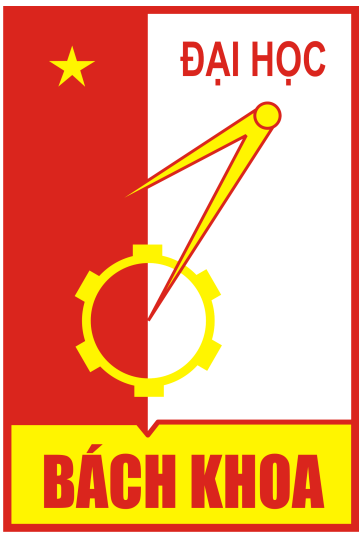
ĐẠI HỌC BÁCH KHOA HÀ NỘI
KHOA TOÁN - TIN



Dự báo giá chứng khoán theo giờ

Hệ hỗ trợ quyết định

Chuyên ngành: Hệ thống thông tin quản lý

Giảng viên hướng dẫn: **Ts. Trần Ngọc Thăng**

Sinh viên thực hiện: **Nguyễn Văn Hội**

Mã sinh viên: **20216926**

HÀ NỘI – 2024

Lời Cảm Ơn

Lời đầu tiên, em xin gửi lời cảm ơn tới Ts.Trần Ngọc Thăng, giảng viên Khoa Toán Tin, Đại học Bách khoa Hà Nội đã tận tình hướng dẫn và chỉ bảo trong quá trình học tập môn học.

Nhờ vào sự chỉ dẫn tận tình, kiến thức sâu rộng và lòng nhiệt huyết của thầy, em đã học hỏi được rất nhiều điều bổ ích và hiểu sâu hơn về các hệ hỗ trợ ra quyết định. Những bài giảng của thầy/cô không chỉ giúp em nắm vững kiến thức lý thuyết mà còn ứng dụng được vào thực tiễn, giúp em tự tin hơn trong học tập cũng như trong công việc tương lai. Em rất trân trọng những kiến thức và kinh nghiệm mà thầy/cô đã truyền đạt, cũng như sự quan tâm và động viên của thầy/cô trong suốt quá trình học tập. Đây sẽ là hành trang quý báu giúp em tiến bước trên con đường học vấn và sự nghiệp.

Một lần nữa, em xin chân thành cảm ơn thầy và chúc thầy luôn dồi dào sức khỏe, hạnh phúc và thành công trong sự nghiệp giảng dạy.

Mục lục

Mở đầu

Tỷ lệ rời bỏ, còn được gọi là tỷ lệ hao hụt khách hàng, là tốc độ mà khách hàng ngừng sử dụng sản phẩm, dịch vụ của doanh nghiệp trong một khoảng thời gian nhất định. Để doanh nghiệp có thể mở rộng thị phần hay tệp khách hàng của mình thì tỷ lệ khách hàng mới phải vượt qua tỷ lệ rời bỏ. Vì thế mà đây là tỉ lệ rất quan trọng đối với một doanh nghiệp. Do vậy việc đánh giá và nâng cao tỷ lệ rời bỏ là điều thiết yếu, doanh nghiệp mong muốn tỷ lệ này có thể thấp nhất có thể.

Bằng cách phân tích lịch sử giao dịch, hành vi mua hàng, tương tác trước đó và các yếu tố khác để đánh giá

Trong bài viết này, tôi sẽ trình bày về cơ sở lý thuyết và các phương pháp dùng để phân tích cũng như giải quyết bài toán

GIỚI THIỆU

Bài toán dự báo giá cổ phiếu theo giờ là một trong những lĩnh vực nghiên cứu và ứng dụng quan trọng trong phân tích tài chính và đầu tư. Việc dự đoán chính xác biến động giá cổ phiếu trong ngắn hạn không chỉ giúp nhà đầu tư tối ưu hóa lợi nhuận mà còn giúp quản lý rủi ro một cách hiệu quả. Tuy nhiên, đây cũng là một nhiệm vụ phức tạp do tính ngẫu nhiên và biến động cao của thị trường chứng khoán.

Phát biểu bài toán

Trong nghiên cứu này, mục tiêu chính là dự báo giá chứng khoán theo giờ, dữ liệu được thu thập công ty IBM, là viết tắt của International Business Machines, là một tập đoàn về công nghệ máy tính đa quốc gia có trụ sở tại Armonk, New York, Mỹ. IBM được thành lập năm 1911 tại Thành phố New York, lúc đầu có tên là Computing Tabulating Recording (CTR) và đổi thành International Business Machines vào năm 1924. Việc dự đoán giá cổ phiếu theo từng giờ trong ngày giao dịch hữu ích cho giao dịch ngắn hạn và chiến lược giao dịch trong ngày.

Bài toán kỹ thuật

Dự báo giá cổ phiếu là bài toán hồi quy, dự đoán giá cổ phiếu tại tương lai dựa trên các dữ liệu lịch sử và yếu tố khác.
Phát triển các mô hình học máy để dự báo dựa trên dữ liệu thu thập được từ các nguồn khác nhau. Xử lí, tích hợp các dữ liệu từ nhiều nguồn và gán nhãn. Đề xuất các tiêu chí đánh giá mô hình phù hợp với dữ liệu. Phát triển nhiều mô hình khác nhau để có thể so sánh đánh giá các mô hình với nhau, đánh giá và điều chỉnh các mô hình để tối ưu và đề xuất được mô hình tốt nhất cho bài toán

Chương 1 Thu thập và xử lý dữ liệu

1.1 Thu thập dữ liệu

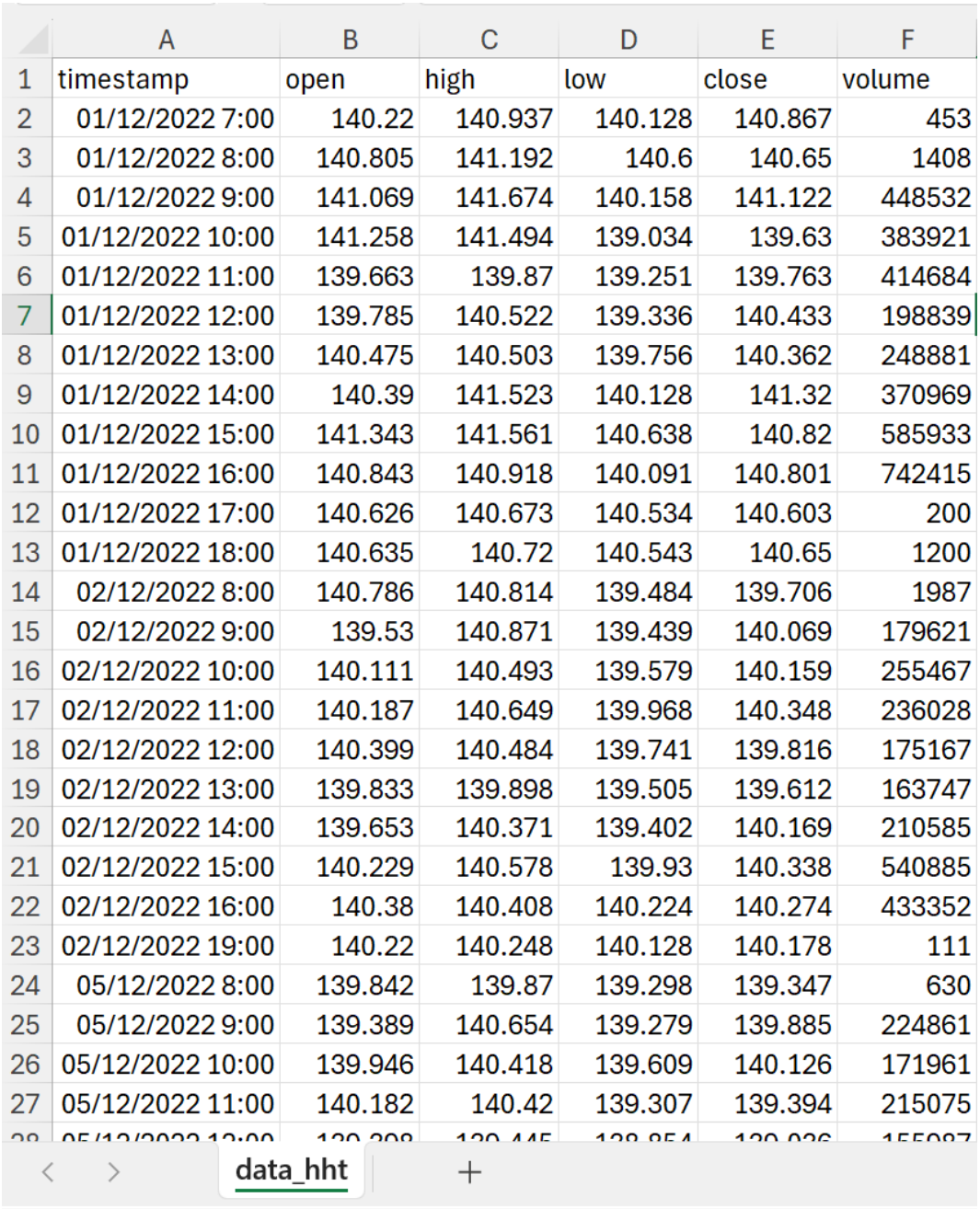
Dữ liệu được thu thập từ API dữ liệu thị trường chứng khoán do Alpha Vantage cung cấp dữ liệu lịch sử và dữ liệu thị trường tài chính theo thời gian thực. Với thời gian từ tháng 1/12/2022 đến thời điểm hiện tại với ngày cuối cùng được lấy là ngày 20/6/2024. Dữ liệu cung cấp thông tin về thời gian, giá mở, giá đóng, giá cao nhất, giá thấp nhất và khối lượng giao dịch trong 1 giờ đó.

craw.py > ...

```python
import requests
import json
import csv

# Define the URL
url = "https://www.alphavantage.co/query?function=TIME_SERIES_INTRADAY&symbol=IBM&interval=60min&month=2022-12&outputsize=full&apikey=421YOM1NM6H8MNN7"

# Fetch the data
response = requests.get(url)

# Check if the request was successful
if response.status_code == 200:
    # Parse the JSON data
    data = response.json()

    # Extracting relevant information
    meta_data = data.get("Meta Data", {})
    time_series = data.get("Time Series (60min)", {})

    # Define the CSV file name
    csv_file = "time_series_data_thang_12_2022.csv"

    # Write the data to a CSV file
    with open(csv_file, mode='w', newline='') as file:
        writer = csv.writer(file)

        # Write the meta data
        writer.writerow(["Meta Data"])
        for key, value in meta_data.items():
            writer.writerow([key, value])

        # Write a blank row for separation
        writer.writerow([])

        # Write the header for time series data
        header = ["timestamp", "open", "high", "low", "close", "volume"]
        writer.writerow(header)
```

Hình 1.1: Lấy dữ liệu từ API

Dữ liệu sau khi được lấy theo từng tháng về sẽ được tổng hợp lại, gộp chung thành 1 file csv. Với tổng số 5979 dữ liệu.



	A	B	C	D	E	F
1	timestamp	open	high	low	close	volume
2	01/12/2022 7:00	140.22	140.937	140.128	140.867	453
3	01/12/2022 8:00	140.805	141.192	140.6	140.65	1408
4	01/12/2022 9:00	141.069	141.674	140.158	141.122	448532
5	01/12/2022 10:00	141.258	141.494	139.034	139.63	383921
6	01/12/2022 11:00	139.663	139.87	139.251	139.763	414684
7	01/12/2022 12:00	139.785	140.522	139.336	140.433	198839
8	01/12/2022 13:00	140.475	140.503	139.756	140.362	248881
9	01/12/2022 14:00	140.39	141.523	140.128	141.32	370969
10	01/12/2022 15:00	141.343	141.561	140.638	140.82	585933
11	01/12/2022 16:00	140.843	140.918	140.091	140.801	742415
12	01/12/2022 17:00	140.626	140.673	140.534	140.603	200
13	01/12/2022 18:00	140.635	140.72	140.543	140.65	1200
14	02/12/2022 8:00	140.786	140.814	139.484	139.706	1987
15	02/12/2022 9:00	139.53	140.871	139.439	140.069	179621
16	02/12/2022 10:00	140.111	140.493	139.579	140.159	255467
17	02/12/2022 11:00	140.187	140.649	139.968	140.348	236028
18	02/12/2022 12:00	140.399	140.484	139.741	139.816	175167
19	02/12/2022 13:00	139.833	139.898	139.505	139.612	163747
20	02/12/2022 14:00	139.653	140.371	139.402	140.169	210585
21	02/12/2022 15:00	140.229	140.578	139.93	140.338	540885
22	02/12/2022 16:00	140.38	140.408	140.224	140.274	433352
23	02/12/2022 19:00	140.22	140.248	140.128	140.178	111
24	05/12/2022 8:00	139.842	139.87	139.298	139.347	630
25	05/12/2022 9:00	139.389	140.654	139.279	139.885	224861
26	05/12/2022 10:00	139.946	140.418	139.609	140.126	171961
27	05/12/2022 11:00	140.182	140.42	139.307	139.394	215075

Hình 1.2: Dữ liệu bài toán

Biến	Mô tả
timestamp	Đây là cột chứa thông tin về thời gian giao dịch diễn ra.
open	Giá mở cửa của cổ phiếu trong khoảng thời gian 1 giờ
high	Giá cao nhất (đỉnh) mà cổ phiếu đạt được trong khoảng thời gian đó.
low	Giá thấp nhất (đáy) mà cổ phiếu đạt được trong khoảng thời gian đó
close	Giá đóng cửa của cổ phiếu trong khoảng thời gian đó
volume	Khối lượng giao dịch trong khoảng thời gian đó.

Bảng 1.1: Bảng dữ liệu

1.1.1 Đánh nhãn dữ liệu

Ở đây, ta sử dụng cột 'close' để làm nhãn của dữ liệu, ta sẽ so sánh giá trị dự báo với giá trị lúc đóng cửa để xem chúng ta dự báo có chính xác hay không.

```
data
```
✓ 0.0s

	close
timestamp	
2022-12-01 07:00:00	140.867
2022-12-01 08:00:00	140.650
2022-12-01 09:00:00	141.122
2022-12-01 10:00:00	139.630
2022-12-01 11:00:00	139.763
...	...
2024-06-20 15:00:00	173.900
2024-06-20 16:00:00	173.900
2024-06-20 17:00:00	173.900
2024-06-20 18:00:00	173.900
2024-06-20 19:00:00	173.920

5585 rows × 1 columns

Hình 1.3: Nhãn

1.1.2 Tiền xử lý dữ liệu

Chuyển dữ liệu 'timestamp' về dạng Datetime Index

```
df['timestamp'] = pd.to_datetime(df['timestamp'], format='%d/%m/%Y %H:%M')
df.set_index('timestamp', inplace=True)
print(df.info())
print(df.head())
```

Kiểm tra dữ liệu bị thiếu

```
df.isnull().sum()
```

Kết quả

```
open      0
high      0
low       0
close     0
volume    0
dtype: int64
```

Kiểm tra dữ liệu trùng lặp

```
df.duplicated().sum()
```

Kết quả trả về không có dữ liệu nào bị trùng lặp

Loại bỏ điểm ngoại lai bằng phương pháp IQR

```
for column in df.select_dtypes(include=[np.number]).columns:
Q1 = df[column].quantile(0.25)
Q3 = df[column].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

df[column] = np.where(df[column] > upper_bound, upper_bound, df[column])
df[column] = np.where(df[column] < lower_bound, lower_bound, df[column])

print("\nDataFrame sau khi xu lyp IQR:")
print(df)
```

1.1.3 Thống kê dữ liệu mẫu

Một số đặc trưng của dữ liệu

```
print(df.describe())
```

Kết quả trả về

```
              open         high          low        close        volume
count  5585.000000  5585.000000  5585.000000  5585.000000  5.585000e+03
mean    149.922331   150.301102   149.495601   149.912088  3.299665e+05
std      23.047833    23.152454    22.942366    23.049758  3.604722e+05
min     116.414000   116.937000   116.270000   116.402000  1.000000e+00
25%     129.635000   129.913000   129.348000   129.621000  1.159000e+03
50%     141.069000   141.523000   140.642000   141.117000  2.538460e+05
75%     168.450000   168.990000   168.000000   168.420000  4.909370e+05
max     197.214000   197.818000   196.056000   196.784000  1.225604e+06
```

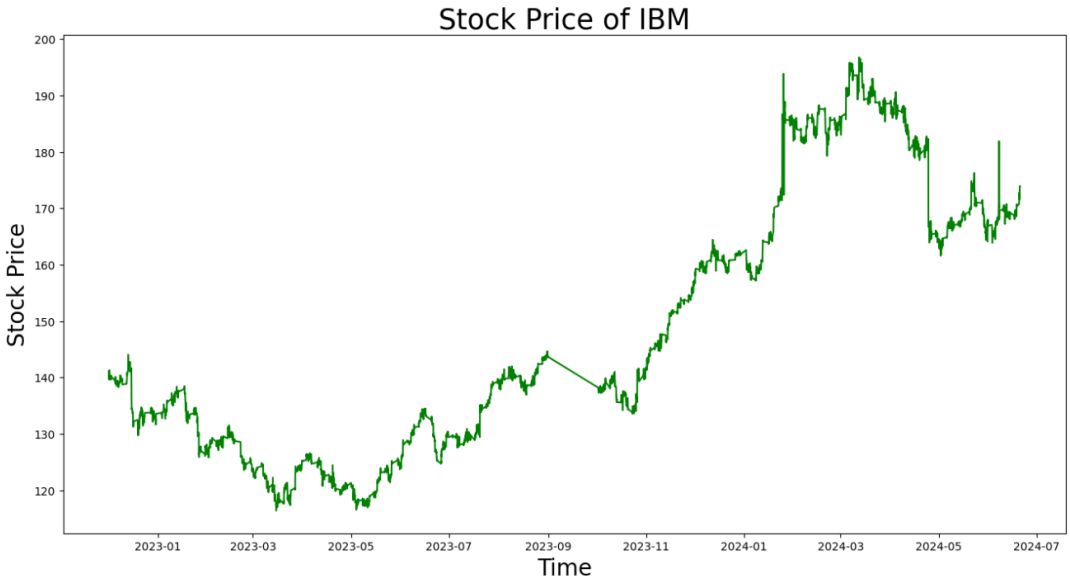
Tương quan giữa biến'close' với các biến còn lại

```
correlations = df.corr().loc['close', :]
print(correlations)
```

Kết quả trả về

```
open      0.999575
high      0.999550
low       0.999143
close     1.000000
volume    0.089623
Name: close, dtype: float64
```

Có thể thấy các biến 'open', 'high', 'low' có tường quan mạnh với biến 'close', trong khi đó 'volume' lại có tương quan yếu
Biểu đồ thể hiện giá cổ phiếu của IBM



Hình 1.4: Biểu đồ thể hiện mức giá và thời gian

Chương 2 Đánh Giá mô hình

Để đánh giá độ chính xác của mô hình, ta sửa dụng một số tiêu chí đánh giá sau

1. MSE (Mean Squared Error)

Là một trong những phép đo phổ biến để đánh giá chất lượng của mô hình dự báo giá chứng khoán. MSE được tính bằng cách lấy trung bình của bình phương sai số (chênh lệch giữa giá dự đoán và giá thực tế) trên tất cả các điểm dữ liệu.

Công thức tính MSE là:

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

Trong đó:

- y_i là giá trị thực tế của điểm dữ liệu thứ i.
- $\hat{y}_i$ là giá trị dự đoán của mô hình cho điểm dữ liệu thứ i.
- n là số lượng điểm dữ liệu.

Ý nghĩa của MSE trong đánh giá mô hình dự báo giá chứng khoán:

- Phản ánh sai số trung bình: MSE cho biết mức độ lớn của sai số bình phương trung bình giữa giá trị dự đoán và giá trị thực tế. Giá trị MSE càng thấp thì mô hình càng chính xác.
- Ưu điểm: MSE đánh giá tốt sự chính xác của mô hình bằng cách đánh giá tất cả các sai số dự đoán, không chỉ xét trọng số lớn hơn của các sai số lớn.
- Nhược điểm: MSE có thể bị ảnh hưởng bởi các giá trị ngoại lệ (outliers) trong dữ liệu, do tính chất bình phương trong công thức. Điều này có thể làm cho MSE không nhạy đủ với những sai số lớn khi các giá trị ngoại lệ xuất hiện.

2. RMSE Root Mean Squared Error

Là một phương pháp quan trọng để đo lường độ chính xác của các mô hình dự báo giá chứng khoán. RMSE là căn bậc hai của MSE (Mean Squared Error) và cung cấp một cái nhìn trực quan hơn về mức độ sai số, giữ nguyên đơn vị của giá trị cần dự đoán.

Công thức tính RMSE cho một tập dữ liệu gồm n điểm là:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Trong đó:

- y_i là giá trị thực tế của điểm dữ liệu thứ i.
- $\hat{y}_i$ là giá trị dự đoán của mô hình cho điểm dữ liệu thứ i.
- n là số lượng điểm dữ liệu.

Ý nghĩa của RMSE trong đánh giá mô hình dự báo giá chứng khoán:

- Phản ánh sai số trung bình: RMSE cung cấp thông tin về sai số trung bình giữa giá trị dự đoán và giá trị thực tế, trong cùng đơn vị với giá trị cần dự đoán, giúp dễ hiểu hơn so với MSE.
- Vì RMSE là căn bậc hai của MSE, nó có cùng đơn vị với biến mục tiêu, giúp dễ dàng so sánh và diễn giải sai số.
- Giống như MSE, RMSE cũng bị ảnh hưởng mạnh bởi các giá trị ngoại lệ, do đó có thể làm sai lệch đánh giá về mô hình nếu dữ liệu có nhiều ngoại lệ.

3. R-squared ($\mathbf{R^2}$)

Hay còn gọi là hệ số xác định, là một chỉ số thống kê dùng để đánh giá độ phù hợp của mô hình hồi quy. R-squared cho biết tỷ lệ phần trăm biến thiên của biến phụ thuộc (y) có thể được giải thích bởi các biến độc lập (x) trong mô hình.

Công thức tính R^2 là:

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

Trong đó:

- $SS_{\text{res}} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ là tổng bình phương của phần dư (Residual Sum of Squares).
- $SS_{\text{tot}} = \sum_{i=1}^{n}(y_i - \bar{y})^2$ là tổng bình phương của tổng (Total Sum of Squares).
- y_i là giá trị thực tế của điểm dữ liệu thứ i.

- $\hat{y}_i$ là giá trị dự đoán của mô hình cho điểm dữ liệu thứ i.
- $\bar{y}$ là giá trị trung bình của các giá trị thực tế y_i.
- n là số lượng điểm dữ liệu.

Ý nghĩa

- R-squared có giá trị từ 0 đến 1.
- $R^2 = 1$ nghĩa là mô hình hoàn toàn giải thích được biến thiên của dữ liệu.
- $R^2 = 0$ nghĩa là mô hình không giải thích được biến thiên của dữ liệu.

Ưu điểm

- R-squared cung cấp một thước đo dễ hiểu về mức độ giải thích của mô hình.
- Có thể so sánh giữa các mô hình để chọn mô hình tốt hơn.

Nhược điểm

- Không thể dùng để so sánh giữa các mô hình có số lượng biến độc lập khác nhau.
- Có thể tăng lên khi thêm biến vào mô hình, ngay cả khi biến đó không có ý nghĩa thống kê.

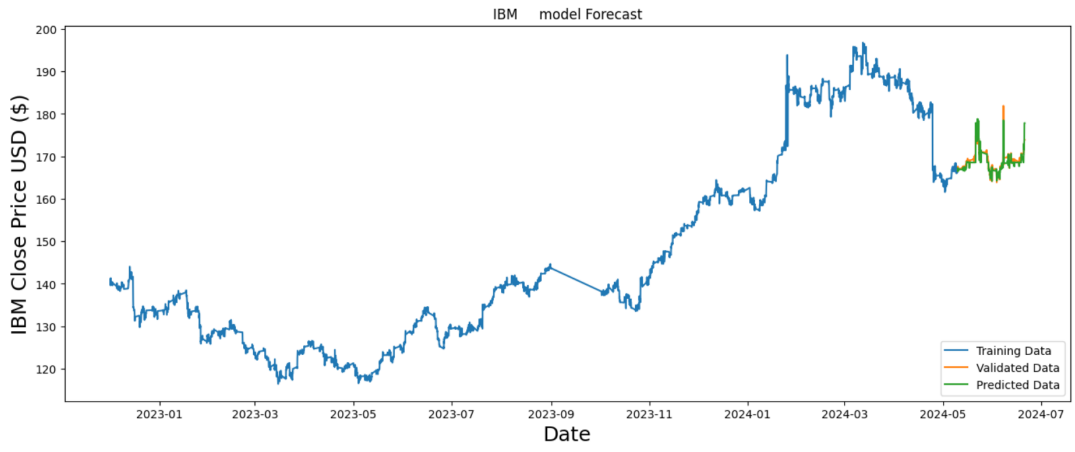
Chương 3 Cải tiến mô hình

Mô hình 1: XGBRegressor

Cấu trúc mô hình

```
from xgboost import XGBRegressor
def build_xgboot_model(x_train,y_train):
    x_train_2d = x_train.reshape((x_train.shape[0], -1))
    model = XGBRegressor(objective='reg:squarederror', n_estimators=100, learning_rate
    =0.1, max_depth=3, random_state=42)
    model.fit(x_train_2d, y_train)
    return model
xgboot_model =build_xgboot_model(x_train, y_train)
```

Biểu đồ thể hiện các giá trị thực tế, giá trị dự báo



Hình 3.1: Mô hình 1

Bảng thể hiện kết quả dự báo và thực tế

```
timestamp        close   Predictions
2024-05-10 06:00:00 166.860 166.567871
2024-05-10 07:00:00 166.770 166.554321
2024-05-10 08:00:00 166.510 166.567871
2024-05-10 09:00:00 167.620 166.567871
2024-05-10 10:00:00 167.046 167.246063
... ... ...
2024-06-20 15:00:00 173.900 177.230972
```

```
2024-06-20 16:00:00 173.900 177.848267
2024-06-20 17:00:00 173.900 177.848267
2024-06-20 18:00:00 173.900 177.848267
2024-06-20 19:00:00 173.920 177.848267
```

Các tiêu chí đánh giá

```
MSE: 2.737367728009499
RMSE: 1.654
R-squared: 0.5949826142322556
```

Mô hình 2: Gradient Boosting Regressor

cấu trúc mô hình

```
from sklearn.ensemble import GradientBoostingRegressor

def build_Gradien_model(x_train, y_train):
    # Chuyen doi x_train tu 3d qua 2d
    x_train_2d = x_train.reshape((x_train.shape[0], -1))

    # Khoi tao mo hinh voi tham so tuy chon
    model = GradientBoostingRegressor(n_estimators=100,learning_rate=0.1, max_depth=10,
    random_state=42)

    # Huan luyen mo hinh
    model.fit(x_train_2d, y_train)
    return model

Gradien_model = build_Gradien_model(x_train, y_train)
```

Kết quả dự đoán của mô hình

```
timestamp       close Predictions
2024-05-10 06:00:00 166.860 167.111083
2024-05-10 07:00:00 166.770 167.414468
2024-05-10 08:00:00 166.510 167.346550
2024-05-10 09:00:00 167.620 167.448806
2024-05-10 10:00:00 167.046 167.576233
... ... ...
2024-06-20 15:00:00 173.900 174.864577
2024-06-20 16:00:00 173.900 174.805142
2024-06-20 17:00:00 173.900 174.890634
2024-06-20 18:00:00 173.900 174.756458
2024-06-20 19:00:00 173.920 174.769353
```

Các tiêu chí đánh giá mô hình

```
MSE: 3.483153263996346
Test RMSE: 1.866
R-squared: 0.4846371516778213
```

Mô hình 3: Gradient Boosting Regressor2

Cấu trúc mô hình

```
from sklearn.ensemble import GradientBoostingRegressor

def build_Gradien_model_2(x_train, y_train):
    x_train_2d = x_train.reshape((x_train.shape[0], -1))

    model =GradientBoostingRegressor(n_estimators=200,learning_rate=0.05, max_depth=5,
    random_state=42)

    model.fit(x_train_2d, y_train)
    return model

Gradien_model_2 = build_Gradien_model(x_train, y_train)
```

Kết quả dự đoán mô hình

```
timestamp       close Predictions
2024-05-10 06:00:00 166.860 166.567871
2024-05-10 07:00:00 166.770 166.554321
2024-05-10 08:00:00 166.510 166.567871
2024-05-10 09:00:00 167.620 166.567871
2024-05-10 10:00:00 167.046 167.246063
... ... ...
2024-06-20 15:00:00 173.900 177.230972
2024-06-20 16:00:00 173.900 177.848267
2024-06-20 17:00:00 173.900 177.848267
2024-06-20 18:00:00 173.900 177.848267
2024-06-20 19:00:00 173.920 177.848267
```

Các tiêu chí đánh giá

```
MSE: 2.737367728009499
RMSE: 1.654
R-squared: 0.5949826142322556
```

Mô hình 4: RandomForestRegressor

Cấu trúc mô hình

```
from sklearn.ensemble import RandomForestRegressor

def build_randomforest_model(x_train, y_train):
    x_train_2d = x_train.reshape((x_train.shape[0], -1))

    model = RandomForestRegressor(n_estimators=100, max_depth=10, random_state=42)

    model.fit(x_train_2d, y_train)

    return model
```

```

RDFR_model = build_randomforest_model(x_train, y_train)
```

Kết quả dự đoán của mô hình

```
timestamp          close Predictions
2024-05-10 06:00:00 166.860 166.850522
2024-05-10 07:00:00 166.770 167.052881
2024-05-10 08:00:00 166.510 167.191766
2024-05-10 09:00:00 167.620 167.145276
2024-05-10 10:00:00 167.046 167.459744
... ... ...
2024-06-20 15:00:00 173.900 173.752127
2024-06-20 16:00:00 173.900 173.488415
2024-06-20 17:00:00 173.900 173.547921
2024-06-20 18:00:00 173.900 173.760975
2024-06-20 19:00:00 173.920 173.382053
```

Các tiêu chí đánh giá mô hình

```
MSE: 1.8127872558877436
Test RMSE: 1.346
R-squared: 0.7317823441037552
```

Mô hình 5: RNN (Recurrent Neural Network)

Cấu trúc mô hình

```
def build_RNN_model(x_train, y_train):

    model = Sequential()
    # lop dau tien
    model.add(SimpleRNN(units = 10,activation='tanh', return_sequences = True,
    input_shape = (x_train.shape[1], 1)))
    model.add(Dropout(0.2))
    # lop an thu 2
    model.add(SimpleRNN(units = 10,activation='tanh', return_sequences = True))
    model.add(Dropout(0.2))
    # lop an thu 3
    model.add(SimpleRNN(units = 10,activation='tanh', return_sequences = True))
    model.add(Dropout(0.2))
    # lop an thu 4
    model.add(SimpleRNN(units = 10))
    model.add(Dropout(0.2))
    # Lop dau ra
    model.add(Dense(units = 1))
    model.compile(optimizer = 'adam', loss = 'mean_squared_error')

    model.fit(x_train, y_train, epochs = 100, batch_size = 256)
    return model

rnn_model = build_RNN_model(x_train, y_train)
```

Kết quả dự báo của mô hình

```
timestamp       close Predictions
2024-05-10 06:00:00 166.860 167.522522
2024-05-10 07:00:00 166.770 167.560059
2024-05-10 08:00:00 166.510 167.651520
2024-05-10 09:00:00 167.620 167.596634
2024-05-10 10:00:00 167.046 167.651230
... ... ...
2024-06-20 15:00:00 173.900 174.265320
2024-06-20 16:00:00 173.900 174.705505
2024-06-20 17:00:00 173.900 175.000320
2024-06-20 18:00:00 173.900 175.071289
2024-06-20 19:00:00 173.920 174.938858
```

Các tiêu chí đánh giá

```
MSE: 1.974449846442725
Test RMSE: 1.405
R-squared: 0.707862956462464
```

Mô hình 6: RNN2 (Recurrent Neural Network)

Cấu trúc mô hình

```
from keras.models import Sequential
from keras.layers import SimpleRNN, Dense, Dropout

def build_improved_RNN_model(x_train, y_train):
    model = Sequential()
    model.add(SimpleRNN(units=64, activation='relu', return_sequences=True, input_shape=(
    x_train.shape[1], 1)))
    model.add(Dropout(0.3))

    model.add(SimpleRNN(units=32, activation='relu', return_sequences=True))
    model.add(Dropout(0.3))

    model.add(SimpleRNN(units=16, activation='relu', return_sequences=False))
    model.add(Dropout(0.3))

    model.add(Dense(units=1))

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

    model.fit(x_train, y_train, epochs=50, batch_size=128, validation_split=0.2)

    return model

# Test the function
improved_rnn_model = build_improved_RNN_model(x_train, y_train)
```

Kết quả dự đoán

```

timestamp        close  Predictions
2024-05-10 06:00:00 166.860 166.737656
2024-05-10 07:00:00 166.770 166.715652
2024-05-10 08:00:00 166.510 166.713409
2024-05-10 09:00:00 167.620 166.714188
2024-05-10 10:00:00 167.046 166.764297
... ... ...
2024-06-20 15:00:00 173.900 172.222900
2024-06-20 16:00:00 173.900 172.466827
2024-06-20 17:00:00 173.900 172.705856
2024-06-20 18:00:00 173.900 172.931335
2024-06-20 19:00:00 173.920 173.137573
```

Các tiêu chí đánh giá

```
MSE: 1.0847059324831971
Test RMSE: 1.041
R-squared: 0.8395083142809727
```

Mô hình 7: GRU (Gated Recurrent Unit)

Cấu trúc mô hình

```
def build_GRU_model(x_train, y_train):
    model = Sequential()
    model.add(GRU(units=50, return_sequences=True, input_shape=(x_train.shape[1], x_train
    .shape[2])))
    model.add(GRU(units=50))
    model.add(Dense(units=1))
    model.compile(optimizer='adam', loss='mean_squared_error')

    model.fit(x_train, y_train, epochs=10, batch_size=128)
    return model

GRU_model = build_GRU_model(x_train, y_train)
```

Kết quả dự đoán của mô hình

```
timestamp       close Predictions
2024-05-10 06:00:00 166.860 166.504105
2024-05-10 07:00:00 166.770 166.627884
2024-05-10 08:00:00 166.510 166.731934
2024-05-10 09:00:00 167.620 166.744705
2024-05-10 10:00:00 167.046 166.966034
... ... ...
2024-06-20 15:00:00 173.900 173.074631
2024-06-20 16:00:00 173.900 173.409790
2024-06-20 17:00:00 173.900 173.653488
2024-06-20 18:00:00 173.900 173.806427
2024-06-20 19:00:00 173.920 173.892868
```

Các tiêu chí đánh giá

```
MSE: 0.682824411277829
Test RMSE: 0.826
R-squared: 0.8989701839601779
```

Mô hình 8: LSTM(Long Short-Term Memory)

Cấu trúc mô hình

```
def build_lstm_model(x_train,y_train):
    # Build the LSTM model
    model = Sequential()
    model.add(LSTM(128, return_sequences=True, input_shape= (x_train.shape[1], 1)))
    model.add(LSTM(64, return_sequences=False))
    model.add(Dense(25))
    model.add(Dense(1))

    # Compile the model
    model.compile(optimizer='adam', loss='mean_squared_error')
    # Train the model
    model.fit(x_train, y_train, batch_size=256, epochs=10)
    return model

#Test the function
lstm_model = build_lstm_model(x_train,y_train)
```

Kết quả dự đoán của mô hình

```
timestamp       close Predictions
2024-05-10 06:00:00 166.860 166.921051
2024-05-10 07:00:00 166.770 166.897461
2024-05-10 08:00:00 166.510 166.892136
2024-05-10 09:00:00 167.620 166.889221
2024-05-10 10:00:00 167.046 166.932648
... ... ...
2024-06-20 15:00:00 173.900 172.256851
2024-06-20 16:00:00 173.900 172.495010
2024-06-20 17:00:00 173.900 172.729431
2024-06-20 18:00:00 173.900 172.951645
2024-06-20 19:00:00 173.920 173.156036
```

Các tiêu chí đánh giá

```
MSE: 1.1603836331739121
Test RMSE: 1.077
R-squared: 0.8283111396445365
```

Mô hình 9: LSTM2(Long Short-Term Memory)

Cấu trúc mô hình

```
def build_lmst2_model(x_train,y_train):
    model=Sequential()
    model.add(LSTM(20,return_sequences=True,input_shape=(x_train.shape[1], x_train.shape
    [2])))
    model.add(Dropout(0.2))
    model.add(BatchNormalization())
    model.add(LSTM(15,return_sequences=True))
    model.add(Dropout(0.2))
    model.add(BatchNormalization())
    model.add(LSTM(15))
    model.add(Dropout(0.2))
    model.add(BatchNormalization())
    model.add(Dense(16, activation='sigmoid'))
    model.add(Dense(1))

    adam = optimizers.Adam(0.01)
    model.compile(loss='mean_squared_error',optimizer=adam)
    model.fit(x_train, y_train, epochs=50, batch_size=256)
    return model
lmst2_model =  build_lmst2_model(x_train, y_train)
```

Kết quả dự báo mô hình

```
timestamp       close Predictions
2024-05-10 06:00:00 166.860 165.946960
2024-05-10 07:00:00 166.770 165.982986
2024-05-10 08:00:00 166.510 166.042664
2024-05-10 09:00:00 167.620 166.101807
2024-05-10 10:00:00 167.046 166.221283
... ... ...
2024-06-20 15:00:00 173.900 177.460892
2024-06-20 16:00:00 173.900 177.999039
2024-06-20 17:00:00 173.900 178.478195
2024-06-20 18:00:00 173.900 178.875351
2024-06-20 19:00:00 173.920 179.183670
```

Các tiêu chí đánh giá

```
MSE: 5.78190092441239
Test RMSE: 2.405
R-squared: 0.1445174233582236
```

Mô hình 10: LSTM3(Long Short-Term Memory)

Cấu trúc mô hình

```
def build_lstm_model_3(x_train,y_train):
    # Build the LSTM model
    model = Sequential()
    model.add(LSTM(128, return_sequences=True, input_shape= (x_train.shape[1], 1)))
    model.add(Dropout(0.2))
    model.add(LSTM(64, return_sequences=False))
```

```
    model.add(Dense(50, activation='sigmoid'))
    model.add(Dense(1))

    # Compile the model
    model.compile(optimizer='adam', loss='mean_squared_error')
    # Train the model
    model.fit(x_train, y_train, batch_size=256, epochs=10)
    return model

#Test the function
lstm_model_3 = build_lstm_model_3(x_train,y_train)
```

Kết qủa dự đoán mô hình

```
timestamp       close Predictions
2024-05-10 06:00:00 166.860 166.869446
2024-05-10 07:00:00 166.770 166.857071
2024-05-10 08:00:00 166.510 166.861755
2024-05-10 09:00:00 167.620 166.865372
2024-05-10 10:00:00 167.046 166.919907
... ... ...
2024-06-20 15:00:00 173.900 172.287323
2024-06-20 16:00:00 173.900 172.526169
2024-06-20 17:00:00 173.900 172.759720
2024-06-20 18:00:00 173.900 172.978622
2024-06-20 19:00:00 173.920 173.177383
```

Các tiêu chí đánh giá

```
MSE: 1.1048375455791 79
Test RMSE: 1.051
R-squared: 0.8365296668657966
```

KẾT LUẬN

Tổng Quan

Báo cáo này đã dùng các mô hình học máy tiên tiến để đánh giá, dự báo giá của cổ phiếu công ty IBM theo giờ trong thời gian từ 12/2022 đến 20/6/2024. Đưa ra các mô hình với độ chính xác cao và đáng tin cậy

Kết Quả Đạt Được

Xử Lý Dữ Liệu

- Dữ liệu được thu thập từ API tài chính do Alpha Vantage cung cấp với tổng số 5979 điểm dữ liệu
- Sau khi gán nhãn, dữ liệu được chuyển đổi về dạng số và chia thành các tập train và test với tỉ lệ 90% và 10%.

Áp Dụng Mô Hình

- 10 mô hình đã được áp dụng để đánh giá và đều đưa ra những kết quả rất tốt

Đóng Góp Của Đồ Án

Báo cáo đã đưa ra được dự đoán gần như chính xác về giá cổ phiếu theo giờ của IBm, từ đó có thể giúp các nhà đầu tư có thể đưa ra quyết định trong thời gian ngắn hiệu quả hơn.

Hướng Phát Triển Tương Lai

Trong tương lai, việc mở rộng và cải thiện mô hình dự báo sự rời bỏ khách hàng có thể bao gồm:

- Áp dụng nhiều mô hình tiên tiến khác nhau để tăng độ chính xác và hiệu quả của mô hình.

- Thu thập và phân tích thêm nhiều nguồn dữ liệu khác nhau để có cái nhìn tổng quan hơn về bài toán
- Tìm hiểu và thêm vào mô hình các yếu tố mới có thể ảnh hưởng

Bằng cách thực hiện các bước cải tiến này, chúng ta có thể nâng cao hiệu suất và độ chính xác của mô hình dự đoán