TRƯỜNG ĐẠI HỌC KINH TẾ - LUẬT
ĐẠI HỌC QUỐC GIA TP.HCM

ĐỒ ÁN CUỐI KỲ:
ỨNG DỤNG PHÂN TÍCH KỸ THUẬT, MACHINE LEARNING VÀ DEEP LEARNING VÀO DỰ ĐOÁN GIÁ CHỨNG KHOÁN

Giảng viên: Ngô Phú Thanh
Bộ môn: Gói phần mềm ứng dụng cho tài chính 1
Họ và tên: Phạm Duy Tùng
MSSV: K214140961

TP. Hồ Chí Minh, ngày 10 tháng 01 năm 2023

ỨNG DỤNG PHÂN TÍCH KỸ THUẬT, MACHINE LEARNING VÀ DEEP LEARNING VÀO DỰ ĐOÁN GIÁ CHỨNG KHOÁN

I. Thông tin chung

Tên đề tài: Ứng dụng phân tích kỹ thuật, machine learning và deep learning vào dự đoán giá chứng khoán.

Ngôn ngữ lập trình: Python

Nền tảng lập trình: Jupyter Notebook

Các file nằm trong thư mục:

+ file ipynb: Stock Prediction Models.ipynb (file source code chính)

+ file pdf: README.pdf (file diễn giải và trình bài kết quả nghiên cứu)

+ file csv: microsoft_data.csv (file dữ liệu)

Giới thiệu:

Trong đề tài này, nhiều mô hình khác nhau sẽ được sử dụng để dự báo giá Đóng cửa điều chỉnh cho dữ liệu giá cổ phiếu của Tập đoàn Microsoft. Từ các chỉ số phân tích kỹ thuật đơn giản (như đường Trung bình động – Moving Average), thuật toán Hồi quy tuyến tính (Linear Regression), các mô hình máy học từ đơn giản (k-Nearest Neighbor) đến phức tạp hơn (Prophet), và đến cuối cùng là thuật toán học sâu (LSTM - Long Short-Term Memory). Các phương pháp sẽ được ứng dụng lần lượt và so sánh kết quả để tìm ra được đâu là mô hình mang lại hiệu quả tối ưu nhất.

Dữ liệu:

Dữ liệu trong bài làm là dữ liệu giá đóng cửa điều chỉnh của Tập đoàn Microsoft, với interval cho mỗi điểm dữ liệu là 1 ngày, được lấy trong giai đoạn từ ngày 01/01/2010 đến ngày 10/01/2023 (là thời điểm mới nhất). Dữ liệu được lấy trực tiếp từ Yahoo Finance thông qua API của yfinance.

Các mô hình trong bài làm có thể được áp dụng đối với bất kỳ dữ liệu cổ phiếu nào được niêm yết công khai.

II. Nội dung

Install/import thư viện:

Các thư viện được sử dụng: yfinance, prophet, tensorflow, numpy, pandas, matplotlib, seaborn, re, csv, statsmodel, sklearn, datetime, warnings, keras.

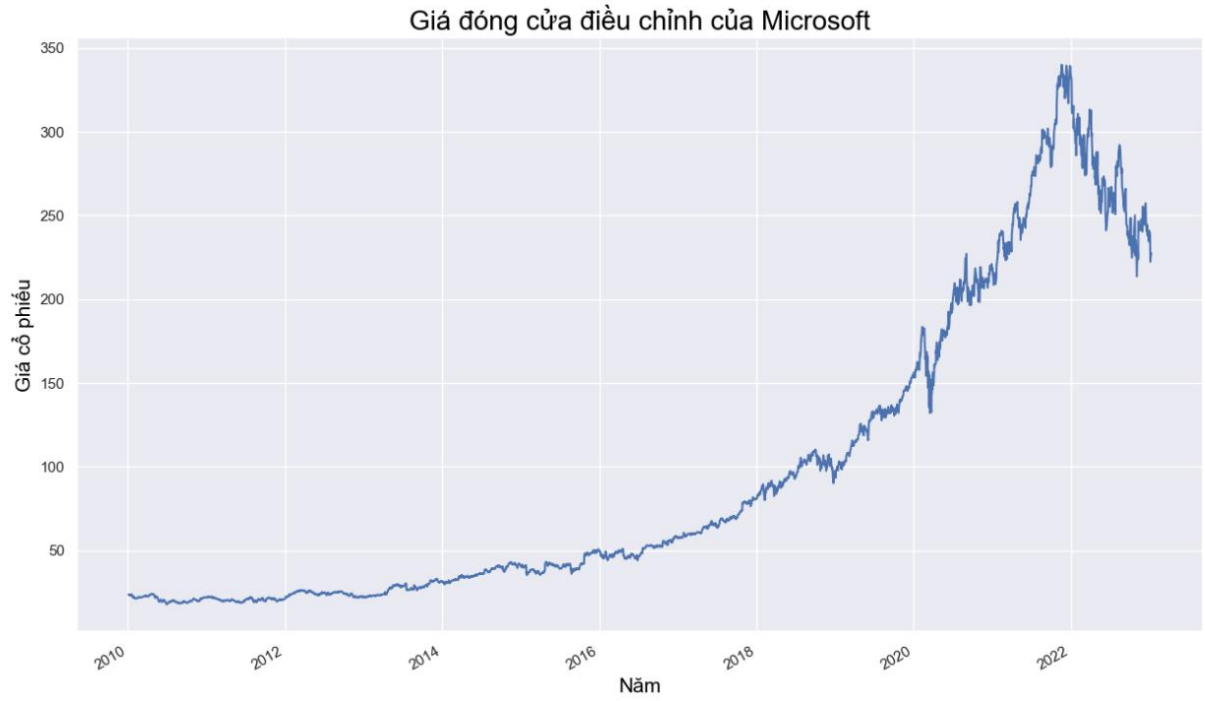
Import dữ liệu:

```
In [9]: msft_df = yf.download("MSFT", start="2010-01-01", end="2023-01-10")[['Adj Close','Open', 'High', 'Low', 'Close', 'Volume']].round
        msft_df.to_csv("microsoft_data.csv")

[**********100%**********]  1 of 1 completed

In [10]: msft_df
```

Out[10]:

Date	Adj Close	Open	High	Low	Close	Volume
2010-01-04	23.68	30.62	31.10	30.59	30.95	38409100
2010-01-05	23.69	30.85	31.10	30.64	30.96	49749600
2010-01-06	23.55	30.88	31.08	30.52	30.77	58182400
2010-01-07	23.30	30.63	30.70	30.19	30.45	50559700
2010-01-08	23.46	30.28	30.88	30.24	30.66	51197400
...	...	...	...	...	...	...
2023-01-03	239.58	243.08	245.75	237.40	239.58	25740000
2023-01-04	229.10	232.28	232.87	225.96	229.10	50623400
2023-01-05	222.31	227.20	227.55	221.76	222.31	39585600
2023-01-06	224.93	223.00	225.76	219.35	224.93	43597700

Hiện thị dữ liệu giá đóng cửa điều chỉnh của Microsoft giai đoạn 01/01/2010 – 10/01/2023.



1. Đường trung bình động – Moving Average

Đường trung bình động là một chỉ số phân tích kỹ thuật đơn giản, theo đó quan sát tiếp theo là giá trị trung bình của tất cả các quan sát trong quá khứ tạo ra một mức giá trung bình được cập nhật liên tục.

In [19]:
```
# Tạo DataFrame cho Giá đóng cửa điều chỉnh

msft_adj = msft_df[['Adj Close']]
msft_adj
```

Out[19]:

Date	Adj Close
2010-01-04	23.68
2010-01-05	23.69
2010-01-06	23.55
2010-01-07	23.30
2010-01-08	23.46
...	...
2023-01-03	239.58
2023-01-04	229.10
2023-01-05	222.31
2023-01-06	224.93
2023-01-09	227.43

3277 rows × 1 columns

Split dữ liệu

Chúng ta sẽ chia dữ liệu thành train set và test set để xác minh kết quả dự đoán. Vì đây là dữ liệu dạng Time series nên ta không thể chia ngẫu nhiên được. Vậy nên là ta sẽ chia 80% đầu là train set và 20% còn lại là test set.

Bộ dữ liệu được phân chia với set data từ 04/01/2010 – 02/06/2020 là train set và từ 03/06/2020 – 09/01/2023 là test set

In [23]:
```
train.index.min(), train.index.max(), test.index.min(), test.index.max()
```

Out[23]:
```
(Timestamp('2010-01-04 00:00:00'),
 Timestamp('2020-06-02 00:00:00'),
 Timestamp('2020-06-03 00:00:00'),
 Timestamp('2023-01-09 00:00:00'))
```

Chạy dự đoán cho dataset và tính chỉ số RMSE để so sánh kết quả với các mô hình khác

Tạo dự đoán cho test set

In [24]:
```
# Tạo dự đoán
preds = []
for i in range(0,656):
    a = train[len(train)-656+i:].sum() + sum(preds)
    b = a/656
    preds.append(b)
```

Tính chỉ số RMSE

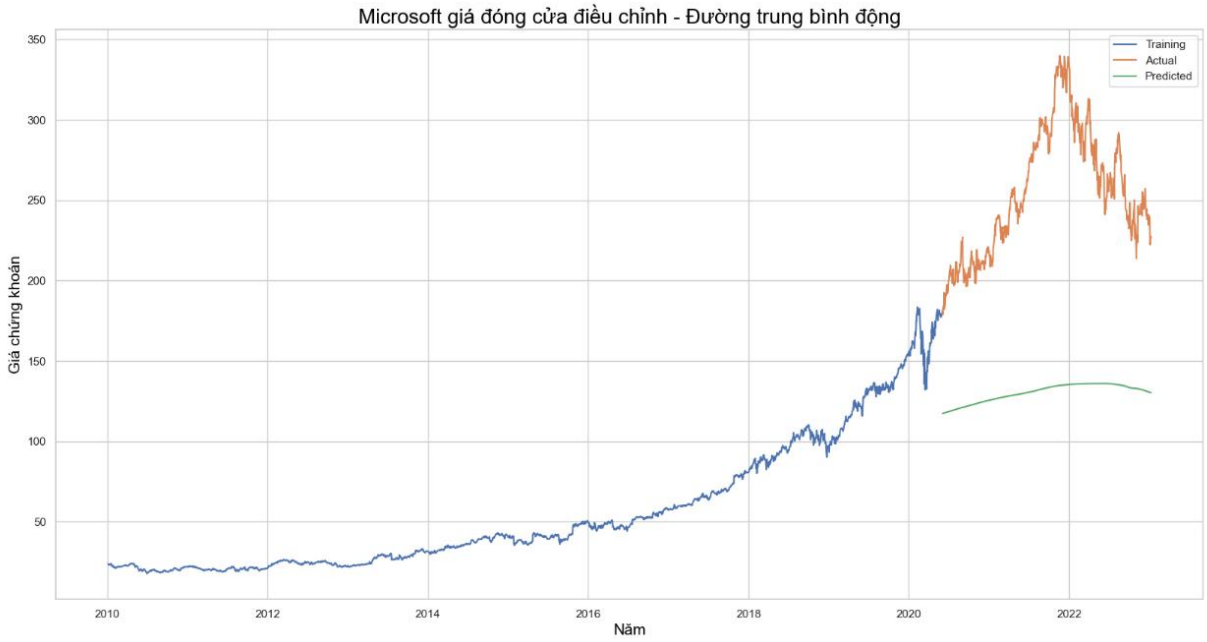
Chúng ta sẽ đánh giá mô hình bằng cách tính toán chỉ số RMSE hoặc Độ lệch gốc-trung bình-bình phương (Root-mean-square deviation), một thước đo tốt để xem xét mức độ chính xác của các mô hình dự đoán phản hồi.

In [26]:
```
# Tính rmse
rmse = np.sqrt(np.mean(np.power((np.array(test)-preds),2)))
rmse
```

Out[26]:
```
129.66982647342456
```

Chúng ta sẽ đánh giá mô hình bằng cách tính toán chỉ số RMSE hoặc Độ lệch gốc-trung bình-bình phương (Root-mean-square deviation), một thước đo tốt để xem xét mức độ chính xác của các mô hình dự đoán phản hồi. Chỉ số RSME càng nhỏ đồng nghĩa với việc tồn tại sai số nhỏ, chứng tỏ hiệu quả càng cao của mô hình. Chỉ số RSME của Đường trung bình động trả về là **129.66982647342456**.

Trực quan hóa các dữ liệu dự đoán và thực tế



Với đường màu xanh lam là dữ liệu training, đường màu cam là dữ liệu thực test và đường màu xanh lục là đường trung bình động.

Các giá trị dự đoán ban đầu cho thấy xu hướng tăng, sau đó chững lại và bắt đầu giảm dần.

2. Hồi quy tuyến tính - Linear Regression

Mô hình này áp dụng cách tiếp cận tuyến tính để mô hình hóa mối quan hệ giữa biến phụ thuộc và (các) biến độc lập, đồng thời cũng là hình thức học máy đơn giản nhất.

Feature engineering (tạm dịch: Trích chọn đặc trưng)

Giả thuyết cho rằng, những ngày đầu tiên và cuối cùng trong tuần có khả năng ảnh hưởng nhiều đến giá đóng cửa điều chỉnh. Vì vậy chúng ta sẽ thêm một tính năng để phân loại các ngày dựa vào ngày trong tuần của chúng.

Nếu số ngày là 0 (thứ hai) hoặc 4(thứ sáu) thì cột ghi chú sẽ hiện thị 1, và ngược lại, nếu ngày rơi vào thứ ba, tư, năm, cột hiện thị sẽ ghi là 0.

In [35]:
```
# Tạo tính năng
msft_adj['mon_fri'] = 0
for i in range(0,len(msft_adj)):
    if (msft_adj['Dayofweek'][i] == 0 or msft_adj['Dayofweek'][i] == 4):
        msft_adj['mon_fri'][i] = 1
    else:
        msft_adj['mon_fri'][i] = 0
```

Split data thành tập train và test

Cũng giống như mô hình trên, chúng ta sẽ chọn tỉ lệ 80%/20% cho tập train và test của bộ dữ liệu

Split data thành train và test set

```
In [36]: split = int(0.8*len(msft_adj))

In [37]: train, test = msft_adj[:split], msft_adj[split:]

In [38]: msft_adj.shape, train.shape, test.shape
Out[38]: ((3277, 14), (2621, 14), (656, 14))

In [39]: X_train = train.drop('Adj Close', axis=1)
         y_train = train['Adj Close']
         X_test = test.drop('Adj Close', axis=1)
         y_test = test['Adj Close']

In [40]: X_train.shape, y_train.shape, X_test.shape, y_test.shape
Out[40]: ((2621, 13), (2621,), (656, 13), (656,))
```

Tạo, train và test mô hình Hồi quy tuyến tính

Tạo và train mô hình Hồi quy tuyến tính

```
In [41]: # Tạo model
         model = LinearRegression()

         # Train model
         model.fit(X_train,y_train)
Out[41]: LinearRegression()
```

Test model

```
In [42]: # Tạo dự báo
         preds = model.predict(X_test)
```

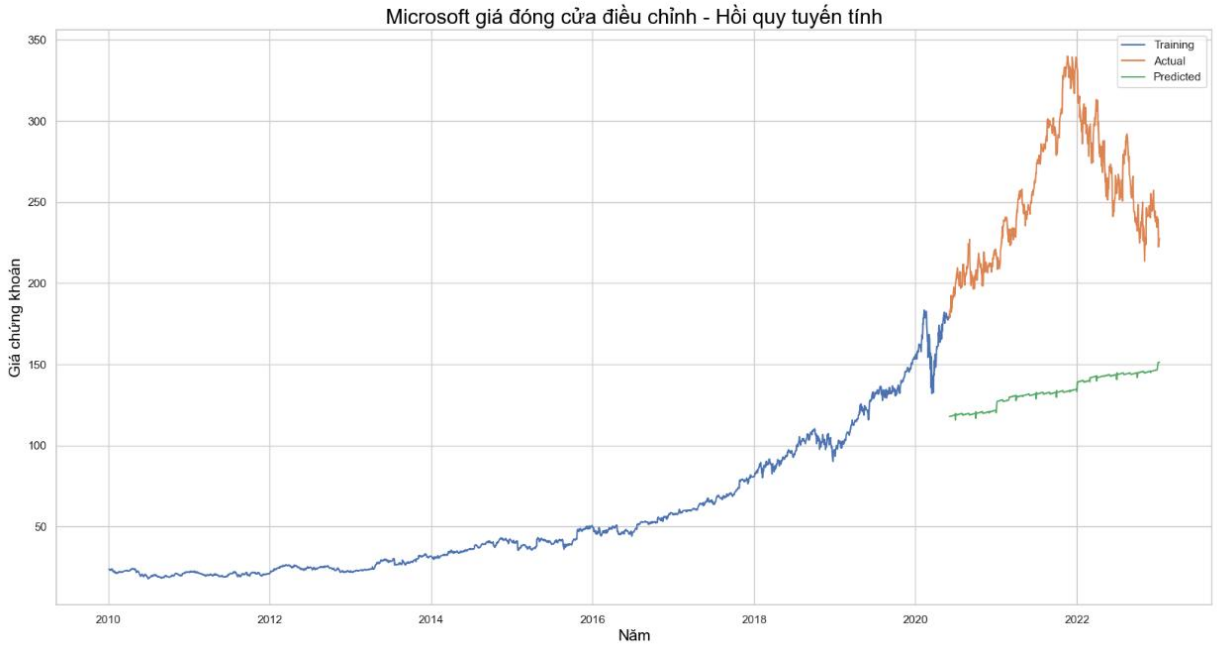
Tính chỉ số RSME của mô hình

Tính chỉ số RMSE

```
In [43]: rmse = np.sqrt(np.mean(np.power((np.array(y_test)-np.array(preds)),2)))
         rmse
Out[43]: 126.67440240138559
```

Chỉ số RSME của mô hình Hồi quy tuyến tình trả về là **126.67440240138559** (thấp hơn so với Đường trung bình động).

Trực quan hóa các dữ liệu dự đoán và thực tế



Giá trị RMSE thấp hơn so với Đường trung bình động, nhưng đồ thị cho thấy mô hình Hồi quy tuyến tính vẫn có vẻ không quá hứa hẹn.

3. k-Nearest Neighbours

Dựa trên các biến độc lập, k-Nearest Neighbors tìm thấy sự giống nhau giữa các điểm dữ liệu mới, cũ và giả định rằng các điểm dữ liệu tương tự tồn tại ở gần nhau. Nó chọn k cho "hàng xóm" gần nhất và sau đó dựa trên những "hàng xóm" này để dự đoán một giá trị cho quan sát mới. k-Nearest Neighbours được biết đến như là một mô hình machine learning đơn giản.

Để có thể chạy được mô hình này, trước hết chúng ta phải Normalisation (Tiêu chuẩn hoá dữ liệu) các thông số của dataset thành tỉ lệ để các dữ liệu đều nằm trong miền có giá trị từ 0 đến 1. Sau đó thực hiện GridSearch để tìm ra tham số tối ưu cho model.

Feature scaling (tạm dịch: Scaling trích chọn)

```
In [46]: # Normalisation - thay đổi tỷ lệ dữ liệu để tất cả các giá trị nằm trong phạm vi 0 và 1
         scaler = MinMaxScaler(feature_range=(0, 1))

         X_train_scaled = scaler.fit_transform(X_train)
         X_train = pd.DataFrame(X_train_scaled)
         X_test_scaled = scaler.transform(X_test)
         X_test = pd.DataFrame(X_test_scaled)
```

GridSearch tìm tham số tối ưu

```
In [47]: # Tạo thư viện cho tham số
         params = {'n_neighbors':[2,3,4,5,6,7,8,9]}
         # Tạo model
         knn = KNeighborsRegressor()
         # Dùng GridSearch để tìm tham số tối ưu
         model = GridSearchCV(knn, params, cv=5)
```

Train, test mô hình và tính chỉ số RSME

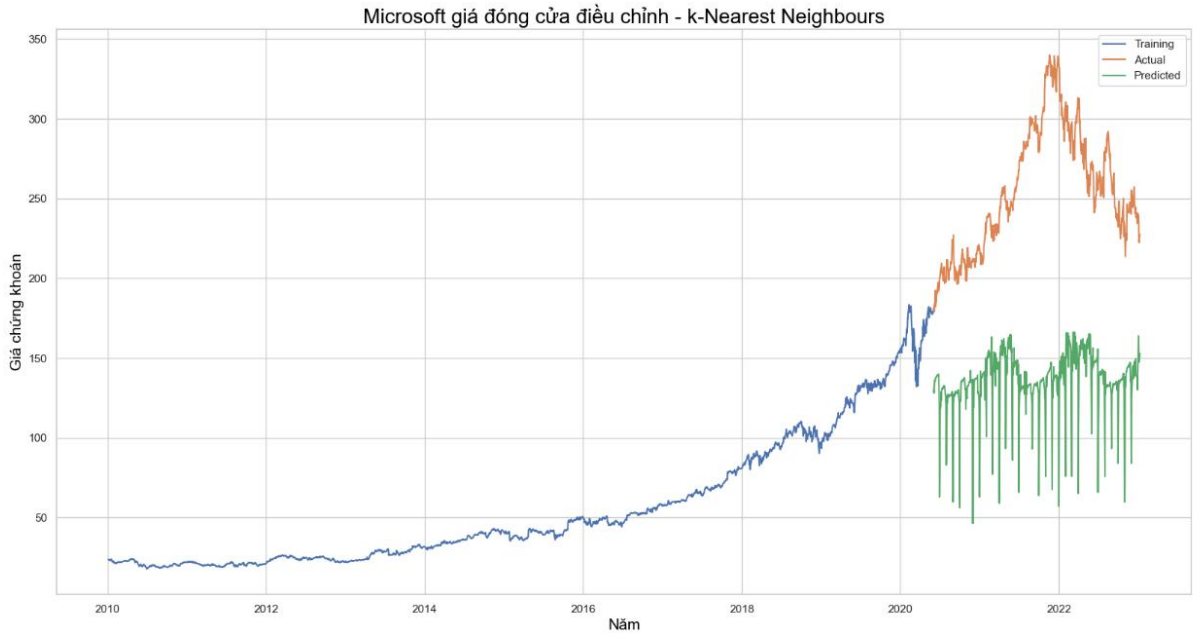
Train và test model

```
In [48]: # Fit model và tạo dự đoán
         model.fit(X_train,y_train)
         preds = model.predict(X_test)
```

Tính chỉ số RMSE

```
In [50]: rmse = np.sqrt(np.mean(np.power((np.array(y_test)-np.array(preds)),2)))
         rmse
Out[50]: 125.69569336103844
```

Mô hình k-Nearest Neighbours trả về giá trị RMSE là **125.69569336103844** (thấp hơn cả Đường trung động và Hồi quy tuyến tính nhưng không đáng kể)



Từ ba biểu đồ trên, có vẻ như các thuật toán hồi quy không thực sự hoạt động tốt trên tập dữ liệu chuỗi thời gian này.

Bằng cách sử dụng dữ liệu trong quá khứ để phân tích và hiểu được các hình mẫu trong chuỗi thời gian, các mô hình đã nắm bắt được xu hướng ngày càng tăng trong chuỗi và đạt đến được RMSE thấp nhất cho đến hiện tại. Tuy nhiên tỉ lệ sai số vẫn còn khá cao và hiệu suất vẫn còn thấp.

4. Prophet

Tiếp theo, chúng ta sẽ sử dụng một mô hình phân tích và dự đoán của dữ liệu chuỗi thời gian có tính đến cả xu hướng và tính thời vụ của tập dữ liệu. Prophet là một gói dự báo dữ liệu chuỗi thời gian dựa trên mô hình cộng trong đó các xu hướng phi tuyến tính phù hợp với tính thời vụ hàng năm, hàng tuần và hàng ngày, cộng với cả các tác nhân ảnh hưởng bên ngoài như các dịp đặc biệt.

Dữ liệu đầu vào của Prophet được cài đặt là một khung dữ liệu có hai cột: ds và y. Cột ds (datestamp) phải được định dạng theo pandas là YYYY-MM-DD cho điểm dữ liệu. Cột y là số và đại diện cho phép đo mà chúng ta muốn dự báo.

Chuẩn bị và định dạng dữ liệu

```
In [52]: # Tạo data set với cột Giá đóng cửa điều chỉnh và cột ngày
         msft_adj = pd.DataFrame(index=range(0,len(msft_df)),columns=['Date', 'Adj Close'])

         for i in range(0,len(msft_df)):
              msft_adj['Date'][i] = msft_df.index[i]
              msft_adj['Adj Close'][i] = msft_df['Adj Close'][i]

In [53]: # Định dạng ngày
         msft_adj['Date'] = pd.to_datetime(msft_adj.Date,format='%Y-%m-%d')
         msft_adj.index = msft_adj['Date']

In [54]: # Định dạng dữ liệu
         msft_adj.rename(columns={'Adj Close': 'y', 'Date': 'ds'}, inplace=True)
```

Split data thành train và test sets

Như thường lệ chúng ta vẫn sẽ chia tập dữ liệu thành 80% cho training set và 20% cho test set.

Split data thành train và test sets

```
In [56]: split = int(0.8*len(msft_adj))

In [57]: train, test = msft_adj[:split], msft_adj[split:]

In [58]: msft_adj.shape, train.shape, test.shape
Out[58]: ((3277, 2), (2621, 2), (656, 2))
```

Tạo, train, test model và tính giá trị chỉ số RSME

Tạo và train model

```
In [59]: # Tạo mô hình Prophet
         model = Prophet()
         # Fit model
         model.fit(train)

         05:15:27 - cmdstanpy - INFO - Chain [1] start processing
         05:15:27 - cmdstanpy - INFO - Chain [1] done processing
Out[59]: <prophet.forecaster.Prophet at 0x1b7ae3d7130>
```

Test model

Sau đó, dự đoán được thực hiện trên DataFrame với cột **ds** chứa ngày dự đoán sẽ được thực hiện.

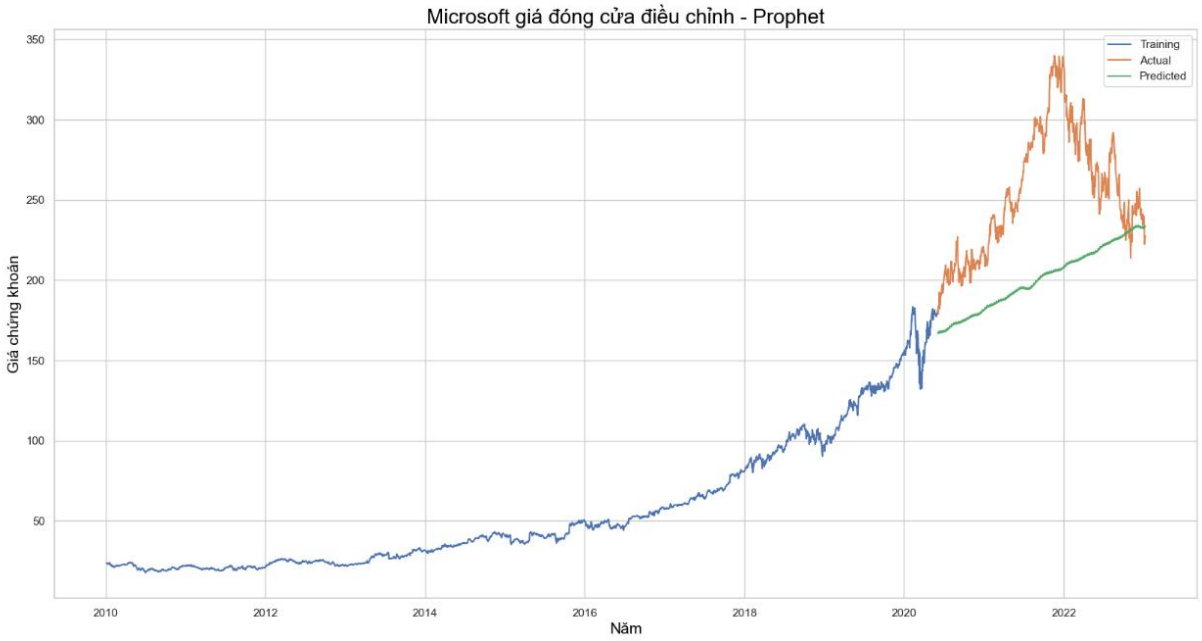
```
In [60]: # Tạo dự báo
         adj_close_prices = model.make_future_dataframe(periods=len(test))
         forecast = model.predict(adj_close_prices)
```

Tính chỉ số RMSE

```
In [61]: forecast_test = forecast['yhat'][split:]
         rmse = np.sqrt(np.mean(np.power((np.array(test['y'])-np.array(forecast_test)),2)))
         rmse
Out[61]: 63.79123610077194
```

Mô hình được khởi tạo và training, sau đó đưa ra dự đoán. Chỉ số RSME của mô hình Prophet được trả về là **63.79123610077194** (thấp hơn đáng kể so với các mô hình hồi quy trước, giảm hơn một nửa, tăng gấp đôi hiệu suất dự đoán).

Trực quan hóa các dữ liệu dự đoán và thực tế



Đặc tính của giá cổ phiếu đó là giá của cổ phiếu sẽ biến động phụ thuộc nhiều vào các lực lượng ảnh hưởng của thị trường, vì vậy thường là chúng sẽ không có một xu hướng thực sự cụ thể trong thời gian dài hay có tính thời vụ. Do đó, các kỹ thuật dự báo như Prophet (hay SARIMA ít nhiều có tính tương đồng) thường sẽ không mang lại kết quả quá khả quan khi áp dụng cho vấn đề này, mặc dù rằng chúng vẫn thể hiện tốt hơn nhiều so với các mô hình hồi quy truyền thống.

Tuy nhiên, nói như vậy không có nghĩ là Prophet hoàn toàn không thể dự đoán được các dạng dữ liệu thời gian không có tính thời vụ hay xu hướng. Thư viện Prophet cung cấp cho người dùng một thuật toán gọi là prophet_make_future_dataframe, trong đó mô hình sẽ dự đoán dữ liệu trong 365 ngày tiếp theo, với các mốc changepoint (dữ liệu thay đổi) trong quá khứ. Tuy không thực sự mang tính thời vụ đều đặn, nhưng khi dự đoán dữ liệu mới qua các mốc thời gian tương ứng ở những năm trước này, Prophet sẽ cân nhắc chúng vào các biến số và ước lượng để cho ra một kết quả dự đoán chính xác hơn.

In [63]:
```
future = model.make_future_dataframe(periods=365)
future.tail()
```

Out[63]:

	ds
2981	2021-05-29
2982	2021-05-30
2983	2021-05-31
2984	2021-06-01
2985	2021-06-02

In [64]:
```
# Tạo dự đoán cho 365 ngày trong tương lai
forecast = model.predict(future)
```

Mô hình dự đoán sẽ gán cho mỗi hàng trong tương lai một giá trị dự đoán và đặt tên nó là yhat. Nếu mô hình chạy vượt qua các ngày này trong lịch sử, nó sẽ cung cấp một sự điều chỉnh đối với mẫu. Đối tượng dự báo ở đây là một DataFrame mới bao gồm một cột yhat với dự báo, cũng như các cột cho các thành phần và các khoảng không chắc chắn.

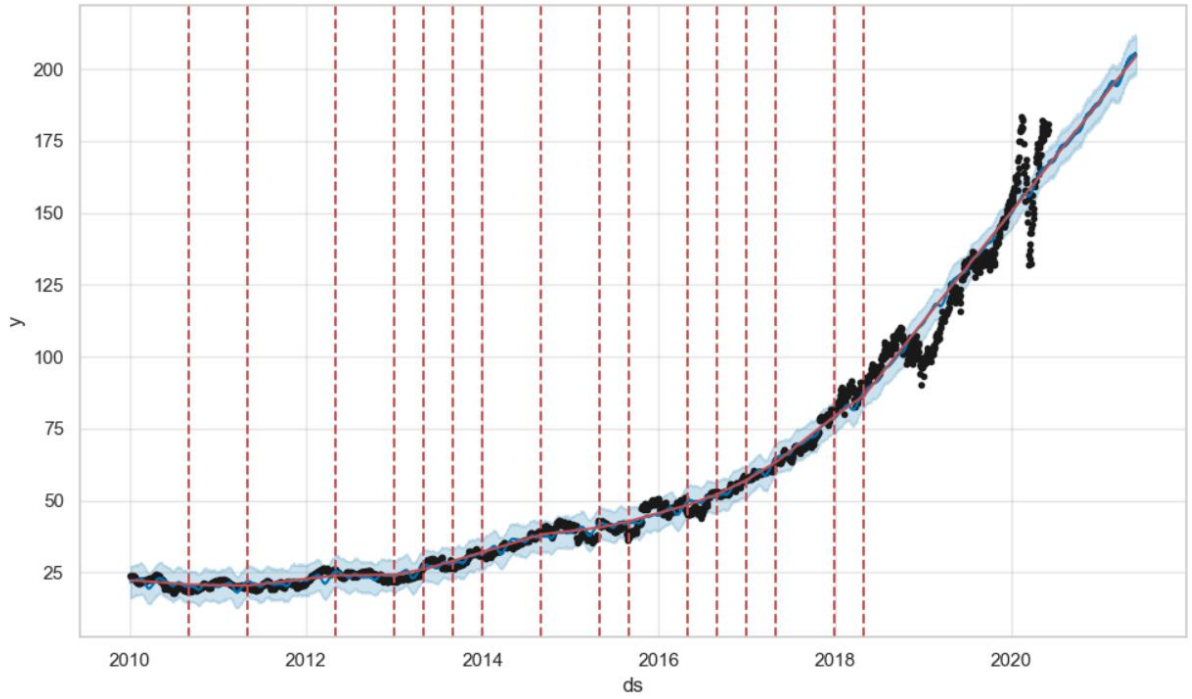
```
In [65]: # Print dự đoán DataFrame

         forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

Out[65]:

	ds	yhat	yhat_lower	yhat_upper
2981	2021-05-29	205.614836	198.838041	212.211196
2982	2021-05-30	205.687877	199.165953	211.985605
2983	2021-05-31	204.734041	197.949175	211.539860
2984	2021-06-01	204.925579	198.518456	211.076302
2985	2021-06-02	204.915615	198.221503	211.161677

Trực quan hóa dự báo và các điểm thay đổi xu hướng

Dữ liệu chuỗi thời gian thường có những thay đổi đột ngột trong khuynh hướng chuyển động của chúng. Vì vậy, Prophet sẽ tự động phát hiện ra những điểm thay đổi này và sẽ cho phép điều chỉnh các xu hướng một cách thích hợp hơn.



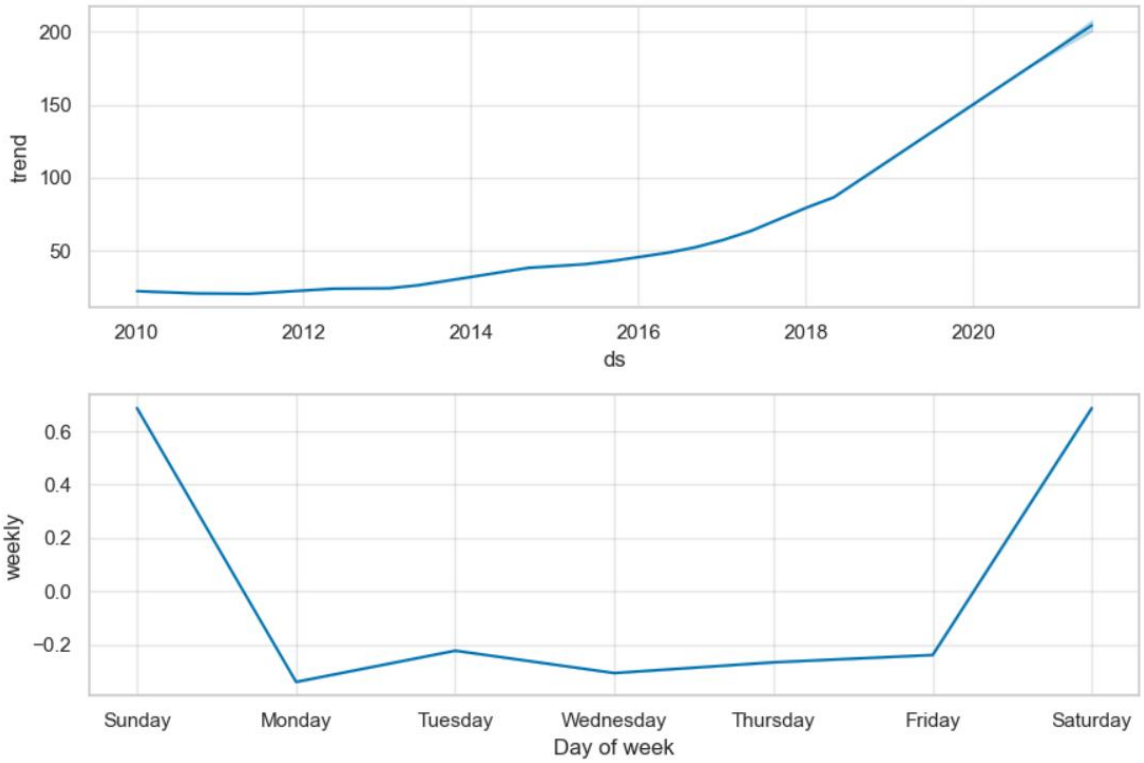
Trích xuất các điểm thay đổi

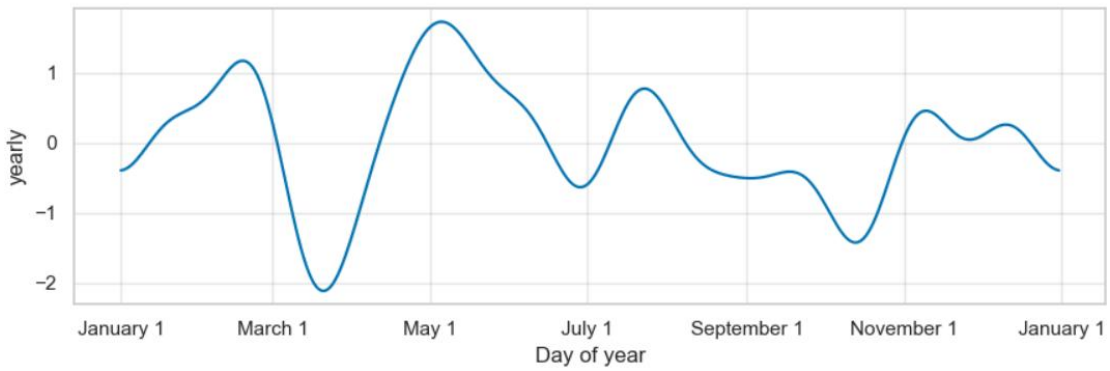
Những điểm dưới đây thể hiện các mốc thay đổi xu hướng của dữ liệu

```
In [67]: model.changepoints
```

```
Out[67]: 84     2010-05-05
         168    2010-09-02
         251    2010-12-31
         335    2011-05-03
         419    2011-08-31
         503    2011-12-30
         587    2012-05-02
         670    2012-08-29
         754    2013-01-02
         838    2013-05-03
         922    2013-09-03
         1006   2014-01-02
         1089   2014-05-02
         1173   2014-09-02
         1257   2014-12-31
         1341   2015-05-04
         1425   2015-09-01
         1508   2015-12-30
         1592   2016-05-02
         1676   2016-08-30
         1760   2016-12-29
         1844   2017-05-02
         1927   2017-08-29
         2011   2017-12-28
         2095   2018-05-01
         Name: ds, dtype: datetime64[ns]
```

Trực quan hoá các thành phần của dự báo





Dựa trên ước lượng về các xu hướng thì có thể thấy từ theo biểu đồ con hằng năm, thông thường giá cổ phiếu Microsoft sẽ chạm đáy vào cuối tháng 3 và chạm đỉnh vào đầu tháng 5. Khối lượng giao dịch của cổ phiếu cũng tập trung nhiều vào đầu tuần và cuối tuần khi sàn vừa mở và đóng cửa, như giải thuyết của chúng ta lúc đầu. Và ở phía trên cùng, biểu đồ con ds cho thấy xu hướng của giá cổ phiếu tăng đều đặn trong thời gian dự đoán.

5. Long Short-Term Memory (LSTM)

LSTM là một Recurrent Neural Network (tạm dịch: Mạng thần kinh tái phát) có thể lưu trữ những thông tin quan trọng trong quá khứ và quên đi những thông tin không quan trọng. Nó sử dụng các quy trình bộ nhớ ngắn hạn để tạo ra bộ nhớ dài hơn và giới thiệu khái niệm về cổng để kiểm soát luồng thông tin trong mạng lưới bằng cách có cơ chế cổng đầu vào, đầu ra và cổng quên.

Split data thành train và test set

Chúng ta vẫn tiếp tục chia mô hình thành train và test set với tỉ lệ 80%/20%

Split data thành train và test set

```
In [71]: split = int(0.8*len(msft_adj_arr))

In [72]: train, test = msft_adj_arr[:split], msft_adj_arr[split:]

In [73]: train.shape, test.shape
Out[73]: ((2621, 1), (656, 1))
```

Feature scaling (tạm dịch: Scaling trích chọn)

Tiêu chuẩn hoá dữ liệu bằng cách scale dữ liệu thành định dạng từ 0 đến 1 để cải thiện việc học và tính hội tụ của mô hình.

Feature scaling (tạm dịch: Scaling trích chọn)

Tiêu chuẩn hoá dữ liệu bằng cách scale dữ liệu thành định dạng từ 0 đến 1 để cải thiện việc học và tính hội tụ của mô hình.

```
In [74]: # Scaling trích chọn và fit dữ liệu scale
         scaler = MinMaxScaler(feature_range=(0, 1))
         scaled_data = scaler.fit_transform(msft_adj_arr)
```

Tạo dữ liệu training

Vì mô hình LSTM yêu cầu những đặc tính cụ thể của dữ liệu đầu vào, nên chúng ta phải xử lý chúng trước khi đưa dữ liệu vào mô hình.

Tạo dữ liệu training

In [75]:
```
# Tạo cấu trúc dữ liệu với 60 time-steps và 1 output

X_train, y_train = [], []
for i in range(60,len(train)):
    X_train.append(scaled_data[i-60:i,0])
    y_train.append(scaled_data[i,0])
```

In [76]:
```
# Chuyển X_train và y_train thành numpy arrays cho mô hình LSTM training

X_train, y_train = np.array(X_train), np.array(y_train)
```

In [77]:
```
# Chuyển dạng dữ liệu thành 3D theo định dạng yêu cầu của LSTM (samples, time steps, features)

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

Out[77]:
```
(2561, 60, 1)
```

Tạo và train model

Chúng ta sẽ triển khai một mô hình đơn giản bao gồm một lớp ẩn với 50 nơ-ron, hình dạng dữ liệu đầu vào với số bước thời gian (number of time steps) là (60) và số chiều (dimensionality) là (1) và một lớp đầu ra với bước thời gian là 1. Mô hình sẽ được biên dịch bằng Mean Squared Error (MSE) và trình tối ưu hóa ADAM (Adaptive Moment Estimation), sau đó sẽ được fit với training set với một epoch và batch size là một.

In [78]:
```
# Tạo và fit mạng LSTM
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1],1)))
model.add(LSTM(units=50))
model.add(Dense(1))

model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(X_train, y_train, epochs=1, batch_size=1, verbose=2)

# Dự đoán 504 giá trị, sự dụng 60 từ tập data train
inputs = msft_adj_arr [len(msft_adj_arr) - len(test) - 60:]
inputs = inputs.reshape(-1,1)
inputs  = scaler.transform(inputs)
```

```
2561/2561 - 29s - loss: 3.3273e-04 - 29s/epoch - 11ms/step
```

Tạo test set và test

Tương tự như training set, test set cũng cần phải được định dạng để phù hợp với mô hình LSTM

Tạo test set và test

In [79]:
```
# Tạo test data set
X_test = []
for i in range(60,inputs.shape[0]):
    X_test.append(inputs[i-60:i,0])

# Chuyển data thành numpy array
X_test = np.array(X_test)

# Chuyển data thành 3-D
X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
adj_closing_price = model.predict(X_test)
adj_closing_price = scaler.inverse_transform(adj_closing_price)
```

```
21/21 [==============================] - 1s 8ms/step
```

In [80]:
```
print(X_test.shape)
```

```
(656, 60, 1)
```

Tính chỉ số RMSE

Tính chỉ số RMSE

In [81]:
```
rmse = np.sqrt(np.mean(np.power((test - adj_closing_price),2)))
rmse
```

Out[81]:
```
7.879194951543048
```

Chỉ số RMSE của mô hình Long Short-Term Memory trả về giá trị là **7.879194951543048** (một con số thấp vượt trội cho thấy mức độ hiệu quả được cải thiện đáng kể của mô hình so với những mô hình dự đoán trước).

Bảng so sánh sai số của các mô hình

Mô hình	RMSE
Đường trung bình động	129.66982647342456
Hồi quy tuyến tính	126.67440240138559
k-Nearest Neighbours	125.69569336103844
Prophet	63.79123610077194
Long Short-Term Memory	7.879194951543048

Mô hình LSTM đã đạt được giá trị RMSE thấp nhất và dự đoán chính xác nhất. Dù rằng đây chỉ là một mô hình deep learning cơ bản song chúng ta có thể tiếp tục điều chỉnh nhiều tham số khác nhau, nhiều lớp có thể được thêm vào với các tính năng để chuẩn hóa và tăng số lượng epochs để mô hình có tính hội tụ tốt hơn.