

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN



HCMUTE

**BÁO CÁO ĐỒ ÁN CUỐI KỲ
ĐỀ TÀI: PHÂN TÍCH DỮ LIỆU CHỨNG KHOÁN
CÓ SỬ DỤNG LTSM DỰ ĐOÁN**

Môn học: Phân Tích Dữ Liệu

GVHD: ThS. Nguyễn Văn Thành

Nhóm sinh viên thực hiện:Nhóm 1

Đinh Đức Nguyên Vũ	20128171
Nguyễn Tân Sương	21133078
Bùi Thị Huỳnh Hân	21133039
Phan Khải Huyền	21133041

DANH SÁCH THÀNH VIÊN THAM GIA

THỰC HIỆN ĐỀ TÀI VÀ VIẾT BÁO CÁO

Môn: Phân Tích Dữ Liệu - HỌC KÌ II – NĂM HỌC 2024 – 2025

STT	HỌ VÀ TÊN	MSSV	TỶ LỆ ĐÓNG GÓP
1	Đinh Đức Nguyên Vũ	20128171	100%
2	Nguyễn Tân Sương	21133078	100%
3	Bùi Thị Huỳnh Hân	21133029	100%
4	Phan Khải Huyền	21133041	100%

Nhận xét của giảng viên:

Ngày ... tháng 05 năm 2024

Giảng viên chấm điểm

Ths. Nguyễn Văn Thành

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành đến **trường Đại học Sư phạm Kỹ thuật TPHCM** đã bồi sung môn học Phân Tích Dữ Liệu vào chương trình giảng dạy. Đặc biệt, chúng em xin bày tỏ lòng biết ơn sâu sắc đến giảng viên bộ môn – Thạc sĩ Nguyễn Văn Thành, đã dạy dỗ và truyền đạt những kiến thức quý báu cho chúng em trong suốt thời gian học tập vừa qua. Trong quá trình tham gia lớp học Phân Tích Dữ Liệu của Thầy, chúng em đã thu thập được nhiều kiến thức hữu ích, và đã phát triển tinh thần học tập hiệu quả và nghiêm túc. Những kiến thức này chắc chắn sẽ đóng vai trò quan trọng và trở thành hành trang để chúng em vững bước trong tương lai. Môn học Phân Tích Dữ Liệu thực sự thú vị, mang lại lợi ích to lớn và có tính ứng dụng cao. Nó đảm bảo cung cấp đầy đủ kiến thức, liên kết với nhu cầu thực tế của sinh viên. Mặc dù chúng em đã cố gắng hết sức, nhưng chắc chắn rằng bài đồ án này khó tránh khỏi những thiếu sót và một số vấn đề chưa chính xác. Kính mong Thầy xem xét và đóng góp ý kiến để bài đồ án của chúng em được hoàn thiện hơn.

Nhóm chúng em xin chân thành biết ơn!

Mục Lục

CHƯƠNG 1: GIỚI THIỆU VỀ ĐỀ TÀI	6
1.1.Lý do chọn đề tài	6
1.2.Tổng quan về tập dữ liệu	6
1.2.1.Nguồn dữ liệu	6
1.2.2.Mô tả chi tiết tập dữ liệu	6
1.3.Các kiến thức liên quan	6
1.3.1. Giới thiệu về mạng nơ ron thần kinh	6
1.3.2. Giới thiệu về LSTM	7
1.3.3. Ứng dụng LSTM trong dự đoán	7
CHƯƠNG 2: PHÂN TÍCH DỮ LIỆU	8
2.1.Preprocessing	8
2.2.Trực quan trên python	8
2.2.1.Import các thư viện cần thiết	8
2.2.2.Đọc file CSV	8
2.2.3.Kiểm tra tệp dữ liệu	9
2.2.4. Trực quan hóa dữ liệu	10
2.3.Trực quan trên PowerBI	26
2.3.1 Tập dữ liệu facebook.csv	26
2.3.2 Tập dữ liệu amazon.csv	27
2.3.3 Tập dữ liệu Apple.csv	28
2.3.4 Tập dữ liệu Google.csv	29
CHƯƠNG 3: DỰ ĐOÁN DỮ LIỆU	30
3.1.Sử dụng LTSM và RNN dự đoán tập dữ liệu Google	30
3.1.1.Import các thư viện cần thiết	30
3.1.2.Đọc file CSV và Đổi kiểu dữ liệu ngày	30
3.1.3.Kiểm tra tệp dữ liệu	30
3.1.4.Chuẩn bị tập train và set	31
3.1.5.Tạo Model	31
3.1.6.Huấn luyện mô hình	34
3.1.7.Dự đoán và đánh giá các chỉ số sai số	35
3.1.8.Trực quan giá trị dự đoán và thực tế	36
3.1.9.Sử dụng RNN dự đoán mô hình	36
3.1.10.So sánh đánh giá 2 mô hình	39
3.2.Sử dụng LTSM và Simple Dense Layer dự đoán tập dữ liệu Apple	39

3.2.1. Import các thư viện cần thiết	39
3.2.2. Đọc file CSV và In 5 dòng đầu	39
3.2.3. Kiểm tra tệp dữ liệu	40
3.2.4. Chuẩn bị tập train và set	40
3.2.5. Tạo Model	43
3.2.6. Huấn luyện mô hình	45
3.2.7. Dự đoán	46
3.2.8. Trực quan giá trị dự đoán và thực tế	46
3.2.9. Sử dụng mô hình Simple Dense Layer để so sánh hiệu suất đánh giá của LSTM	47
3.2.10. So sánh 2 mô hình áp dụng và đưa ra kết luận	50
CHƯƠNG 4: TỔNG KẾT	51
CHƯƠNG 5: TÀI LIỆU THAM KHẢO	51

BẢNG PHÂN CÔNG NHIỆM VỤ NHÓM 1

Nhiệm Vụ	Vũ	Huyền	Sương	Hân
Tìm kiếm tập dữ liệu	X	X	X	X
Tiền xử lý,trực quan trên python		X		
Trực quan trên PowerBI				X
Sử dụng RNN dự đoán tập dữ liệu Google	X			
Sử dụng RNN dự đoán tập dữ liệu Apple			X	
Thực hiện ppt	X	X	X	X
Thực hiện word	X	X	X	X

CHƯƠNG 1: GIỚI THIỆU VỀ ĐỀ TÀI

1.1. Lý do chọn đề tài

- Thị trường chứng khoán là một môi trường đầy thách thức và biến động, với giá cổ phiếu được ảnh hưởng bởi nhiều yếu tố khác nhau như tin tức kinh tế, biến động thị trường toàn cầu và các sự kiện địa phương. Do đó, việc có một phương pháp dự đoán giá cổ phiếu chính xác và hiệu quả là rất quan trọng để giúp nhà đầu tư ra quyết định đúng đắn.
- LSTM (Long Short-Term Memory) là một mô hình mạng nơ-ron thần kinh đặc biệt hiệu quả trong việc xử lý dữ liệu chuỗi thời gian. Khả năng của LSTM trong việc ghi nhớ và học từ dữ liệu lịch sử có thể giúp dự đoán giá cổ phiếu dựa trên các mô hình phức tạp và các yếu tố tương quan.
- Bằng cách áp dụng LSTM vào phân tích giá cổ phiếu, ta có thể nắm bắt được các xu hướng và biến động trong giá cổ phiếu, từ đó đưa ra dự đoán về hướng đi tiềm năng của thị trường. Điều này không chỉ hỗ trợ nhà đầu tư trong việc đưa ra quyết định đầu tư, mà còn mang lại cơ hội cho các nhà nghiên cứu và chuyên gia tài chính trong việc phát triển các chiến lược giao dịch mới và cải thiện hiệu suất đầu tư.

1.2. Tổng quan về tập dữ liệu

1.2.1. Nguồn dữ liệu

- Tập dữ liệu sử dụng gồm 4 file csv về giá cổ phiếu của 4 công ty: Amazon, Apple, Facebook, Google
- Tập dữ liệu được lấy từ Kaggle là một bộ dữ liệu chứa thông tin về giá cổ phiếu lịch sử của 10 công ty phổ biến. Dữ liệu này được tổng hợp từ một số nguồn tin cậy và bao gồm thông tin về giá mở cửa (Open), giá cao nhất (High), giá thấp nhất (Low), giá đóng cửa (Close) và khối lượng giao dịch (Volume) của các cổ phiếu trong một khoảng thời gian nhất định.

1.2.2. Mô tả chi tiết tập dữ liệu

Tập dữ liệu gồm 7 cột : Date, Open, High, Low, Adj close, Volume.

1.3. Các kiến thức liên quan

1.3.1. Giới thiệu về mạng nơ ron thần kinh

- Mạng nơ ron thần kinh (Neural Networks) là một mô hình tính toán được lấy cảm hứng từ cấu trúc và chức năng của bộ não con người. Các mạng này bao gồm các đơn vị xử lý (neurons) được kết nối với nhau theo nhiều lớp: lớp đầu vào, lớp ẩn, và lớp đầu ra. Chức năng chính của mạng nơ ron thần kinh là học từ dữ liệu để nhận diện các mẫu và đưa ra dự đoán.
- Một mạng nơ ron thần kinh cơ bản hoạt động qua ba giai đoạn:
- Lan truyền tiến (Forward Propagation): Dữ liệu đầu vào được truyền qua các lớp của mạng để tạo ra dự đoán đầu ra.
- Tính toán hàm mất mát (Loss Function): Hàm mất mát đánh giá độ chính xác của dự đoán so với giá trị thực.
- Lan truyền ngược (Backward Propagation): Dựa trên giá trị của hàm mất mát, mạng cập nhật các trọng số của các kết nối bằng cách sử dụng các thuật toán tối ưu hóa như Gradient Descent.

1.3.2. Giới thiệu về LSTM

- LSTM (Long Short-Term Memory) là một loại mạng nơ ron hồi quy (Recurrent Neural Network - RNN) được thiết kế để xử lý và dự đoán dữ liệu theo chuỗi thời gian, vượt qua những hạn chế của các RNN thông thường. LSTM đặc biệt hiệu quả trong việc duy trì thông tin trong thời gian dài và xử lý các vấn đề liên quan đến gradient biến mất hoặc bùng nổ.
- Cấu trúc cơ bản của một tế bào LSTM bao gồm:
 - Cổng vào (Input Gate): Quyết định lượng thông tin mới sẽ được thêm vào trạng thái tế bào.
 - Cổng quên (Forget Gate): Quyết định lượng thông tin cần loại bỏ khỏi trạng thái tế bào.
 - Cổng đầu ra (Output Gate): Quyết định lượng thông tin sẽ được xuất ra khỏi tế bào để đưa đến bước tiếp theo.
- Các đặc điểm nổi bật của LSTM:
 - Khả năng ghi nhớ dài hạn: Nhờ cấu trúc các cổng, LSTM có khả năng ghi nhớ và duy trì thông tin qua các bước thời gian dài.
 - Tránh gradient biến mất và bùng nổ: Bằng cách điều chỉnh cách thông tin được lưu trữ và cập nhật, LSTM giúp duy trì sự ổn định của quá trình học.

1.3.3. Ứng dụng LSTM trong dự đoán

- LSTM đã được ứng dụng rộng rãi trong nhiều lĩnh vực yêu cầu dự đoán dựa trên dữ liệu chuỗi thời gian. Một số ứng dụng phổ biến bao gồm:
 - Dự báo tài chính: LSTM được sử dụng để dự đoán giá cổ phiếu, tỷ giá hối đoái và các chỉ số tài chính khác.
 - Xử lý ngôn ngữ tự nhiên: Trong các tác vụ như dự đoán từ tiếp theo trong văn bản, phân loại cảm xúc, và dịch máy.
 - Nhận dạng giọng nói: Giúp cải thiện độ chính xác của các hệ thống nhận dạng giọng nói bằng cách xử lý chuỗi âm thanh.
 - Dự báo nhu cầu: LSTM có thể dự đoán nhu cầu của sản phẩm trong tương lai dựa trên dữ liệu bán hàng trong quá khứ.
 - Phân tích chuỗi thời gian: Ứng dụng trong việc dự đoán xu hướng thời tiết, sản xuất năng lượng, và các hiện tượng khác theo thời gian.
 - Nhờ khả năng xử lý các quan hệ dài hạn và tính năng động trong việc quản lý thông tin, LSTM đã trở thành công cụ quan trọng trong việc phân tích và dự đoán dữ liệu chuỗi thời gian phức tạp.

CHƯƠNG 2: PHÂN TÍCH DỮ LIỆU

2.1. Preprocessing

```
df_final.info()  
✓ 0.1s  
  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 7044 entries, 0 to 7043  
Data columns (total 8 columns):  
 #   Column      Non-Null Count  Dtype     
---  --          -----          ----  
 0   Date        7044 non-null    object    
 1   Open         7044 non-null    float64  
 2   High         7044 non-null    float64  
 3   Low          7044 non-null    float64  
 4   Close        7044 non-null    float64  
 5   Adj Close    7044 non-null    float64  
 6   Volume       7044 non-null    int64     
 7   Company      7044 non-null    object    
dtypes: float64(5), int64(1), object(2)  
memory usage: 440.4+ KB
```

2.2. Trực quan trên python

2.2.1. Import các thư viện cần thiết

```
import numpy as np  
import pandas as pd  
from matplotlib import pyplot as plt  
import seaborn as sns  
import plotly.subplots as sp  
import plotly.graph_objects as go  
import plotly.express as px  
✓ 4.8s
```

2.2.2. Đọc file CSV

```
#Loading the dataset  
  
df1=pd.read_csv("Amazon.csv")  
df2=pd.read_csv("Apple.csv")  
df3=pd.read_csv("Facebook.csv")  
df4=pd.read_csv("Google.csv")  
✓ 0.0s
```

- Sử dụng hàm pd.concat để kết hợp các DataFrame df1, df2, df3, và df4 thành một DataFrame duy nhất df_final.

```
df_final = pd.concat([df1, df2, df3, df4], ignore_index=True)
```

✓ 0.0s

2.2.3.Kiểm tra tệp dữ liệu

```
df_final.info()
```

✓ 0.1s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7044 entries, 0 to 7043
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype  
---  --  
 0   Date        7044 non-null    object 
 1   Open         7044 non-null    float64
 2   High         7044 non-null    float64
 3   Low          7044 non-null    float64
 4   Close        7044 non-null    float64
 5   Adj Close    7044 non-null    float64
 6   Volume       7044 non-null    int64  
 7   Company      7044 non-null    object 
dtypes: float64(5), int64(1), object(2)
memory usage: 440.4+ KB
```

```
df_final.describe()
```

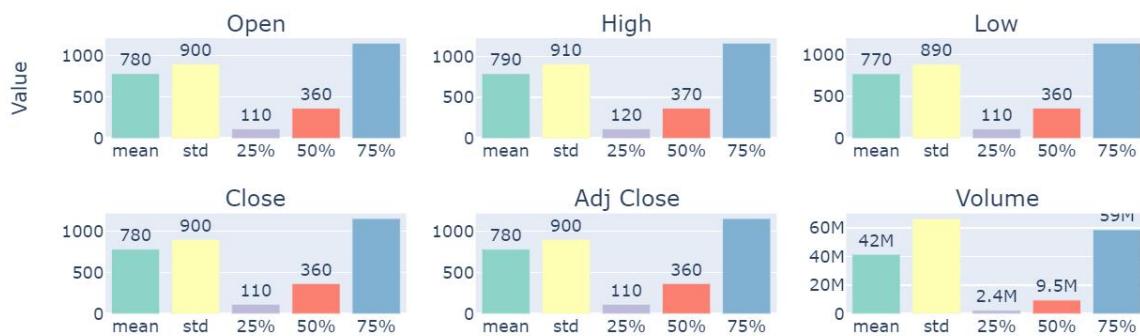
✓ 0.0s

	Open	High	Low	Close	Adj Close	Volume
count	7044.000000	7044.000000	7044.000000	7044.000000	7044.000000	7.044000e+03
mean	780.057697	787.913384	771.869170	780.149782	779.782324	4.150669e+07
std	899.058744	908.331676	889.167146	898.867655	899.168317	6.657481e+07
min	22.500000	22.917500	22.367500	22.584999	21.036304	3.468000e+05
25%	113.932502	115.119997	112.582496	113.889371	113.677501	2.419375e+06
50%	362.975006	367.660004	361.074997	364.595001	364.595001	9.509750e+06
75%	1146.777496	1159.350006	1137.664001	1151.322540	1151.322540	5.873652e+07
max	3744.000000	3773.080078	3696.790039	3731.409912	3731.409912	6.488252e+08

2.2.4. Trực quan hóa dữ liệu

a. Phân phối & Thống kê Dữ liệu

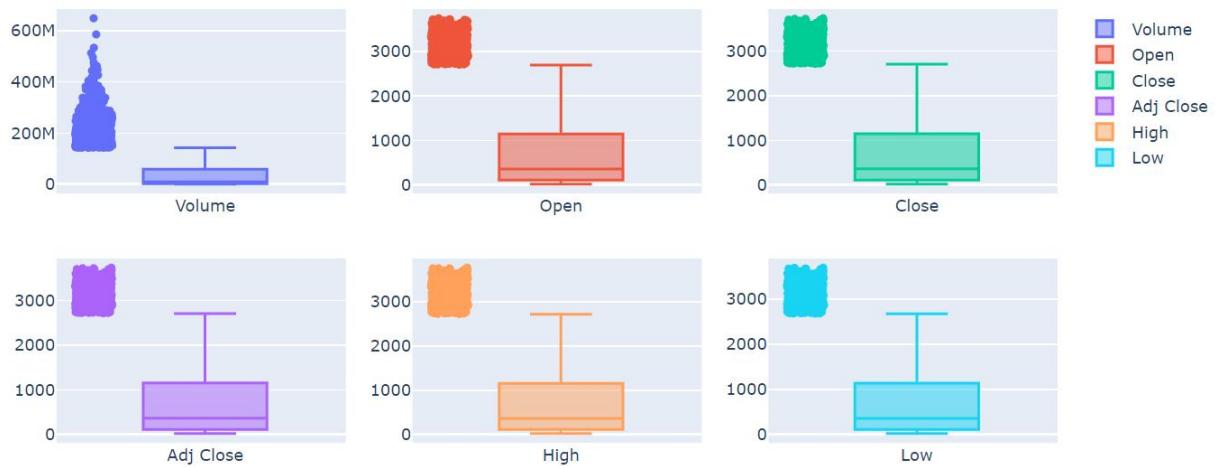
Thống kê mô tả



Phân phối dữ liệu



b. Trực quan hóa phân phối dữ liệu và phát hiện các giá trị ngoại lai

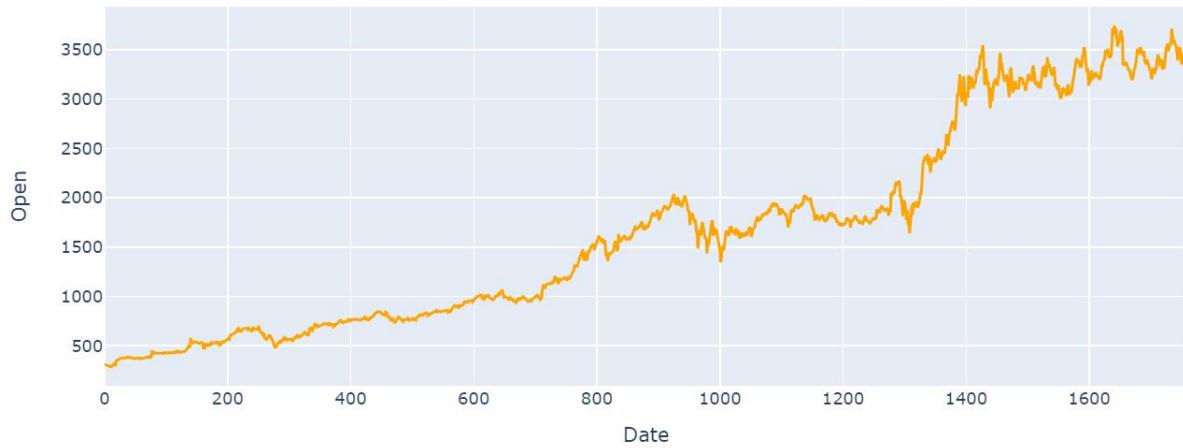


c. Trực quan hóa dữ liệu

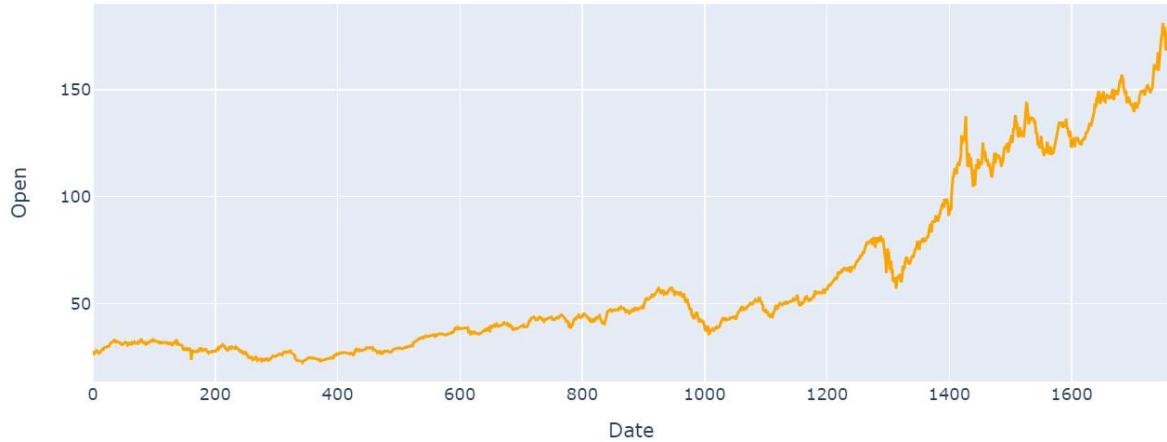
* Phân tích giá cổ phiếu các công ty qua các năm

- Open Price

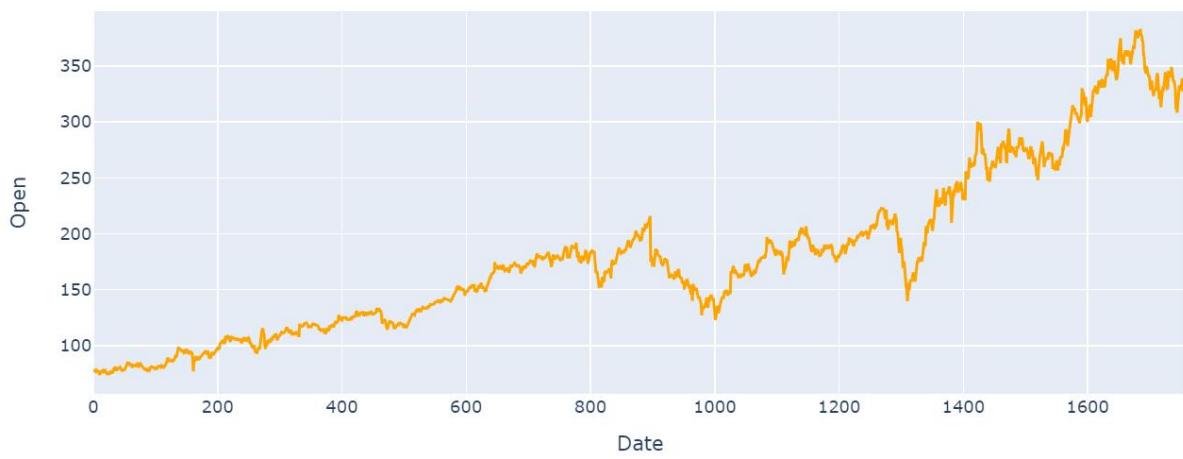
Open Price of Amazon



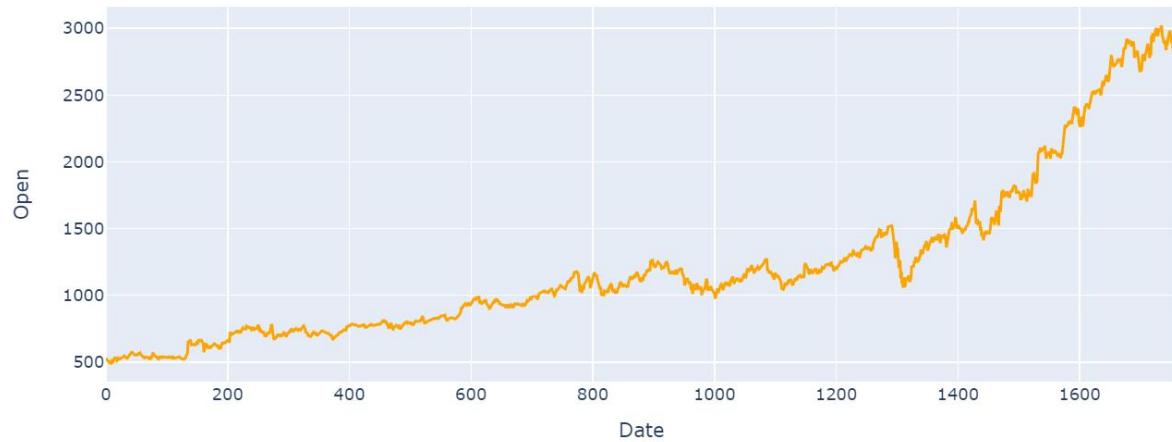
Open Price of Apple



Open Price of Facebook

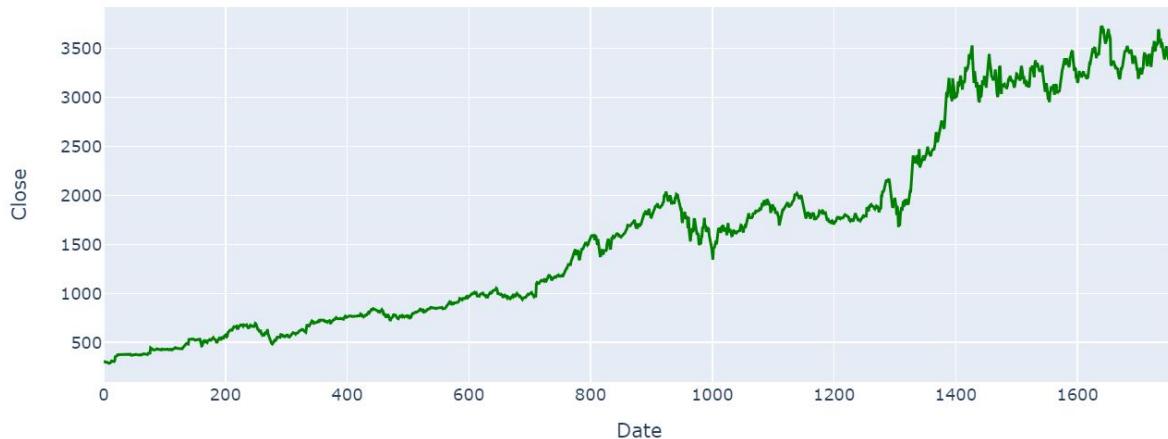


Open Price of Google

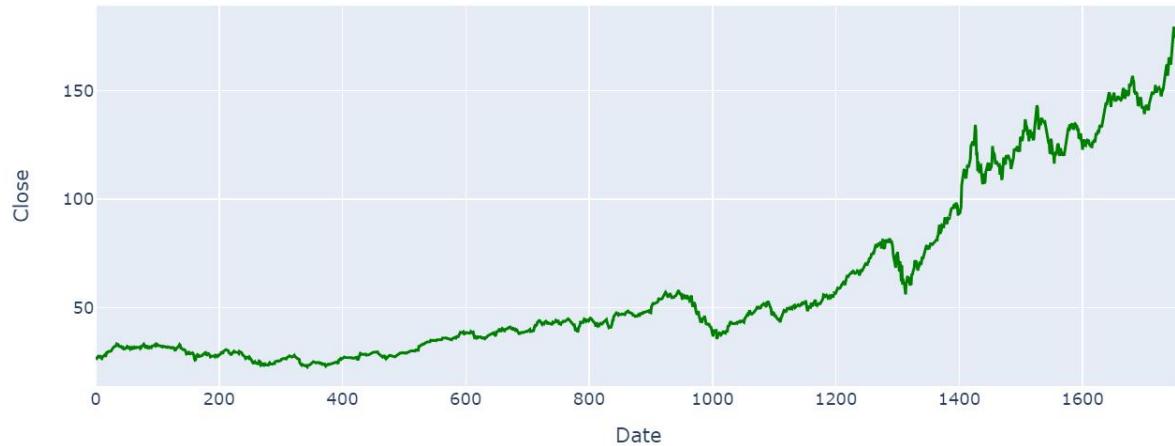


- Close Price

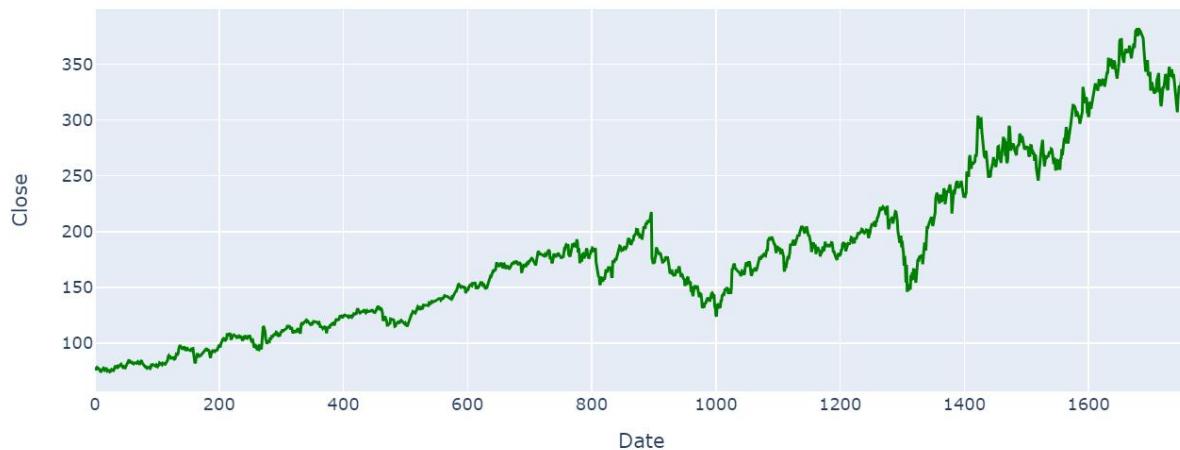
Close Price of Amazon



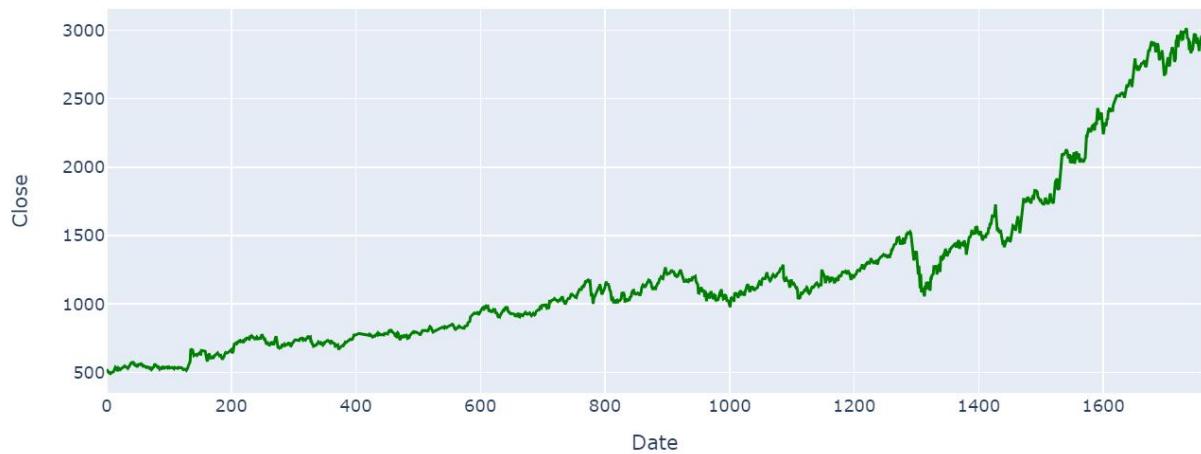
Close Price of Apple



Close Price of Facebook

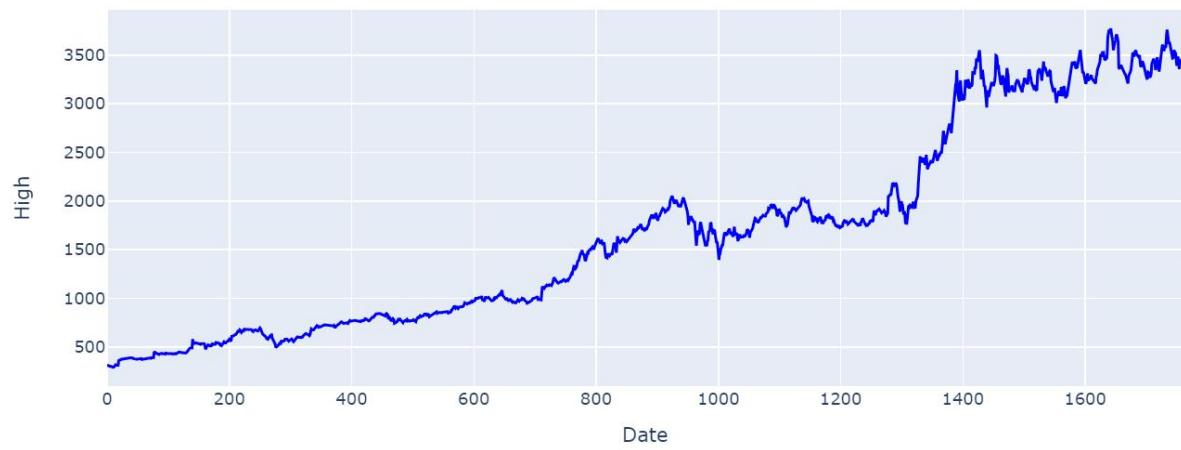


Close Price of Google

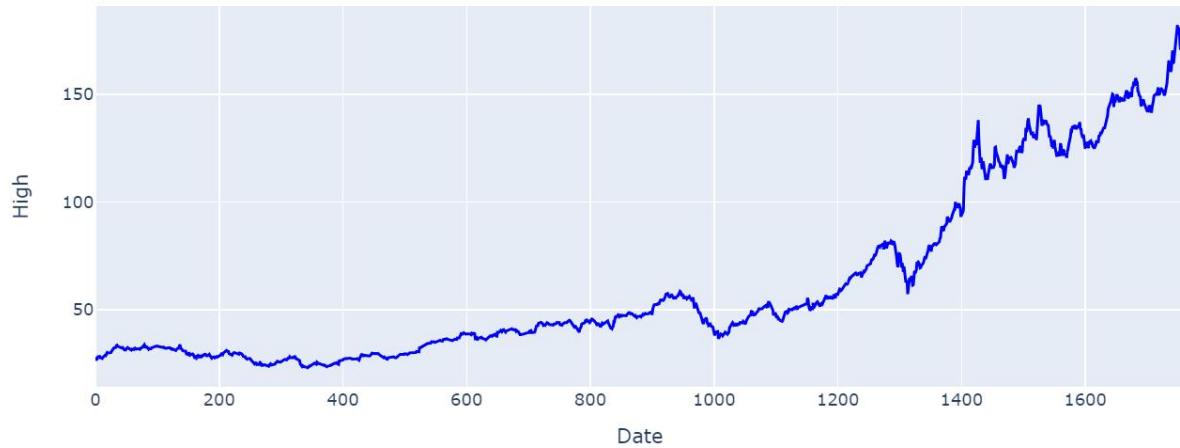


- High Price

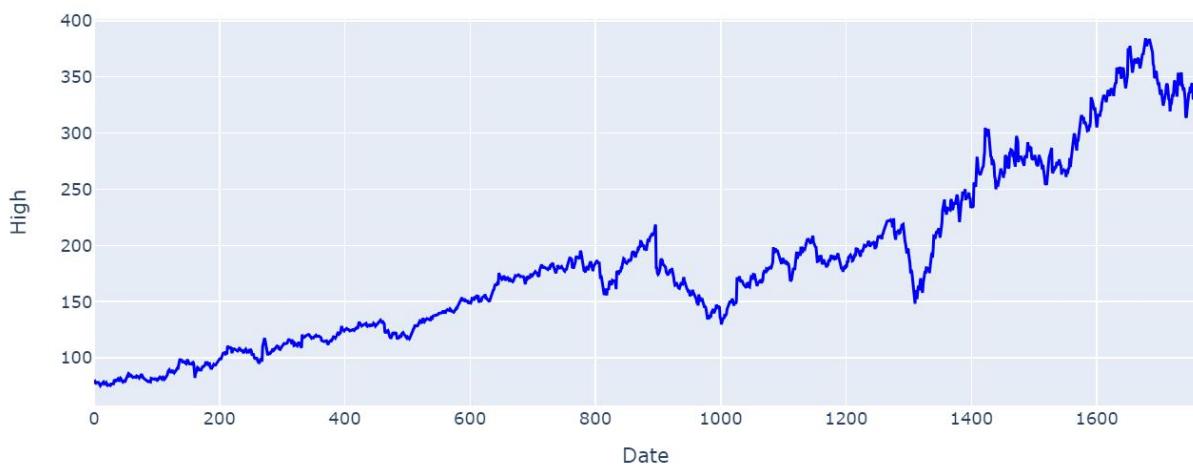
High Price of Amazon



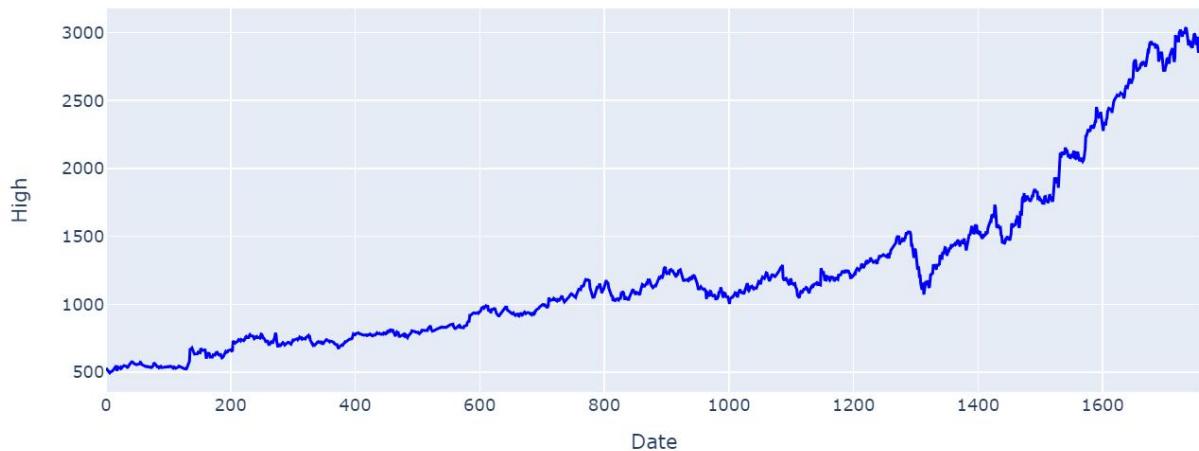
High Price of Apple



High Price of Facebook

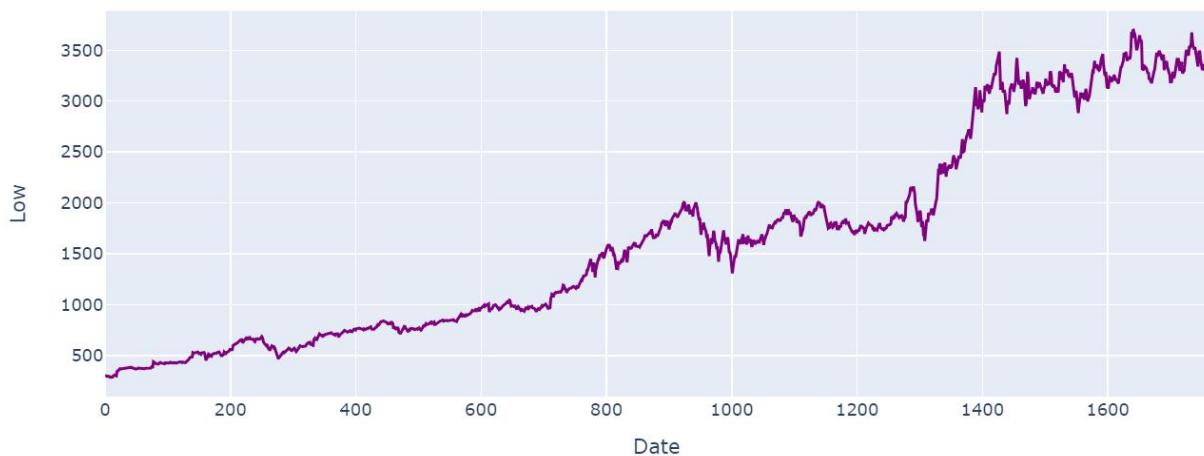


High Price of Google

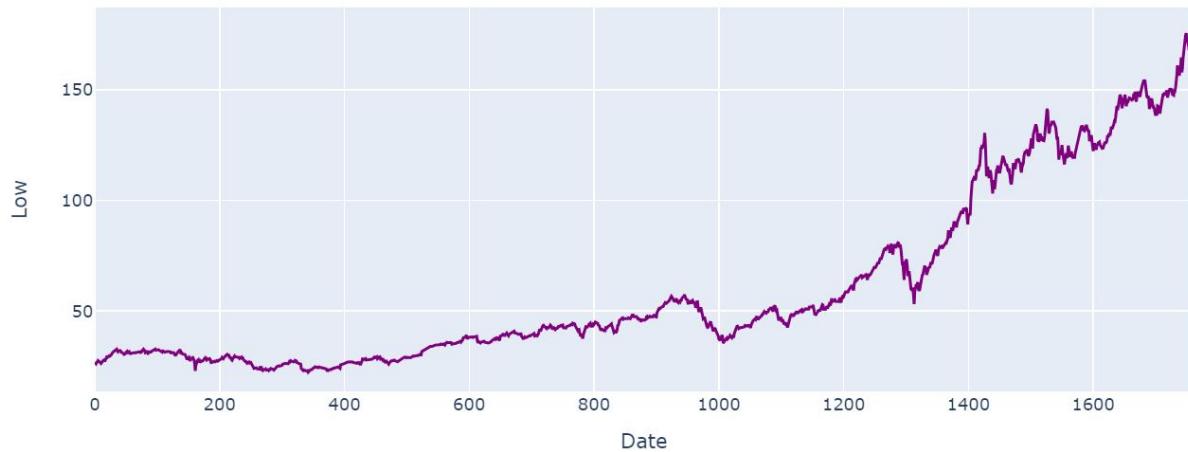


- Low Price

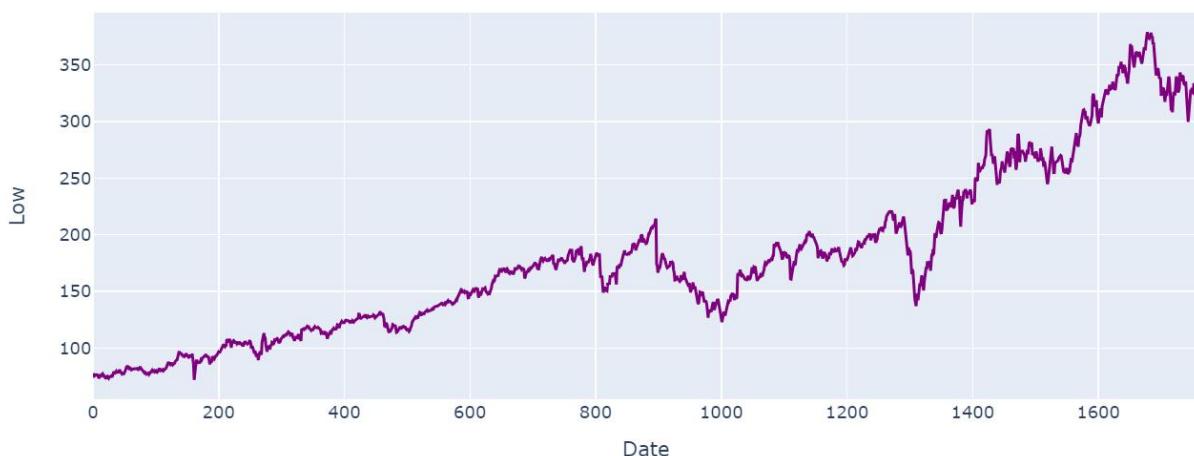
Low Price of Amazon



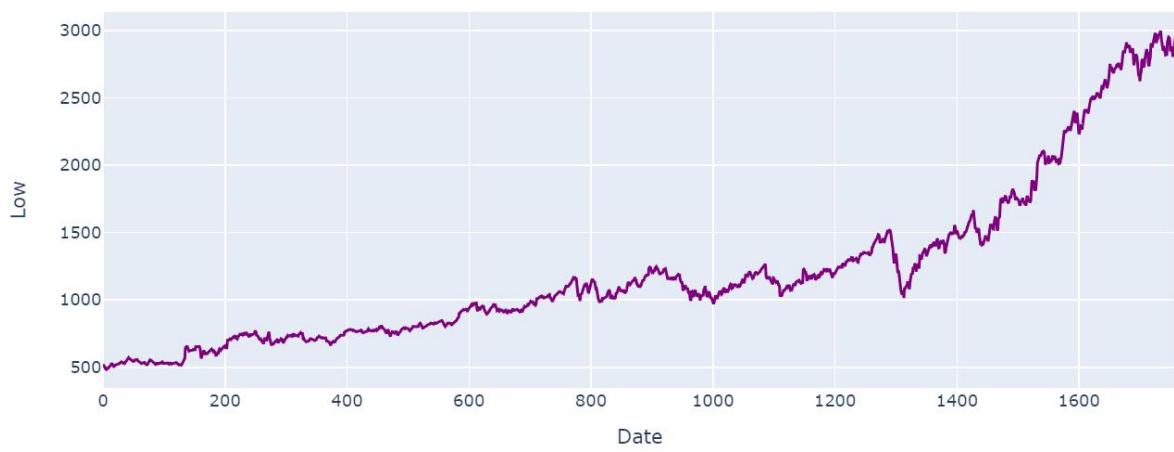
Low Price of Apple



Low Price of Facebook

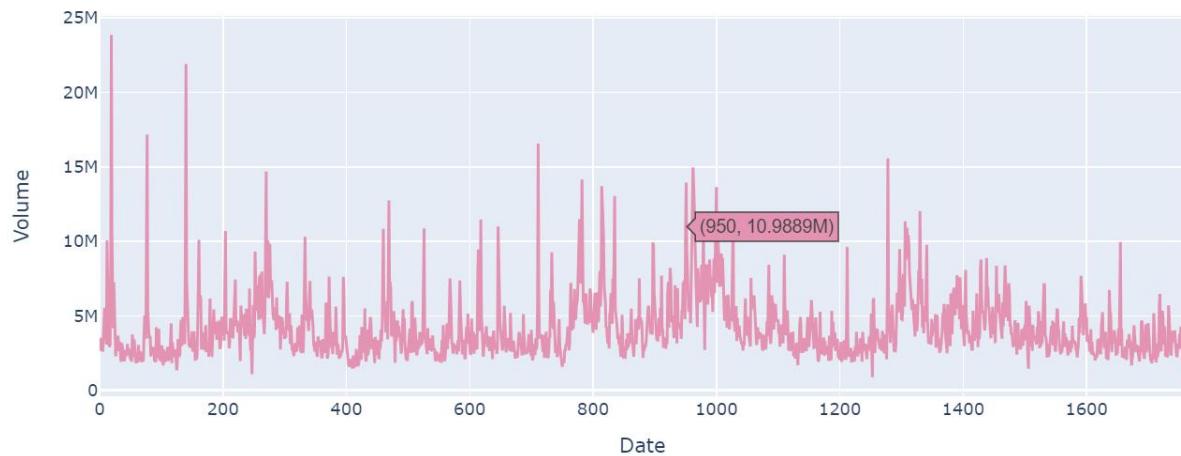


Low Price of Google

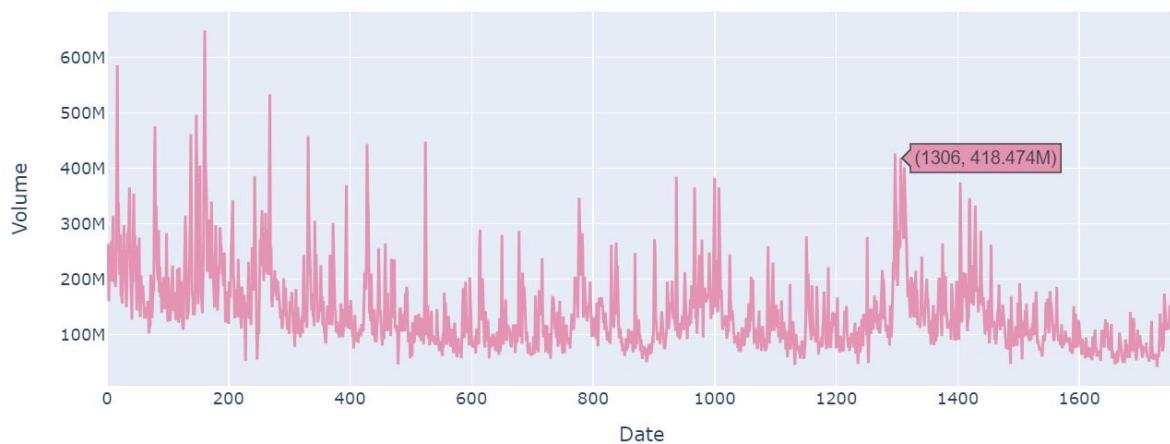


- Volume

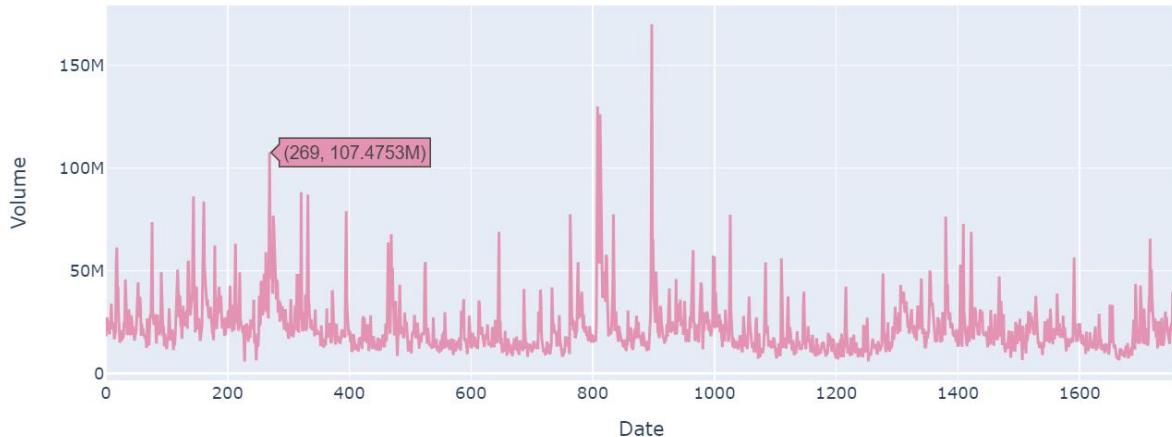
Volume Of Amazon Stock Prices



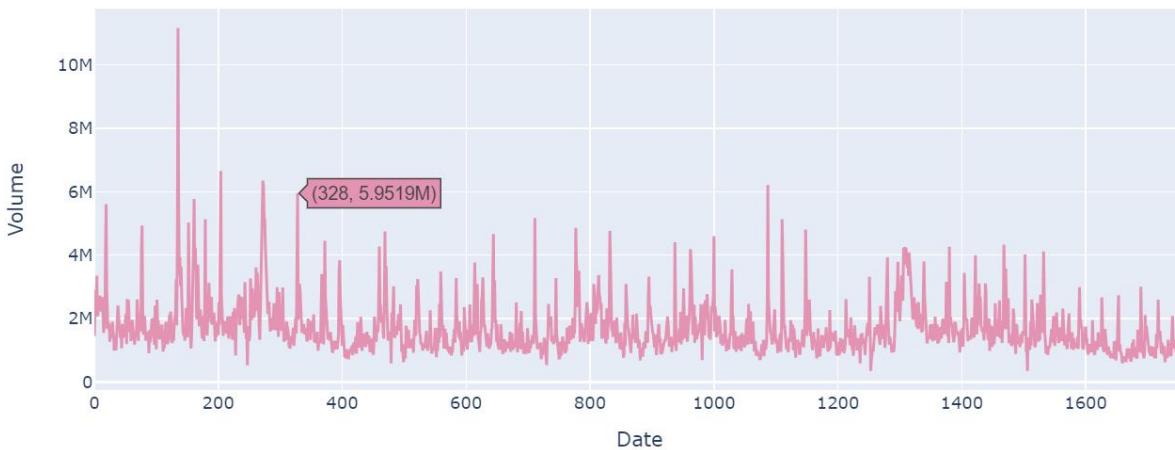
Volume Of Apple Stock Prices



Volume Of Facebook Stock Prices



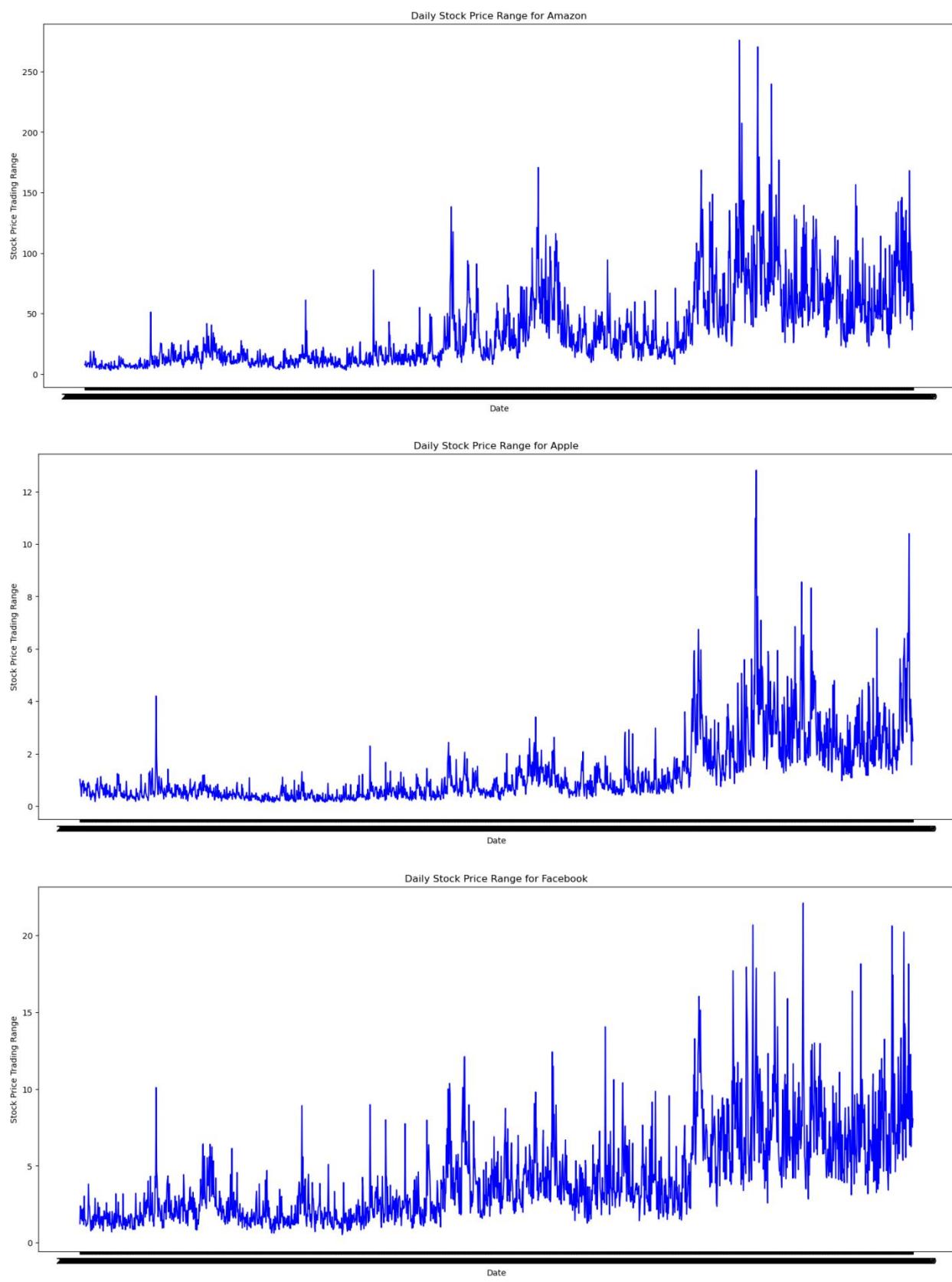
Volume Of Google Stock Prices

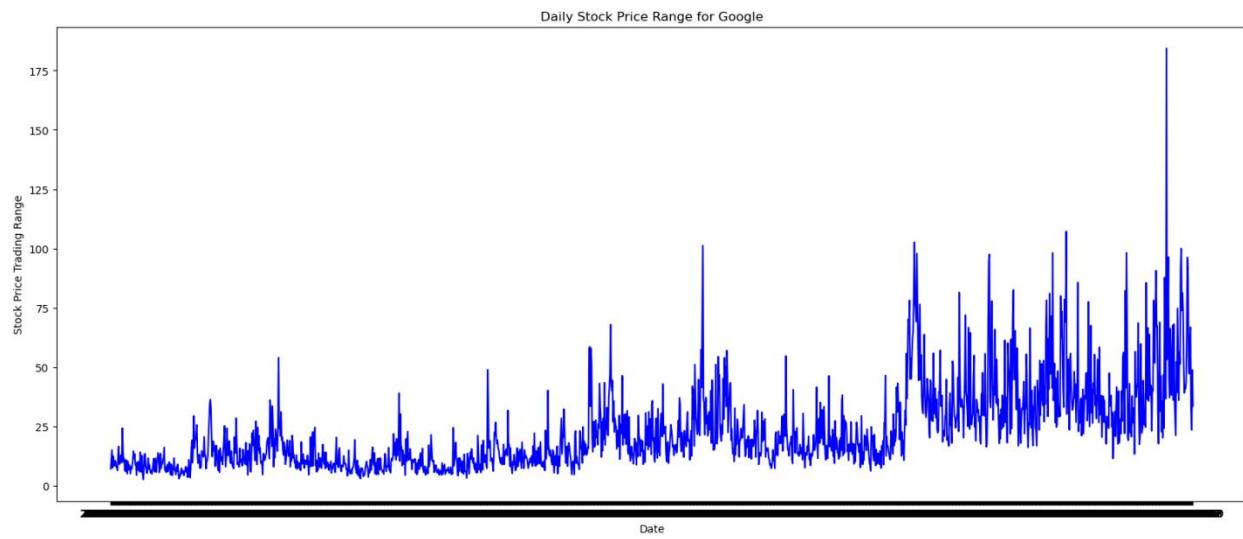


* Phân tích so sánh giữa giá cao nhất và giá thấp nhất của cổ phiếu qua các năm, cho thấy sự thay đổi trong biên độ dao động hàng ngày của giao dịch.

```
def high_low(df,cmp_name):
    plt.figure(figsize=(20,8))
    df['Daily Range']= df['High']-df['Low']
    plt.plot(df['Date'],df['Daily Range'],c='blue')
    plt.xlabel("Date")
    plt.ylabel("Stock Price Trading Range")
    plt.title("Daily Stock Price Range for "+cmp_name)
    plt.show()
```

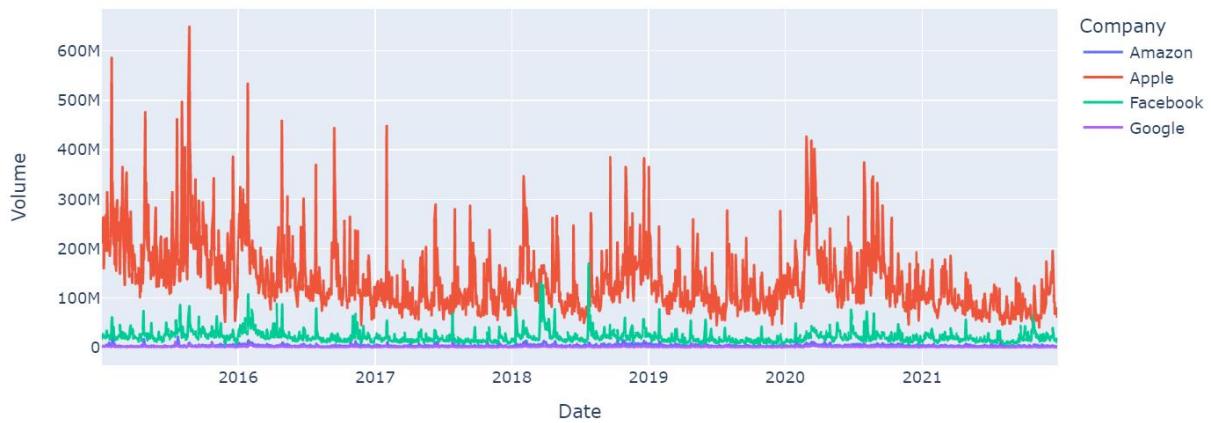
Python





* Lợi nhuận hằng ngày theo thời gian

Daily Returns Over Time for Companies

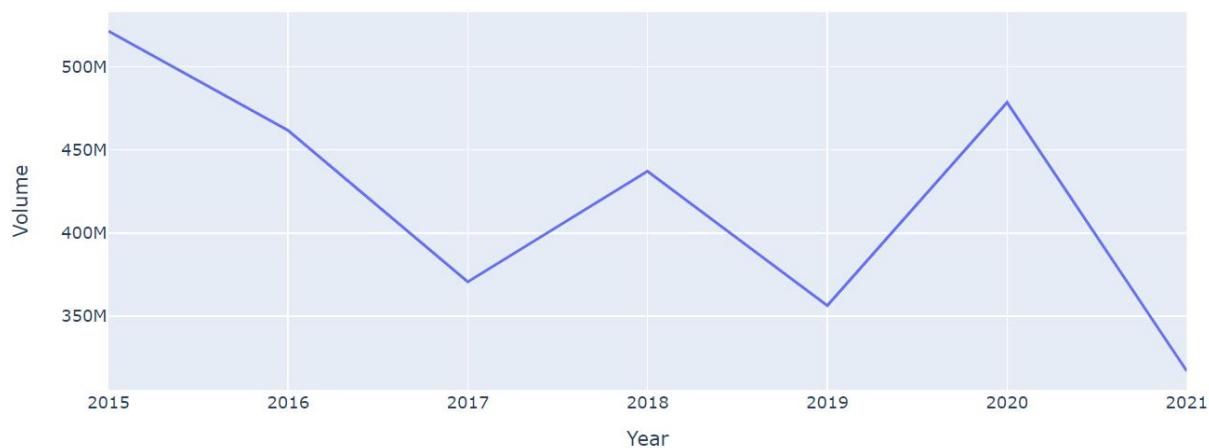


* Khối lượng cổ phiếu giao dịch theo thời gian

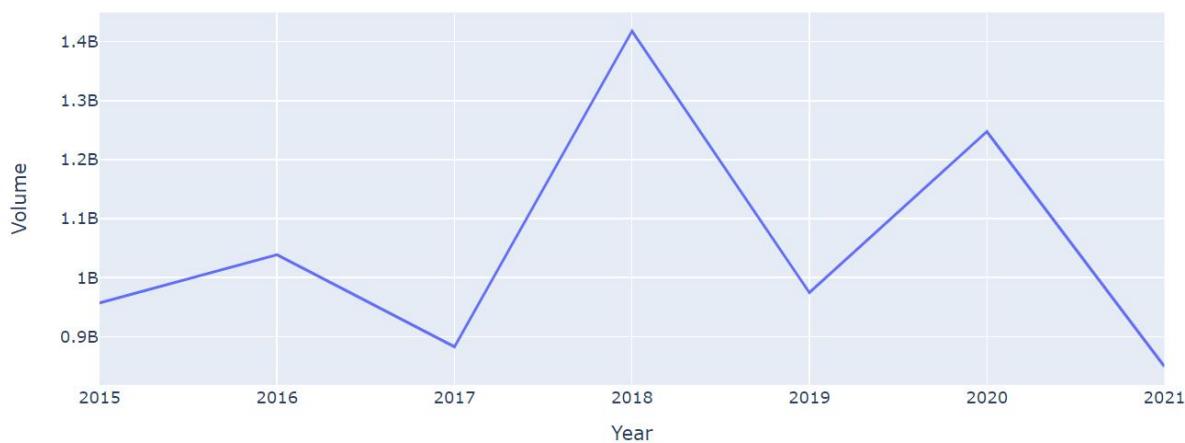
Volume of Stocks Traded Over Time for Apple by Year



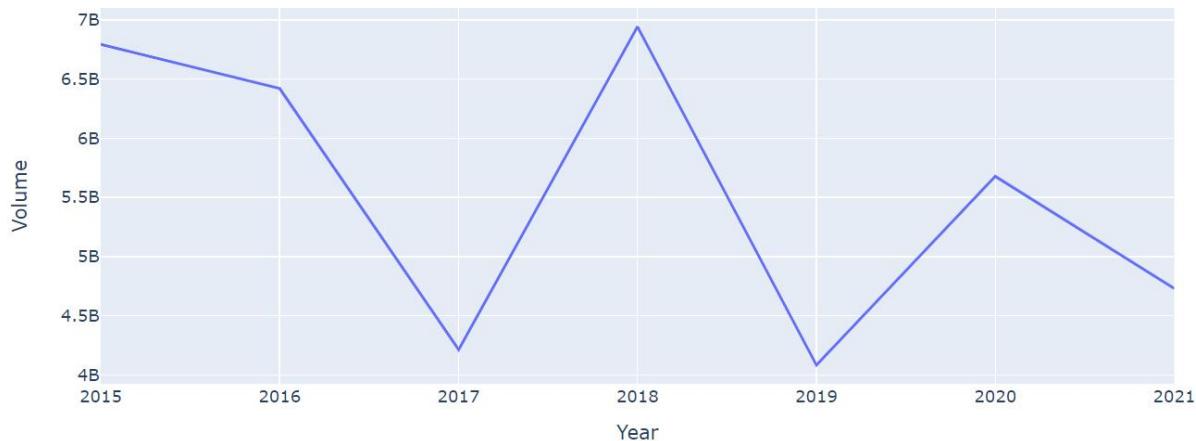
Volume of Stocks Traded Over Time for Google by Year



Volume of Stocks Traded Over Time for Amazon by Year

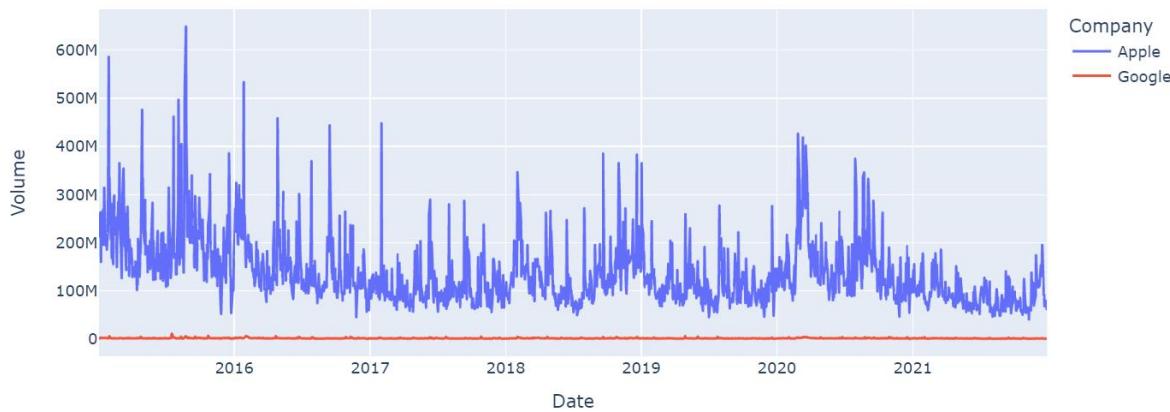


Volume of Stocks Traded Over Time for Facebook by Year



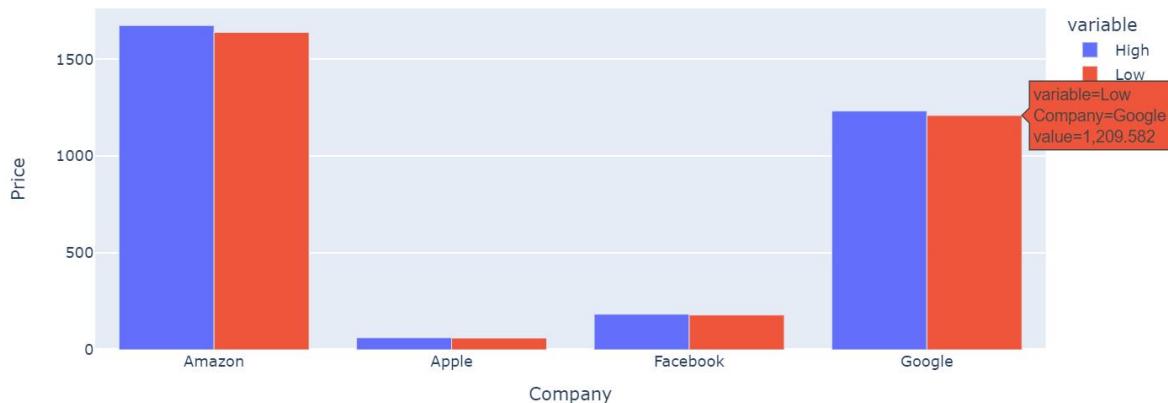
* Khối lượng cổ phiếu giao dịch theo thời gian: Apple so với Google

Volume of Stocks Traded Over Time: Apple vs Google



* Trung bình giá mở cửa và giá đóng cửa theo Công ty

Open and Close Averages by Company



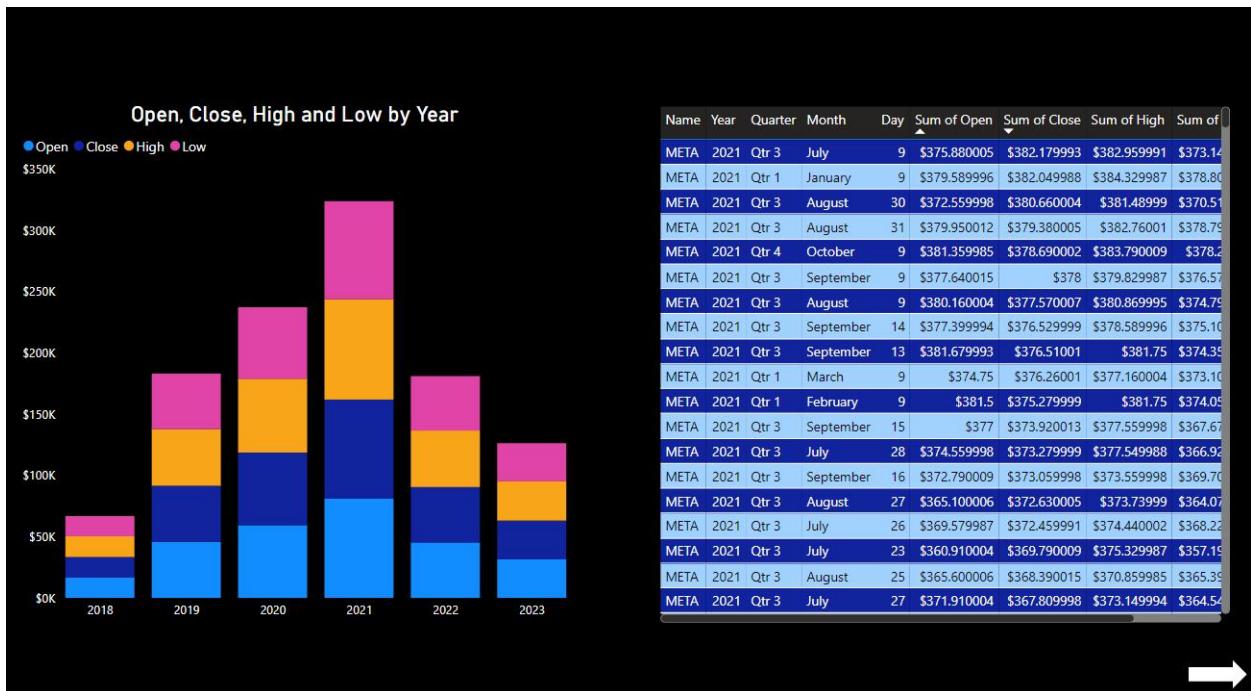
2.3.Trực quan trên PowerBI

2.3.1 Tập dữ liệu facebook.csv

Tính tổng giá mở cửa, đóng cửa, giá đỉnh và giá thấp theo tháng, theo năm, theo quý

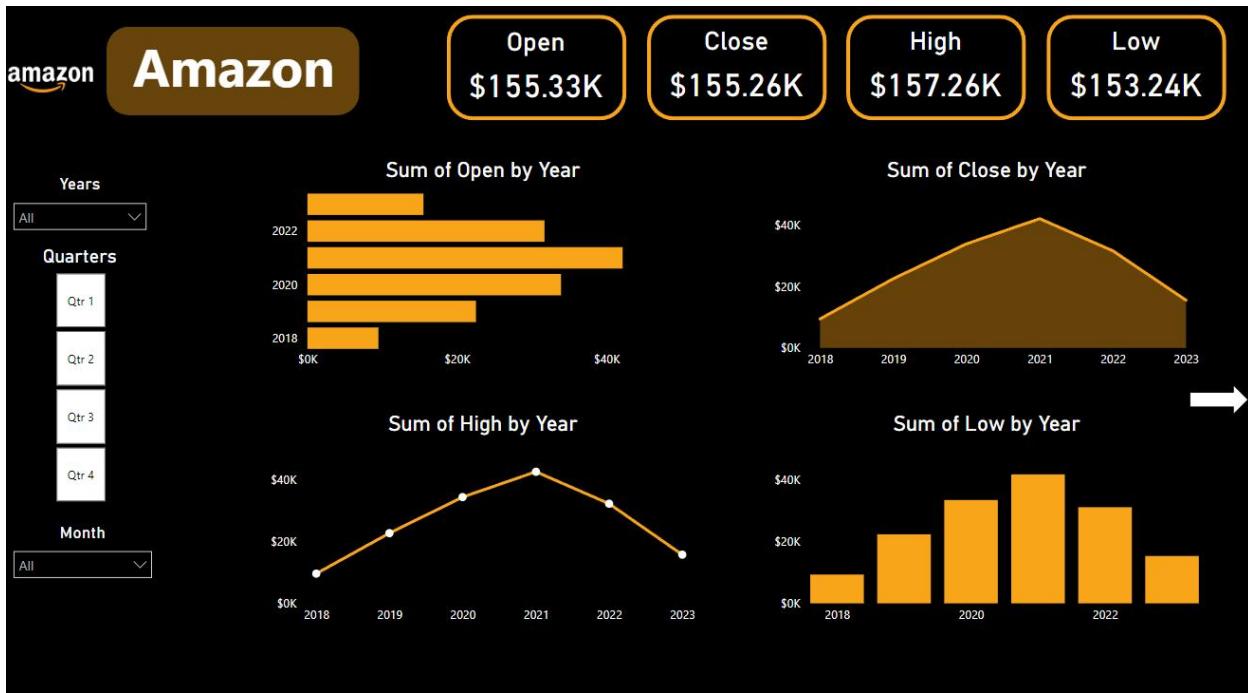


So sánh các tổng giá mở cửa, đóng cửa, giá đỉnh và giá thấp theo năm

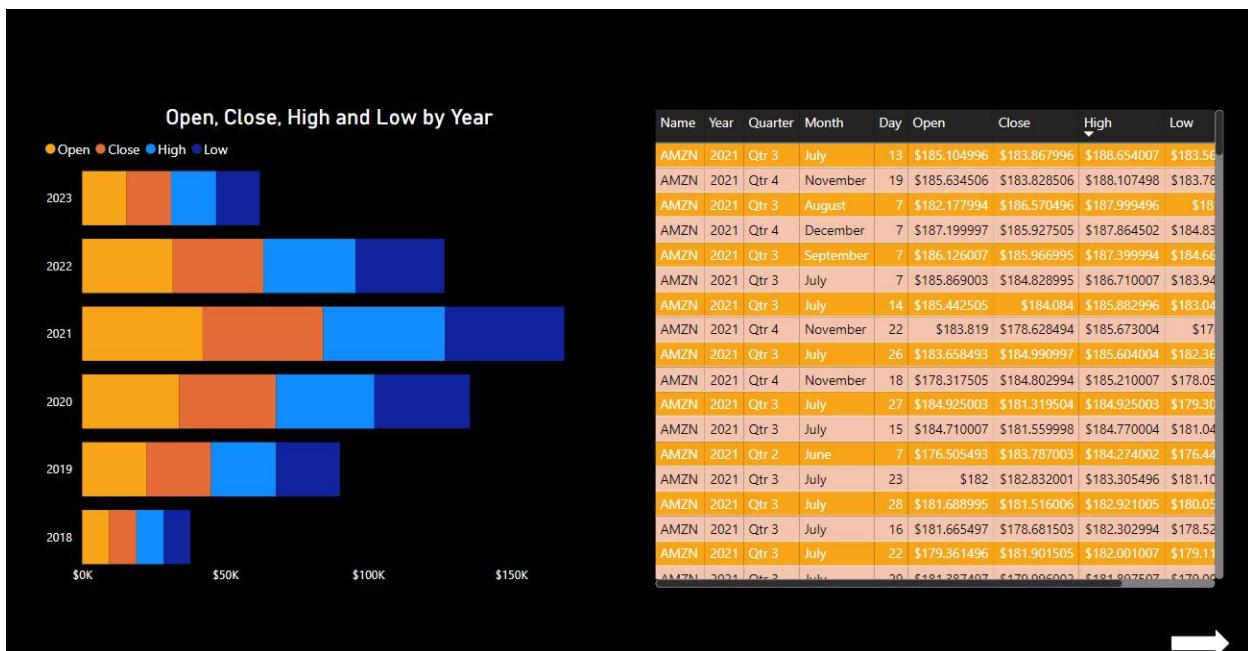


2.3.2 Tập dữ liệu amazon.csv

Tính tổng giá mở cửa, đóng cửa, giá đỉnh và giá thấp theo tháng, theo năm, theo quý

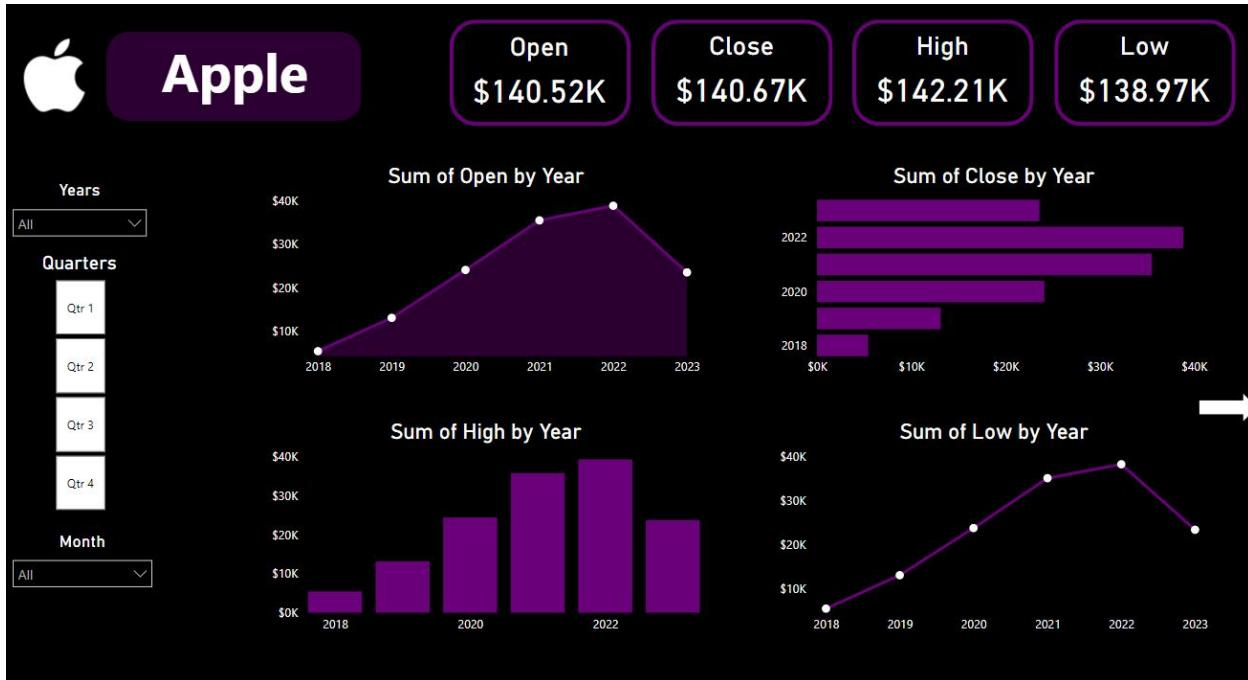


So sánh các tổng giá mở cửa, đóng cửa, giá đỉnh và giá thấp theo năm



2.3.3 Tập dữ liệu Apple.csv

Tính tổng giá mở cửa, đóng cửa, giá đỉnh và giá thấp theo tháng, theo năm, theo quý



So sánh các tổng giá mở cửa, đóng cửa, giá đỉnh và giá thấp theo năm



2.3.4 Tập dữ liệu Google.csv

Tính tổng giá mở cửa, đóng cửa, giá đỉnh và giá thấp theo tháng, theo năm, theo quý



So sánh các tổng giá mở cửa, đóng cửa, giá đỉnh và giá thấp theo năm



CHƯƠNG 3: DỰ ĐOÁN DỮ LIỆU

3.1. Sử dụng LTSM và RNN dự đoán tập dữ liệu Google

3.1.1. Import các thư viện cần thiết

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import LSTM, Dense
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from keras.models import load_model
```

3.1.2. Đọc file CSV và Đổi kiểu dữ liệu ngày

```
df=pd.read_csv('CSV/Google.csv')
df['Date']=pd.to_datetime(df['Date'])
```

3.1.3. Kiểm tra tệp dữ liệu

```
Data.info()
0.0s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1761 entries, 0 to 1760
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Date        1761 non-null    object 
 1   Open         1761 non-null    float64
 2   High         1761 non-null    float64
 3   Low          1761 non-null    float64
 4   Close        1761 non-null    float64
 5   Adj Close    1761 non-null    float64
 6   Volume       1761 non-null    int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 96.4+ KB
```

3.1.4. Chuẩn bị tập train và set

- Chuẩn hoá dữ liệu cột Close giá trị của dữ liệu sẽ được chuẩn hoá nằm trong vùng giá trị từ [0,1]

```
#Chuẩn hoá dữ liệu
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(df['Close'].values.reshape(-1, 1))
scaled_data

array([[0.01275147],
       [0.00842718],
       [0.0037195 ],
       ...,
       [0.97903276],
       [0.96622247],
       [0.96667041]])
```

- Chuẩn bị tập train và test

```
#Tạo tập train và test
X = scaled_data
y = scaled_data

X = np.reshape(X, (X.shape[0], 1, X.shape[1]))

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

3.1.5. Tạo Model

```
model=Sequential(
[
    LSTM(units=60,input_shape=(X.shape[1],1),return_sequences=True),
    LSTM(units=50,return_sequences=True),
    LSTM(units=30,return_sequences=True),
    LSTM(units=20,return_sequences=True),
    LSTM(10),
    Dense(units=1),
])
```

- Lớp đầu tiên

- LSTM(units=60, input_shape=(X.shape[1], 1), return_sequences=True) định nghĩa một lớp LSTM với 60 đơn vị (units).
 - Tham số input_shape=(X.shape[1], 1) xác định hình dạng đầu vào của lớp LSTM. Trong trường hợp này, kích thước của dữ liệu đầu vào là (X.shape[1], 1), trong đó X.shape[1] đại diện cho độ dài của mỗi cửa sổ dữ liệu và 1 đại diện cho số lượng đặc trưng.
 - return_sequences=True cho phép lớp LSTM trả về chuỗi kết quả cho lớp LSTM tiếp theo trong mạng nơ-ron.
- Các lớp LSTM tiếp theo:
 - LSTM(units=50, return_sequences=True) định nghĩa một lớp LSTM với 50 đơn vị (units) và trả về chuỗi kết quả.
 - LSTM(units=30, return_sequences=True) định nghĩa một lớp LSTM với 30 đơn vị (units) và trả về chuỗi kết quả.
 - LSTM(units=20, return_sequences=True) định nghĩa một lớp LSTM với 20 đơn vị (units) và trả về chuỗi kết quả.
- Lớp LSTM cuối cùng:
 - LSTM(10) định nghĩa một lớp LSTM với 10 đơn vị (units).
 - Lớp này không có tham số return_sequences=True, do đó sẽ trả về kết quả duy nhất.
- Lớp Dense:
 - Dense(units=1) định nghĩa một lớp Dense với 1 đơn vị (units).
 - Lớp này sẽ ánh xạ đầu ra từ lớp LSTM cuối cùng sang một giá trị dự đoán duy nhất.

Sử dụng phương pháp tối ưu hóa Adam (Adam optimizer) và hàm mất mát là mean squared error (MSE) để biên dịch (compile) mô hình "Model1".

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 1, 60)	14,880
lstm_1 (LSTM)	(None, 1, 50)	22,200
lstm_2 (LSTM)	(None, 1, 30)	9,720
lstm_3 (LSTM)	(None, 1, 20)	4,080
lstm_4 (LSTM)	(None, 10)	1,240
dense (Dense)	(None, 1)	11

Total params: 52,131 (203.64 KB)

Trainable params: 52,131 (203.64 KB)

Non-trainable params: 0 (0.00 B)

```
model.compile(optimizer = "adam", loss='mean_squared_error')
```

3.1.6. Huấn luyện mô hình

```
model.fit(X_train, y_train, epochs=50, batch_size=1, verbose=2)

1  ⏪ 1m 26.2s

Epoch 1/50
1408/1408 - 10s - 7ms/step - loss: 0.0098
Epoch 2/50
1408/1408 - 4s - 3ms/step - loss: 4.0154e-04
Epoch 3/50
1408/1408 - 3s - 2ms/step - loss: 3.1733e-04
Epoch 4/50
1408/1408 - 4s - 3ms/step - loss: 3.7046e-04
Epoch 5/50
1408/1408 - 4s - 3ms/step - loss: 1.7551e-04
Epoch 6/50
1408/1408 - 3s - 2ms/step - loss: 1.1384e-04
Epoch 7/50
1408/1408 - 4s - 3ms/step - loss: 1.8752e-04
Epoch 8/50
1408/1408 - 4s - 3ms/step - loss: 1.0532e-04
Epoch 9/50
1408/1408 - 4s - 3ms/step - loss: 1.8188e-04
Epoch 10/50
1408/1408 - 3s - 2ms/step - loss: 6.0252e-05
Epoch 11/50
1408/1408 - 4s - 3ms/step - loss: 2.3433e-04
Epoch 12/50
1408/1408 - 3s - 2ms/step - loss: 1.5770e-04
Epoch 13/50
...
1408/1408 - 4s - 3ms/step - loss: 1.3967e-04
Epoch 22/50
1408/1408 - 4s - 3ms/step - loss: 1.3412e-04
Epoch 23/50

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Lưu mô hình lại và load mô hình

```
model.save('lstm_model.h5')
```



```
loaded_model = load_model('lstm_model.h5')
```



3.1.7. Dự đoán và đánh giá các chỉ số sai số

```
import matplotlib.pyplot as plt

# Dự Đoán
predictions = loaded_model.predict(X)

# Chuyển dữ liệu về ban đầu
predictions = scaler.inverse_transform(predictions)
y_actual = scaler.inverse_transform(y)

# Vẽ biểu đồ
plt.figure(figsize=(14, 7))
plt.plot(df['Date'], y_actual, label='Actual', color='blue')
plt.plot(df['Date'], predictions, label='Predicted', color='red')
plt.xlabel('Date')
plt.ylabel('Confirmed Cases')
plt.title('Actual vs Predicted Stock By LTSM')
plt.legend()
plt.show()

# Đánh giá mô hình
mse = mean_squared_error(y_actual, predictions)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_actual, predictions)
r2 = r2_score(y_actual, predictions)

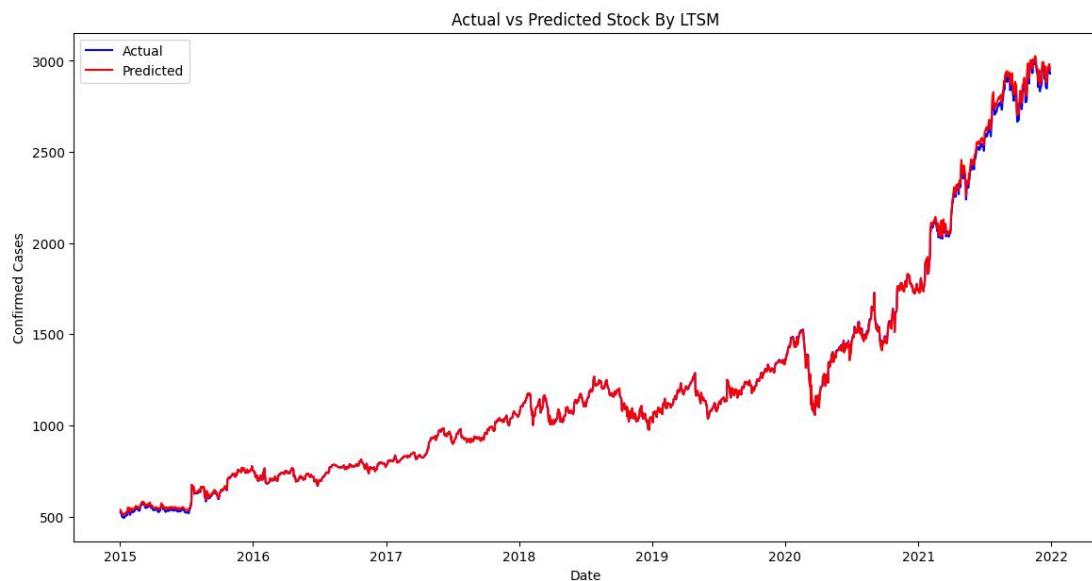
# In kết quả đánh giá
print("Mean Squared Error (MSE): ", mse)
print("Root Mean Squared Error (RMSE): ", rmse)
print("Mean Absolute Error (MAE): ", mae)
print("R2 Score: ", r2)
```



56/56 ━━━━━━━━ 0s 4ms/step

Mean Squared Error (MSE): 112.66388042001495
Root Mean Squared Error (RMSE): 10.614324303506793
Mean Absolute Error (MAE): 5.379887753193098
R2 Score: 0.9996985644983746

3.1.8.Trực quan giá trị dự đoán và thực tế



3.1.9.Sử dụng RNN dự đoán mô hình

- Import thư viện và tạo model

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, SimpleRNN
Model3 = Sequential(
    [
        SimpleRNN(units=30, input_shape=(X.shape[1], 1), activation='relu', return_sequences=True),
        SimpleRNN(units=20, activation='relu', return_sequences=True),
        SimpleRNN(units=10, activation='relu'),
        Dense(units=1)
    ]
)
```

Sử dụng phương pháp tối ưu hóa Adam (Adam optimizer) và hàm mất mát là mean squared error (MSE) để biên dịch (compile) mô hình "Model3".

```
Model3.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
simple_rnn (SimpleRNN)	(None, 1, 30)	960
simple_rnn_1 (SimpleRNN)	(None, 1, 20)	1,020
simple_rnn_2 (SimpleRNN)	(None, 10)	310
dense_1 (Dense)	(None, 1)	11

Total params: 2,301 (8.99 KB)

Trainable params: 2,301 (8.99 KB)

Non-trainable params: 0 (0.00 B)

```
Model3.compile(optimizer = "adam", loss='mean_squared_error')
```

- Thực hiện training sau đó sao lưu và load lại mô hình

```
#Model3.fit(X_train, y_train, epochs=50, batch_size=1, verbose=2)
```

```
#Model3.save('RNN_model.h5')
```

```
from tensorflow.keras.models import load_model  
  
loaded_model3 = load_model('RNN_model.h5')
```

- Thực hiện việc dự đoán và đánh giá các chỉ số mô hình

```

import matplotlib.pyplot as plt

# Dự Đoán
predictions = loaded_model3.predict(X)

# Chuyển dữ liệu về ban đầu
predictions = scaler.inverse_transform(predictions)
y_actual = scaler.inverse_transform(y)

# Vẽ biểu đồ
plt.figure(figsize=(14, 7))
plt.plot(df['Date'], y_actual, label='Actual', color='blue')
plt.plot(df['Date'], predictions, label='Predicted', color='red')
plt.xlabel('Date')
plt.ylabel('Confirmed Cases')
plt.title('Actual vs Predicted Stock By RNN')
plt.legend()
plt.show()

# Đánh giá mô hình
mse = mean_squared_error(y_actual, predictions)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_actual, predictions)
r2 = r2_score(y_actual, predictions)

# In kết quả đánh giá
print("Mean Squared Error (MSE): ", mse)
print("Root Mean Squared Error (RMSE): ", rmse)
print("Mean Absolute Error (MAE): ", mae)
print("R2 Score: ", r2)

```

Mean Squared Error (MSE): 3.286940891250584
Root Mean Squared Error (RMSE): 1.8129922479841396
Mean Absolute Error (MAE): 1.6755161055419425
R2 Score: 0.9999912056936733

3.1.10.So sánh đánh giá 2 mô hình

Kết Luận

Model	MSE	MAE	R2 Score	RMSE
LTSM	112.66388042001495	5.379887753193098	0.9996985644983746	10.614324303506793
RNN	3.286940891250584	1.6755161055419425	0.9999912056936733	1.8129922479841396

- Về MSE: LSTM có xu hướng dự đoán sai lệch hơn so với mô hình RNN.
- Về MAE: LSTM có xu hướng dự đoán sai lệch hơn so với mô hình RNN.
- Về RMSE: LSTM có khả năng giải thích dữ liệu tốt hơn mô hình RNN.
- Về R2 Score:LSTM có xu hướng dự đoán sai lệch hơn so với mô hình RNN.

-> Dựa trên ba chỉ số MSE, MAE và RMSE, có thể thấy rằng mô hình RNN có hiệu suất tốt hơn mô hình LSTM. Tuy nhiên, mô hình LSTM có giá trị R2 Score cao hơn, cho thấy khả năng giải thích dữ liệu tốt hơn.

3.2.Sử dụng LTSM và Simple Dense Layer dự đoán tập dữ liệu

Apple

3.2.1.Import các thư viện cần thiết

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler,MinMaxScaler
import numpy as np
import tensorflow as tf
from tensorflow import keras
from keras import models, layers
from keras.models import Sequential
from keras.layers import LSTM,Dense,Dropout
from keras.utils import plot_model
plt.style.use('fivethirtyeight')
from sklearn.metrics import mean_squared_error
```

3.2.2.Đọc file CSV và In 5 dòng đầu

```
Data=pd.read_csv(r'./Apple.csv')
```

3.2.3.Kiểm tra tệp dữ liệu

```
Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1761 entries, 0 to 1760
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Date        1761 non-null    object  
 1   Open         1761 non-null    float64 
 2   High         1761 non-null    float64 
 3   Low          1761 non-null    float64 
 4   Close        1761 non-null    float64 
 5   Adj Close    1761 non-null    float64 
 6   Volume       1761 non-null    int64   
dtypes: float64(5), int64(1), object(1)
memory usage: 96.4+ KB
```

3.2.4.Chuẩn bị tập train và set

- Đặt Date làm chỉ mục

```
Data=Data.set_index('Date')
```

Data

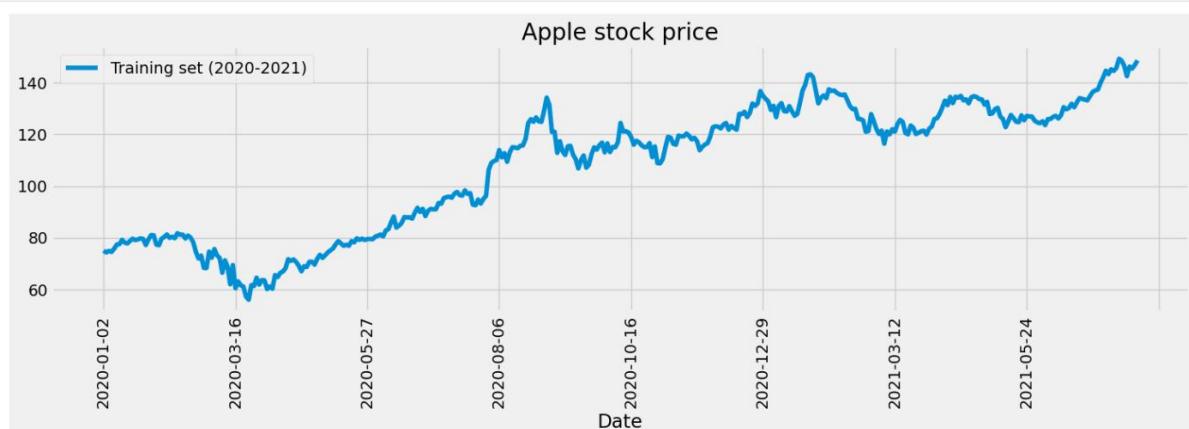
	Open	High	Low	Close	Adj Close	Volume
Date						
2015-01-02	27.847500	27.860001	26.837500	27.332500	24.745996	212818400
2015-01-05	27.072500	27.162500	26.352501	26.562500	24.048864	257142000
2015-01-06	26.635000	26.857500	26.157499	26.565001	24.051125	263188400
2015-01-07	26.799999	27.049999	26.674999	26.937500	24.388372	160423600
2015-01-08	27.307501	28.037500	27.174999	27.972500	25.325430	237458000
...
2021-12-22	173.039993	175.860001	172.149994	175.639999	175.639999	92135300
2021-12-23	175.850006	176.850006	175.270004	176.279999	176.279999	68227500
2021-12-27	177.089996	180.419998	177.070007	180.330002	180.330002	74919600
2021-12-28	180.160004	181.330002	178.529999	179.289993	179.289993	79144300
2021-12-29	179.330002	180.630005	178.139999	179.380005	179.380005	62231200

1761 rows × 6 columns

- Tạo tập train và set theo thời gian và tiến hành chuẩn hoá dữ liệu

```
plt.figure(figsize=(10,5))
Data['Close'][['2020':'2021-07-25']].plot(figsize=(16,4),legend=True)
plt.xticks(rotation=90)
plt.legend(['Training set (2020-2021)'])
plt.title('Apple stock price')
plt.show()
```

Python

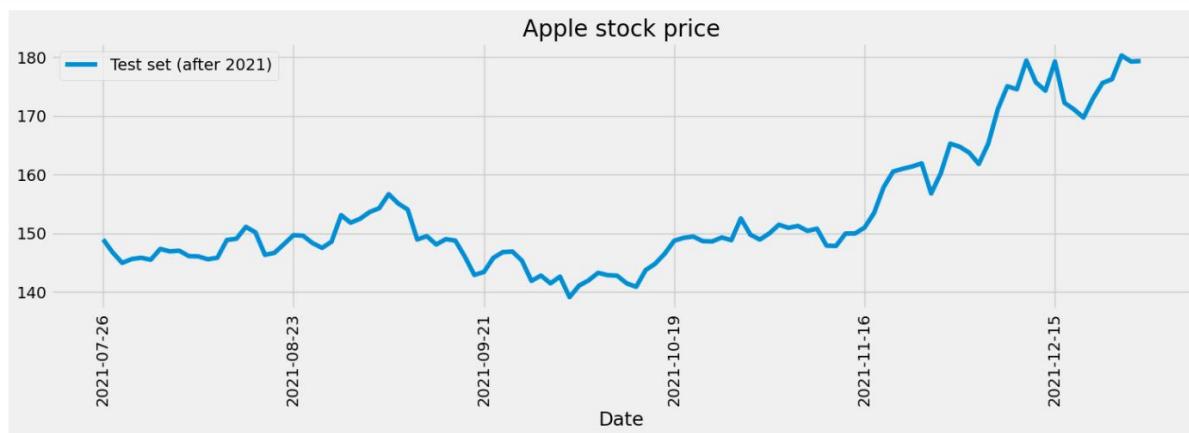


Tạo ra một biểu đồ giá cổ phiếu của Apple trong khoảng thời gian từ năm 2020 đến ngày 25 tháng 7 năm 2021, với chú thích chỉ ra rằng đây là tập dữ liệu huấn luyện.

- Tạo tập test và set theo thời gian và tiến hành chuẩn hóa dữ liệu

```
Data['Close']['2021-07-25':].plot(figsize=(16,4),legend=True)
plt.xticks(rotation=90)
plt.legend(['Test set (after 2021)'])
plt.title('Apple stock price')
plt.show()
```

Python



Tạo ra một biểu đồ giá cổ phiếu của Apple từ ngày 25 tháng 7 năm 2021 đến hiện tại, với chú thích chỉ ra rằng đây là tập test sau năm 2021.

```
#Extract High column data
Train_set=Data['Close']['2020-03-10':'2021-07-25'].values
Test_set=Data['Close']['2021-07-25':].values
S=MinMaxScaler()
scaled_train=S.fit_transform(Train_set.reshape(-1,1))
scaled_test=S.transform(Test_set.reshape(-1,1))
```

Python

Trích xuất dữ liệu cột "Close" từ DataFrame "Data" trong khoảng thời gian từ ngày 10 tháng 3 năm 2020 đến ngày 25 tháng 7 năm 2021. Dữ liệu này được lưu trữ trong biến "Train_set". Và trích xuất dữ liệu cột "Close" từ DataFrame "Data" từ ngày 25 tháng 7 năm 2021 đến hết dữ liệu. Dữ liệu này được lưu trữ trong biến "Test_set".

```
Test_set_to_prediction=np.concatenate([scaled_train[-21:],scaled_test],axis=0)
```

Python

Chuẩn bị dữ liệu để dự đoán trên tập test bằng mô hình Recurrent Neural Network (RNN). Trong đó, RNN hoạt động trên dữ liệu chuỗi, và để dự đoán trên tập test, mô hình cần nhìn thấy dữ liệu từ tập train. Cụ thể, 20 dữ liệu cuối cùng từ tập train sẽ được sử dụng để dự đoán điểm dữ liệu đầu tiên trong tập test.

Dòng mã scaled_train[-21:] lấy 20 dữ liệu cuối cùng từ tập train, và scaled_test là tập test. Hàm np.concatenate được sử dụng để ghép hai tập dữ liệu này lại với nhau theo trục 0 (axis=0), tức là ghép

theo chiều dọc. Kết quả chúng ta nhận được là Test_set_to_prediction, đó là tập dữ liệu đã được chuẩn bị để dự đoán trên tập test.

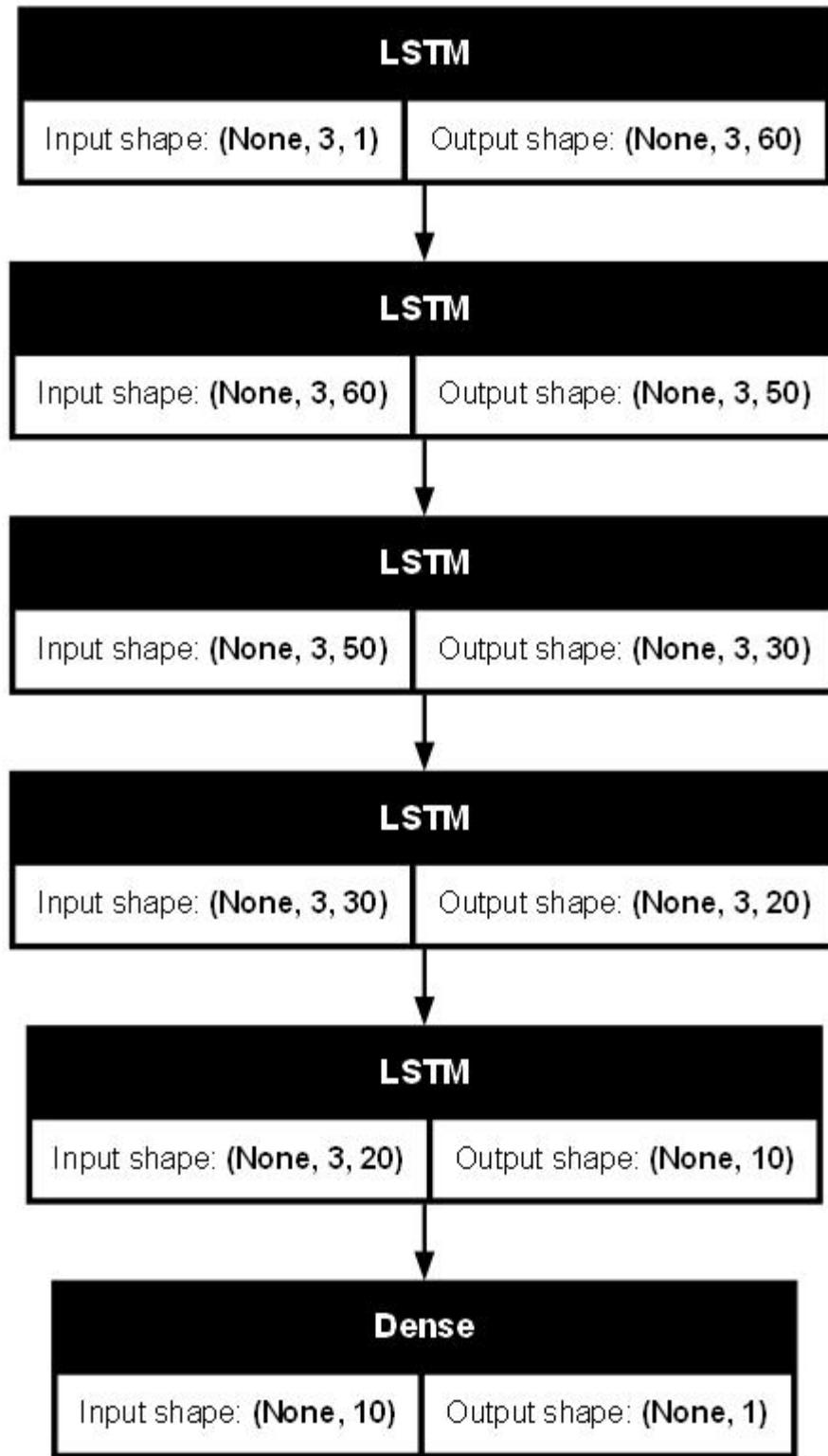
3.2.5.Tạo Model

```
from keras.layers import Input  
  
Model1 = Sequential()  
Model1.add(Input(shape=[1], 1))  
Model1.add(LSTM(units=60, return_sequences=True))  
Model1.add(LSTM(units=50, return_sequences=True))  
Model1.add(LSTM(units=30, return_sequences=True))  
Model1.add(LSTM(units=20, return_sequences=True))  
Model1.add(LSTM(units=10))  
Model1.add(Dense(units=1))
```

Tạo ra một mô hình RNN đã được xây dựng và sẵn sàng để được huấn luyện và sử dụng cho các tác vụ dự đoán dữ liệu chuỗi.

```
plot_model(Model1, show_shapes = True, expand_nested = True, dpi = 80)
```

Vẽ biểu đồ kiến trúc của mô hình Model1



3.2.6. Huấn luyện mô hình

Huấn luyện mô hình Model1 sử dụng dữ liệu X và y trong epochs vòng lặp.

```
callbacks=tf.keras.callbacks.ModelCheckpoint(  
    'Model.keras',  
    monitor= 'loss',  
    save_best_only=True)  
history=Model1.fit(X,y,epochs=350,callbacks=[callbacks])#350
```

```
Epoch 1/350  
11/11 5s 13ms/step - loss: 0.4128  
Epoch 2/350  
11/11 0s 9ms/step - loss: 0.3563  
Epoch 3/350  
11/11 0s 11ms/step - loss: 0.1984  
Epoch 4/350  
11/11 0s 9ms/step - loss: 0.0495  
Epoch 5/350  
11/11 0s 10ms/step - loss: 0.0392  
Epoch 6/350  
11/11 0s 10ms/step - loss: 0.0333  
Epoch 7/350  
11/11 0s 9ms/step - loss: 0.0305  
Epoch 8/350  
11/11 0s 9ms/step - loss: 0.0205  
Epoch 9/350  
11/11 0s 10ms/step - loss: 0.0141  
Epoch 10/350  
11/11 0s 10ms/step - loss: 0.0091  
  
...  
Epoch 349/350  
11/11 0s 6ms/step - loss: 0.0012  
Epoch 350/350  
11/11 0s 11ms/step - loss: 9.6894e-04
```

Output is truncated. View as a [scrollable element](#) or open in a [text editor](#). Adjust cell output [settings](#)...

3.2.7. Dự đoán

```
Output1=Model1.predict(X_test)
```

4/4 ━━━━━━━━ 1s 4ms/step

```
Model1.evaluate(y_test,Output1)
```

4/4 ━━━━━━━━ 1s 2ms/step - loss: 0.6629

0.7107938528060913

```
mean_squared_error(S.inverse_transform(y_test),S.inverse_transform(Outpu1))
```

15.635469961472793

- Sử dụng mô hình "Model1" để dự đoán đầu ra cho dữ liệu kiểm thử "X_test". Kết quả dự đoán được lưu vào biến "Output1".
- Phương thức evaluate() trong mô hình Keras được sử dụng để đánh giá hiệu suất của mô hình.
- Để tính toán mean squared error (MSE) giữa dự đoán của mô hình và nhãn thực tế trên dữ liệu kiểm thử.

3.2.8. Trực quan giá trị dự đoán và thực tế

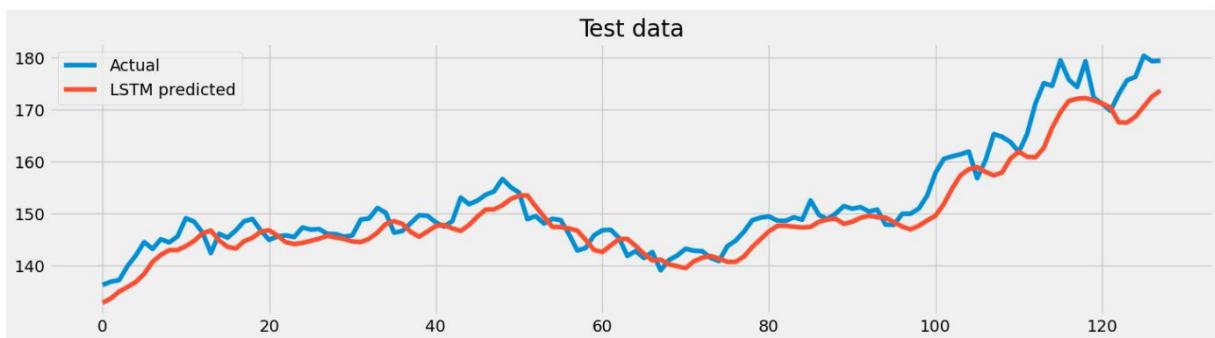
Đồ thị so sánh giữa dữ liệu thực tế và dự đoán của mô hình LSTM trên tập dữ liệu kiểm tra.

```

x=np.arange(0,len(y_test))
plt.figure(figsize=(16,4))
plt.title("Test data", fontsize=20)
plt.plot(x,S.inverse_transform(y_test),label="Actual")
plt.plot(x,S.inverse_transform(Output1),label="LSTM predicted")
plt.legend()
plt.show()

```

Python



Đồ thị so sánh giữa dữ liệu thực tế và dự đoán của mô hình trên tập dữ liệu huấn luyện.

```

q=np.arange(0,len(y))
plt.figure(figsize=(16,4))
plt.title("Training data", fontsize=20)
plt.plot(q,S.inverse_transform(y),label='Actual')
plt.plot(q,S.inverse_transform(Out),label='Predicted')
plt.legend()
plt.show()

```

Python

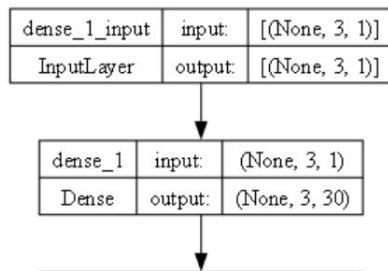


3.2.9. Sử dụng mô hình Simple Dense Layer để so sánh hiệu suất đánh giá của LSTM

Simple Dense Layer

```
In [25]: Model2=Sequential(  
[  
    Dense(units=30,input_shape=(X.shape[1],1),activation='relu'),  
    Dense(units=20,activation='relu'),  
    Dense(units=10),  
    Dense(units=1)  
])  
  
In [26]: plot_model(Model2, show_shapes = True,expand_nested = True,dpi = 80)
```

Out[26]:



Tạo ra một mô hình neural network gồm 4 tầng Dense, và sau đó vẽ ra biểu đồ mô tả cấu trúc của mô hình, giúp người dùng dễ dàng hình dung và hiểu rõ về cấu trúc của mô hình này.

```
In [28]: Model2.compile(optimizer='adam',loss='mse')  
  
In [29]: history2=Model2.fit(X,y,epochs=100,callbacks=[callbacks])
```

```
Epoch 1/100  
11/11 [=====] - 1s 2ms/step - loss: 0.2870  
Epoch 2/100  
11/11 [=====] - 0s 3ms/step - loss: 0.1107  
Epoch 3/100  
11/11 [=====] - 0s 3ms/step - loss: 0.0277  
Epoch 4/100  
11/11 [=====] - 0s 3ms/step - loss: 0.0125  
Epoch 5/100  
11/11 [=====] - 0s 3ms/step - loss: 0.0121  
Epoch 6/100  
11/11 [=====] - 0s 3ms/step - loss: 0.0098  
Epoch 7/100  
11/11 [=====] - 0s 3ms/step - loss: 0.0086  
Epoch 8/100  
11/11 [=====] - 0s 3ms/step - loss: 0.0072  
Epoch 9/100  
11/11 [=====] - 0s 3ms/step - loss: 0.0056  
Epoch 10/100
```

Sau khi thực hiện các bước trên, mô hình Model2 sẽ được huấn luyện sử dụng thuật toán tối ưu hóa Adam và hàm mất mát MSE. Quá trình huấn luyện sẽ diễn ra trong 100 epochs, trong đó các callbacks sẽ được gọi để theo dõi và thực hiện các tác vụ tương ứng. Lịch sử huấn luyện của mô hình sẽ được lưu lại trong biến history2.

Cuối cùng, có thể sử dụng các thông tin trong history2 để đánh giá và theo dõi quá trình huấn luyện của mô hình.

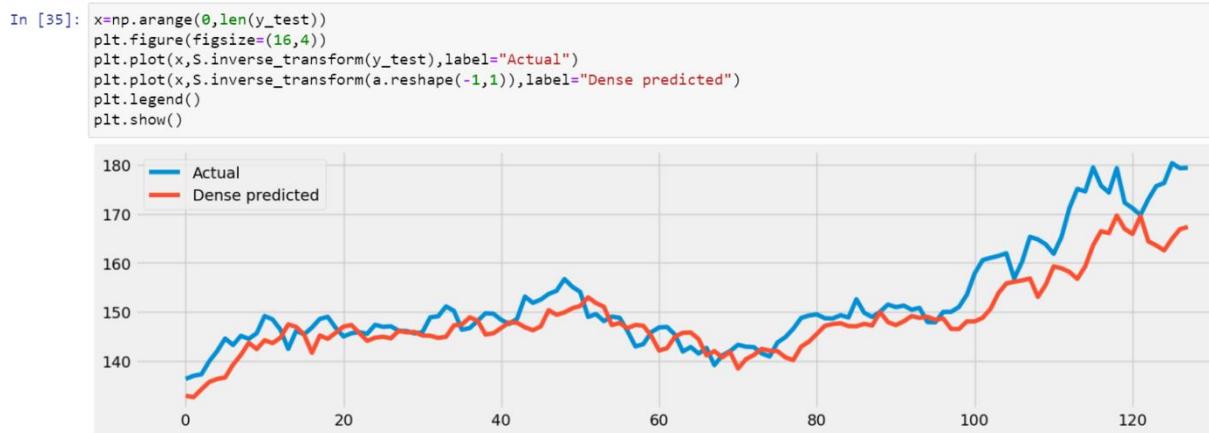
```
In [30]: Output2=Model2.predict(X_test)
l=Output2.shape[0]
a=[]
for i in range(l):
    a.append(Output2[i][0])
a=np.array(a)

4/4 [=====] - 0s 2ms/step

In [31]: mean_squared_error(S.inverse_transform(y_test),S.inverse_transform(a.reshape(-1,1)))

Out[31]: 32.77540584601422
```

Sử dụng mô hình Model2 để dự đoán trên tập dữ liệu kiểm tra X_test. Lấy kết quả dự đoán và lưu vào biến Output2. Tách riêng các giá trị dự đoán từ Output2 và lưu vào một mảng NumPy a.



Vẽ một đồ thị so sánh giữa giá trị thực tế (y_test) và giá trị dự đoán (a) trên tập dữ liệu kiểm tra. Đồ thị sẽ hiển thị đường thực tế và đường dự đoán, cho phép người dùng so sánh và đánh giá kết quả của mô hình.

3.2.10. So sánh 2 mô hình áp dụng và đưa ra kết luận

```
: # Chuyển đổi Output2 về dạng 1D array
1 = Output2.shape[0]
a = []
for i in range(1):
    a.append(Output2[i][0])
a = np.array(a)

# Tính các thông số so sánh
mse1 = mean_squared_error(S.inverse_transform(y_test), S.inverse_transform(Output1))
mse2 = mean_squared_error(S.inverse_transform(y_test), S.inverse_transform(a.reshape(-1,1)))

rmse1 = np.sqrt(mse1)
rmse2 = np.sqrt(mse2)

r2_1 = r2_score(S.inverse_transform(y_test), S.inverse_transform(Output1))
r2_2 = r2_score(S.inverse_transform(y_test), S.inverse_transform(a.reshape(-1,1)))

mae1 = mean_absolute_error(S.inverse_transform(y_test), S.inverse_transform(Output1))
mae2 = mean_absolute_error(S.inverse_transform(y_test), S.inverse_transform(a.reshape(-1,1)))

print("Model LSTM:")
print(f"MSE: {mse1:.3f}")
print(f"RMSE: {rmse1:.3f}")
print(f"R-squared: {r2_1:.3f}")
print(f"MAE: {mae1:.3f}")

print("\nModel Simple Dense Layer :")
print(f"MSE: {mse2:.3f}")
print(f"RMSE: {rmse2:.3f}")
print(f"R-squared: {r2_2:.3f}")
```

Tính các thông số so sánh giữa mô hình "LSTM" và mô hình "Dense"

Kết quả:

Model	MSE	MAE	R2 Score	RMSE
LSTM	19.087	3.583	0.9996985644983746	4.369
Dense Layer	32.775	4.292	0.9999912056936733	5.725

Khi so sánh 2 mô hình này, ta có thể thấy rằng:

MSE và RMSE của Model LSTM thấp hơn so với Model Simple Dense Layer, điều này chỉ ra rằng Model LSTM có độ lỗi trung bình thấp hơn, có nghĩa là dự đoán của nó gần với giá trị thực tế hơn.

Giá trị R-squared của Model LSTM là 0.827, cao hơn so với 0.704 của Model Simple Dense Layer. R-squared là một chỉ số đánh giá mức độ phù hợp của mô hình, giá trị càng cao

(gần 1) thì mô hình càng tốt. Vì vậy, Model LSTM có khả năng giải thích và dự đoán tốt hơn so với Model Simple Dense Layer.

MAE của Model LSTM là 3.583, thấp hơn so với 4.292 của Model Simple Dense Layer. MAE đo lường độ lệch trung bình tuyệt đối giữa giá trị dự đoán và giá trị thực tế, do đó, Model LSTM có độ chính xác cao hơn.

Kết luận:

Dựa trên các chỉ số đánh giá, Model LSTM có hiệu suất tốt hơn so với Model Simple Dense Layer.

Mô hình LSTM thể hiện khả năng dự đoán và giải thích cao hơn, với độ lỗi trung bình thấp hơn và độ chính xác cao hơn.

Do đó, có thể kết luận rằng Model LSTM là mô hình tốt hơn để giải quyết bài toán này.

CHƯƠNG 4: TỔNG KẾT

- Trong nghiên cứu "Phân tích giá cổ phiếu áp dụng LSTM để dự đoán giá cổ phiếu," chúng ta đã tiến hành một cuộc nghiên cứu toàn diện nhằm ứng dụng mô hình LSTM (Long Short-Term Memory) vào việc dự đoán giá cổ phiếu dựa trên dữ liệu lịch sử. Sử dụng tập dữ liệu lịch sử giá cổ phiếu của 4 công ty phổ biến từ Kaggle, chúng ta đã thực hiện các bước tiền xử lý dữ liệu cần thiết, bao gồm làm sạch và chuẩn hóa dữ liệu. Mô hình LSTM đã được huấn luyện và tinh chỉnh để tối ưu hóa khả năng dự đoán.
- Kết quả cho thấy mô hình LSTM có khả năng ghi nhớ và học từ dữ liệu lịch sử, giúp dự đoán giá cổ phiếu với độ chính xác tương đối cao, mặc dù vẫn còn một số sai sót do biến động khó lường của thị trường. Các chỉ số đánh giá như MAE, RMSE và R² đã khẳng định hiệu suất của mô hình. Nghiên cứu này không chỉ minh chứng cho tiềm năng ứng dụng của LSTM trong lĩnh vực tài chính mà còn cung cấp nền tảng cho các nghiên cứu và phát triển tiếp theo.
- Hơn nữa, nghiên cứu đã giúp chúng ta hiểu sâu hơn về cách áp dụng LSTM vào dữ liệu chuỗi thời gian, từ đó mở ra nhiều hướng phát triển mới như tích hợp thêm các yếu tố kinh tế vĩ mô và dữ liệu tin tức để cải thiện chất lượng dự đoán. Tóm lại, việc áp dụng LSTM trong dự đoán giá cổ phiếu đã cho thấy tiềm năng to lớn trong việc hỗ trợ các nhà đầu tư và nhà phân tích tài chính ra quyết định chính xác hơn, đồng thời góp phần vào sự phát triển của các công nghệ dự đoán hiện đại trong lĩnh vực tài chính.

CHƯƠNG 5: TÀI LIỆU THAM KHẢO

[1] LSTM là gì?, Hai's blog, 20/10/2017, <https://dominhhai.github.io/vi/2017/10/what-is-lstm/>

[2] Machine Learning to Predict Stock Prices , Roshan Adusumilli, Dec 26, 2019,
<https://towardsdatascience.com/predicting-stock-prices-using-a-keras-lstm-model-4225457f0233>

[3] Stock Market Predictions with LSTM in Python, DataCamp, 2020 Januarary,
<https://www.datacamp.com/tutorial/lstm-python-stock-market>

[4] LSTM for Apple stock prediction, kaggle, <https://www.kaggle.com/code/anikbhowmick/lstm-for-apple-stock-prediction>

