BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

NGÀNH HỆ THỐNG THÔNG TIN

PHÂN TÍCH TÌNH HÌNH KINH DOANH VÀ GIÁ TRỊ CỔ PHIẾU WALMART BẰNG NGÔN NGỮ PYTHON, CÔNG CỤ POWER BI

CBHD : ThS. Nguyễn Thị Thanh Huyền

Sinh viên : Trịnh Thu Phương

Mã số sinh viên : 2021600948

Hà Nội – 2025

LỜI CẢM ƠN

Em xin chân thành cảm ơn quý thầy cô Trường Đại học Công Nghiệp Hà Nội đã tận tình giảng dạy và truyền đạt cho em những kiến thức nền tảng cũng như kinh nghiệm thực tiễn trong suốt quá trình học tập. Đây là hành trang quan trọng giúp em hoàn thành tốt chương trình học và thực hiện đồ án tốt nghiệp.

Em đặc biệt gửi lời cảm ơn đến cô **ThS. Nguyễn Thị Thanh Huyền** – người đã trực tiếp hướng dẫn, hỗ trợ và góp ý cho em trong suốt quá trình làm đồ án. Những đóng góp và định hướng của cô đã giúp em nâng cao tư duy, kỹ năng làm việc và hoàn thành đề tài một cách hiệu quả.

Em cũng xin ghi nhận và cảm ơn sự hỗ trợ từ Nhà trường cùng các phòng ban chức năng đã tạo điều kiện thuận lợi về môi trường học tập, cơ sở vật chất và các hoạt động liên quan, giúp em có cơ hội tiếp cận và ứng dụng kiến thức vào thực tế.

Quãng thời gian bốn năm đại học đã mang lại cho em nhiều trải nghiệm giá trị, từ kiến thức chuyên môn đến kỹ năng làm việc và những mối quan hệ tích cực. Đây sẽ là nền tảng để em tiếp tục phát triển bản thân trong chặng đường sắp tới.

Cuối cùng, em xin kính chúc quý thầy cô sức khỏe, luôn giữ vững tinh thần nhiệt huyết với nghề. Chúc các bạn cùng khóa hoàn thành tốt đồ án tốt nghiệp và đạt được những mục tiêu mà mình đặt ra trong tương lai.

Em xin chân thành cảm ơn!

MỤC LỤC

THUẬT NGỮ VIẾT TẮT

Chữ viết tắt	Chữ đầy đủ	Diễn giải
MA	Moving Average	Phương pháp trung bình trượt để làm mượt chuỗi thời gian và phát hiện xu hướng.
MAE	Mean Absolute Error	Sai số tuyệt đối trung bình
MAPE	Mean Absolute Percentage Error	Sai số phần trăm tuyệt đối trung bình
MSE	Mean Squared Error	Sai số bình phương trung bình
RMSE	Root Mean Squared Error	Căn bậc hai của MSE
SES	Simple Exponential Smoothing	Phương pháp làm mượt số liệu chuỗi thời gian bằng cách áp dụng hệ số mũ giảm dần theo thời gian.

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

MỞ ĐẦU

1. Lý do chọn đề tài

Trong bối cảnh công nghệ phát triển vượt bậc, việc khai thác và phân tích dữ liệu trong lĩnh vực kinh doanh – tài chính ngày càng trở nên quan trọng trong việc ra quyết định chiến lược và đầu tư. Đề tài "Phân tích dữ liệu kinh doanh và giá cổ phiếu Walmart" nhằm đưa ra cái nhìn tổng quan về hiệu suất kinh doanh của doanh nghiệp, đồng thời áp dụng các mô hình học máy để dự đoán xu hướng cổ phiếu trong tương lai.

Đề tài này sử dụng ngôn ngữ Python là ngôn ngữ chính để thực hiện các bước từ xử lý, phân tích dữ liệu, trực quan hóa và xây dựng mô hình dự báo. Qua đó, em mong muốn xây dựng một hệ thống phân tích toàn diện, giúp hiểu rõ hơn về tình hình kinh doanh và biến động cổ phiếu, đồng thời đưa ra được mô hình dự báo hỗ trợ đề xuất xu hướng giá trị cổ phiếu đầu tư.

Đề tài không chỉ có ý nghĩa học thuật mà còn mang tính thực tiễn cao. Đây cũng là cơ hội để em áp dụng kiến thức đã học vào bài toán thực tế, giúp rèn luyện kỹ năng làm việc với dữ liệu lớn, nâng cao khả năng phân tích, trực quan hóa và tư duy hệ thống. Đồng thời, tạo nền tảng vững chắc cho những định hướng nghề nghiệp sau này.

2. Đối tượng và phạm vi nghiên cứu

- Đối tượng nghiên cứu: Tình hình kinh doanh và giá cổ phiếu của chuỗi siêu thị bán lẻ Walmart.
- Phạm vi nghiên cứu:
  + Dữ liệu sử dụng: Đề tài này sử dụng kết hợp 2 bộ dataset về Walmart từ nguồn Kaggle.com, bao gồm:
    - Dữ liệu liên quan đến hoạt động kinh doanh của 45 cửa hàng Walmart tại Hoa Kỳ từ tháng 2/2010 đến 10/2012, bao gồm doanh thu, chi phí,

tỷ lệ thất nghiệp, giá nhiên liệu, chỉ số giá tiêu dùng (CPI), và tác động của các ngày lễ đến doanh thu.

- Dữ liệu giá cổ phiếu Walmart trên thị trường chứng khoán (1972 - 2024), phản ánh sự biến động và các yếu tố ảnh hưởng đến giá trị cổ phiếu.

+ Ngôn ngữ và các công cụ sử dụng:
  - Ngôn ngữ lập trình Python: Xử lý và phân tích dữ liệu, trực quan hóa và xây dựng mô hình dự báo.
  - Công cụ Power BI: Trực quan hóa dữ liệu, tạo dashboard và báo cáo phân tích dữ liệu.
  - Công cụ Google Colab: Viết và chạy code Python trực quan, kết hợp với biểu đồ.
  - Phần mềm Pycharm Community: Soạn thảo, quản lý và chạy chương trình dự báo bằng ngôn ngữ Python.
+ Kết quả mong đợi: Xây dựng báo cáo phân tích và trực quan hóa dữ liệu kinh doanh, đồng thời dự báo giá cổ phiếu Walmart thông qua các mô hình học máy.

3. Mục tiêu nghiên cứu

- Tìm hiểu và ứng dụng các công cụ, ngôn ngữ hỗ trợ phân tích dữ liệu như Python, Power BI, PyCharm Community, Google Colab.
- Xử lý và phân tích dữ liệu từ các bộ dữ liệu đầu vào liên quan đến doanh thu và giá cổ phiếu của Walmart.
- Tìm hiểu các thuật toán dự báo và xây dựng mô hình dự báo để dự đoán xu hướng giá cổ phiếu.
- Vận dụng kiến thức để đưa ra nhận định, đề xuất chiến lược kinh doanh và đầu tư phù hợp.

4. Kết quả dự kiến

- Nắm vững kiến thức và sử dụng thành thạo ngôn ngữ và các công cụ phân tích dữ liệu.

- Xây dựng được bộ dữ liệu sạch, phân tích và trực quan hóa dữ liệu dưới dạng các biểu đồ về tình hình kinh doanh và biến động cổ phiếu.
- Ứng dụng các thuật toán dự báo để dự đoán xu hướng giá cổ phiếu trong tương lai.
- Dựa trên kết quả phân tích và dự báo, đưa ra nhận định và đề xuất chiến lược đầu tư phù hợp.

5. Phương pháp nghiên cứu

- Tìm hiểu, quan sát, tìm kiếm tài liệu.
- Phân tích, thiết kế hệ thống.
- Sử dụng các công cụ hỗ trợ trong quá trình phân tích.
- Thống kê, đánh giá và viết báo cáo.

6. Cấu trúc của báo cáo

Đề tài gồm 3 chương:

- Chương 1: Tổng quan về ngôn ngữ lập trình (từ trang 4 đến trang 8), trình bày về lịch sử phát triển, ứng dụng và ưu nhược điểm của ngôn ngữ Python.
- Chương 2: Phân tích dữ liệu và các mô hình dự báo (từ trang 9 đến trang), trình bày về quy trình thực hiện, phân tích, trực quan hóa dữ liệu bằng Python, tạo doashboard và báo cáo trên Power BI.
- Chương 3: Phân tích thiết kế và cài đặt bài toán (từ trang đến trang), trình bày về các mô hình dự báo, phân tích thiết kế hệ thống, cài đặt chương trình và kiểm thử.

CHƯƠNG 1. TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH

1.1. Lịch sử phát triển ngôn ngữ lập trình Python

Python là một ngôn ngữ lập trình bậc cao, mã nguồn mở và đa nền tảng. Python được sử dụng rộng rãi để phát triển các ứng dụng web, phát triển phần mềm, khoa học dữ liệu và học máy (Machine Learning).

Python được tạo ra bởi Guido van Rossum vào cuối những năm 1980, với phiên bản đầu tiên ra mắt vào năm 1991. Tính đến thời điểm hiện tại, Python đã được nâng cấp và phát triển nhiều phiên bản khác nhau và ngày càng hoàn thiện hơn. Python đã có tổng cộng 6 phiên bản và luôn nằm trong top những ngôn ngữ lập trình phổ biến, được nhiều người sử dụng nhiều nhất.

- Python 2.0 được phát hành năm 2000 và có nhiều tính năng mới như hỗ trợ Unicode.
- Python 3.0 được phát hành năm 2008 không tương thích hoàn toàn với phiên bản 2.0 nhưng lại có công cụ hỗ trợ chuyển đổi từ phiên bản này sang phiên bản kia.
- Phiên bản mới nhất 3.9 được ra mắt vào 2020 được nâng cấp cải tiến các vấn đề bảo mật.

Python được thiết kế với tư tưởng giúp người học dễ đọc, dễ hiểu và dễ nhớ. Vì thế ngôn ngữ Python có hình thức rất đơn giản, cấu trúc rõ ràng, thuận tiện cho người mới học. Cấu trúc của Python cho phép người dùng viết mã lệnh với số lần gõ phím tối thiểu, nói cách khác thì so với các ngôn ngữ lập trình khác, người dùng có thể sử dụng ít dòng code hơn để viết ra một chương trình trong Python.

Python sử dụng một trình thông dịch để chạy mã. Khi viết mã Python, người dùng không cần phải biên dịch nó thành mã máy trước khi chạy. Thay vào đó, trình thông dịch sẽ đọc và thực thi mã trực tiếp, từng dòng một.

Python có sẵn các cấu trúc dữ liệu mạnh mẽ như list, dictionary, tuple, giúp dễ dàng xử lý và lưu trữ dữ liệu. Ngoài ra, không cần phải khai báo kiểu dữ liệu cho các biến, Python sẽ tự động xác định kiểu dữ liệu dựa trên giá trị của biến.

Những thư viện Python giúp tính toán thống kê, trực quan hóa và ứng dụng trong mô hình dự báo phổ biến:

Bảng 1.1. Các thư viện Python phổ biến

	Thư viện	**Chức năng**
Trực quan hóa dữ liệu	Numpy	Thư viện cung cấp mảng đa chiều, các phép toán nhanh trên mảng bao gồm toán học, logic, sắp xếp, thống kê cơ bản và nhiều tính năng khác.
	Pandas	Thư viện làm việc với dữ liệu dạng bảng (DataFrame). Thư viện hỗ trợ nhập và xuất dữ liệu từ nhiều định dạng (CSV, Excel, SQL, JSON, v.v.), tính toán thống kê, làm sạch, lọc, nhóm và biến đổi dữ liệu.
Trực quan hóa dữ liệu	Matplotlib	Thư viện đồ họa mạnh mẽ và linh hoạt trong Python, thường được sử dụng để tạo các đồ thị và hình ảnh, thích hợp cho việc tạo các biểu đồ 2D.
	Seaborn	Thư viện đồ họa dựa trên Matplotlib và cung cấp một giao diện cao cấp hơn để tạo các biểu đồ thống kê.
Mô hình dự báo	Statsmodel.tsa.seasonal	Chứa các lớp mô hình và hàm hữu ích cho phân tích chuỗi thời gian.
	Statsmodels.tsa.holtwinters	Chứa các lớp mô hình và hàm hữu ích cho làm mịn theo hàm mũ.
	Sklearn.metrics	Thư viện cung cấp các chỉ số đánh giá độ sai lệch của các mô hình dự

		báo bao gồm sai số bình phương trung bình (MSE) và sai số trung bình tuyết đối (MAE).

1.2. Ứng dụng của ngôn ngữ lập trình Python

Python là ngôn ngữ lập trình được sử dụng rộng rãi nhờ cú pháp đơn giản, thư viện phong phú và khả năng linh hoạt trong nhiều lĩnh vực. Dưới đây là một số ứng dụng nổi bật của Python, bao gồm:

- Phát triển Web: Python là một ngôn ngữ phổ biến trong phát triển web nhờ vào các framework mạnh mẽ như Django và Flask.
- Khoa học dữ liệu và phân tích dữ liệu: Thư viện Pandas cung cấp các công cụ mạnh mẽ để thao tác và phân tích dữ liệu. NumPy cho phép thực hiện các tính toán số học với hiệu suất cao. Matplotlib và Seaborn giúp tạo ra các biểu đồ đẹp mắt và dễ hiểu. Người dùng có thể dễ dàng làm việc với các tập dữ liệu lớn, thực hiện các phân tích phức tạp và trực quan hóa kết quả.
- Trí tuệ nhân tạo và học máy: TensorFlow và Keras giúp xây dựng các mô hình học sâu (Deep Learning) mạnh mẽ. PyTorch nổi bật với tính linh hoạt và dễ sử dụng, là lựa chọn ưa thích của nhiều nhà nghiên cứu. Scikit-learn cung cấp các công cụ đơn giản nhưng hiệu quả để thực hiện các thuật toán học máy truyền thống.
- Phát triển Game: Pygame là thư viện nổi tiếng giúp phát triển Game đơn giản, cung cấp công cụ để xử lý đồ họa, âm thanh và tương tác người dùng.
- Mạng và bảo mật: Người dùng có thể viết các script để phân tích gói tin, kiểm thử bảo mật, hoặc thậm chí phát triển các công cụ bảo mật riêng.
- Internet of Things (IoT): Python phổ biến trong phát triển các ứng dụng IoT, đặc biệt là với Raspberry Pi. Người dùng có thể dễ dàng viết mã để điều khiển các cảm biến, thiết bị và thu thập dữ liệu từ môi trường.

- Xử lý hình ảnh và video: OpenCV là thư viện xử lý hình ảnh và video trong Python. Cho phép người dùng thực hiện các tác vụ từ nhận diện khuôn mặt, theo dõi đối tượng, đến xử lý video thời gian thực.

1.3. Ưu nhược điểm của ngôn ngữ lập trình Python

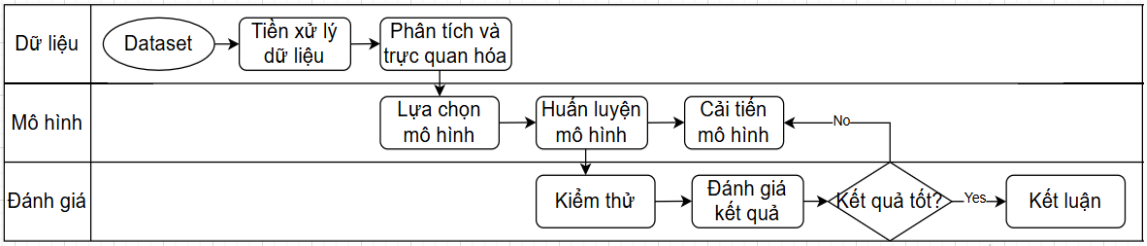
- Python là ngôn ngữ lập trình được sử dụng phổ biến nhất trên thế giới, nhờ các ưu điểm sau:
  + Cú pháp đơn giản và dễ đọc: Cú pháp của Python rất giống với ngôn ngữ Tiếng Anh tự nhiên, dễ học và dễ đọc, giúp lập trình viên tập trung vào giải quyết vấn đề hơn là việc ghi nhớ cú pháp phức tạp.
  + Đa dụng: Python có thể được sử dụng trong nhiều lĩnh vực khác nhau như phát triển web, khoa học dữ liệu, trí tuệ nhân tạo, tự động hóa, phân tích dữ liệu, và nhiều ứng dụng khác.
  + Thư viện phong phú: Python có một kho thư viện đồ sộ, hỗ trợ nhiều tác vụ khác nhau. Các thư viện như NumPy, Pandas, TensorFlow, và Django giúp lập trình viên tiết kiệm thời gian và công sức khi phát triển các ứng dụng phức tạp.
  + Đa nền tảng: Python có thể chạy trên nhiều hệ điều hành khác nhau như Windows, macOS, Linux, Raspberry Pi,… giúp cho việc phát triển và triển khai ứng dụng trở nên dễ dàng.
  + Khả năng mở rộng và tích hợp tốt: Python có thể dễ dàng tích hợp với các ngôn ngữ lập trình khác và các công nghệ hiện có, giúp nó trở thành một lựa chọn lý tưởng cho nhiều dự án khác nhau.
  + Hỗ trợ từ các tổ chức lớn: Nhiều công ty và tổ chức lớn như Google, Facebook, NASA sử dụng Python và đóng góp vào việc phát triển ngôn ngữ này, làm tăng uy tín và sự tin cậy của nó.
- Cùng với những ưu điểm, Python cũng có một số hạn chế trong các lĩnh vực. Sau đây là một số nhược điểm đáng kể của việc sử dụng Python:

+ Tốc độ thực thi chậm: Python là một ngôn ngữ thông dịch, hoạt động với trình thông dịch, không phải với trình biên dịch. Do đó, Python thực thi tương đối chậm hơn so với C, C ++, Java và nhiều ngôn ngữ khác.
+ Tiêu thụ bộ nhớ lớn: Các cấu trúc của Python đòi hỏi nhiều không gian bộ nhớ hơn. Ngôn ngữ này không thích hợp để sử dụng cho sự phát triển trong điều kiện bộ nhớ hạn chế.
+ Không thích hợp cho phát triển trò chơi và thiết bị di động***:*** Python chủ yếu được sử dụng trong phát triển máy tính để bàn và web phía máy chủ. Python không phù hợp để phát triển ứng dụng di động và trò chơi do tiêu tốn nhiều bộ nhớ hơn và tốc độ xử lý chậm so với các ngôn ngữ lập trình khác.
+ Phát hiện lỗi trong mã: Vì Python được thực thi thông qua trình thông dịch thay vì trình biên dịch, nên không thể phát hiện lỗi trong quá trình biên dịch và điều đó không tốt cho các nhà phát triển.
+ Quyền truy cập cơ sở dữ liệu: Python được coi là không an toàn cao và có nguy cơ bảo mật. Có một số hạn chế khi sử dụng Python để truy cập cơ sở dữ liệu. So với các công nghệ phổ biến khác như JDBC và ODBC, lớp truy cập cơ sở dữ liệu Python hơi kém phát triển và sơ khai.
+ Khó kiểm tra: Vì là một ngôn ngữ dựa trên trình thông dịch nên tất cả các lỗi chỉ xuất hiện trong thời gian chạy, điều này khiến việc kiểm tra các đoạn mã được viết bằng Python rất khó khăn.

CHƯƠNG 2. PHÂN TÍCH DỮ LIỆU

2.1. Quy trình thực hiện

Quy trình phân tích dữ liệu gồm các bước chính sau:



Hình 1.1. Quy trình thực hiện

Bước 1: Tiến hành thu thập dữ liệu từ hai tập tin chính là walmart_sales.csv (dữ liệu bán hàng) và walmart_stock.csv (dữ liệu giá cổ phiếu). Sau khi thu thập, dữ liệu được đưa vào bước tiền xử lý, bao gồm làm sạch dữ liệu, xử lý giá trị thiếu, chuyển đổi kiểu dữ liệu và chuẩn hóa để chuẩn bị cho các bước phân tích tiếp theo.

Bước 2: Phân tích và trực quan hóa dữ liệu. Ở giai đoạn này, dữ liệu được thống kê mô tả và biểu diễn thông qua các biểu đồ trực quan (line chart, bar chart, heatmap...) nhằm đưa ra được cái nhìn tổng quan về tình hình kinh doanh, và biến động giá trị cổ phiếu.

Bước 3: Lựa chọn mô hình dự báo phù hợp, chẳng hạn như Moving Average, *Simple* Exponential Smoothing, *Holt, Holt-Winters*,... Sau đó, mô hình được huấn luyện trên tập dữ liệu huấn luyện, và có thể được cải tiến thông qua điều chỉnh tham số, chọn lại đặc trưng đầu vào hoặc thử nghiệm với mô hình khác để nâng cao độ chính xác.

Bước 4: Kiểm thử mô hình trên tập dữ liệu. Kết quả mô hình sẽ được đánh giá dựa trên các tiêu chí như sai số trung bình tuyệt đối (MAE), căn sai số bình phương trung bình (RMSE), hệ số xác định (R^2),... Nếu mô hình cho kết quả tốt, quá trình thực hiện kết thúc với phần kết luận, trong đó nêu bật những phát hiện chính và đề xuất hướng phát triển tiếp theo. Nếu kết quả chưa đạt yêu cầu, mô hình sẽ được cải tiến và quy trình huấn luyện – kiểm thử được lặp lại.

2.2. Tiền xử lý dữ liệu

2.2.1. Thu thập dữ liệu

Cài đặt các thư viện cần thiết:

```
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib
import matplotlib.pyplot as plt
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.tsa.holtwinters import Holt
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

Hình 2.1. Cài đặt các thư viện cần thiết

Đề tài sử dụng kết hợp hai bộ dữ liệu được thu thập từ nền tảng Kaggle.com nhằm phục vụ phân tích hoạt động kinh doanh và biến động giá cổ phiếu của Walmart, bao gồm:

- Walmart_Sales.csv: Cung cấp thông tin về kết quả hoạt động kinh doanh hàng tuần của chuỗi 45 siêu thị của Walmart từ tháng 2/2010 đến tháng 10/2012, bao gồm các trường:
  + Stores: Mã cửa hàng
  + Date: Ngày bắt đầu tuần bán hàng
  + Weekly_Sales: Doanh số bán hàng
  + Holiday_Flag: Tuần lễ (0 – không có, 1 – có)
  + Temperature: Nhiệt độ trung bình trong tuần
  + Fuel_Price: Giá nhiên liệu
  + CPI: Chỉ số giá tiêu dùng
  + Unemployment: Tỷ lệ thất nghiệp
- Walmart_Stock.csv: Cung cấp thông tin về biến động giá trị của công ty Walmart trên sàn giao dịch từ năm 1972 đến năm 2024, bao gồm các trường:
  + Date: Thời gian giao dịch
  + Open: Giá mở cửa

+ High: Giá cao nhất trong ngày
+ Low: Giá thấp nhất trong ngày
+ Close: Giá đóng cửa
+ Adj Close: Giá đóng cửa điều chỉnh
+ Volume: Khối lượng giao dịch trong ngày

Đọc dữ liệu từ file CSV và in ra 5 dòng đầu tiên

```
sales_df = pd.read_csv("Walmart_Sales.csv")
sales_df.head()
```

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	8.106
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	8.106
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	8.106
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	8.106
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	8.106

```
stock_df = pd.read_csv("Walmart_Stock.csv")
stock_df.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	1972-08-25	0.021159	0.021566	0.021159	0.021484	0.011664	7526400
1	1972-08-28	0.021484	0.021647	0.021403	0.021403	0.011620	2918400
2	1972-08-29	0.021322	0.021322	0.021159	0.021159	0.011488	5836800
3	1972-08-30	0.021159	0.021159	0.020996	0.021159	0.011488	1228800
4	1972-08-31	0.020996	0.020996	0.020833	0.020833	0.011311	2611200

Hình 2.2. Tổng quan về bộ dữ liệu

2.2.2. Làm sạch dữ liệu

Để thực hiện việc làm sạch dữ liệu cần thống kê các dữ liệu thiếu, dữ liệu trống và trùng lặp.

```
print("Dữ liệu thiếu: ", sales_df.isnull().values.sum())
print("Dữ liệu trống: ", sales_df.isna().values.any())
print("Dữ liệu trùng lặp: ", sales_df.duplicated().values.sum())
```

```
Dữ liệu thiếu:  0
Dữ liệu trống:  False
Dữ liệu trùng lặp:  0
```

```
print("Dữ liệu thiếu: ", stock_df.isnull().values.sum())
print("Dữ liệu trống: ", stock_df.isna().values.any())
print("Dữ liệu trùng lặp: ", stock_df.duplicated().values.sum())
```

```
Dữ liệu thiếu:  0
Dữ liệu trống:  False
Dữ liệu trùng lặp:  0
```

Hình 2.3. Kiểm tra dữ liệu

Kết quả kiểm tra cho thấy không có dữ liệu bị thiếu, trống hay trùng lặp. Đọc thông tin toàn bộ bảng dữ liệu, ta sẽ thấy kiểu dữ liệu của từng trường.

```
sales_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Store         6435 non-null   int64  
 1   Date          6435 non-null   object 
 2   Weekly_Sales  6435 non-null   float64
 3   Holiday_Flag  6435 non-null   int64  
 4   Temperature   6435 non-null   float64
 5   Fuel_Price    6435 non-null   float64
 6   CPI           6435 non-null   float64
 7   Unemployment  6435 non-null   float64
dtypes: float64(5), int64(2), object(1)
memory usage: 402.3+ KB
```

```
stock_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13145 entries, 0 to 13144
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       13145 non-null  object 
 1   Open       13145 non-null  float64
 2   High       13145 non-null  float64
 3   Low        13145 non-null  float64
 4   Close      13145 non-null  float64
 5   Adj Close  13145 non-null  float64
 6   Volume     13145 non-null  int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 719.0+ KB
```

Hình 2.4. Thông tin toàn bộ hai bảng dữ liệu

Chuyển đổi dữ liệu trong cột ‘Date’ từ kiểu string sang kiểu datetime.

```
sales_df['Date'] = pd.to_datetime(sales_df['Date'], format='%d-%m-%Y')
if 'Date' in sales_df.columns:
    sales_df['Date'] = pd.to_datetime(sales_df['Date'])
    sales_df['Year'] = sales_df['Date'].dt.year
    sales_df['Month'] = sales_df['Date'].dt.month
    sales_df['Day'] = sales_df['Date'].dt.day
    sales_df['Weekday'] = sales_df['Date'].dt.weekday
    sales_df['WeekOfYear'] = sales_df['Date'].dt.isocalendar().week
```

```
stock_df['Date'] = pd.to_datetime(stock_df['Date'])
stock_df.set_index('Date', inplace=True)
```

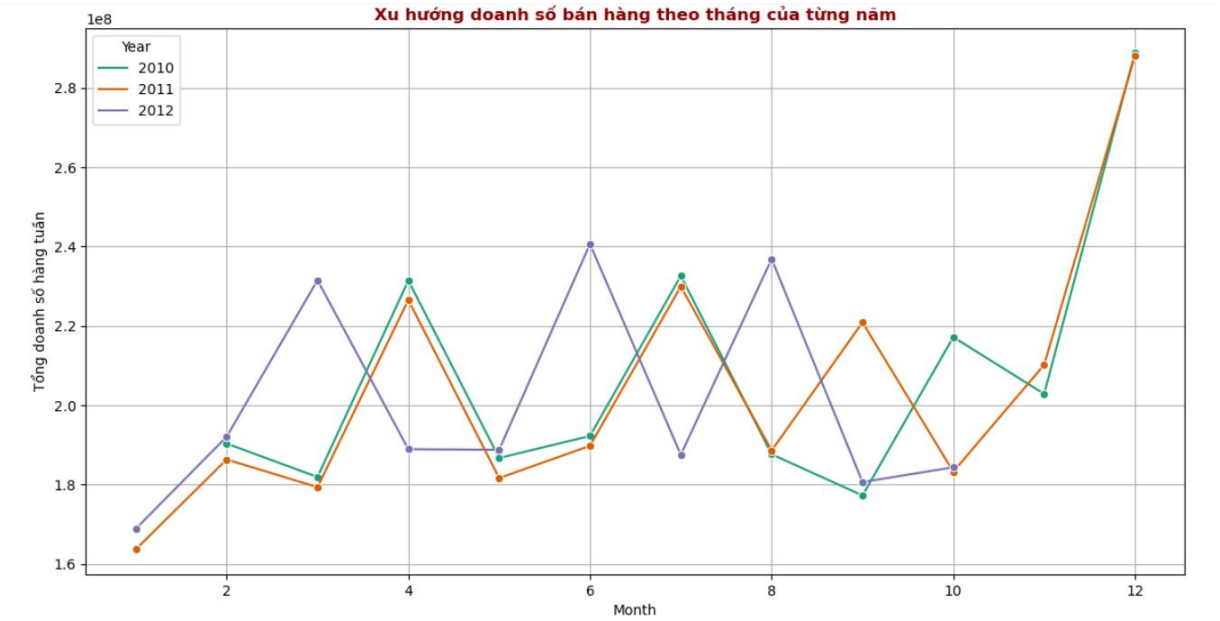
Hình 2.5. Chuyển đổi dữ liệu cột ‘Date’

2.3. Phân tích và trực quan hóa dữ liệu bằng Python

2.3.1. Phân tích dữ liệu kinh doanh

- Phân tích xu hướng doanh số bán hàng theo tháng của từng năm:

```
monthly_sales = sales_df.groupby(['Year', 'Month'])['Weekly_Sales'].sum().reset_index()
plt.figure(figsize=(14, 7))
sb.lineplot(data=monthly_sales, x='Month', y='Weekly_Sales', hue='Year', marker='o', palette='Dark2')
plt.title('Xu hướng doanh số bán hàng theo tháng của từng năm', fontsize='12', fontweight='bold', color='darkred')
plt.xlabel('Month')
plt.ylabel('Tổng doanh số hàng tuần')
plt.grid(True)
plt.show()
```



Hình 2.6. Biểu đồ xu hướng doanh số hàng tháng theo từng năm

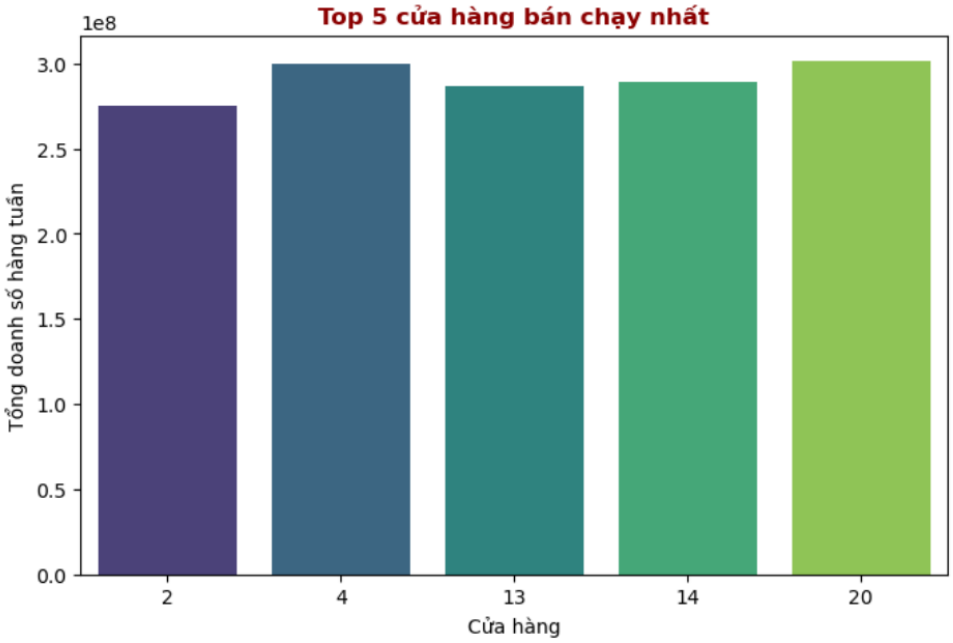
Tháng 11 và 12 luôn đạt tổng doanh số hàng tuần cao nhất trong nhiều năm. Chẳng hạn, tháng 12/2011 ghi nhận doanh số cao nhất là 288,7 triệu USD – cao

hơn đáng kể so với doanh số bán hàng trung bình hàng tháng khoảng 204 triệu USD. Điều này khẳng định mùa lễ cuối năm là động lực chính thúc đẩy doanh số của doanh nghiệp.

Khuyến nghị: Walmart nên ưu tiên tăng tồn kho, đẩy mạnh khuyến mãi và tối ưu nhân sự trong tháng 11 và 12 để tận dụng thời gian mua sắm cao điểm. Bên cạnh đó, triển khai các chương trình khuyến mãi phù hợp với xu hướng mùa lễ sẽ giúp gia tăng doanh số hơn.

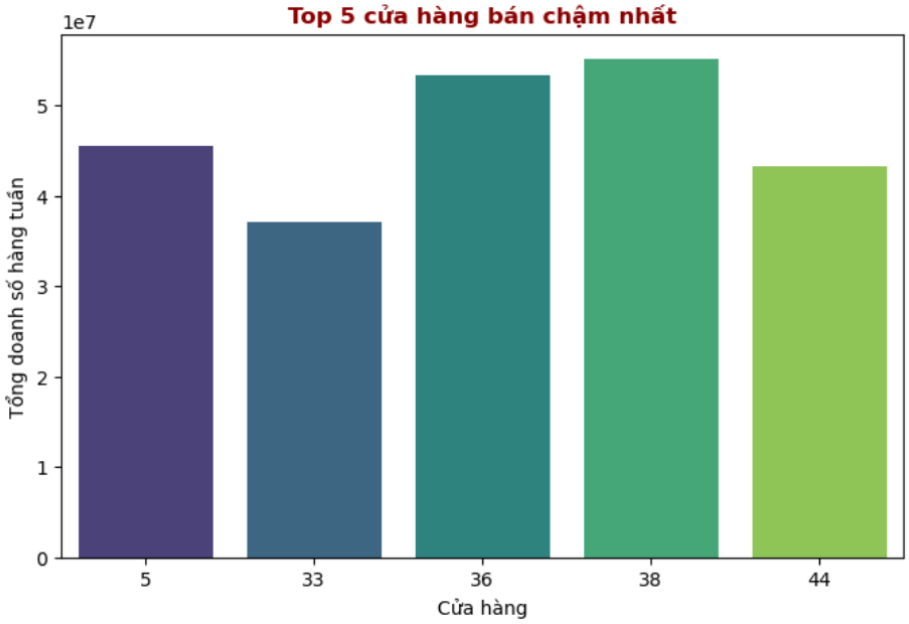
- Phân tích hiệu suất cửa hàng:

```
peak_stores = sales_df.groupby('Store')['Weekly_Sales'].sum().reset_index()
peak_stores = peak_stores.sort_values(by='Weekly_Sales', ascending=False).head(5)
plt.figure(figsize=(8,5))
sb.barplot(x=peak_stores['Store'], y=peak_stores['Weekly_Sales'], palette='viridis')
plt.xlabel('Cửa hàng')
plt.ylabel('Tổng doanh số hàng tuần')
plt.title('Top 5 cửa hàng bán chạy nhất', fontsize='12', fontweight='bold', color='darkred')
plt.show()
```



Hình 2.7. Top 5 cửa hàng bán chạy nhất

```
low_stores = sales_df.groupby('Store')['Weekly_Sales'].sum().reset_index()
low_stores = low_stores.sort_values(by='Weekly_Sales', ascending=True).head(5)
plt.figure(figsize=(8,5))
sb.barplot(x=low_stores['Store'], y=low_stores['Weekly_Sales'], palette='viridis')
plt.xlabel('Cửa hàng')
plt.ylabel('Tổng doanh số hàng tuần')
plt.title('Top 5 cửa hàng bán chậm nhất',fontsize='12', fontweight='bold', color='darkred')
plt.show()
```



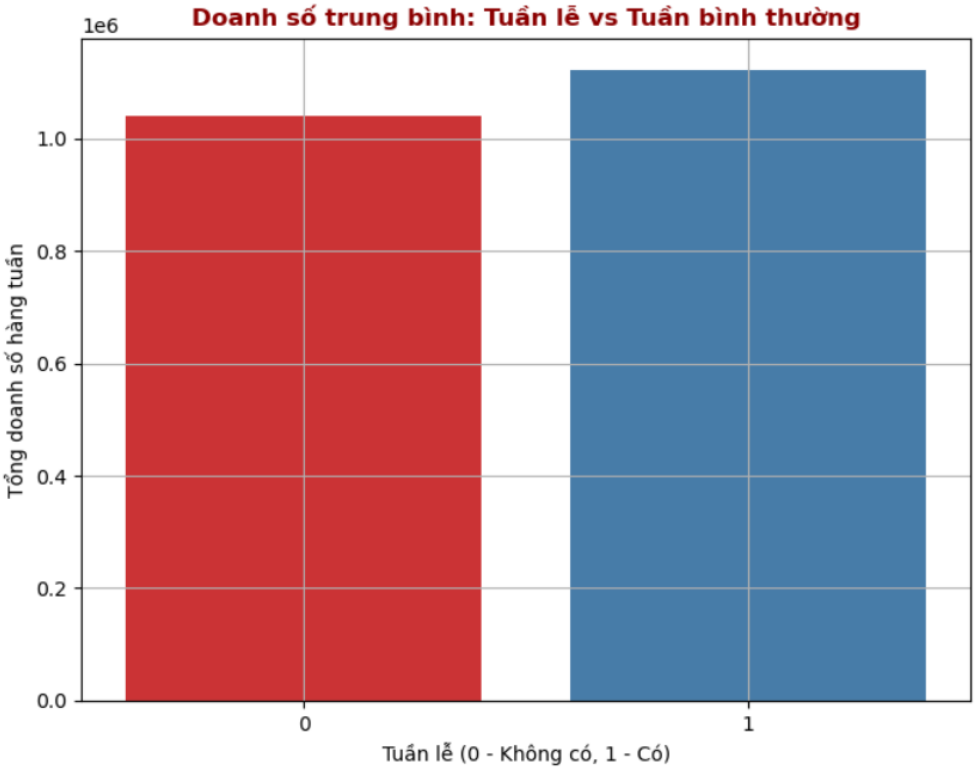
Hình 2.8. Top 5 cửa hàng bán chậm nhất

Doanh số giữa các cửa hàng có sự chênh lệch đáng kể. Ví dụ, cửa hàng 33 có tổng doanh thu chỉ dao động khoảng 35-37 triệu USD, thấp hơn khoảng 5-8 lần so với nhóm bán chạy. Điều này phản ánh sự không đồng đều về hiệu quả vận hành trong mạng lưới cửa hàng Walmart. Các yếu tố như vị trí địa lý, quy mô cửa hàng, chiến lược marketing,... có thể là nguyên nhân chính.

Khuyến nghị: Cần phân tích chuyên sâu các cửa hàng trong nhóm bán chậm để xác định các chiến lược kinh doanh phù hợp.

- Phân tích tác động của tuần lễ:

```
holiday_sales = sales_df.groupby(['Holiday_Flag'])['Weekly_Sales'].mean().reset_index()
plt.figure(figsize=(8, 6))
sb.barplot(data=holiday_sales, x='Holiday_Flag', y='Weekly_Sales', palette='Set1')
plt.title('Doanh số trung bình: Tuần lễ vs Tuần bình thường',fontsize='12', fontweight='bold', color='darkred')
plt.xlabel('Tuần lễ (0 - Không có, 1 - Có)')
plt.ylabel('Tổng doanh số hàng tuần')
plt.grid(True)
plt.show()
```



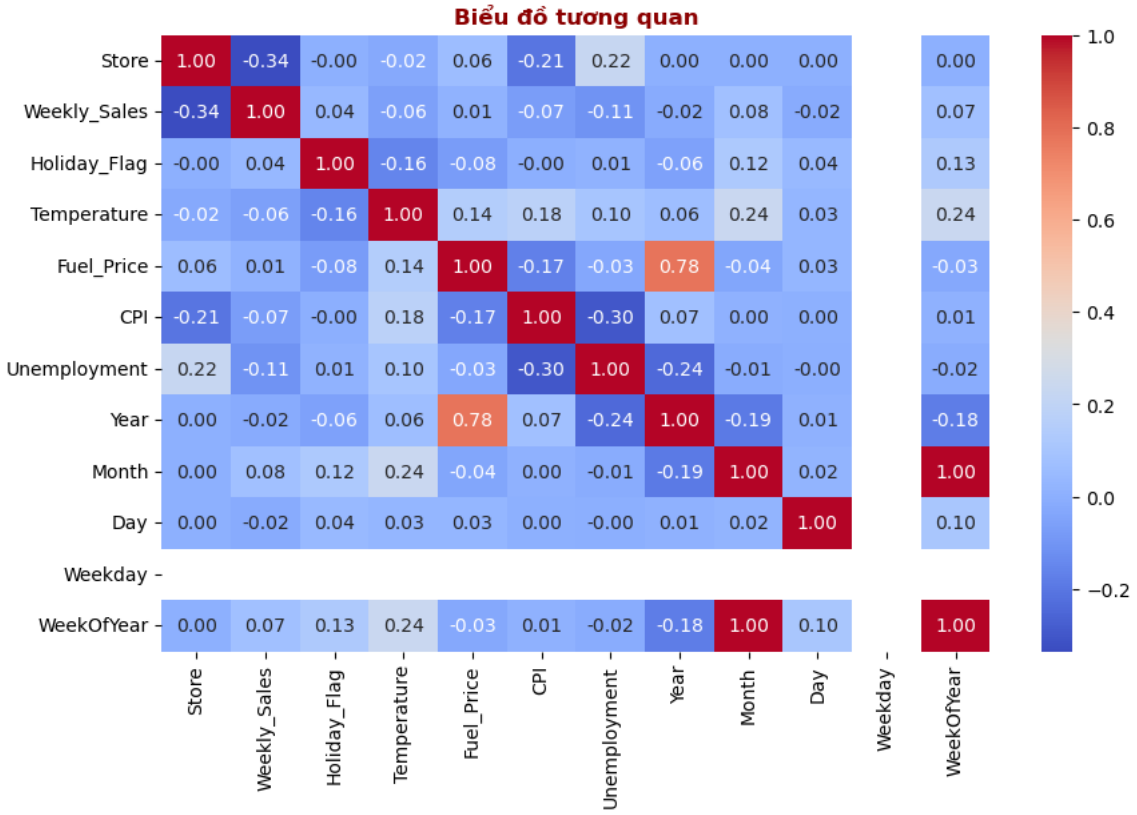
Hình 2.9. Doanh số trung bình vào tuần lễ và tuần bình thường

Các tuần có ngày lễ giúp doanh số trung bình tăng thêm 7,8% so với các tuần bình thường. Đây chính là cơ hội chiến lược để Walmart có thể đẩy mạnh doanh số.

Khuyến nghị: Triển khai chiến dịch marketing mạnh, ưu đãi đặc biệt và sự kiện thu hút khách hàng đúng vào các tuần có ngày lễ.

- Phân tích các yếu tố bên ngoài:

```
plt.figure(figsize=(10, 6))
corr = sales_df.corr(numeric_only=True)
sb.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Biểu đồ tương quan', fontsize=12, fontweight='bold', color='darkred')
plt.show()
```



Hình 2.10. Biểu đồ tương quan

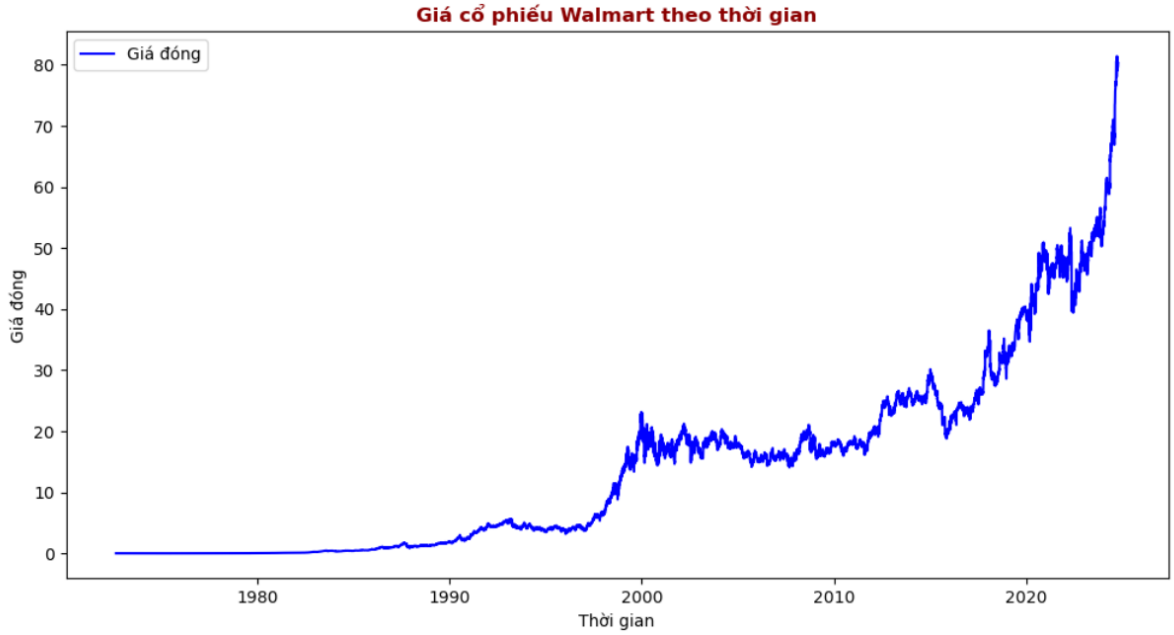
Mối tương quan giữa nhiệt độ và doanh số hàng tuần rất nhẹ (hệ số ≈ 0,06), thời tiết ấm hơn có thể làm doanh số tăng nhẹ. Ngược lại, chỉ số CPI và tỷ lệ thất nghiệp hầu như không tương quan với doanh số, cho thấy các chỉ số này không làm ảnh hưởng tới doanh thu bán hàng.

Khuyến nghị: Nhiệt độ chỉ ảnh hưởng nhẹ nên chưa cần điều chỉnh chiến lược kinh doanh, nên tập trung các yếu tố trên để thúc đẩy doanh số. Vẫn nên theo các biến kinh tế vĩ mô, nhưng không xem là yếu tố quyết định kế hoạch bán hàng.

2.3.2. Phân tích giá cổ phiếu

- Phân tích xu hướng dựa trên ngày:

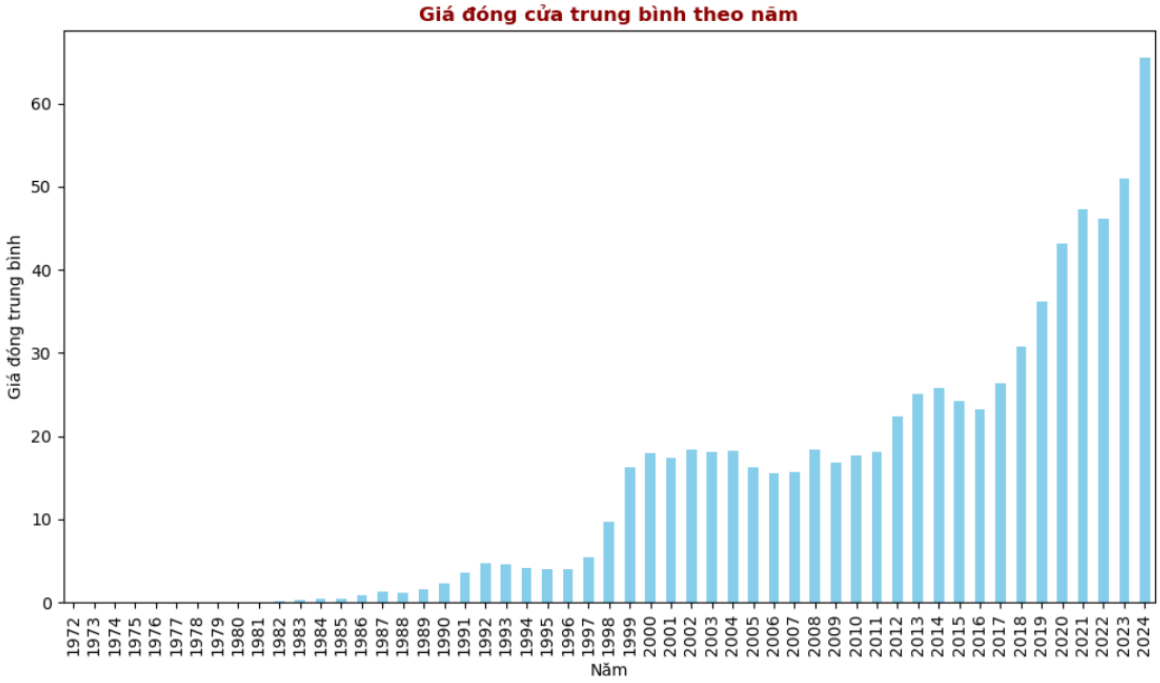
```
plt.figure(figsize=(12, 6))
plt.plot(stock_df.index, stock_df['Close'], color='blue', label='Giá đóng')
plt.title('Giá cổ phiếu Walmart theo thời gian', fontsize=12, fontweight='bold', color='darkred')
plt.xlabel('Thời gian')
plt.ylabel('Giá đóng')
plt.legend()
plt.show()
```



Hình 2.11. Giá cổ phiếu Walmart theo thời gian

Phân tích trên cho thấy giá cổ phiếu Walmart có xu hướng tăng liên tục kể từ đầu những năm 1990, phản ánh sự tăng trưởng mạnh mẽ và niềm tin của nhà đầu tư. Mặc dù, biến động đáng kể xuất hiện trong giai đoạn suy thoái kinh tế, điển hình là cuộc khủng hoảng tài chính năm 2008, nhưng cổ phiếu đã nhanh chóng phục hồi.

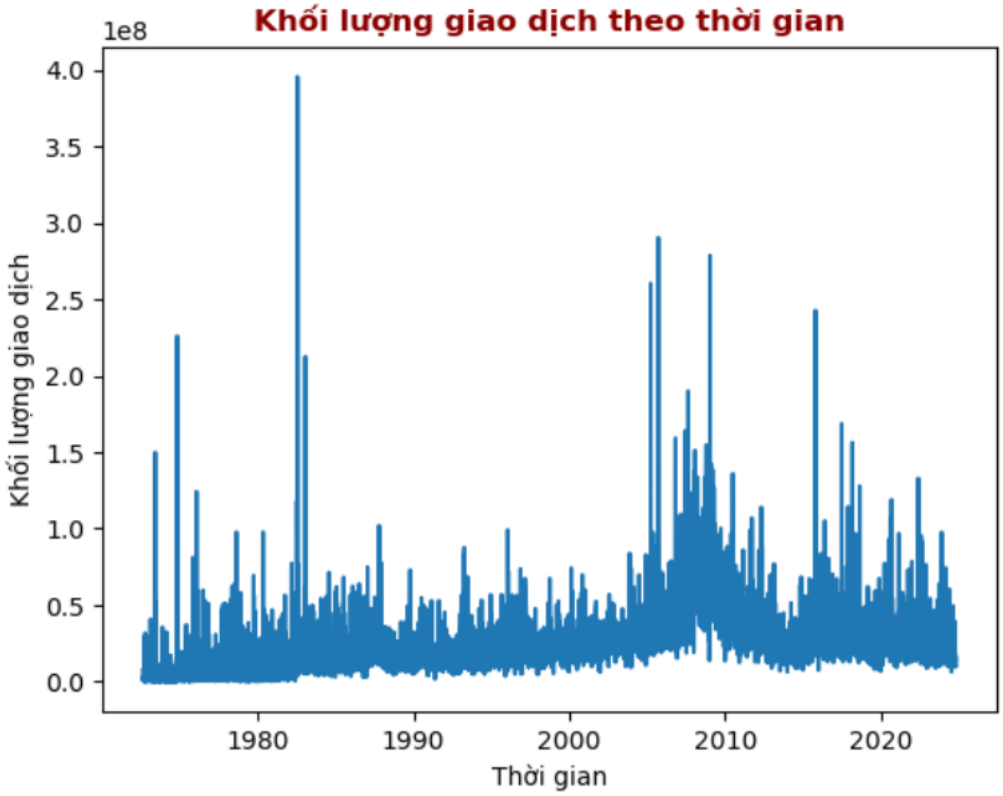
```
stock_df['Year'] = stock_df.index.year
avg_close_per_year = stock_df.groupby('Year')['Close'].mean()
plt.figure(figsize=(10, 6))
avg_close_per_year.plot(kind='bar', color='skyblue')
plt.title('Giá đóng cửa trung bình theo năm', fontsize=12, fontweight='bold', color='darkred')
plt.xlabel('Năm')
plt.ylabel('Giá đóng trung bình')
plt.tight_layout()
plt.show()
```



Hình 2.12. Giá đóng cửa trung bình theo năm

- Phân tích khối lượng cổ phiếu giao dịch:

```
plt.plot(stock_df.index.date, stock_df['Volume'])
plt.title('Khối lượng giao dịch theo thời gian', fontsize=12, fontweight='bold', color='darkred')
plt.xlabel('Thời gian')
plt.ylabel('Khối lượng giao dịch')
plt.show()
```

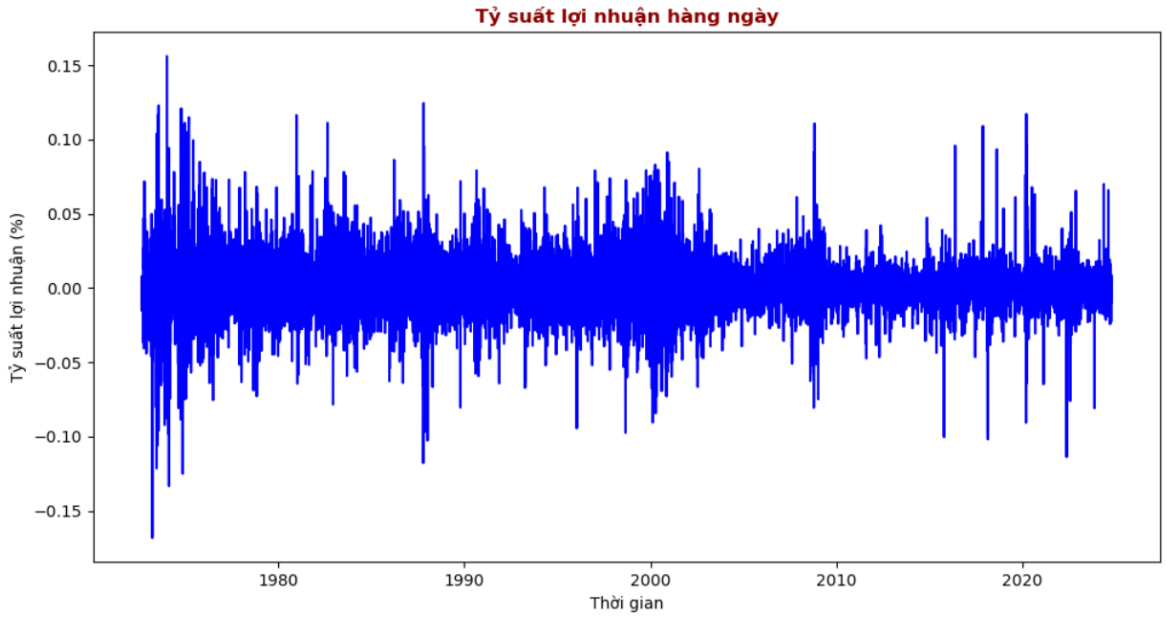


Hình 2.13. Lượng cổ phiếu giao dịch theo thời gian

Lượng cổ phiếu giao dịch theo ngày được mô tả trong biểu đồ dưới đây. Ngày giao dịch với lượng cổ phiếu đạt cao nhất nằm trong năm 1982 đạt gần 400 triệu cổ phiếu/ngày. Năm 1982 cũng là năm có nhiều phiên giao dịch với khối lượng cổ phiếu nhất so với các năm về sau.

- Tỷ suất lợi nhuận:

```
stock_df['Daily Return'] = stock_df['Close'].pct_change()
plt.figure(figsize=(12, 6))
plt.plot(stock_df.index, stock_df['Daily Return'], color='blue')
plt.title('Tỷ suất lợi nhuận hàng ngày', fontsize=12, fontweight='bold', color='darkred')
plt.xlabel('Thời gian')
plt.ylabel('Tỷ suất lợi nhuận (%)')
plt.show()
```

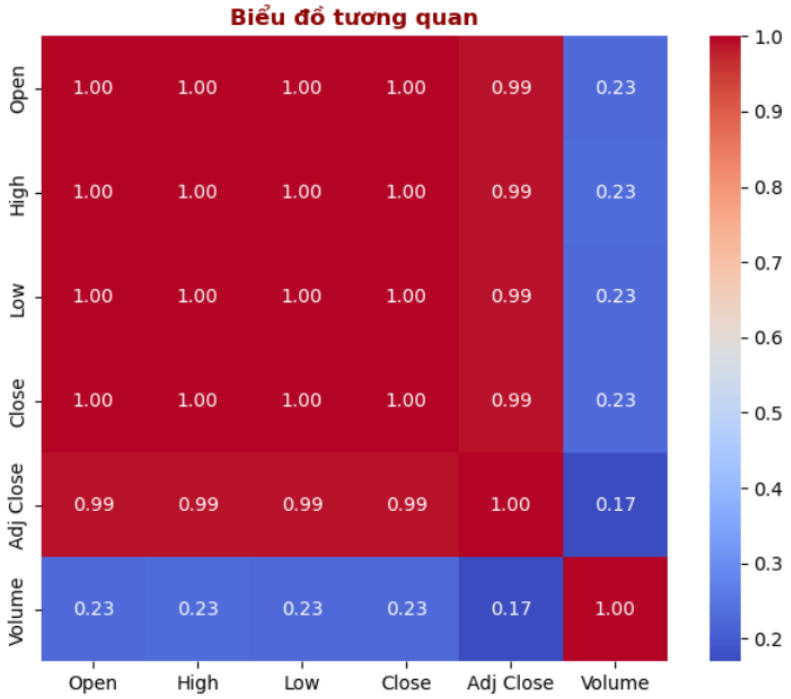


Hình 2.14. Biểu đồ tỷ suất lợi nhuận

Nhìn chung, tỷ suất lợi nhuận tập trung chủ yếu quanh giá trị 0. Điều này cho thấy đa số các ngày giao dịch, tỷ suất lợi nhuận của Walmart biến động không lớn. Một số ngày có tỷ suất lợi nhuận âm thấp nhất là -0,17 và dương lớn khoảng 0,15. Tuy nhiên, tần suất xảy ra thấp, thể hiện các biến động bất thường là hiếm.

- Phân tích tương quan:

```
correlation_data = stock_df[['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']]
correlation_matrix = correlation_data.corr()
plt.figure(figsize=(8,6))
sb.heatmap(correlation_matrix, annot=True, fmt='.2f', cmap='coolwarm', square=True)
plt.title('Biểu đồ tương quan', fontweight='bold', color='darkred')
plt.show()
```



Hình 2.15. Biểu đồ tương quan

Phân tích tương quan cho thấy mối quan hệ dương mạnh giữa giá đóng cửa và khối lượng giao dịch của Walmart, tức là hoạt động giao dịch cao hơn thường đi kèm với giá tăng.

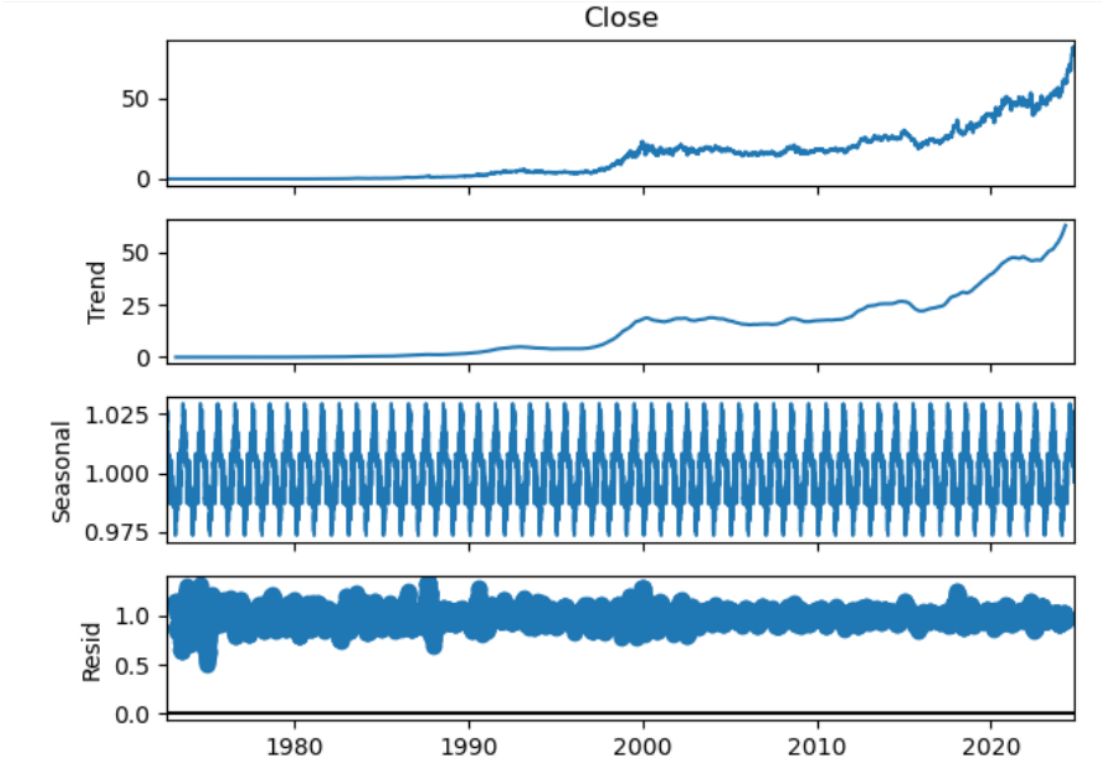
Ngoài ra, phân tích cho thấy mối tương quan chặt chẽ giữa giá mở cửa, giá cao, giá thấp và giá đóng cửa, phản ánh sự dịch chuyển đồng pha theo xu hướng thị trường.

- Dự báo chuỗi thời gian:

Để dự báo chuỗi thời gian, cần khai thác dữ liệu quá khứ để dự đoán biến động giá trong tương lai, sử dụng hàm seasonal_decompose() với chu kỳ 252 phiên (xấp xỉ 1 năm giao dịch).

Trong phân tích dữ liệu chuỗi thời gian, xu hướng, mùa vụ và chu kỳ là 3 thành phần quan trọng giúp hiểu rõ cấu trúc của dữ liệu.

```
import statsmodels.api as sm
decomposition = sm.tsa.seasonal_decompose(stock_df['Close'], model='multiplicative', period=252)
plt.figure(figsize=(10, 8))
decomposition.plot()
plt.show()
```



Hình 2.16. Biểu đồ dự báo chuỗi thời gian

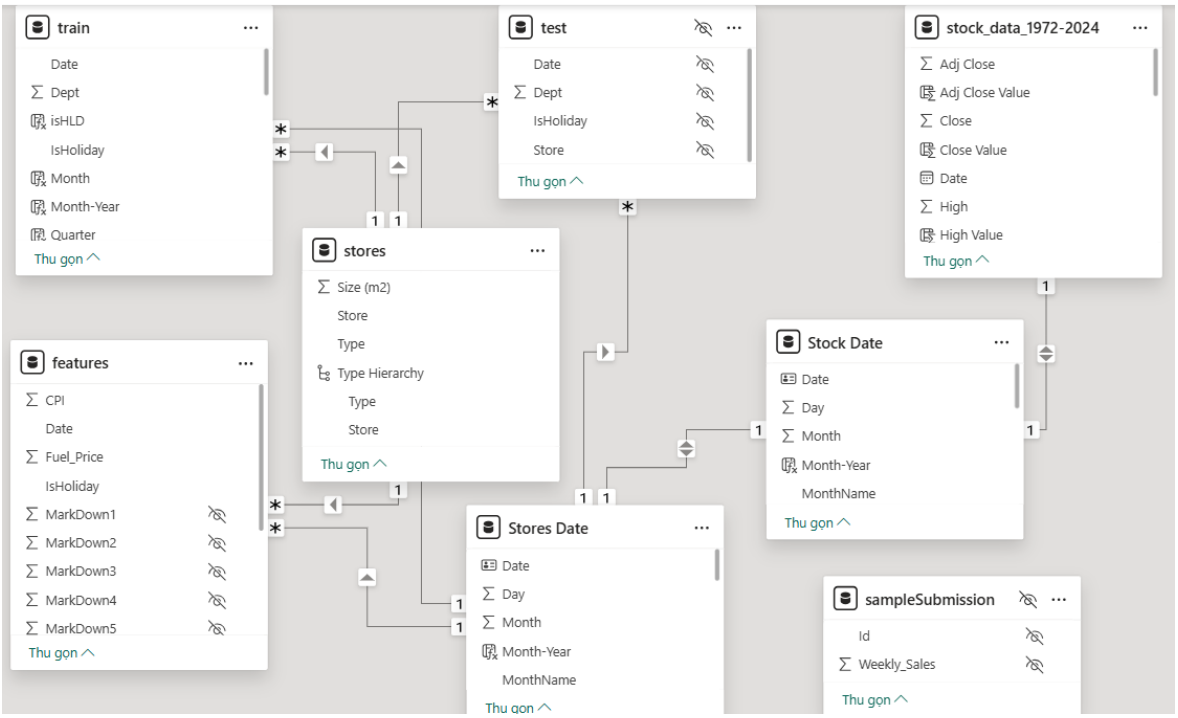
Phân tích cho thấy cổ phiếu của Walmart có xu hướng tăng rõ rệt, đặc biệt mạnh từ giai đoạn sau năm 2000. Điều này cho thấy yếu tố xu hướng (trend) chiếm vai trò quan trọng trong biến động dữ liệu.

Thành phần mùa vụ xuất hiện với chu kỳ dao động đều đặn, biên độ nhỏ (dao động quanh 1.0 từ 0.975 đến 1.025). Mặc dù mức độ biến động không lớn, sự lặp lại theo chu kỳ cho thấy dữ liệu có yếu tố mùa vụ nhẹ.

2.4. Phân tích và trực quan hóa dữ liệu với Power BI

Dashboard được xây dựng trên nền tảng Power BI nhằm mục tiêu trực quan hóa dữ liệu kinh doanh và dữ liệu cổ phiếu của Walmart. Giao diện người dùng được thiết kế sinh động, trực quan, giúp người dùng dễ dàng tương tác và theo dõi các biến động kinh doanh cũng như thị trường chứng khoán của Walmart.

2.4.1. Mô hình hóa dữ liệu



Hình 2.17. Mô hình hóa dữ liệu trên Power BI

- Sau khi tải và xử lý dữ liệu sơ bộ bằng Python để loại bỏ giá trị thiếu, làm sạch và chuẩn hóa định dạng thời gian. Dữ liệu sẽ được kết nối với Power BI để xây dựng các mô hình quan hệ.
- Áp dụng mô hình dạng sao (star schema), gồm các bảng chính:
  + train, test: Chứa dữ liệu bán hàng theo tuần của từng cửa hàng.
  + features: Thông tin các nhân tố khác ảnh hưởng đến doanh số như CPI, fuel price, temperature,...
  + stock_data_1972-2024: Dữ liệu cổ phiếu Walmart.
  + stores, Stock Date, Stores Date: Các bảng thời gian và thông tin cửa hàng phục vụ làm khóa chính kết nối.
- Các bảng dữ liệu được kết nối thông qua các khóa chính như Date, Store, và Dept.

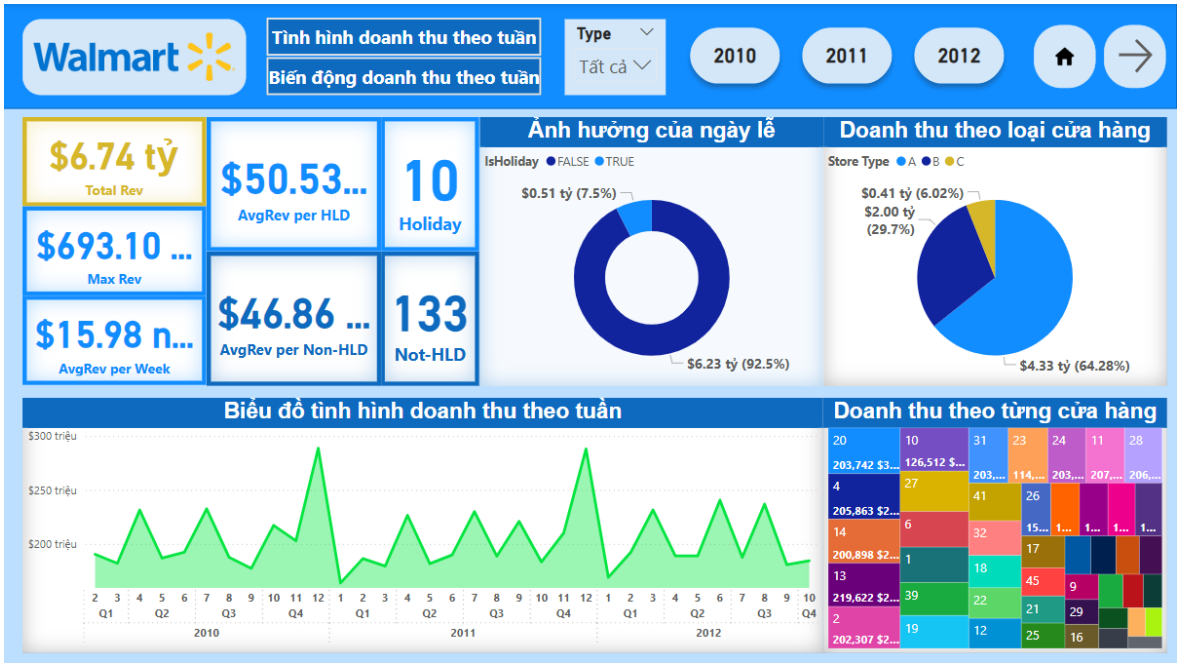
2.4.2. Các thành phần chính trong Dashboard

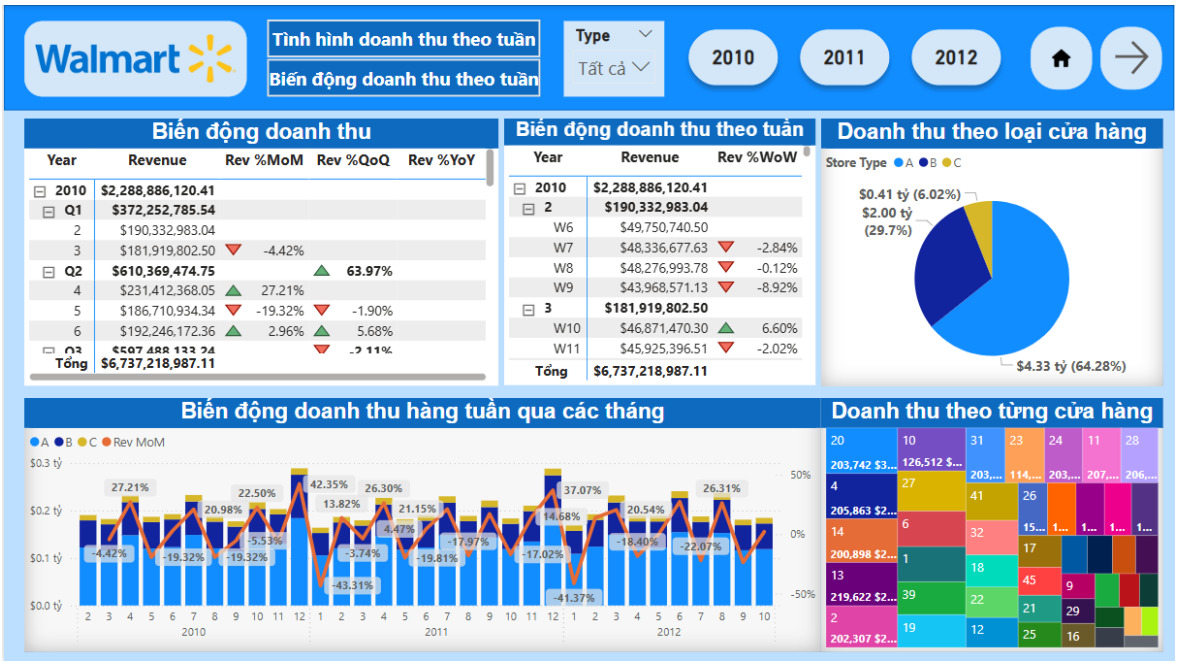
2.4.2.1. Giới thiệu tổng quan

Hình 2.18. Trang giới thiệu tổng quan trên Power BI

- Trang giới thiệu tổng quan trình bày tên đề tài, dữ liệu đầu vào và mục tiêu của đề tài.
- Các nút menu như: SALES, STOCK giúp điều hướng người dùng sang trang phân tích doanh thu và cổ phiếu.

2.4.2.2. Phân tích doanh số bán hàng

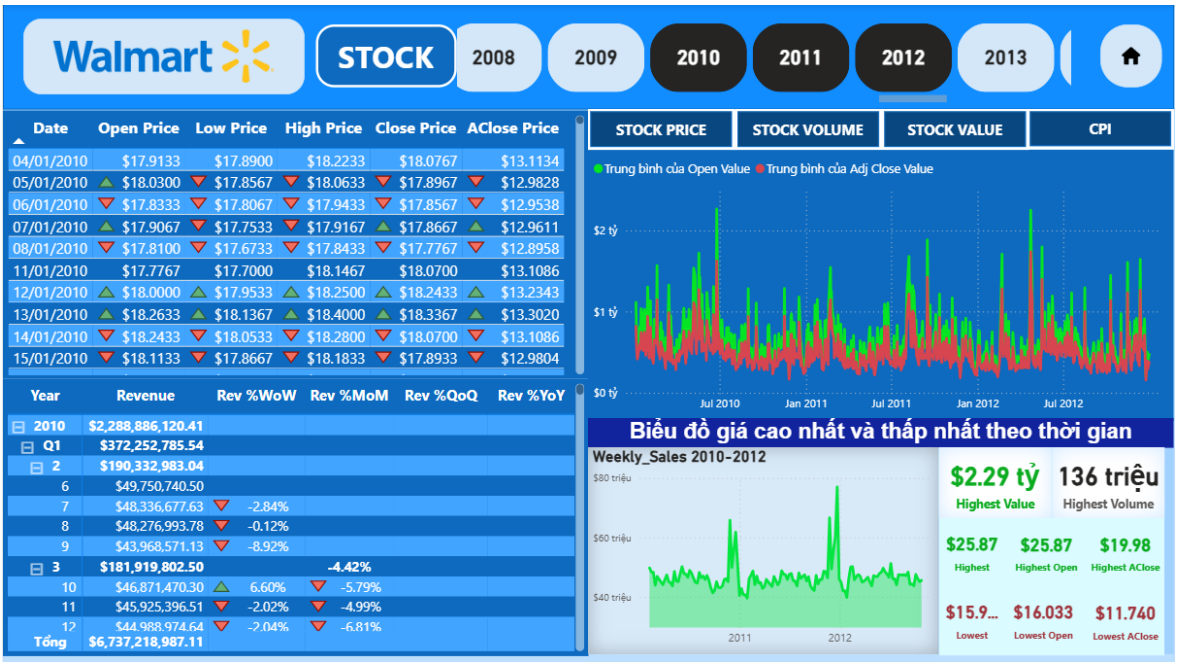




Hình 2.19. Trang phân tích doanh số bán hàng trên Power BI

- Biểu đồ doanh thu theo quý, tháng, tuần: Thể hiện sự biến động doanh thu trong giai đoạn 2010–2012. Ví dụ, Quý 2 năm 2010 ghi nhận tăng trưởng 63.97% doanh thu so với quý trước.
- Tỷ lệ tăng trưởng, gồm:
  + Rev %MoM: Tăng trưởng theo tháng.
  + Rev %QoQ: Theo quý.
  + Rev %YoY: Theo năm.
- Biểu đồ biến động doanh thu theo tháng: Biểu đồ cột kết hợp đường cho thấy các thời điểm tăng/giảm doanh thu mạnh. Ví dụ: tháng 12 năm 2010 giảm mạnh -43.31%, do ảnh hưởng kỳ nghỉ lễ và khuyến mãi.
- Phân tích theo loại cửa hàng và từng cửa hàng:
  + Doanh thu theo loại cửa hàng: Doanh thu được phân theo 3 loại cửa hàng A, B, C. Ví dụ, cửa hàng loại A đóng góp lớn nhất với 64.28% tổng doanh thu.
  + Doanh thu theo từng cửa hàng: Biểu đồ dạng cây (Tree Map) giúp xác định các cửa hàng có doanh thu cao nhất, như cửa hàng số 20, 4 và 14.

2.4.2.3. Phân tích giá trị cổ phiếu



Hình 2.20. Trang phân tích giá trị cổ phiếu trên Power BI

Dashboard này gồm 3 phần chính:

- Bảng dữ liệu giá cổ phiếu: Hiển thị chi tiết các chỉ số giá cổ phiếu như Open Price, Close Price, AClose Price cho từng ngày giao dịch.
- Biểu đồ giá cổ phiếu theo thời gian: So sánh trung bình các giá Open và Aclose Price từ năm 2010 đến 2012. Cho thấy các đỉnh cao bất thường tương ứng với các sự kiện kinh doanh đặc biệt.
- Thống kê cao/thấp: Ghi nhận mức giá cổ phiếu cao nhất là $25.87 và thấp nhất là $15.90, trong khi AClose Price thấp nhất là $11.74. Khối lượng giao dịch cao nhất đạt 136 triệu cổ phiếu.

2.5. Đánh giá nhận xét

Việc kết hợp Python và Power BI trong quá trình phân tích dữ liệu giúp tận dụng tốt sức mạnh của cả hai công cụ:

- Python hỗ trợ xử lý dữ liệu linh hoạt, mạnh mẽ, đặc biệt hiệu quả trong việc làm sạch dữ liệu, phân tích thống kê và xây dựng mô hình dự báo (như mô hình Holt-Winters cho cổ phiếu Walmart).

- Power BI cung cấp giao diện trực quan, dễ sử dụng, cho phép xây dựng các dashboard tương tác, từ đó giúp người dùng phi kỹ thuật (như quản lý doanh nghiệp) dễ dàng tiếp cận thông tin và đưa ra quyết định nhanh chóng.

Nhìn chung, sự kết hợp này mang lại quy trình phân tích toàn diện — từ xử lý dữ liệu thô đến trình bày kết quả trực quan, hiệu quả và có tính ứng dụng cao trong thực tiễn doanh nghiệp.

CHƯƠNG 3. MÔ HÌNH DỰ BÁO VÀ CÀI ĐẶT BÀI TOÁN

3.1. Bài toán

Bài toán xây dựng hệ thống dự báo giá cổ phiếu Walmart được thực hiện dựa trên bộ dữ liệu Walmart_Stock.csv, bao gồm các trường dữ liệu đầu vào như: Thời gian giao dịch, giá mở cửa, giá đóng cửa, giá cao nhất, thấp nhất, giá đóng cửa điều chỉnh và khối lượng giao dịch. Trong bài toán này, mục tiêu chính là dự báo giá đóng cửa (Close) của cổ phiếu Wallmart theo thời gian.

Bài toán áp dụng mô hình Holt-Winters – một mô hình dự báo chuỗi thời gian có khả năng xử lý dữ liệu có xu hướng và mùa vụ, phù hợp với đặc điểm dao động theo chu kỳ của giá cổ phiếu trên thị trường tài chính. Mô hình Holt-Winters được áp dụng với các thành phần xu hướng và mùa vụ, kết hợp với quá trình tối ưu hóa tham số (alpha, beta, gamma) để giảm sai số dự báo.

Bài toán được thực hiện bằng ngôn ngữ lập trình Python, một ngôn ngữ mạnh mẽ và linh hoạt với nhiều thư viện hỗ trợ phân tích và dự báo chuỗi thời gian như pandas, statsmodels, matplotlib và scikit-learn. Trong quá trình triển khai, bài toán thêm sử dụng các hàm có sẵn của thư viện statsmodels để huấn luyện mô hình, tính toán các chỉ số đánh giá như MAE, MSE, RMSE và MAPE để đánh giá hiệu suất dự báo.

Sau khi mô hình đạt được kết quả tối ưu, chương trình dự báo được xây dựng trên nền tảng Streamlit, cho phép người dùng có thể chọn khoảng thời gian cần phân tích và tương tác với số liệu biểu đồ. Qua đó, giúp người dùng có được cái nhìn trực quan và hỗ trợ ra quyết định trong đầu tư cổ phiếu.

3.2. Các mô hình dự báo

3.2.1. Mô hình dự báo Moving Average

3.2.1.1. Ý tưởng thuật toán

Moving Average (MA) là một kỹ thuật làm mịn chuỗi thời gian bằng cách lấy trung bình cộng các giá trị trong một khoảng trượt (window size) cố định.

Mục tiêu của MA là loại bỏ nhiễu ngắn hạn để làm nổi bật xu hướng chính trong dữ liệu (ví dụ: biến động giá cổ phiếu).

3.2.1.2. Vấn đề giải pháp

Dữ liệu chuỗi thời gian thực tế (như giá cổ phiếu, doanh số bán hàng) thường chứa nhiễu lớn và dao động thất thường. Các mô hình dự báo dễ bị nhiễu làm sai lệch dự đoán nếu không làm mịn dữ liệu đầu vào.

Để giải quyết vấn đề dữ liệu chuỗi thời gian bị nhiễu và dao động bất thường, giải pháp được đề xuất là áp dụng thuật toán MA nhằm làm mịn dữ liệu. Bằng cách tính trung bình các giá trị trong một khoảng trượt (window size) cố định, từ đó giảm thiểu tác động của các biến động ngắn hạn và làm nổi bật xu hướng chính của dữ liệu.

Việc lựa chọn kích thước khoảng trượt phù hợp đóng vai trò rất quan trọng: khoảng trượt nhỏ sẽ cho phép mô hình phản ứng nhanh với những thay đổi mới, nhưng có thể giữ lại nhiều nhiễu, trong khi khoảng trượt lớn sẽ giúp dữ liệu mượt hơn, dễ nhận diện xu hướng nhưng phản ứng chậm hơn với các biến động mới.

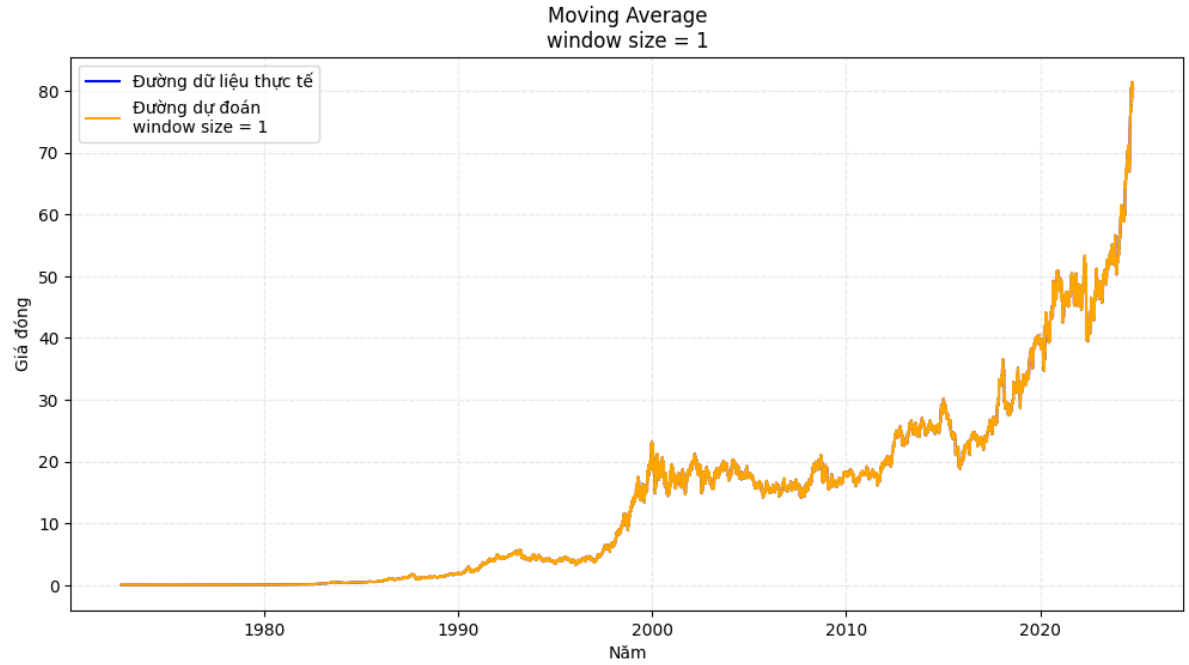
Bằng cách điều chỉnh linh hoạt kích thước khoảng trượt, MA có thể hỗ trợ hiệu quả cho việc phân tích và dự báo chuỗi thời gian.

3.2.1.3. Triển khai thuật toán

- Mô hình MA với khoảng trượt (window size) bằng 1:

Với khoảng trượt (window size) bằng 1, mỗi điểm giá trị trung bình chính là giá trị gốc, do đó không có bất kỳ làm mịn nào xảy ra và đường dự đoán khớp hoàn toàn với dữ liệu thực. Mô hình này sử dụng để tham khảo và so sánh với các mô hình khác.

```python
#Mô hình Moving Average với window size = 1
#Tính giá trị trung bình động
window_size = 1
MA_ws1 = stock_df['Close'].rolling(window=window_size).mean()
#Vẽ biểu đồ
plt.figure(figsize=(12,6))
plt.plot(stock_df.index, stock_df['Close'], color='blue', label='Đường dữ liệu thực tế')
plt.plot(MA_ws1, label=f'Đường dự đoán\nwindow size = {window_size}', color='orange')
plt.title(f'Moving Average\nwindow size = {window_size}')
plt.xlabel('Năm')
plt.ylabel('Giá đóng')
plt.legend()
plt.grid(True, which='both', linestyle='--', alpha=0.3)
plt.show()
#Các chỉ số đo độ lệch
mae_ws1 = mean_absolute_error(stock_df['Close'], MA_ws1)
mse_ws1 = mean_squared_error(stock_df['Close'], MA_ws1)
rmse_ws1 = np.sqrt(mse_ws1)
print(f'Mean Absolute Error (MAE): ', mae_ws1)
print(f'Mean Squared Error (MSE): ', mse_ws1)
print(f'Root Mean Squared Error (RMSE): ', rmse_ws1)
```



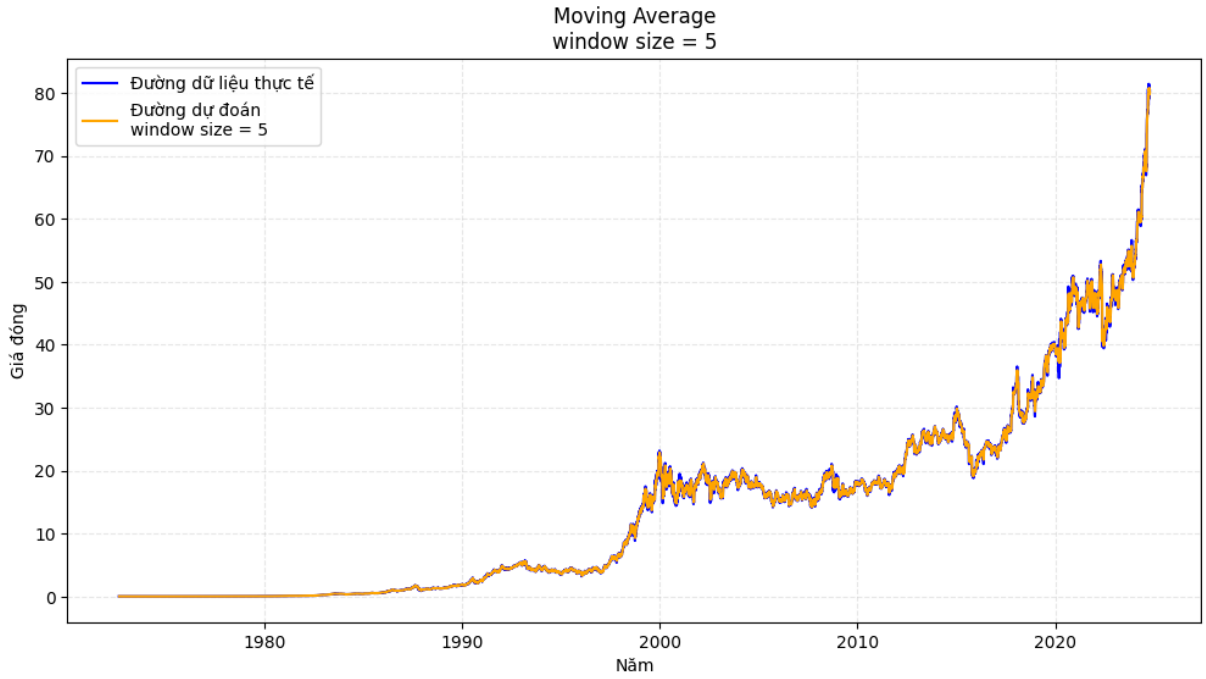

```
Mean Absolute Error (MAE):  0.0
Mean Squared Error (MSE):  0.0
Root Mean Squared Error (RMSE):  0.0
```

Hình 3.1. Biểu đồ mô hình MA với khoảng trượt bằng 1

- Mô hình MA với khoảng trượt (window size) bằng 5:

Với khoảng trượt (window size) bằng 5, mô hình chỉ tập trung vào dữ liệu gần nhất (5 phiên), điều này giúp giảm nhiễu ngắn hạn nhưng vẫn phản ứng nhanh với thay đổi mới.

```
#Mô hình Moving Average với window size = 5
#Tính giá trị trung bình động
window_size = 5
MA_ws5 = stock_df['Close'].rolling(window=window_size).mean()
#Vẽ biểu đồ
plt.figure(figsize=(12,6))
plt.plot(stock_df.index, stock_df['Close'], color='blue', label='Đường dữ liệu thực tế')
plt.plot(MA_ws5, label=f'Đường dự đoán\nwindow size = {window_size}', color='orange')
plt.title(f'Moving Average\nwindow size = {window_size}')
plt.xlabel('Năm')
plt.ylabel('Giá đóng')
plt.legend()
plt.grid(True, which='both', linestyle='--', alpha=0.3)
plt.show()
#Các chỉ số đo độ lệch
mae_ws5 = mean_absolute_error(stock_df['Close'][4:], MA_ws5.dropna())
mse_ws5 = mean_squared_error(stock_df['Close'][4:], MA_ws5.dropna())
rmse_ws5 = np.sqrt(mse_ws5)
print(f'Mean Absolute Error (MAE): {mae_ws5}')
print(f'Mean Squared Error (MSE): {mse_ws5}')
print(f'Root Mean Squared Error (RMSE): {rmse_ws5}')
```



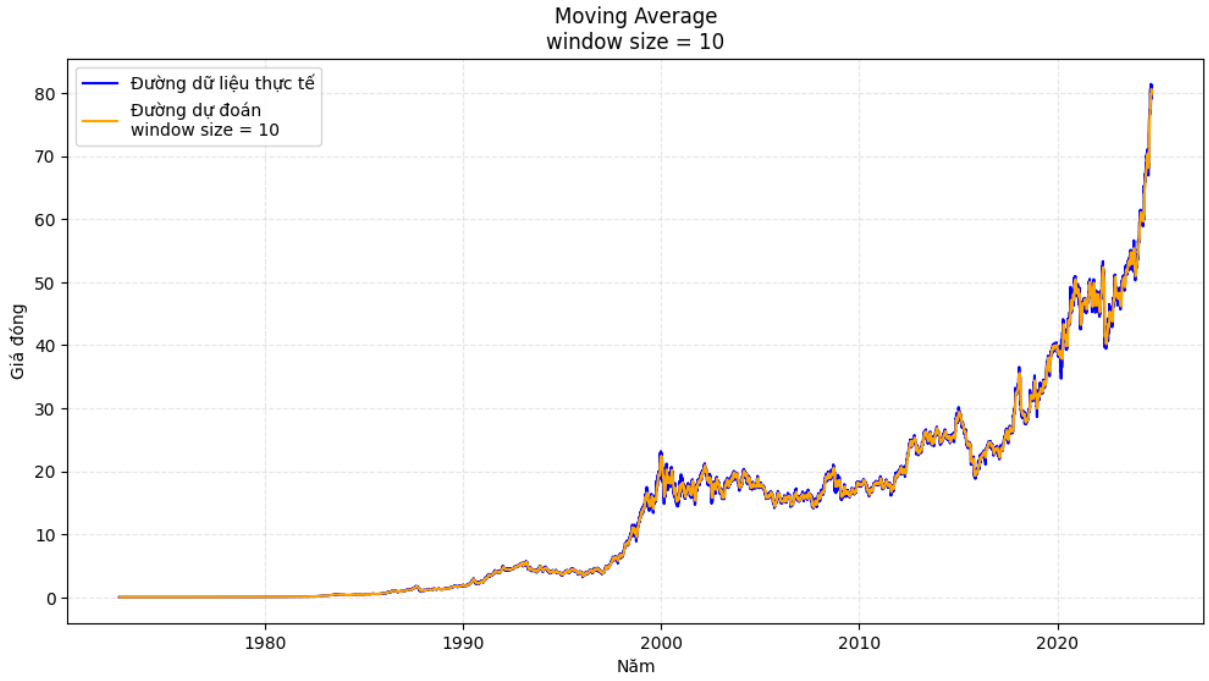

```
Mean Absolute Error (MAE): 0.14924038722742564
Mean Squared Error (MSE): 0.0903935441423181
Root Mean Squared Error (RMSE): 0.3006551914441494
```

Hình 3.2. Biểu đồ mô hình MA với khoảng trượt bằng 5

- Mô hình MA với khoảng trượt (window size) bằng 10:

Với khoảng trượt (window size) bằng 10, mô hình mịn hơn so với giá thực tế, giúp phát hiện xu hướng ngắn hạn rõ ràng hơn và dễ sử dụng trong thực tế.

```
#Mô hình Moving Average với window size = 10
#Tính giá trị trung bình động
window_size = 10
MA_ws10 = stock_df['Close'].rolling(window=window_size).mean()
#Vẽ biểu đồ
plt.figure(figsize=(12,6))
plt.plot(stock_df.index, stock_df['Close'], color='blue', label='Đường dữ liệu thực tế')
plt.plot(MA_ws10, label=f'Đường dự đoán\nwindow size = {window_size}', color='orange')
plt.title(f'Moving Average\nwindow size = {window_size}')
plt.xlabel('Năm')
plt.ylabel('Giá đóng')
plt.legend()
plt.grid(True, which='both', linestyle='--', alpha=0.3)
plt.show()
#Các chỉ số đo độ lệch
mae_ws10 = mean_absolute_error(stock_df['Close'][9:], MA_ws10.dropna())
mse_ws10 = mean_squared_error(stock_df['Close'][9:], MA_ws10.dropna())
rmse_ws10 = np.sqrt(mse_ws10)
print(f'Mean Absolute Error (MAE): {mae_ws10}')
print(f'Mean Squared Error (MSE): {mse_ws10}')
print(f'Root Mean Squared Error (RMSE): {rmse_ws10}')
```




```
Mean Absolute Error (MAE): 0.2268693631514464
Mean Squared Error (MSE): 0.19876894248603497
Root Mean Squared Error (RMSE): 0.44583510683439337
```

Hình 3.3. Biểu đồ mô hình MA với khoảng trượt bằng 10

3.2.1.4. Ưu, nhược điểm

- Ưu điểm:
  + Là một công cụ đơn giản, dễ dàng tính toán và hiểu để dự báo xu hướng.
  + Giúp xác định hướng của giá (tăng, giảm hoặc đi ngang).

+ Có thể loại bỏ các tín hiệu giao dịch nhiễu, giúp nhà đầu tư tập trung vào xu hướng chính.
+ Có thể được sử dụng để xác định các mức giá tiềm năng làm hỗ trợ hoặc kháng cự cho giá.

- Nhược điểm:
  + Là một chỉ báo trễ, nghĩa là nó phản ứng sau khi giá đã thay đổi, không thể phản ánh chính xác các biến động ngắn hạn.
  + Do tính chất trễ, nên không phản ánh được các biến động mạnh hoặc đột ngột của giá trong ngắn hạn.
  + Chu kỳ của MA (ví dụ: MA20, MA50) sẽ ảnh hưởng đến độ trễ và độ chính xác của chỉ báo.
  + Có thể tạo ra các tín hiệu giả, đặc biệt là trong các thị trường có biến động mạnh hoặc giao dịch nhỏ.

3.2.2. Mô hình dự báo Simple Exponential Smoothing

3.2.2.1. Ý tưởng thuật toán

Mô hình Simple Exponential Smoothing (SES) giúp làm mịn dữ liệu bằng cách giảm trọng số của các quan sát cũ và tập trung vào những quan sát gần đây. Trọng số được xác định bởi tham số alpha (α), alpha (α) càng nhỏ đường càng mượt, alpha (α) càng lớn đường càng nhạy cảm với thay đổi.

Phương pháp này ưu tiên dữ liệu gần đây hơn, giúp dự báo tốt hơn xu hướng ngắn hạn và giảm độ trễ so với các phương pháp trung bình trượt truyền thống.

3.2.2.2. Vấn đề giải pháp

Khi phân tích chuỗi thời gian, đặc biệt là dữ liệu giá cổ phiếu, có thể có nhiều biến động ngắn hạn và nhiễu không cần thiết. Điều này làm cho việc dự báo tương lai dựa trên dữ liệu quá khứ trở nên khó khăn và không chính xác.

Bằng việc sử dụng mô hình SES để làm mượt chuỗi thời gian, giúp giảm ảnh hưởng của những biến động ngắn hạn (nhiễu) và giữ lại xu hướng chính. Thuật toán này sử dụng hệ số trọng số alpha (α) để điều chỉnh mức độ "mượt" của dữ liệu, từ đó đưa ra dự báo chính xác hơn.

Alpha (α) là thông số quan trọng trong SES, giúp kiểm tra độ quan trọng của dữ liệu mới và dữ liệu cũ. Alpha (α) sẽ nằm trong khoảng từ 0.0 đến 1.0, độ quan trọng sẽ thể hiện trực tiếp qua trọng số:

- Alpha (α) cao: Tăng trọng số cho dữ liệu mới => Tăng độ quan trọng cho dữ liệu mới.
- Alpha (α) thấp: Tăng trọng số cho dữ liệu cũ => Tăng độ quan trọng cho dữ liệu cũ.

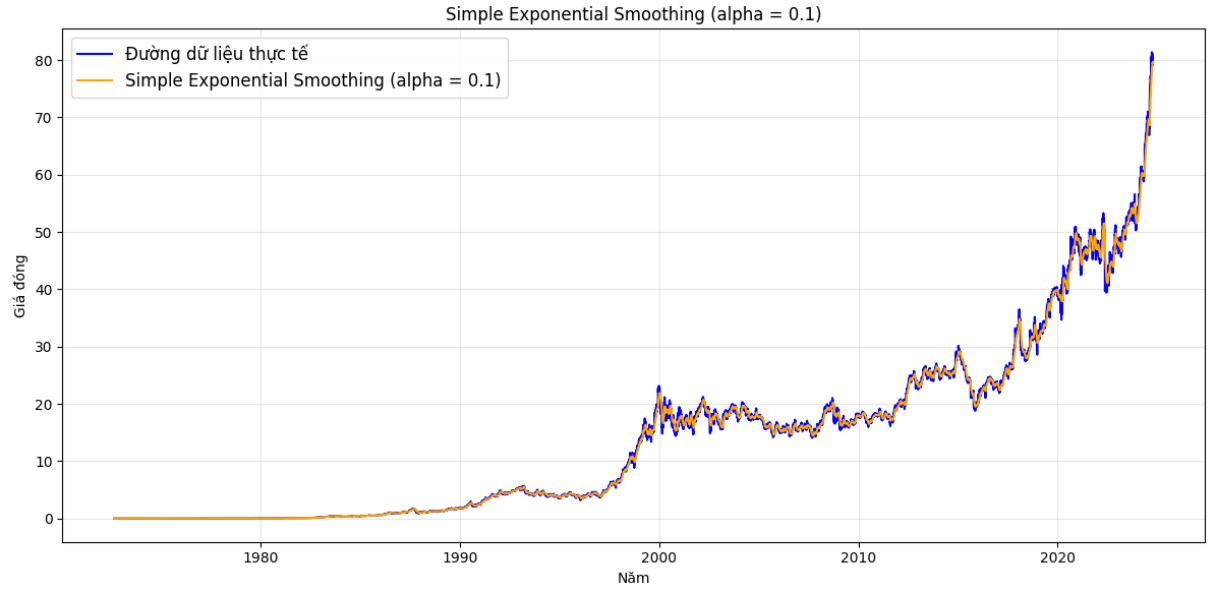
  (Điểm cần lưu ý là dữ liệu cũ sẽ không có trọng số cao hơn dữ liệu mới).

3.2.2.3. Triển khai thuật toán

- Mô hình SES với hệ số alpha α = 0.1:

Mô hình SES với hệ số alpha α = 0.1 làm mịn tốt và đường dự báo loại bỏ được phần lớn nhiễu ngắn hạn. Với alpha (α) nhỏ, mô hình phản ứng chậm với biến động lớn hoặc thay đổi bất thường trong giá cổ phiếu.

```
#Mô hình Simple Exponential Smoothing với hệ số alpha = 0.1
alpha = 0.1
model = SimpleExpSmoothing(stock_df['Close']).fit(smoothing_level=alpha, optimized=False)
SES = model.fittedvalues
#Vẽ biểu đồ
plt.figure(figsize=(12,6))
plt.plot(stock_df.index, stock_df['Close'], label='Đường dữ liệu thực tế', color='blue')
plt.plot(SES, label=f'Simple Exponential Smoothing (alpha = {alpha})', color = 'orange')
plt.title('Simple Exponential Smoothing (alpha = 0.1)')
plt.xlabel('Năm')
plt.ylabel('Giá đóng')
plt.legend(fontsize=12)
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
#Các chỉ số đo độ lệch
mae_alpha1 = mean_absolute_error(stock_df['Close'], SES)
mse_alpha1 = mean_squared_error(stock_df['Close'], SES)
rmse_alpha1 = np.sqrt(mse_alpha1)
print(f'Mean Absolute Error (MAE): {mae_alpha1}')
print(f'Mean Squared Error (MSE): {mse_alpha1}')
print(f'Root Mean Squared Error (RMSE): {rmse_alpha1}')
```




```
Mean Absolute Error (MAE): 0.3090460904500785
Mean Squared Error (MSE): 0.35721593914520977
Root Mean Squared Error (RMSE): 0.5976754463295357
```

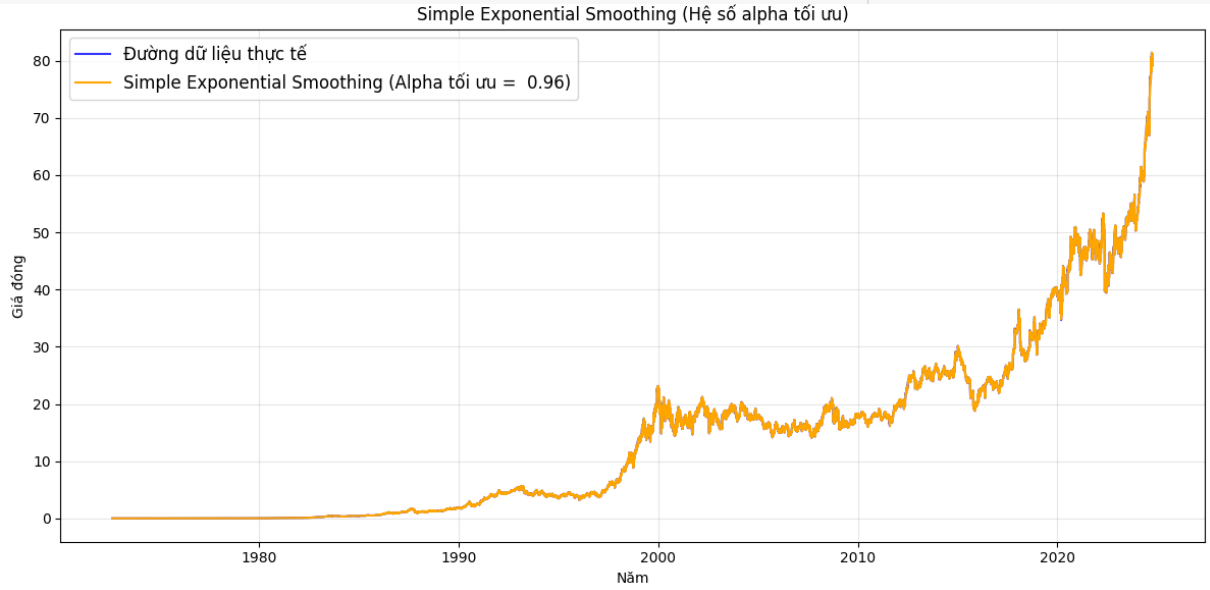
Hình 3.4. Biểu đồ mô hình SES với hệ số alpha bằng 0.1

- Mô hình SES với hệ số alpha (α) tối ưu:

Mô hình SES với alpha (α) tối ưu tối ưu hóa độ mịn, phản ứng hiệu quả với xu hướng của chuỗi thời gian nhưng vẫn duy trì độ ổn định, loại bỏ nhiễu ngắn hạn.

```
#Mô hình Simple Exponential Smoothing với hệ số alpha tối ưu
#Hàm tìm alpha tối ưu
def optimize_alpha (data, alpha_values):
  best_alpha = None
  best_mse = float('inf')
  for alpha in alpha_values:
    model = SimpleExpSmoothing(data).fit(smoothing_level=alpha, optimized=False)
    fitted_values = model.fittedvalues
    mse = mean_squared_error(data, fitted_values)
    if mse < best_mse:
      best_mse = mse
      best_alpha = alpha
  return best_alpha, best_mse
#Tập giá trị alpha để thử nghiệm
alpha_range = np.linspace(0.01, 1.0, 100)
optimal_alpha, optimal_mse = optimize_alpha(stock_df['Close'], alpha_range)
#Áp dụng mô hình với alpha tối ưu
model = SimpleExpSmoothing(stock_df['Close']).fit(smoothing_level=optimal_alpha, optimized=False)
SES_opt = model.fittedvalues
```

```
#Vẽ biểu đồ
plt.figure(figsize=(12,6))
plt.plot(stock_df['Close'], label='Đường dữ liệu thực tế', color='blue', alpha=0.8)
plt.plot(SES_opt, label=f'Simple Exponential Smoothing (Alpha tối ưu = {optimal_alpha: .2f})', color = 'orange')
plt.title('Simple Exponential Smoothing (Hệ số alpha tối ưu)')
plt.xlabel('Năm')
plt.ylabel('Giá đóng')
plt.legend(fontsize=12)
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
#Các chỉ số đo độ lệch
mae_SES_opt = mean_absolute_error(stock_df['Close'], SES_opt)
mse_SES_opt = mean_squared_error(stock_df['Close'], SES_opt)
rmse_SES_opt = np.sqrt(mse_SES_opt)
print(f'Mean Absolute Error (MAE): {mae_SES_opt}')
print(f'Mean Squared Error (MSE): {mse_SES_opt}')
print(f'Root Mean Squared Error (RMSE): {rmse_SES_opt}')
```




```
Mean Absolute Error (MAE): 0.137704269836691
Mean Squared Error (MSE): 0.08073033634355828
Root Mean Squared Error (RMSE): 0.28413084370331615
```

Hình 3.5. Biểu đồ mô hình SES với hệ số alpha tối ưu

3.2.2.4. Ưu, nhược điểm

- Ưu điểm:
  + Công thức tính toán đơn giản, dễ dàng triển khai trong các công cụ phân tích dữ liệu như Excel.
  + Mô hình hoạt động tốt khi dự báo cho các khoảng thời gian ngắn.
  + Được sử dụng rộng rãi trong các lĩnh vực như dự báo giá cổ phiếu, dự báo doanh số, dự báo nhu cầu,...
- Nhược điểm:
  + Chỉ phù hợp với chuỗi thời gian không có sự biến động quá mạnh.
  + Khó mô hình hóa mùa vụ và xu hướng phức tạp.
  + Việc lựa chọn giá trị alpha (α) thích hợp có thể gây khó khăn.

3.2.3. Mô hình dự báo Holt

3.2.3.1. Ý tưởng thuật toán

Mô hình Holt, hay còn gọi là phương pháp làm mịn hàm mũ kép, là một kỹ thuật dự báo chuỗi thời gian được sử dụng để dự đoán các giá trị trong tương lai dựa trên dữ liệu quá khứ.

Nó được mở rộng từ mô hình Exponential Smoothing bằng cách không chỉ làm mịn dữ liệu hiện tại, mà còn ước lượng thêm xu hướng tuyến tính của chuỗi thời gian.

Mô hình Holt dùng hai hệ số:

- Alpha (α): kiểm soát độ mượt của mức giá trị (level).
- Beta (β): kiểm soát độ mượt của xu hướng (trend).

3.2.3.2. Vấn đề giải pháp

Các chuỗi thời gian thường không ổn định, có xu hướng tăng hoặc giảm theo thời gian.

Các mô hình chỉ làm mịn giá trị (như mô hình Exponential Smoothing) không nắm được xu hướng, nên dễ dự báo sai khi chuỗi tăng hoặc giảm mạnh.

Mô hình Holt khắc phục bằng cách:

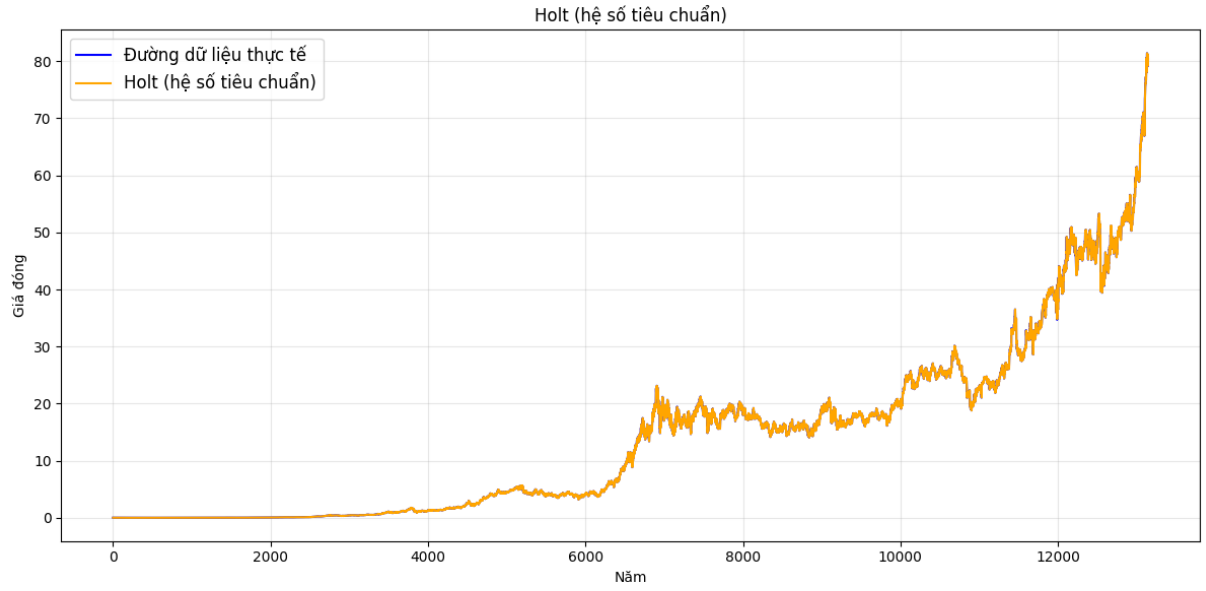
- Tách riêng phần "giá trị hiện tại" và "xu hướng" để dự báo.
- Cập nhật liên tục cả mức và xu hướng theo từng bước thời gian, giúp dự báo bám sát biến động thực tế hơn.

3.2.3.3. Triển khai thuật toán

- Mô hình Holt với hệ số tiêu chuẩn:

Mô hình Holt phù hợp để dự báo chuỗi thời gian có cả mức giá trị và xu hướng tuyến tính. Mô hình sử dụng hai hệ số làm trơn là alpha và beta, tương ứng với mức độ giá trị hiện tại và xu hướng.

```python
#Mô hình Holt với hệ số tiêu chuẩn
model = ExponentialSmoothing(stock_df['Close'], trend='add', damped_trend=True).fit()
Holt = model.fittedvalues
#Vẽ biểu đồ
plt.figure(figsize=(12,6))
plt.plot(stock_df.index, stock_df['Close'], label='Đường dữ liệu thực tế', color='blue')
plt.plot(Holt, label=f'Holt (hệ số tiêu chuẩn)', color = 'orange')
plt.title('Holt (hệ số tiêu chuẩn)')
plt.xlabel('Năm')
plt.ylabel('Giá đóng')
plt.legend(fontsize=12)
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
#Các chỉ số đo độ lệch
mae_holt = mean_absolute_error(stock_df['Close'], Holt)
mse_holt = mean_squared_error(stock_df['Close'], Holt)
rmse_holt = np.sqrt(mse_holt)
print(f'Mean Absolute Error (MAE): {mae_holt}')
print(f'Mean Squared Error (MSE): {mse_holt}')
print(f'Root Mean Squared Error (RMSE): {rmse_holt}')
```




```
Mean Absolute Error (MAE): 0.13769124153047282
Mean Squared Error (MSE): 0.08072475983320294
Root Mean Squared Error (RMSE): 0.28412103025507096
```

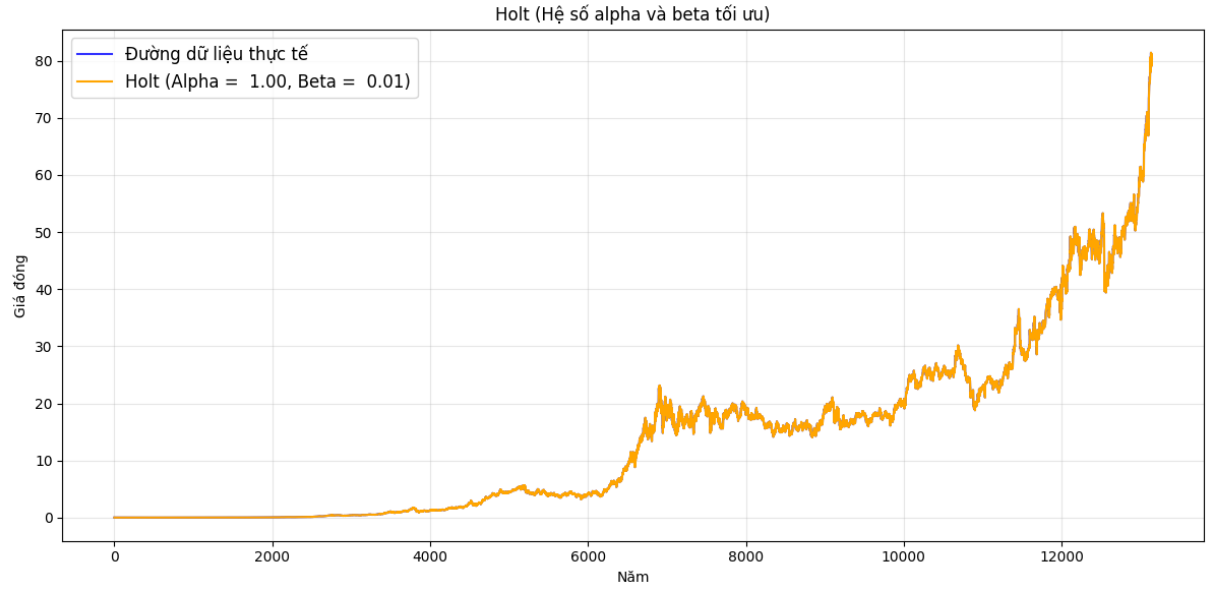
Hình 3.6. Biểu đồ mô hình Holt với hệ số tiêu chuẩn

- Mô hình Holt với hệ số tối ưu:

Mô hình Holt với hệ số tối ưu là một phương pháp mở rộng của Mô hình Holt được cải tiến bằng cách tối ưu hóa các hệ số alpha (α) và beta (β) để đạt được kết quả dự báo tốt nhất. Mô hình này đặc biệt hữu ích khi bạn muốn tự động điều chỉnh các tham số của mô hình để cải thiện độ chính xác của dự báo.

```
#Mô hình Holt với hệ số tối ưu
#Hàm tìm alpha và beta tối ưu
def optimize_holt (data, alpha_values, beta_values):
  best_alpha = None
  best_beta = None
  best_mse = float('inf')
  for alpha in alpha_values:
    for beta in beta_values:
      model = ExponentialSmoothing(data, trend='add', damped_trend=True).fit(
          smoothing_level=alpha, smoothing_trend=beta)
      fitted_values = model.fittedvalues
      mse = mean_squared_error(data, fitted_values)
      if mse < best_mse:
        best_mse = mse
        best_alpha = alpha
        best_beta = beta
  return best_alpha, best_beta, best_mse
#Tập giá trị alpha và beta để thử nghiệm
alpha_range = np.linspace(0.01, 1.0, 10)
beta_range = np.linspace(0.01, 1.0, 10)
```

```
#Tìm alpha và beta tối ưu
optimal_alpha, optimal_beta, optimal_mse = optimize_holt(stock_df['Close'], alpha_range,beta_range)
#Áp dụng mô hình với alpha và beta tối ưu
final_model = ExponentialSmoothing(stock_df['Close'], trend='add', damped_trend=True).fit(
    smoothing_level=optimal_alpha, smoothing_trend=optimal_beta)
Holt_opt = final_model.fittedvalues
#Vẽ biểu đồ
plt.figure(figsize=(12,6))
plt.plot(stock_df['Close'], label='Đường dữ liệu thực tế', color='blue', alpha=0.8)
plt.plot(Holt_opt, label=f'Holt (Alpha = {optimal_alpha: .2f}, Beta = {optimal_beta: .2f})', color = 'orange')
plt.title('Holt (Hệ số alpha và beta tối ưu)')
plt.xlabel('Năm')
plt.ylabel('Giá đóng')
plt.legend(fontsize=12)
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
#Các chỉ số đo độ lệch
mae_holt_opt = mean_absolute_error(stock_df['Close'], Holt_opt)
mse_holt_opt = mean_squared_error(stock_df['Close'], Holt_opt)
rmse_holt_opt = np.sqrt(mse_holt_opt)
print(f'Mean Absolute Error (MAE): {mae_holt_opt}')
print(f'Mean Squared Error (MSE): {mse_holt_opt}')
print(f'Root Mean Squared Error (RMSE): {rmse_holt_opt}')
```




```
Mean Absolute Error (MAE): 0.13790735211430374
Mean Squared Error (MSE): 0.08096361690605168
Root Mean Squared Error (RMSE): 0.2845410636552336
```

Hình 3.7. Biểu đồ mô hình Holt với hệ số tối ưu

3.2.3.4. Ưu, nhược điểm

- Ưu điểm:
  + Dự báo tốt với dữ liệu có xu hướng tuyến tính.
  + Đơn giản, dễ áp dụng và dễ tối ưu.
  + Có thể tự động tối ưu alpha và beta.
- Nhược điểm:
  + Không xử lý được dữ liệu có tính mùa vụ (theo chu kỳ).
  + Nếu xu hướng dữ liệu không ổn định (biến dạng phi tuyến mạnh), dự báo sẽ kém chính xác.
  + Nếu dữ liệu nhiễu mạnh, Holt có thể overfit nếu không điều chỉnh hệ số tốt.

3.2.4. Mô hình dự báo Holt-Winters

3.2.4.1. Ý tưởng thuật toán

Holt-Winters là mô hình phát triển mở rộng của mô hình Holt. Nó không chỉ dự báo xu hướng (trend) mà còn xử lý được tính mùa vụ (seasonality) trong chuỗi dữ liệu.

Mỗi giá trị dự báo được tính dựa trên ba thành phần chính:

- Mức độ (level): Giá trị trung bình của chuỗi thời gian.
- Xu hướng (trend): Biến đổi của chuỗi thời gian theo xu hướng.
- Mùa vụ (seasonality): Các dao động tuần hoàn trong chuỗi thời gian.

3.2.4.2. Vấn đề giải pháp

Holt-Winters thích hợp cho các chuỗi thời gian có xu hướng tăng/giảm hoặc có tính mùa vụ rõ ràng. Nếu không có các đặc điểm này, kết quả dự báo có thể không chính xác.

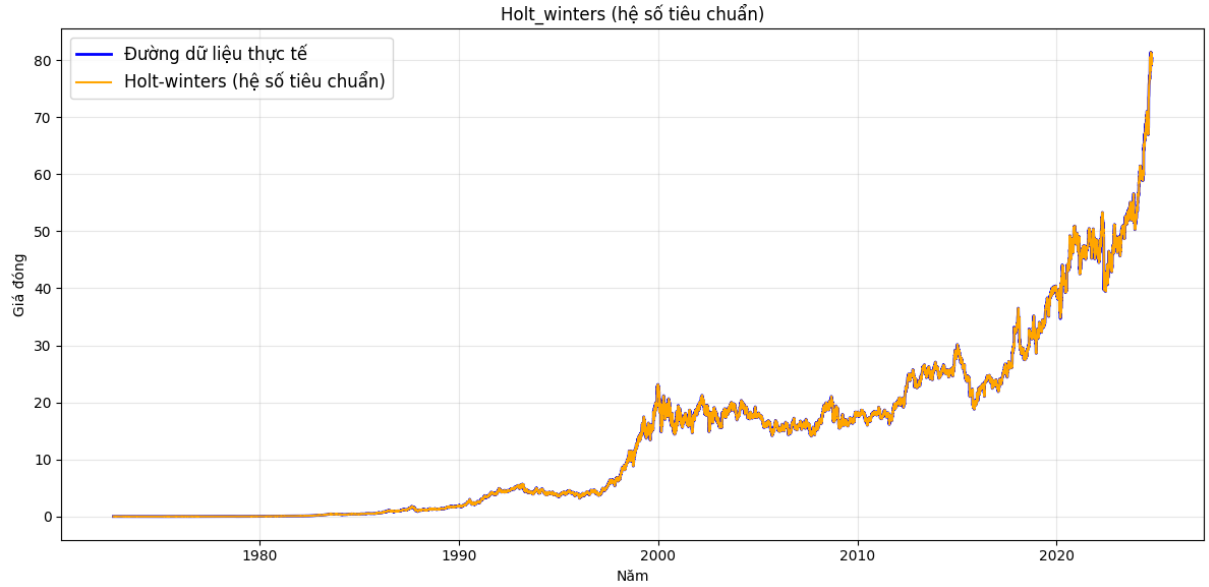
Các tham số như alpha (α) – mức độ, beta (β) – xu hướng, gamma (γ) – mùa vụ, cần được tối ưu để đạt được kết quả tốt nhất. Việc tối ưu tham số này có thể là một công việc tốn thời gian và phức tạp.

3.2.4.3. Triển khai thuật toán

- Mô hình Holt-Winters với hệ số tiêu chuẩn:

Mô hình Holt-Winter là phương pháp dự báo chuỗi thời gian khi dữ liệu có xu hướng (trend) và tính chất mùa vụ (seasonality). Mô hình Holt-Winter hệ số tiêu chuẩn sử dụng các giá trị mặc định hoặc gần đúng để áp dụng nhanh, không tối ưu hóa tham số dựa trên dữ liệu cụ thể.

```
#Mô hình Holt-Winters với hệ số tiêu chuẩn
model = ExponentialSmoothing(stock_df['Close'], seasonal='add', seasonal_periods=7, trend='add', damped_trend=True).fit()
Holt_winters = model.fittedvalues
#Vẽ biểu đồ
plt.figure(figsize=(12,6))
plt.plot(stock_df.index, stock_df['Close'], label='Đường dữ liệu thực tế', color='blue', linewidth=2)
plt.plot(Holt_winters, label=f'Holt-winters (hệ số tiêu chuẩn)', color = 'orange')
plt.title('Holt_winters (hệ số tiêu chuẩn)')
plt.xlabel('Năm')
plt.ylabel('Giá đóng')
plt.legend(fontsize=12)
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
#Các chỉ số đo độ lệch
mae_holt_winters = mean_absolute_error(stock_df['Close'], Holt_winters)
mse_holt_winters = mean_squared_error(stock_df['Close'], Holt_winters)
rmse_holt_winters = np.sqrt(mse_holt_winters)
print(f'Mean Absolute Error (MAE): {mae_holt_winters}')
print(f'Mean Squared Error (MSE): {mse_holt_winters}')
print(f'Root Mean Squared Error (RMSE): {rmse_holt_winters}')
```




```
Mean Absolute Error (MAE): 0.13863737227307227
Mean Squared Error (MSE): 0.08068395113618543
Root Mean Squared Error (RMSE): 0.28404920548416507
```

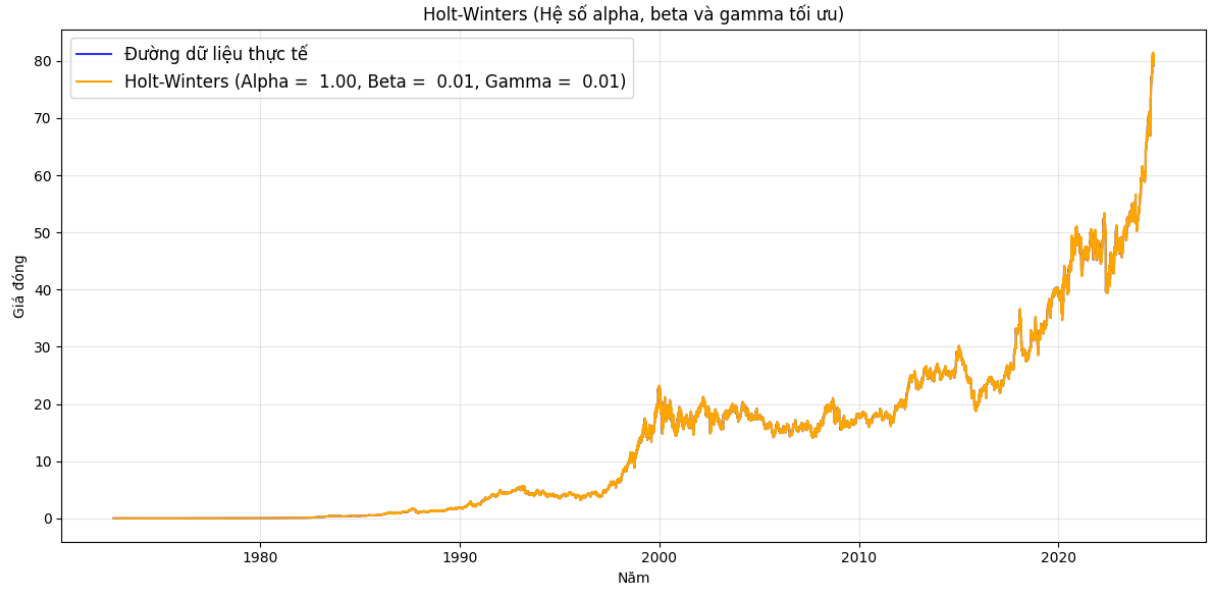
Hình 3.8. Biểu đồ mô hình Holt-Winters với hệ số tiêu chuẩn

- Mô hình Holt-Winters với hệ số tối ưu:

Với phiên bản hệ số tối ưu, các tham số của mô hình được điều chỉnh để phù hợp nhất với dữ liệu thực tế, từ đó tăng độ chính xác của dự báo.

```
#Mô hình Holt-Winters với hệ số tối ưu
#Hàm tìm alpha, beta và gamma tối ưu
def optimize_holt_winters (data, alpha_values, beta_values, gamma_values, seasonal_periods):
  best_alpha = None
  best_beta = None
  best_gamma = None
  best_mse = float('inf')
  for alpha in alpha_values:
    for beta in beta_values:
      for gamma in gamma_values:
        try:
          model = ExponentialSmoothing(data, trend='add', seasonal='add', seasonal_periods=seasonal_periods).fit(
              smoothing_level=alpha, smoothing_trend=beta, smoothing_seasonal=gamma)
          fitted_values = model.fittedvalues
          mse = mean_squared_error(data, fitted_values)
          if mse < best_mse:
            best_mse = mse
            best_alpha = alpha
            best_beta = beta
            best_gamma = gamma
        except Exception as e:
          #Bỏ qua các trường hợp không hợp lệ
          pass
  return best_alpha, best_beta, best_gamma, best_mse
#Tập giá trị alpha, beta và gamma để thử nghiệm
alpha_range = np.linspace(0.01, 1.0, 10)
beta_range = np.linspace(0.01, 1.0, 10)
gamma_range = np.linspace(0.01, 1.0, 10)
```

```
#Tìm alpha, beta và gamma tối ưu
optimal_alpha, optimal_beta, optimal_gamma,optimal_mse = optimize_holt_winters(
    stock_df['Close'], alpha_range, beta_range, gamma_range, seasonal_periods=7)
#Áp dụng mô hình với alpha, beta và gamma tối ưu
final_model = ExponentialSmoothing(stock_df['Close'], trend='add', seasonal='add', seasonal_periods=7).fit(
    smoothing_level=optimal_alpha, smoothing_trend=optimal_beta, smoothing_seasonal=optimal_gamma)
Holt_winters_opt = final_model.fittedvalues
#Vẽ biểu đồ
plt.figure(figsize=(12,6))
plt.plot(stock_df.index, stock_df['Close'], label='Đường dữ liệu thực tế', color='blue', alpha=0.8)
plt.plot(Holt_winters_opt,
         label=f'Holt-Winters (Alpha = {optimal_alpha: .2f}, Beta = {optimal_beta: .2f}, Gamma = {optimal_gamma: .2f})',
         color = 'orange')
plt.title('Holt-Winters (Hệ số alpha, beta và gamma tối ưu)')
plt.xlabel('Năm')
plt.ylabel('Giá đóng')
plt.legend(fontsize=12)
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
#Các chỉ số đo độ lệch
mae_holt_winters_opt = mean_absolute_error(stock_df['Close'], Holt_winters_opt)
mse_holt_winters_opt = mean_squared_error(stock_df['Close'], Holt_winters_opt)
rmse_holt_winters_opt = np.sqrt(mse_holt_winters_opt)
print(f'Mean Absolute Error (MAE): {mae_holt_winters_opt}')
print(f'Mean Squared Error (MSE): {mse_holt_winters_opt}')
print(f'Root Mean Squared Error (RMSE): {rmse_holt_winters_opt}')
```




```
Mean Absolute Error (MAE): 0.1389355917016146
Mean Squared Error (MSE): 0.08140073012275645
Root Mean Squared Error (RMSE): 0.28530813189034143
```

Hình 3.9. Biểu đồ mô hình Holt-Winters với hệ số tối ưu

3.2.4.4. Ưu, nhược điểm

- Ưu điểm:
  + Đơn giản và dễ triển khai, rất thích hợp cho các chuỗi thời gian có tính xu hướng hoặc mùa vụ.
  + Có thể thích ứng với các chuỗi thời gian có xu hướng, tính theo mùa hoặc cả hai.
- Nhược điểm:

+ Phương pháp cần dữ liệu lịch sử đủ lớn để ước tính các tham số của mô hình.
+ Mặc dù mô hình đơn giản, nhưng việc tối ưu hóa các tham số có thể gặp khó khăn, đặc biệt khi dữ liệu có nhiều biến động.

3.3. Đánh giá các mô hình

Thống kê các chỉ số đo độ lệch:

```
#Thống kê các chỉ số đo độ lệch
models = ['MA_ws1', 'MA_ws5', 'MA_ws10', 'SES', 'SES_opt', 'Holt', 'Holt_opt', 'Holt_winters', 'Holt_winters_opt']
mae_scores = [mae_ws1, mae_ws5, mae_ws10, mae_alpha1, mae_SES_opt, mae_holt, mae_holt_opt, mae_holt_winters, mae_holt_winters_opt]
mse_scores = [mse_ws1, mse_ws5, mse_ws10, mse_alpha1, mse_SES_opt, mse_holt, mse_holt_opt, mse_holt_winters, mse_holt_winters_opt]
mape_scores = [rmse_ws1, rmse_ws5, rmse_ws10, rmse_alpha1, rmse_SES_opt, rmse_holt, rmse_holt_opt, rmse_holt_winters, rmse_holt_winters_opt]
#Tạo bảng thống kê các chỉ số
evaluation = pd.DataFrame({'Model':models, 'MSE': mse_scores, 'MAE': mae_scores, 'MAPE': mape_scores})
evaluation.set_index('Model', inplace=True)
evaluation
```

Model	MSE	MAE	MAPE
MA_ws1	0.000000	0.000000	0.000000
MA_ws5	0.090394	0.149240	0.300655
MA_ws10	0.198769	0.226869	0.445835
SES	0.357216	0.309046	0.597675
SES_opt	0.080730	0.137704	0.284131
Holt	0.080725	0.137691	0.284121
Holt_opt	0.080964	0.137907	0.284541
Holt_winters	0.080684	0.138637	0.284049
Holt_winters_opt	0.081401	0.138936	0.285308

Hình 3.10. Thống kê các chỉ số đo độ lệch

Từ bảng thống kê đo lường các chỉ số đo độ lệch, có thể rút ra được một số nhận xét như sau:

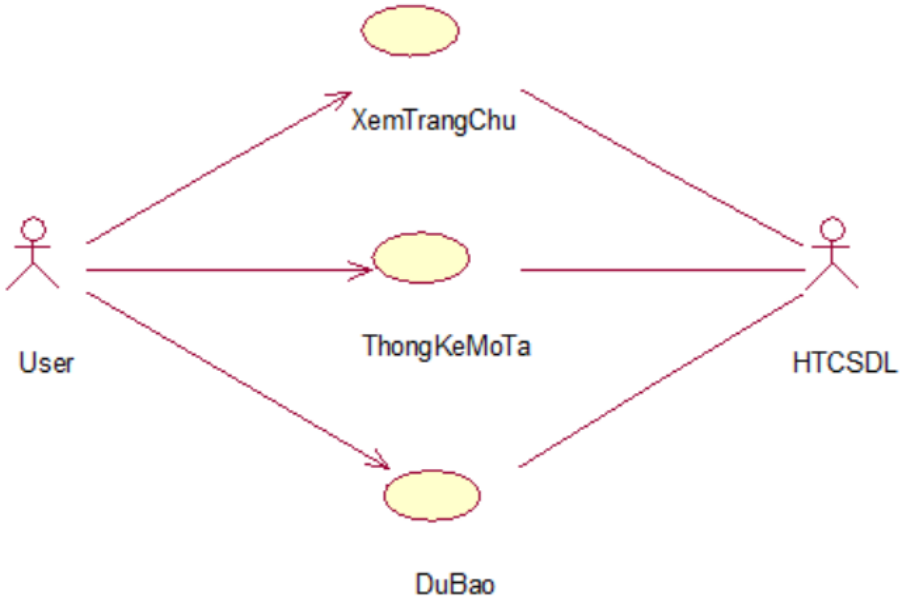
- Mô hình Moving Average (MA): Dự báo tốt trong thời gian ngắn. Tuy nhiên, khi tăng kích thước khoảng trượt (window size), độ chính xác giảm đáng kể.
- Mô hình Simple Exponential Smoothing (SES): Mô hình phù hợp với dự báo ngắn hạn, nhưng không hiệu quả bằng mô hình MA do sai số MSE và MAPE cao hơn.

- Mô hình Holt và Holt-winter: Hai mô hình này thích hợp cho các dự báo dài hạn. Trong đó, Holt-winter dự báo tốt hơn do có khả năng xử lý đồng thời cả yếu tố xu hướng và mùa vụ.

Từ kết quả trên, mô hình dự báo Holt-Winters sau khi tối ưu tham số được lựa chọn để xây dựng chương trình dự báo giá cổ phiếu, vì nó mang lại độ chính xác cao và phù hợp với đặc điểm chuỗi thời gian có xu hướng và khả năng biến động theo mùa.

3.4. Phân tích thiết kế hệ thống

3.4.1. Biểu đồ use case hệ thống



Hình 3.11. Biểu đồ use case hệ thống

1. Xem trang chủ: Cho phép người dùng xem trang chủ của hệ thống.
2. Thống kê mô tả: Cho phép người dùng xem các biểu đồ trực quan hóa về biến động giá cổ phiếu của Walmart.
3. Dự báo: Cho phép người dùng lựa chọn khoảng thời gian dự báo và hiển thị kết quả dự báo giá đóng cửa trong tương lai.

3.4.2. Mô tả chi tiết use case

3.4.2.1. Mô tả use case Xem trang chủ

- Tên use case: Xem trang chủ

- Mô tả vắn tắt: Use case này cho phép người dùng xem trang chủ của hệ thống.
- Luồng sự kiện:
  + Luồng cơ bản:
    1. Use case này bắt đầu khi người dùng truy cập vào trang web hoặc kích chọn "Trang chủ" trên thanh menu chính. Hệ thống sẽ lấy thông tin dữ liệu trong bảng COPHIEU bao gồm: Id, Date, Open, High, Low, Close, Adj_Close, Volume và hiển thị lên màn hình.
    2. Người dùng kích chọn khoảng thời gian muốn hiển thị, bao gồm ngày bắt đầu và ngày kết thúc. Hệ thống sẽ hiển thị lại dữ liệu tương ứng với thời gian được chọn lên màn hình.

    Use case kết thúc.
  + Luồng rẽ nhánh:

  1. Tại bước 2 trong luồng cơ bản, nếu người dùng chọn thời gian bắt đầu nhỏ hơn thời gian kết thúc. Hệ thống sẽ hiển thị lên màn hình thông báo lỗi: "Thời gian bắt đầu phải lớn hơn thời gian kết thúc".
  2. Tại bất kỳ thời điểm nào trong quá trình thực hiện use case nếu không kết nối được với cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc.

- Các yêu cầu đặc biệt: Không có.
- Tiền điều kiện: Không có.
- Hậu điều kiện: Không có.
- Điểm mở rộng: Không có.

3.4.2.2. Mô tả use case Thống kê mô tả

- Tên use case: Thống kê mô tả
- Mô tả vắn tắt: Use case này cho phép người dùng xem các biểu đồ trực quan hóa về biến động giá cổ phiếu của Walmart.
- Luồng sự kiện:
  + Luồng cơ bản:

1. Use case này bắt đầu khi người dùng kích chọn “Thống kê mô tả” trên thanh menu chính. Hệ thống sẽ hiển thị lần lượt các biểu đồ trực quan hóa bao gồm: biểu đồ nến, biểu đồ giá đóng trung bình theo năm và biểu đồ khối lượng giao dịch theo thời gian lên màn hình.
2. Người dùng kích chọn khoảng thời gian muốn hiển thị, bao gồm ngày bắt đầu và ngày kết thúc. Hệ thống sẽ hiển thị lại biểu đồ tương ứng với thời gian được chọn lên màn hình.

   Use case kết thúc.

+ Luồng rẽ nhánh:
   1. Tại bước 2 trong luồng cơ bản, nếu người dùng chọn thời gian bắt đầu nhỏ hơn thời gian kết thúc. Hệ thống sẽ hiển thị lên màn hình thông báo lỗi: “Thời gian bắt đầu phải lớn hơn thời gian kết thúc”.
   2. Tại bất kỳ thời điểm nào trong quá trình thực hiện use case nếu không kết nối được với cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc.

- Các yêu cầu đặc biệt: Không có.
- Tiền điều kiện: Không có.
- Hậu điều kiện: Không có.
- Điểm mở rộng: Không có.

3.4.2.3. Mô tả use case Dự báo

- Tên use case: Dự báo
- Mô tả vắn tắt: Use case này cho phép người dùng lựa chọn khoảng thời gian dự báo và hiển thị kết quả dự báo giá đóng cửa trong tương lai.
- Luồng sự kiện:
  + Luồng cơ bản:
    1. Use case này bắt đầu khi người dùng kích chọn “Dự báo cổ phiếu” trên thanh menu chính. Hệ thống hiển thị lên màn hình thanh lựa chọn khoảng thời gian dự báo.

2. Người dùng kích chọn khoảng thời gian mong muốn: tuần, tháng, năm. Hệ thống sẽ lấy thông tin từ bảng KETQUA bao gồm: Id, Forecast_Date, Forecast_Value, MAE, MSE, RMSE cùng biểu đồ dự đoán giá đóng, kết quả đánh giá sai số và hiển thị lên màn hình.
   Use case kết thúc.

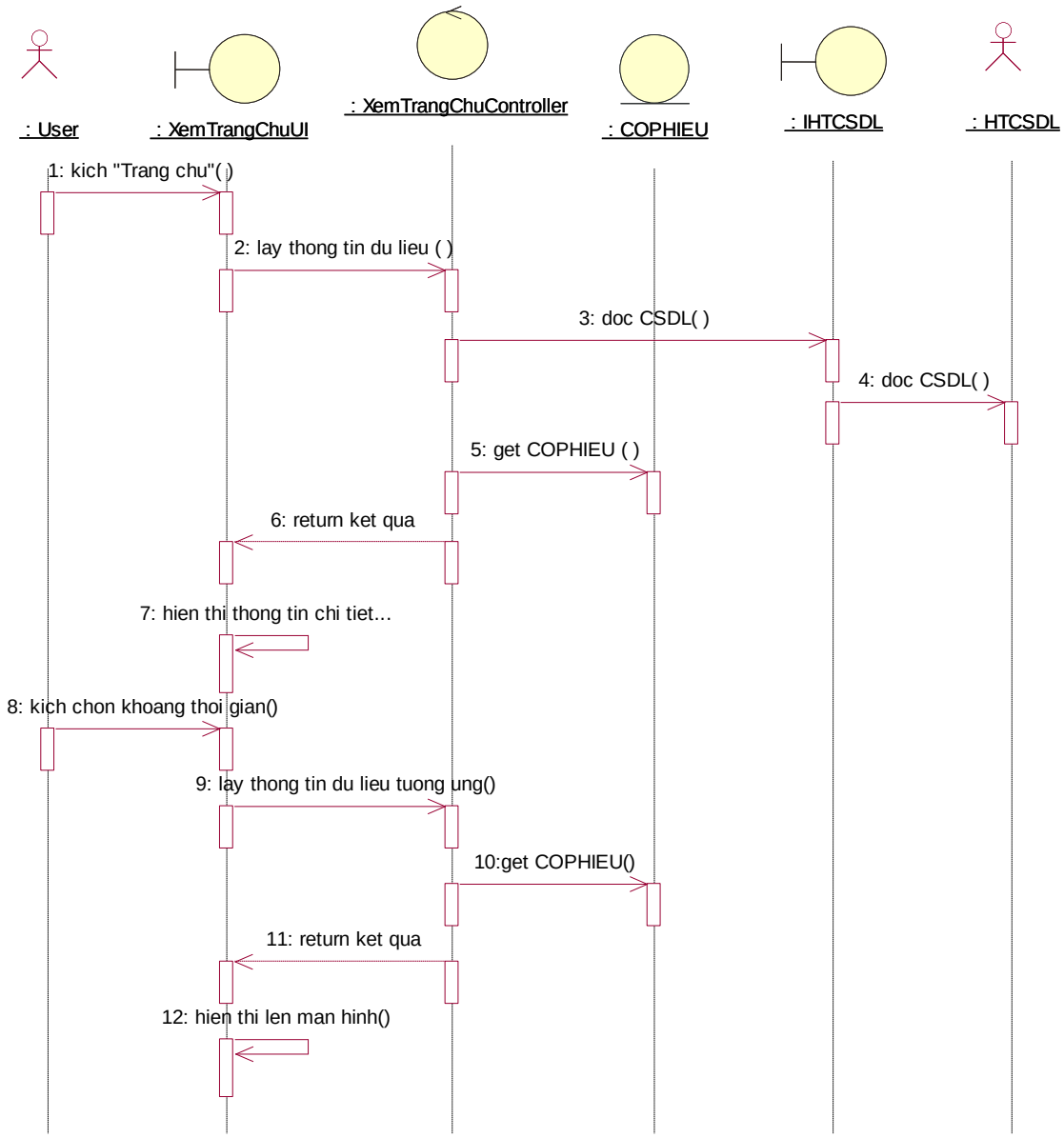
+ Luồng rẽ nhánh:
   1. Tại bất kỳ thời điểm nào trong quá trình thực hiện use case nếu không kết nối được với cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc.

- Các yêu cầu đặc biệt: Không có.
- Tiền điều kiện: Không có.
- Hậu điều kiện: Không có.
- Điểm mở rộng: Không có.

3.4.3. Phân tích use case

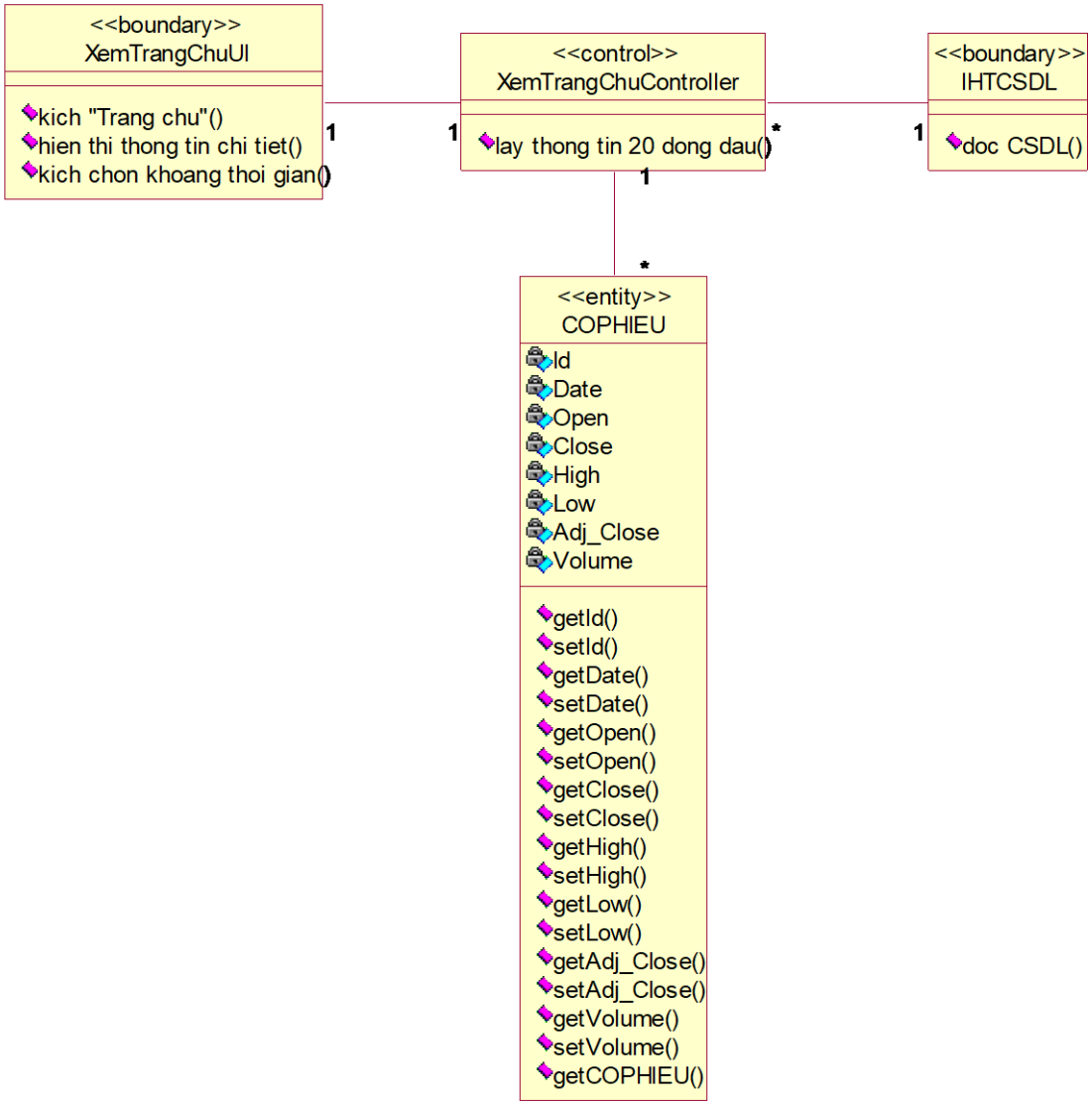
3.4.3.1. Phân tích use case Xem trang chủ

- Biểu đồ trình tự:



Hình 3.12. Biểu đồ trình tự use case Xem trang chủ

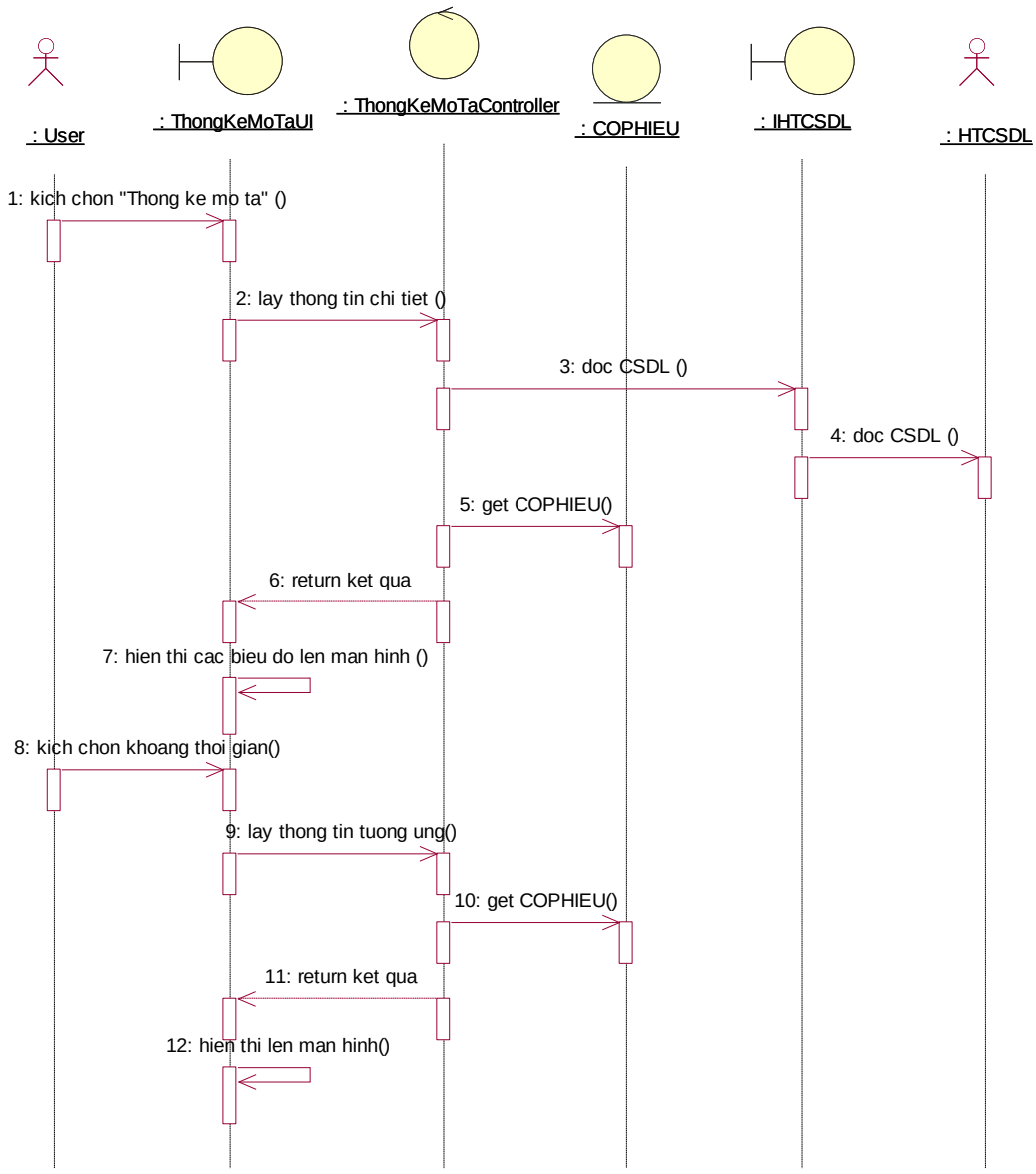
- Biểu đồ lớp phân tích:



Hình 3.13. Biểu đồ lớp phân tích use case Xem trang chủ

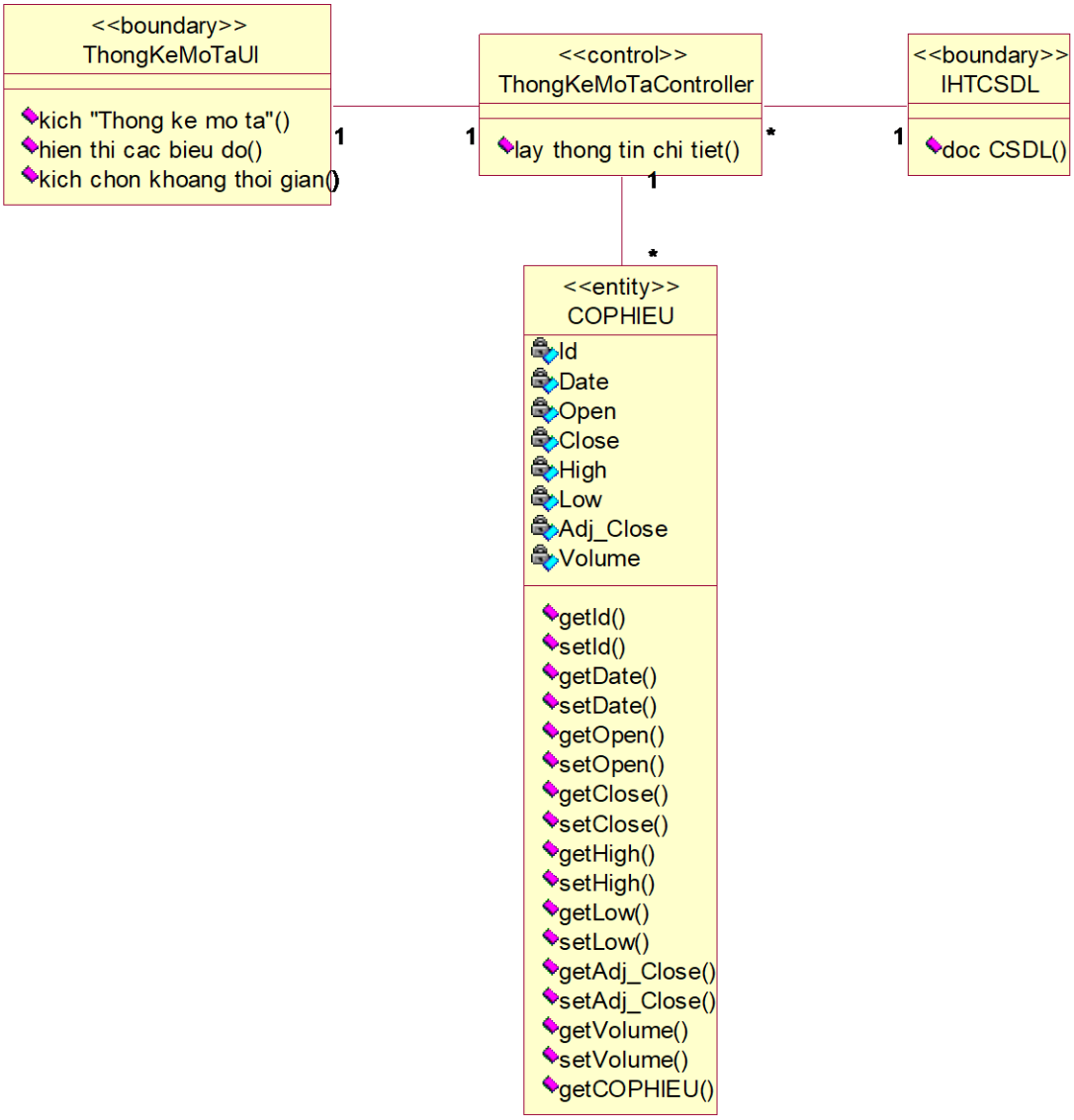
3.4.3.2. Phân tích use case Thống kê mô tả

- Biểu đồ trình tự:



Hình 3.14. Biểu đồ trình tự use case Thống kê mô tả

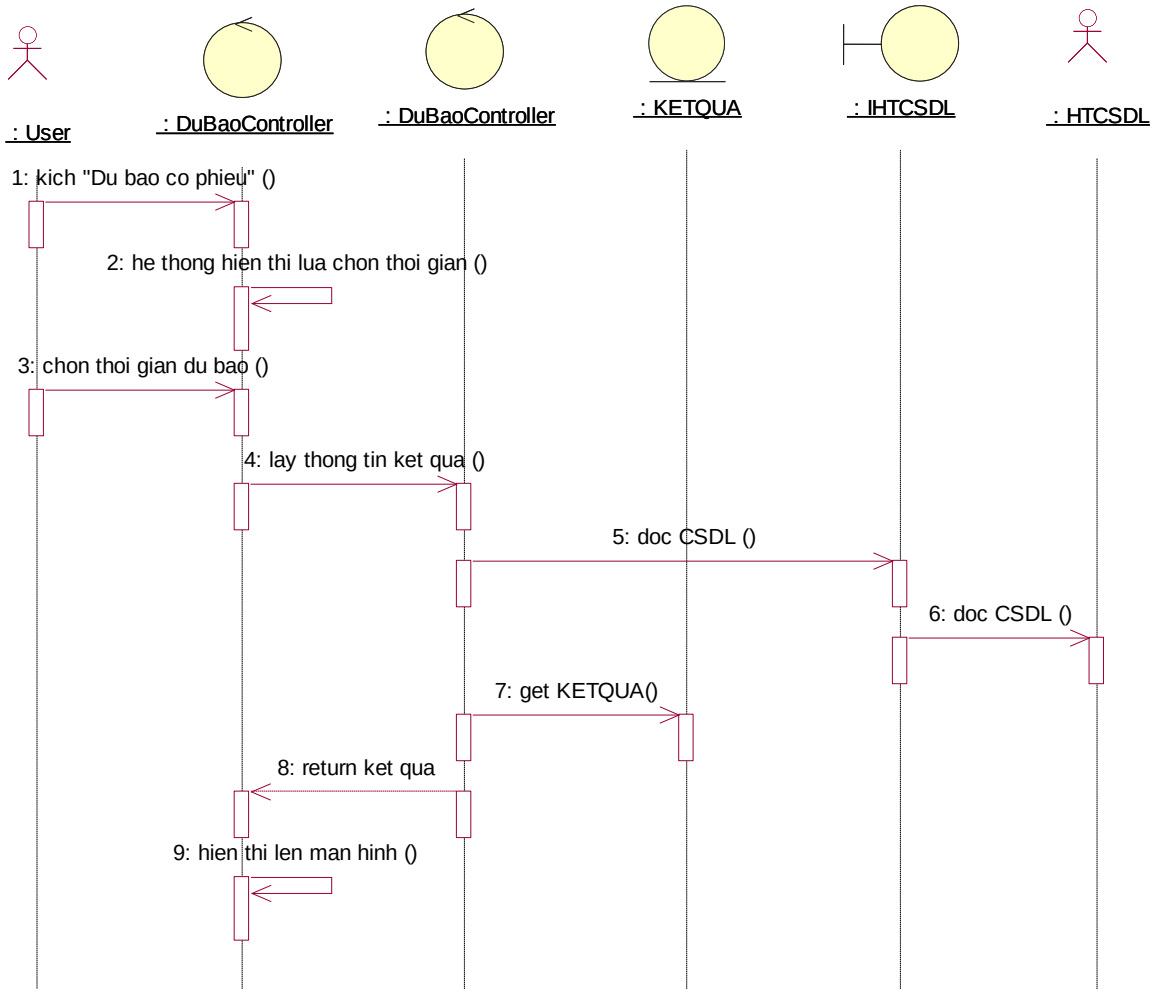
- Biểu đồ lớp phân tích:



Hình 3.15. Biểu đồ lớp phân tích use case Thống kê mô tả

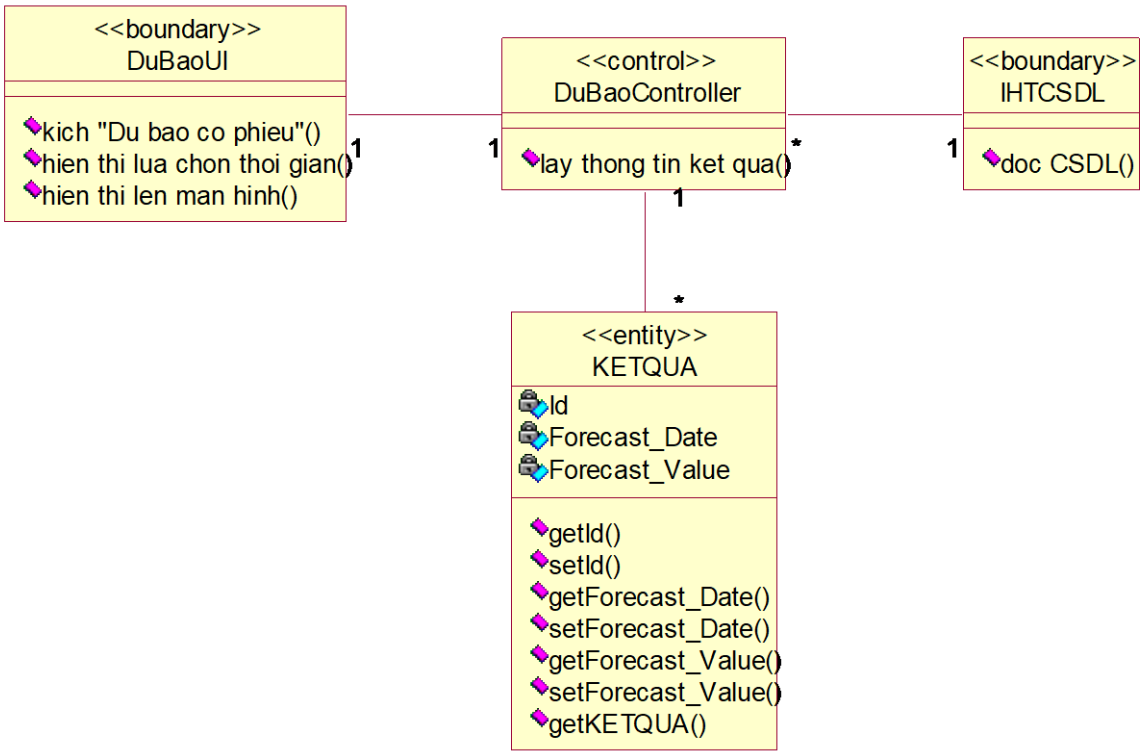
3.4.3.3. Phân tích use case Dự báo

- Biểu đồ trình tự:



Hình 3.16. Biểu đồ trình tự use case Dự báo

- Biểu đồ lớp phân tích:



Hình 3.17. Biểu đồ lớp phân tích use case Dự báo

3.4.4. Thiết kế cơ sở dữ liệu

- Thiết kế bảng thực thể COPHIEU:

Entity Name: COPHIEU Table Name: COPHIEU

Attributes | Properties | Advanced | Indexes | Alternate Keys | Others | Relationship | Storage | Comments | Notes | DDL

	Key	Name	Column Name	Datatype	Not null	Unique	Description
1		Id	Id	Integer	Not null		
2		Date	Date	Datetime			
3		Open	Open	Float			
4		High	High	Float			
5		Low	Low	Float			
6		Close	Close	Float			
7		Adj_Close	Adj_Close	Float			
8		Volume	Volume	Bigint			

Hình 3.18. Cấu trúc bảng dữ liệu COPHIEU

- Thiết kế bảng thực thể KETQUA:

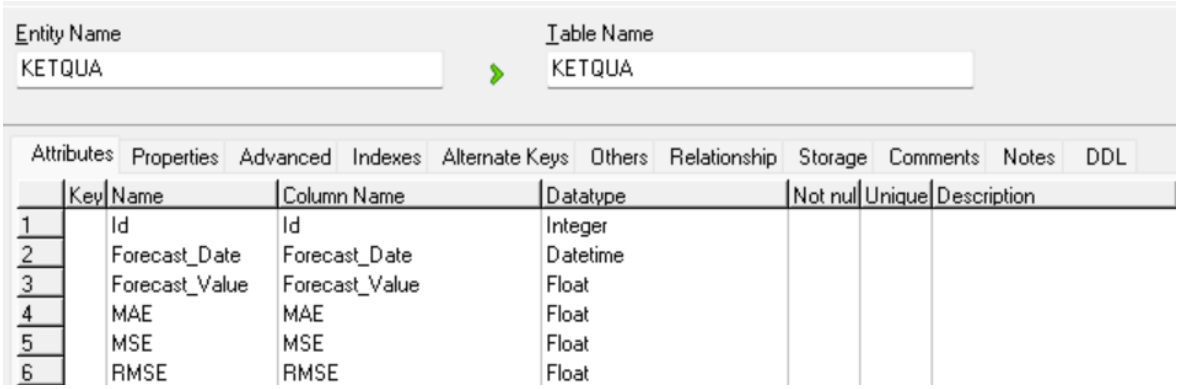

Entity Name: KETQUA

Table Name: KETQUA

Attributes | Properties | Advanced | Indexes | Alternate Keys | Others | Relationship | Storage | Comments | Notes | DDL

	Key	Name	Column Name	Datatype	Not null	Unique	Description
1		Id	Id	Integer			
2		Forecast_Date	Forecast_Date	Datetime			
3		Forecast_Value	Forecast_Value	Float			
4		MAE	MAE	Float			
5		MSE	MSE	Float			
6		RMSE	RMSE	Float			

Hình 3.19. Cấu trúc bảng dữ liệu KETQUA

3.5. Cài đặt chương trình

3.5.1. Yêu cầu phần cứng, phần mềm

- Hệ thống yêu cầu một máy tính có cấu hình từ mức trung bình trở lên, đảm bảo khả năng xử lý dữ liệu và vận hành các công cụ phân tích:
  + Bộ vi xử lý (CPU): Intel Core i5 hoặc tương đương trở lên, nhằm đảm bảo hiệu suất khi thực hiện các thuật toán xử lý và dự báo.
  + Bộ nhớ RAM: Tối thiểu 8GB; khuyến nghị 16GB để đảm bảo xử lý mượt mà dữ liệu và chạy mô hình học máy.
  + Dung lượng ổ cứng: Tối thiểu 100GB dung lượng trống để lưu trữ dữ liệu, mã nguồn và môi trường phát triển.
  + Kết nối Internet: Ổn định, phục vụ việc sử dụng các công cụ trực tuyến như Google Colab và triển khai ứng dụng web bằng Streamlit.
- Về mặt phần mềm, đề tài sử dụng các công cụ và thư viện phục vụ cho việc phân tích, trực quan hóa và xây dựng hệ thống dự báo như sau:
  + Hệ điều hành: Windows 10/11, Ubuntu 20.04+ hoặc macOS.
  + Ngôn ngữ lập trình chính: Python 3.9 trở lên, phục vụ xử lý dữ liệu, xây dựng mô hình học máy và trực quan hóa.
  + Thư viện Python: pandas, numpy, matplotlib, plotly, statsmodels, streamlit, hỗ trợ toàn bộ quy trình từ phân tích dữ liệu đến triển khai web.
  + Google Colab: Dùng để thực hiện thử nghiệm mô hình, vẽ biểu đồ và lưu trữ code online.

+ PyCharm Community Edition: Công cụ phát triển mã nguồn, tổ chức dự án và xây dựng ứng dụng dự báo.
+ Power BI Desktop: Phục vụ trực quan hóa dữ liệu kinh doanh và tạo báo cáo phân tích.
+ Trình duyệt web: Chrome, Edge hoặc Firefox để chạy giao diện người dùng của ứng dụng web.

3.5.2. Giới thiệu ứng dụng

Chương trình "Dự báo giá cổ phiếu Walmart" được xây dựng nhằm phục vụ phân tích và dự đoán giá trị cổ phiếu của công ty Walmart thông qua các công cụ hiện đại như Python, các thư viện phân tích số liệu, và nền tảng Streamlit.

Chương trình tích hợp các chức năng chính bao gồm:

- Hiển thị dữ liệu cổ phiếu gốc: Cho phép người dùng xem nhanh dữ liệu đóng cửa, khối lượng giao dịch và giá cổ phiếu theo thời gian.
- Thống kê mô tả trực quan: Cung cấp các biểu đồ như biểu đồ nến, biểu đồ giá trung bình theo năm, và biểu đồ khối lượng giao dịch giúp người dùng nắm bắt xu hướng thị trường.
- Dự báo giá cổ phiếu: Sử dụng mô hình Holt-Winters để dự đoán giá đóng cửa trong tương lai theo tuần, tháng hoặc năm.
- Đánh giá sai số mô hình: Tính toán các chỉ số sai số như MAE, MSE và RMSE để đánh giá độ chính xác của mô hình dự báo.

Người dùng có thể tương tác linh hoạt với ứng dụng thông qua các bộ lọc thời gian, lựa chọn khoảng dự đoán và theo dõi biểu đồ dự báo một cách trực quan, sinh động.

Walmart Walmart Inc.

Trang chủ Thống kê mô tả Dự báo cổ phiếu

Chọn khoảng thời gian hiển thị:

Từ ngày

1972/08/25

Đến ngày

2024/10/14

Dữ liệu gốc trong khoảng thời gian đã chọn:

Date	Open	High	Low	Close	Adj Close	Volume
1972-08-25 00:00:00	0.0212	0.0216	0.0212	0.0215	0.0117	7526400
1972-08-28 00:00:00	0.0215	0.0216	0.0214	0.0214	0.0116	2918400
1972-08-29 00:00:00	0.0213	0.0213	0.0212	0.0212	0.0115	5836800
1972-08-30 00:00:00	0.0212	0.0212	0.021	0.0212	0.0115	1228800
1972-08-31 00:00:00	0.021	0.021	0.0208	0.0208	0.0113	2611200
1972-09-01 00:00:00	0.0209	0.021	0.0209	0.021	0.0114	768000
1972-09-05 00:00:00	0.021	0.021	0.0208	0.0208	0.0113	1689600

Hình 3.20. Giao diện Trang chủ

Walmart Walmart Inc.

Trang chủ Thống kê mô tả Dự báo cổ phiếu

Thống kê mô tả

Chọn khoảng thời gian phân tích:

Từ ngày

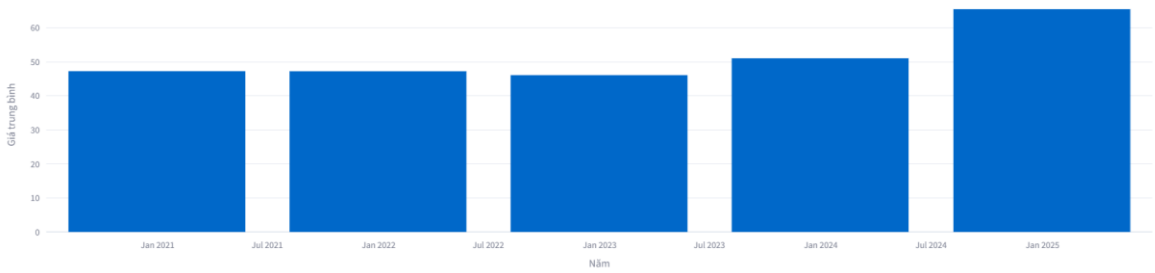
2020/08/01

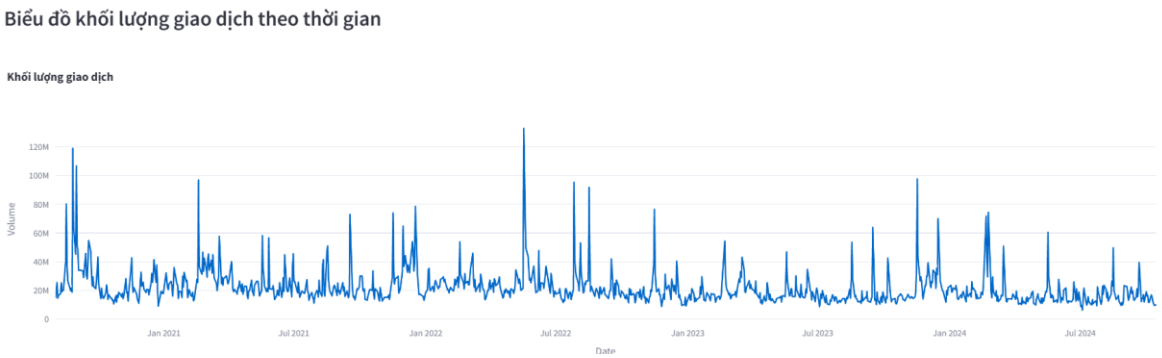
Đến ngày

2024/10/14

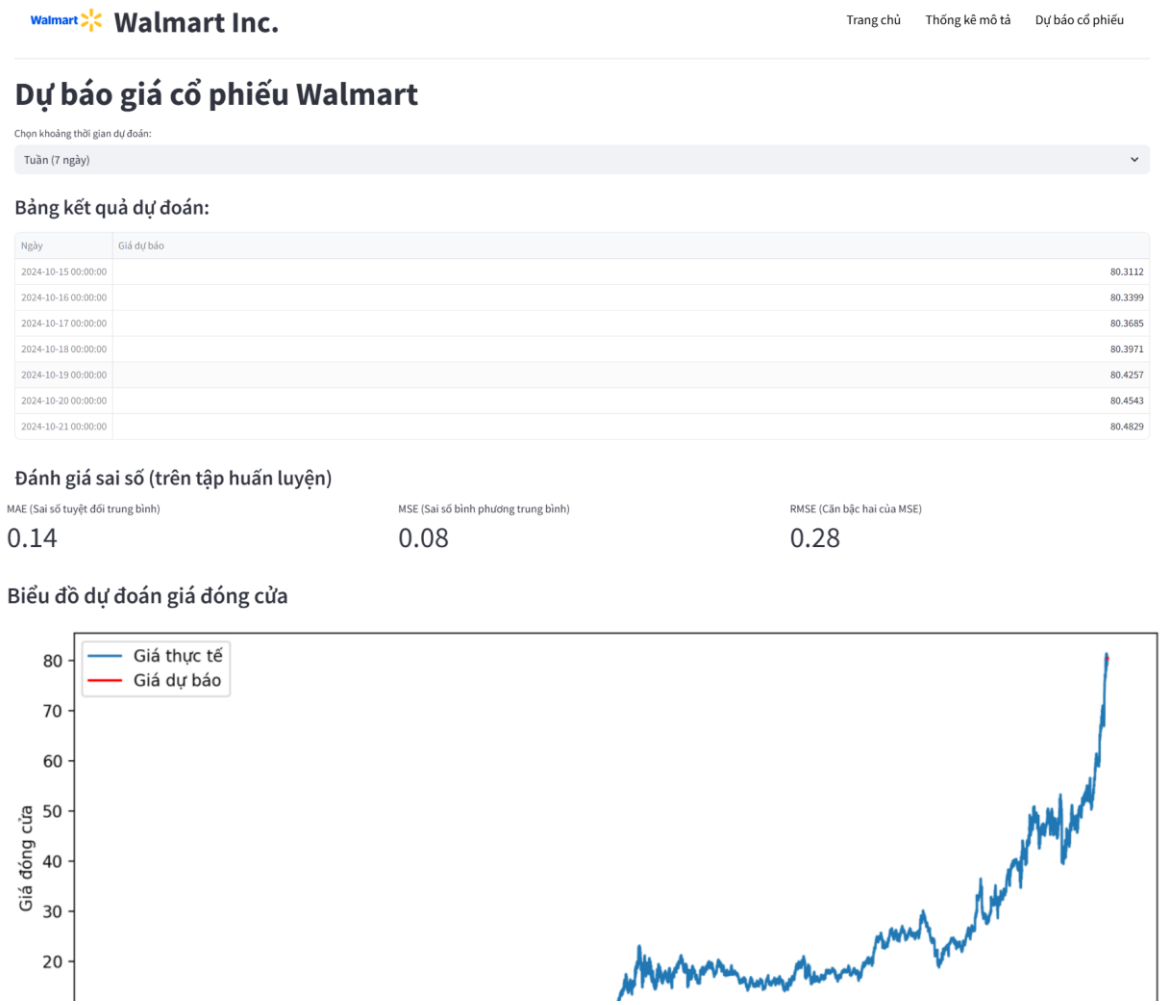
Biểu đồ nến

Biểu đồ giá đóng cửa trung bình theo năm





Hình 3.21. Giao diện trang Thống kê mô tả



Hình 3.22. Giao diện trang Dự báo cổ phiếu

3.6. Kiểm thử

Để đảm bảo hệ thống hoạt động chính xác, đầy đủ chức năng và mang lại trải nghiệm tốt cho người dùng, đề tài được tiến hành kiểm thử theo hai nhóm chính: Kiểm thử chức năng và kiểm thử phi chức năng.

- Mục tiêu kiểm thử chức năng là đảm bảo các chức năng chính của hệ thống hoạt động đúng theo yêu cầu đã đề ra. Các ca kiểm thử tiêu biểu như sau:

Bảng 3.1. Kiểm thử hệ thống

STT	Chức năng	Mô tả kiểm thử	Kết quả mong đợi	Kết quả thực tế
1	Tải dữ liệu	Kiểm tra khi ứng dụng khởi chạy, dữ liệu có được đọc và hiển thị không	Hiển thị đúng bảng dữ liệu	Pass
2	Bộ lọc thời gian	Chọn khoảng ngày bắt đầu – kết thúc và xem dữ liệu lọc	Chỉ hiển thị dữ liệu trong khoảng chọn	Pass
		Chọn ngày bắt đầu lớn hơn ngày kết thúc	Hiển thị thông báo lỗi hoặc cảnh báo yêu cầu chọn đúng khoảng thời gian hợp lệ	Pass
3	Dự báo theo tuần, tháng, năm	Chọn khoảng dự báo và xem kết quả	Hiển thị bảng và biểu đồ dự báo	Pass
4	Biểu đồ thống kê mô tả	Kiểm tra biểu đồ nến, khối lượng, giá trung bình	Hiển thị biểu đồ tương ứng	Pass
5	Tính sai số MAE, MSE, RMSE	Kiểm tra sai số trên tập huấn luyện	Hiển thị đúng giá trị	Pass
6	Giao diện và điều hướng	Nhấp các menu (Trang chủ, Thống kê, Dự báo)	Chuyển đúng nội dung từng trang	Pass

- Ngoài các chức năng chính, hệ thống còn được kiểm thử về các yếu tố phi chức năng như sau:
  + Hiệu năng: Hệ thống phản hồi nhanh, thời gian tải dữ liệu dưới 2 giây đối với bộ dữ liệu hiện tại.
  + Tính ổn định: Không phát sinh lỗi trong quá trình chuyển trang hoặc tương tác với biểu đồ.
  + Tính dễ sử dụng: Giao diện trực quan, menu dễ hiểu, người dùng mới có thể thao tác dễ dàng mà không cần hướng dẫn.
- Kết quả kiểm thử cho thấy hệ thống hoạt động ổn định, đáp ứng đầy đủ các yêu cầu chức năng và phi chức năng. Các chức năng chính đều cho kết quả chính xác, không phát hiện lỗi nghiêm trọng. Hệ thống sẵn sàng đưa vào sử dụng hoặc phát triển thêm tính năng mở rộng trong tương lai.

KẾT LUẬN

Qua quá trình thực hiện đồ án tốt nghiệp, em đã có cơ hội vận dụng các kiến thức chuyên ngành, đặc biệt là trong lĩnh vực khoa học dữ liệu và phân tích dữ liệu kinh doanh. Đề tài giúp em rèn luyện kỹ năng lập trình, trực quan hóa dữ liệu và xây dựng mô hình dự báo, đồng thời tiếp cận các công cụ hiện đại như Python và Power BI. Những trải nghiệm này là nền tảng quan trọng để em phát triển sự nghiệp trong tương lai.

Kết quả đạt được:

- Hoàn thành việc thu thập, làm sạch và xử lý dữ liệu từ dữ liệu thô.
- Vận dụng hiệu quả các thư viện Python như Pandas, Matplotlib, Statsmodels để phân tích và xây dựng mô hình dự báo bằng phương pháp Holt-Winters.
- Xây dựng hệ thống báo cáo trực quan bằng Power BI giúp trực quan hóa các xu hướng doanh thu, cũng như biến động giá cổ phiếu một cách dễ hiểu.
- Xây dựng được chương trình dự báo giá cổ phiếu, hỗ trợ cho việc đưa ra quyết định kinh doanh và đầu tư hiệu quả hơn.

Hướng phát triển:

- Mở rộng phạm vi dữ liệu: Tích hợp thêm dữ liệu kinh tế vĩ mô, xu hướng ngành và dữ liệu từ đối thủ cạnh tranh để có góc nhìn toàn diện hơn.
- Nâng cấp mô hình dự báo: Ứng dụng các thuật toán học máy và Deep Learning nhằm tăng độ chính xác và khả năng dự đoán dài hạn cho giá cổ phiếu.
- Phân tích nâng cao: Thực hiện phân tích hồi quy đa biến, kiểm định thống kê và mô hình nhân quả để hiểu sâu hơn về các yếu tố ảnh hưởng đến hiệu quả kinh doanh.
- Xây dựng hệ thống báo cáo thời gian thực: Tự động cập nhật dữ liệu và cảnh báo khi có biến động lớn, giúp doanh nghiệp phản ứng nhanh chóng với thị trường.

TÀI LIỆU THAM KHẢO

[1] Nguyễn Văn Hiệp, Trần Văn Lâm (2020), *Phân tích dữ liệu với Python*, NXB Khoa học và Kỹ thuật.

[2] Python pandas documentation. (n.d.). Truy cập ngày 15/03/2025 từ https://pandas.pydata.org/docs/

[3] Microsoft Power BI Documentation. (n.d.). Truy cập ngày 20/03/2025 từ https://learn.microsoft.com/en-us/power-bi/

[4] Brockwell, P. J., & Davis, R. A. (2016), *Introduction to Time Series and Forecasting* (3rd ed.), Springer.

[5] Hyndman, R. J., & Athanasopoulos, G. (2018), *Forecasting: Principles and Practice* (2nd ed.), OTexts.

[6] Nguyễn Thị Thanh Huyền, Ngô Thị Bích Thuý, Phạm Kim Phượng (2011), Giáo trình Phân tích thiết kế hệ thống, NXB Giáo dục Việt Nam.

[7] Hoàng Quang Huy (2016), "Giáo trình kiểm thử phần mềm", Nhà xuất bản Thống kê.