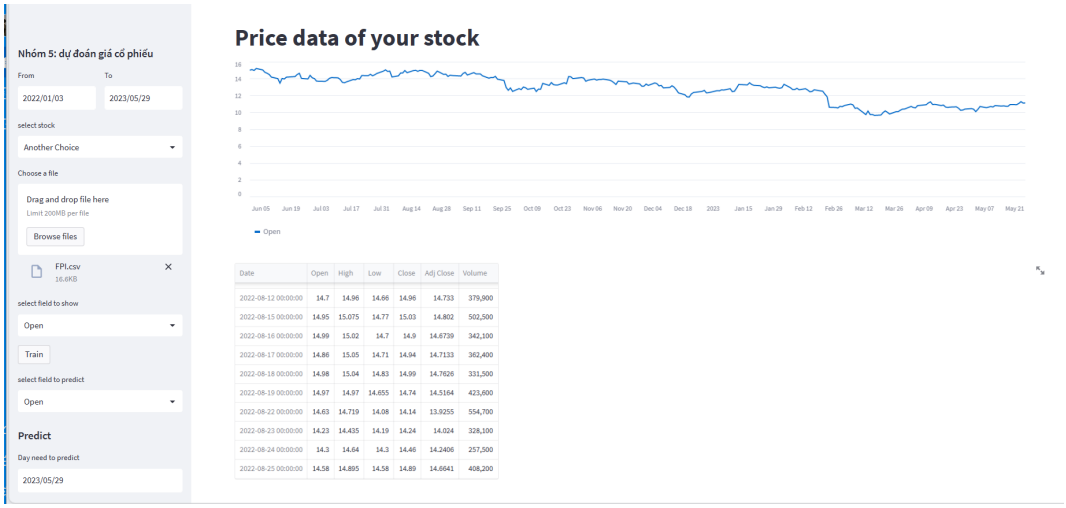
- Khi người dùng chọn "Another Choice"

```
else:
    uploaded_file = window_selection_c.file_uploader("Choose a file")
    stock = pd.DataFrame()
    if uploaded_file is not None:
        stock = pd.read_csv(uploaded_file,index_col=0,parse_dates=True,infer_datetime_format=True)
        # print(stock)

        sl = stock.columns
        fd = window_selection_c.selectbox("select field to show", sl)
        st.title('Price data of your stock')
        xuatdothi_1(stock[fd])
        st.write(stock)
        SYMB = 'Your_stock'
        r_t,mean = fig_1(stock,fd,SYMB)
        fig_3(stock,fd,SYMB,r_t,mean)
        Size = stock[fd].shape[0]
        stock.reset_index(inplace=True)
        Button = window_selection_c.button("Train")
        if Button:
            train(stock,fd)
            #for ld in sl:
            #    train(stock,ld)
        day = got_day()
        fdd = window_selection_c.selectbox("select field to predict",sl)

        predict(stock[:30],fd,'Your_stock',day)
```

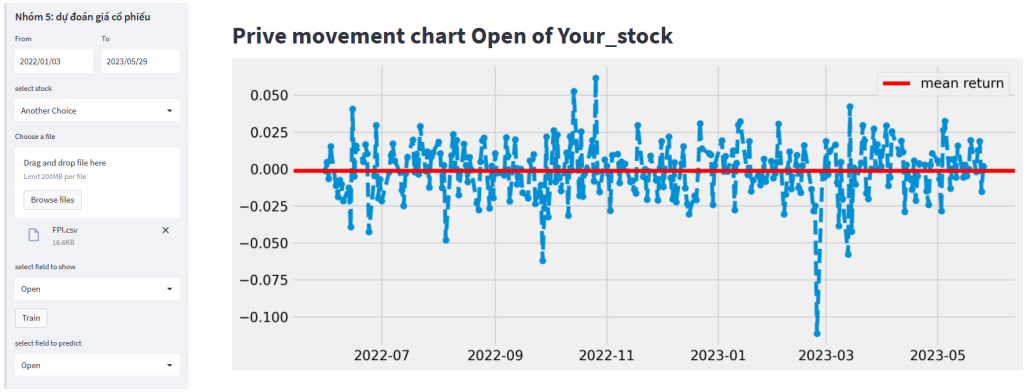
Hình 39: Code chọn mã cổ phiếu khác (người dùng tự upload file dữ liệu mã cổ phiếu)

- Cho người dùng tải file từ máy lên khi chọn "Another Choice" và nút "Train" hiện lên. Cho phép người dùng click là bắt đầu train qua hàm train() và thực hiện train bằng mô hình LSTM bên trên (Mục 3. Đánh giá mô hình).

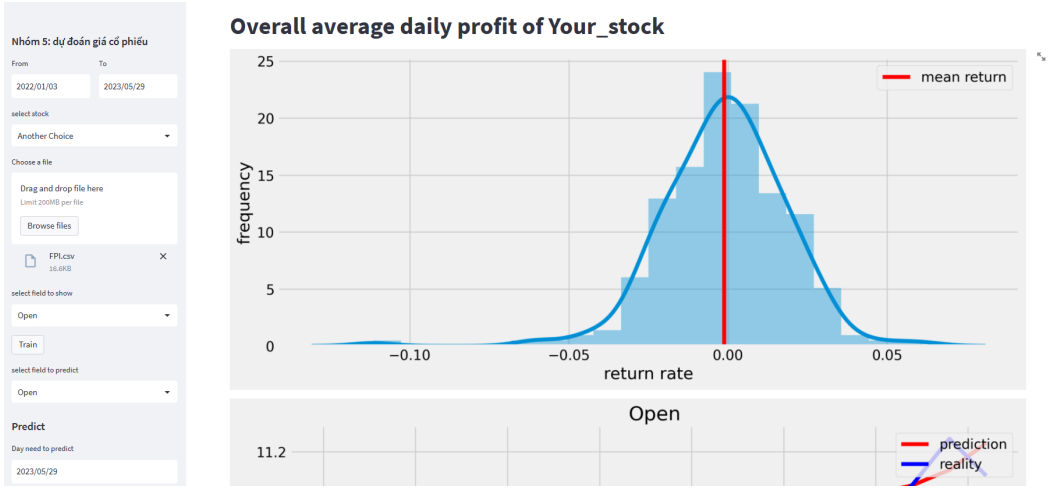
  Kết quả:



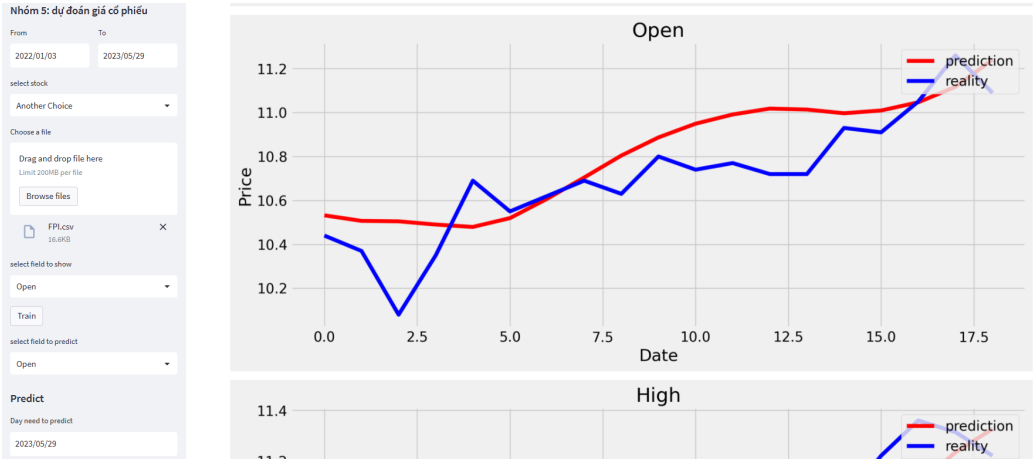
Hình 40: Dữ liệu mã cổ phiếu của bạn vừa tải lên



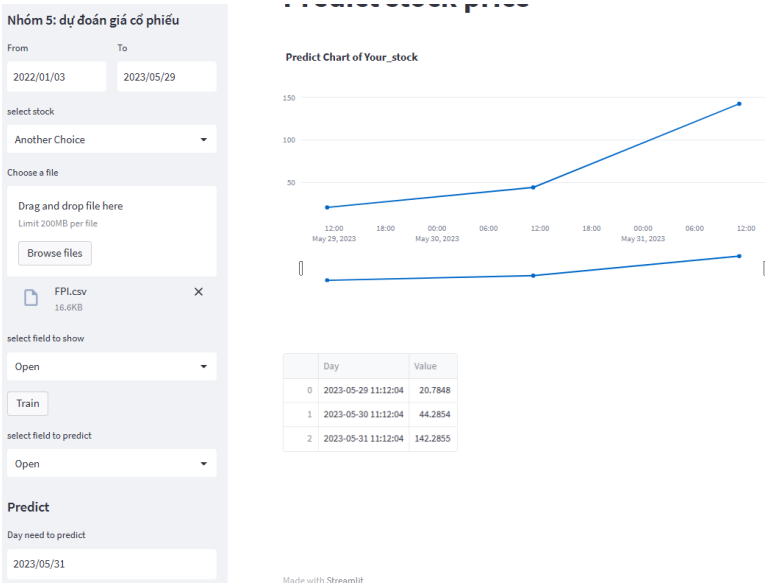
Hình 41: Biểu đồ chuyển động giá Open của mã cổ phiếu của bạn vừa tải lên



Hình 42: Tổng lợi nhuận trung bình hàng ngày của mã cổ phiếu bạn vừa tải lên



Hình 43: Biểu đồ training field Open của mã cổ phiếu bạn vừa tải lên



Hình 44: Dự đoán mã cổ phiếu của bạn của tải lên

III. Tổng kết

Trong bài báo cáo này, chúng em đã xây dựng một trang web dự đoán giá cổ phiếu bằng cách sử dụng thuật toán LSTM (Long Short-Term Memory). Chúng em đã trình bày quá trình xử lý dữ liệu, huấn luyện mô hình LSTM và triển khai trang web để cho phép người dùng dự đoán giá cổ phiếu trong tương lai.

Kết quả thực nghiệm cho thấy mô hình LSTM đã đạt được độ chính xác cao trong việc dự đoán giá cổ phiếu. Qua quá trình huấn luyện, mô hình đã học được mẫu chuyển động và xu hướng của dữ liệu giá cổ phiếu trong quá khứ. Điều này cho phép nó dự đoán giá cổ phiếu một cách chính xác hơn.

Trang web mà chúng em xây dựng cung cấp giao diện thân thiện và dễ sử dụng cho người dùng. Người dùng có thể nhập vào các thông tin liên quan đến cổ phiếu và thời gian dự đoán, sau đó nhận được kết quả dự đoán giá cổ phiếu dựa trên mô hình LSTM. Điều này cung cấp cho người dùng một công cụ hữu ích để đưa ra quyết định đầu tư dựa trên dự báo giá cổ phiếu.

Tuy nhiên, cần lưu ý rằng dự đoán giá cổ phiếu luôn tiềm ẩn những rủi ro và không đảm bảo chính xác tuyệt đối. Ngoài ra, mô hình LSTM chỉ dựa trên dữ liệu lịch sử và không đánh giá được những yếu tố bên ngoài có thể ảnh hưởng đến giá cổ phiếu như tin tức hoặc sự kiện xã hội. Do đó, việc sử dụng dự đoán giá cổ phiếu từ mô hình LSTM nên được kết hợp với các phân tích và thông tin thêm để đưa ra quyết định đầu tư có trách nhiệm.

Như vậy, công nghệ LSTM đã cho thấy khả năng dự đoán tốt trong việc xác định xu hướng giá cổ phiếu. Việc sử dụng trang web dự đoán giá cổ phiếu bằng thuật toán LSTM có thể hỗ trợ người dùng trong việc đưa ra quyết định đầu tư thông minh, tuy nhiên cần được sử dụng cùng với phân tích và thông tin bổ sung khác.

IV. Tài liệu tham khảo

1. Sách Deep Learning cơ bản - Nguyễn Thanh Tuấn - Phiên bản 2, tháng 8 năm 2020

2. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow - Aurélien Géron - tháng 9 năm 2019