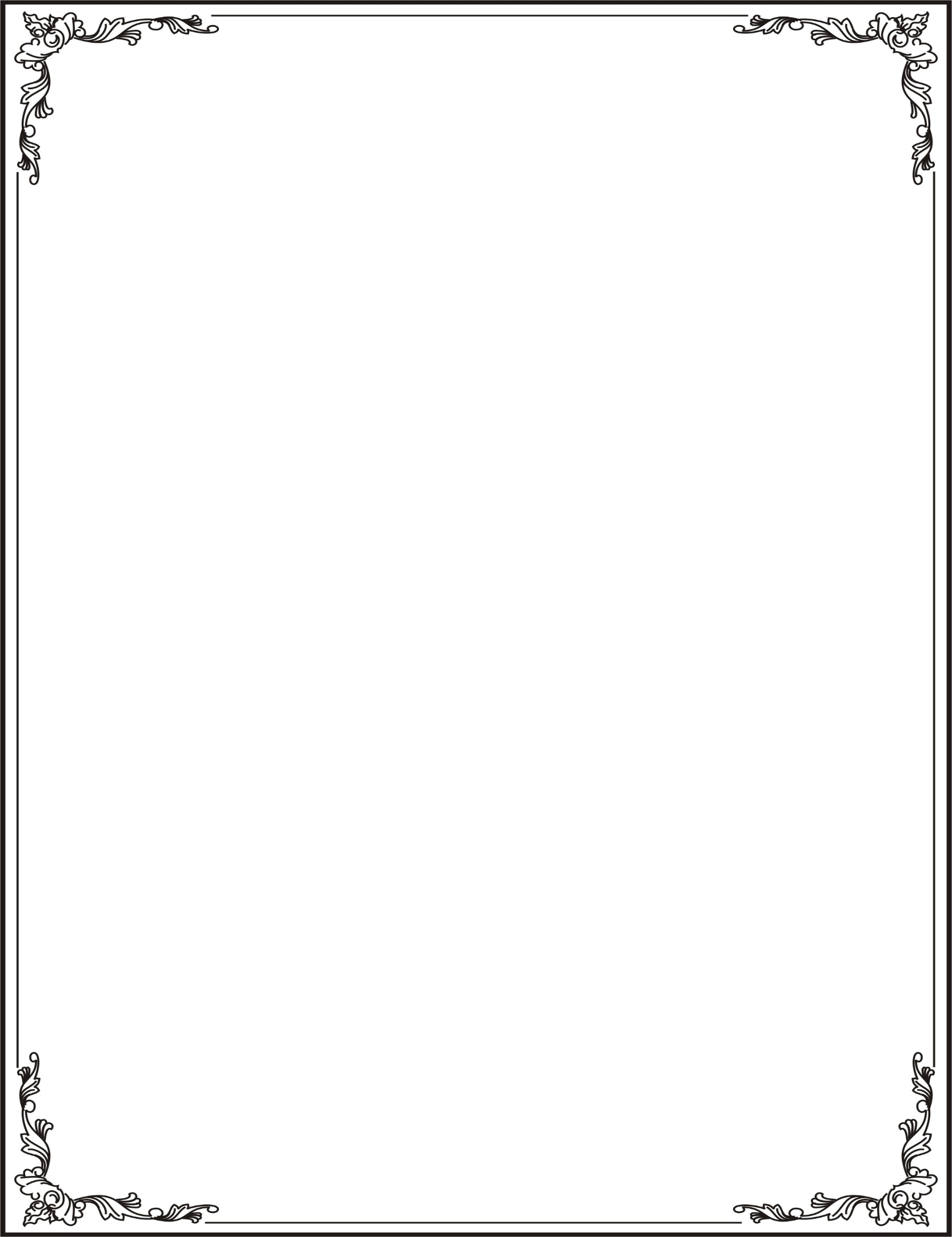
**TRƯỜNG ĐẠI HỌC THỦY LỢI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**A blue and white logo

Description automatically generated**

**BÁO CÁO BÀI TẬP LỚN**

**HỌC PHẦN HỌC MÁY**

**ĐỀ TÀI:XÂY DỰNG MÔ HÌNH DỰ BÁO GIÁ VÀNG**

**BẰNG CÁC THUẬT TOÁN HỒI QUY**

**VÀ MẠNG NƠ-RON NHÂN TẠO**

**Nhóm thực hiện:** 64HTTT1-10

|  |  |
| --- | --- |
| 1.Vũ Xuân Huy | – MSV: 2251172378 |
| 2.Nguễn Thị Vân Anh | – MSV: 2251161943 |
| 3.Phạm Thị Kim Huệ | – MSV: 2251162020 |
| 3.Trần Văn Minh Quốc | – MSV: 2251162130 |

**Thành viên nhóm:**

**Giảng viên phụ trách môn học:** PGS.TS Nguyễn Quang Hoan

***Hà Nội, tháng 01 năm 2025***

**MỤC LỤC**

[**LỜI NÓI ĐẦU** 1](#_Toc181187151)

[**CHƯƠNG 1 :TECHNICAL REPORT – TRIỂN KHAI MÃ NGUỒN** 2](#_Toc181187152)

[**PHẦN 1: TỔNG QUAN ĐỀ TÀI** 2](#_Toc181187153)

[1. Giới thiệu chung 2](#_Toc181187154)

[2. Phạm vi tìm hiểu 2](#_Toc181187155)

[**PHẦN 2: QUY TRÌNH TRIỂN KHAI MÃ NGUỒN** 3](#_Toc181187156)

[1. Lý do chọn nền tảng Render 3](#_Toc181187157)

[2. Chuẩn bị mã nguồn 4](#_Toc181187158)

[*2.1 Cấu trúc mã nguồn* 4](#_Toc181187159)

[*2.2 Cài đặt các dependencies* 4](#_Toc181187160)

[*2.3 Tạo file Dockerfile hoặc Procfile (nếu có)* 5](#_Toc181187161)

[*2.4 Cấu hình biến môi trường (.env)* 5](#_Toc181187162)

[*2.5 Kiểm tra mã nguồn trước khi triển khai* 5](#_Toc181187163)

[3. Đẩy mã nguồn lên GitHub 6](#_Toc181187164)

[4. Cấu hình trên Render 7](#_Toc181187165)

[*4.1 Truy cập Render:* 7](#_Toc181187166)

[*4.2 Tạo Web Service mới:* 7](#_Toc181187167)

[*4.3 Cấu hình Web Service:* 8](#_Toc181187168)

[*4.4 Cấu hình biến môi trường:* 9](#_Toc181187169)

[5. Triển khai ứng dụng trên Render 9](#_Toc181187170)

[*5.1 Kiểm tra quá trình triển khai:* 9](#_Toc181187171)

[*5.2 Kiểm tra ứng dụng:* 10](#_Toc181187172)

[6. Cấu hình tên miền tùy chỉnh (nếu có) 11](#_Toc181187173)

[*6.1. Thêm tên miền vào Render:* 11](#_Toc181187174)

[*6.2. Cấu hình DNS trên trang quản lý tên miền:* 11](#_Toc181187175)

[*6.3. Cài đặt chứng chỉ SSL (Secure Socket Layer):* 11](#_Toc181187176)

[7. Theo dõi và bảo trì ứng dụng trên Render 11](#_Toc181187177)

[*7.1. Kiểm tra log ứng dụng:* 11](#_Toc181187178)

[*7.2. Cập nhật ứng dụng:* 11](#_Toc181187179)

[**PHẦN 3: KẾT LUẬN** 12](#_Toc181187180)

[**CHƯƠNG 2: LÝ THUYẾT CÁC THUẬT TOÁN** 13](#_Toc181187181)

[**PHẦN 1: GIỚI THIỆU VỀ CÁC THUẬT TOÁN HỌC MÁY** 13](#_Toc181187182)

[1. Định nghĩa và ứng dụng 13](#_Toc181187183)

[2. Phân loại các thuật toán học máy 14](#_Toc181187184)

[**PHẦN 2: TIỀN XỬ LÝ DỮ LIỆU** 14](#_Toc181187185)

[1. Định nghĩa hàm preprocess\_data: 14](#_Toc181187186)

[2. Loại bỏ các hàng chứa giá trị null: 14](#_Toc181187187)

[3. Xóa các cột không cần thiết: 15](#_Toc181187188)

[4. Trả về DataFrame đã được tiền xử lý: 15](#_Toc181187189)

[5. Gọi hàm preprocess data: 15](#_Toc181187190)

[**PHẦN 3: CÁC THUẬT TOÁN CỤ THỂ ĐƯỢC SỬ DỤNG** 15](#_Toc181187191)

1. [Hồi quy tuyến tính. 15](#_Toc181187192)

[2. Hồi quy Lasso 16](#_Toc181187193)

[3. Neural Network (Mạng nơ-ron nhân tạo ) 18](#_Toc181187194)

[**CHƯƠNG 3: PHÂN TÍCH VÀ ĐÁNH GIÁ BIỂU ĐỒ PHÂN BỔ DỮ LIỆU VÀ MÔ HÌNH HỌC KẾT HỢP** 21](#_Toc181187195)

[**PHẦN 1: PHÂN TÍCH ĐÁNH GIÁ BIỂU ĐỒ PHÂN BỔ DỮ LIỆU PAIR PLOT** 21](#_Toc181187196)

[**PHẦN 2: XÂY DỰNG MÔ HÌNH HỌC KẾT HỢP** 29](#_Toc181187197)

[**CHƯƠNG 4: PHÂN TÍCH BỘ THAM SỐ** 31](#_Toc181187198)

[1. Tham số trong Hồi quy tuyến tính 31](#_Toc181187199)

[2. Tham số trong Hồi quy Lasso 32](#_Toc181187200)

[3. Tham số trong Neural Network (MLPRegressor) 33](#_Toc181187201)

[4. Hàm bagging 35](#_Toc181187202)

[**CHƯƠNG 5: PHÂN TÍCH KẾT QUẢ KHI CHẠY ĐƯỢC MÔ HÌNH** 37](#_Toc181187203)

[**1.** **Kết quả sau khi đã nhập giá mở cửa:** 37](#_Toc181187204)

[1.1 Hồi Quy Tuyến Tính và Hồi Quy Lasso: 38](#_Toc181187205)

[1.2 Mô Hình Neural Network: 39](#_Toc181187206)

[1.3 Mô Hình Bagging: 39](#_Toc181187207)

[**2. Đánh giá MSE và R²** 39](#_Toc181187208)

[2.1 Mean Squared Error (MSE) 39](#_Toc181187209)

[2.2 R² (Coefficient of Determination) 40](#_Toc181187210)

[**KẾT LUẬN** 40](#_Toc181187211)

[1. Kết quả đạt được 40](#_Toc181187212)

[2. Hướng phát triển 41](#_Toc181187213)

[**LỜI CẢM ƠN** 42](#_Toc181187214)

[**TÀI LIỆU THAM KHẢO** 43](#_Toc181187215)

# 

# **LỜI NÓI ĐẦU**

Trong thời đại công nghệ số hiện nay, trí tuệ nhân tạo (AI) và học máy (Machine learning) đang trở thành những lĩnh vực được quan tâm và nghiên cứu sâu rộng. Học máy không chỉ là một ngành khoa học mà còn là một công cụ mạnh mẽ, giúp chúng ta giải quyết những bài toán phức tạp và tìm ra những mẫu hình tiềm ẩn trong dữ liệu.

Môn học máy cung cấp cho chúng ta những kiến thức nền tảng về cách xây dựng các mô hình dự đoán, phân loại và phân tích dữ liệu. Từ những thuật toán hồi quy đơn giản đến những mô hình phức tạp như mạng nơ-ron, học máy mở ra cơ hội cho chúng ta ứng dụng trong nhiều lĩnh vực như tài chính, y tế, marketing và nhiều ngành khác.

Trong khuôn khổ môn học này, chúng em đã có cơ hội tìm hiểu và thực hành với nhiều phương pháp học máy khác nhau. Chúng em đã được trang bị những kỹ năng cần thiết để xây dựng và tối ưu hóa mô hình, từ việc thu thập dữ liệu, tiền xử lý cho đến việc đánh giá và triển khai mô hình.

Nhận thức được những lợi ích to lớn của học máy, nhóm chúng em đã quyết định thực hiện đề tài *“Xây dựng mô hình dự đoán giá vàng năm 2025”* với ba thuật toán: *hồi quy tuyến tính, hồi quy Lasso và mạng nơ-ron nhân tạo*. Đề tài này nhằm phát triển một hệ thống có khả năng dự đoán chính xác giá vàng, giúp người dùng có thông tin đáng tin cậy để đưa ra quyết định đầu tư hợp lý.

Chúng em xin chân thành cảm ơn thầy *PGS.TS Nguyễn Quang Hoan* đã cung cấp cho chúng em những kiến thức cơ bản và tận tình hướng dẫn chúng em từng bước thực hiện đề tài này. Mặc dù chúng em đã cố gắng hết sức nhưng cũng không tránh khỏi sai sót. Chúng em rất mong nhận được sự cảm thông và góp ý của thầy để chúng em từng bước hoàn thiện hơn.

# **CHƯƠNG 1: TECHNICAL REPORT – TRIỂN KHAI MÃ NGUỒN**

## **PHẦN 1: TỔNG QUAN ĐỀ TÀI**

### ***Giới thiệu chung***

- Triển khai (deploy) mã nguồn là quá trình chuyển mã từ môi trường phát triển lên môi trường thực tế hoặc môi trường thử nghiệm để người dùng có thể truy cập và sử dụng ứng dụng. Quy trình này đóng vai trò quan trọng trong vòng đời phát triển phần mềm, giúp đảm bảo rằng sản phẩm đáp ứng các tiêu chuẩn về hiệu suất, tính ổn định và bảo mật khi đưa vào sử dụng.

- Trong bối cảnh công nghệ ngày càng phát triển, việc triển khai mã nguồn không chỉ đơn thuần là việc chuyển đổi mã từ môi trường này sang môi trường khác, mà còn bao gồm nhiều bước phức tạp và chi tiết. Từ việc chuẩn bị mã nguồn, cấu hình môi trường, triển khai tự động, đến giám sát và bảo trì sau khi triển khai, mỗi bước đều đòi hỏi sự chính xác và cẩn thận để đảm bảo rằng ứng dụng hoạt động một cách hiệu quả và an toàn.

- Mục tiêu của báo cáo là trình bày các bước cơ bản trong quy trình triển khai mã nguồn, từ việc chuẩn bị mã đến cấu hình môi trường và giám sát sau khi triển khai.

### ***Phạm vi tìm hiểu***

- Báo cáo sẽ tập trung vào quy trình triển khai mã nguồn trên nền tảng Render, bao gồm các bước thiết lập, triển khai và quản lý ứng dụng.

- Phân tích các phương pháp triển khai ứng dụng thủ công và tự động trên Render, ưu và nhược điểm của từng phương pháp.

- Tìm hiểu về các tính năng của Render như triển khai tự động qua Git, auto-scaling, SSL tự động và giám sát hoạt động của ứng dụng.

- Nghiên cứu về quy trình giám sát sau triển khai và bảo trì hệ thống trên Render, nhằm đảm bảo ứng dụng hoạt động ổn định và có thể phát hiện, xử lý sự cố kịp thời.

## **PHẦN 2: QUY TRÌNH TRIỂN KHAI MÃ NGUỒN**

### **Lý do chọn nền tảng Render.**

Render (truy cập tại <https://render.com>) là một nền tảng đám mây hiện đại, cung cấp các giải pháp triển khai tự động, ổn định và dễ sử dụng.

* **Tự động hóa triển khai**: Render hỗ trợ triển khai tự động mỗi khi có thay đổi trong mã nguồn (auto-deploy) nhờ tích hợp với GitHub, GitLab, và Bitbucket. Điều này giúp duy trì tốc độ phát triển liên tục và đảm bảo cập nhật nhanh chóng mà không cần thao tác thủ công sau mỗi lần cập nhật.
* **Dễ sử dụng và giao diện thân thiện**: Giao diện của Render trực quan và dễ thao tác, phù hợp cho cả người mới bắt đầu và các nhà phát triển giàu kinh nghiệm. Quy trình triển khai trên Render ít bước, không đòi hỏi thiết lập phức tạp, giúp giảm thời gian và công sức.
* **Tích hợp sẵn các tính năng quản lý và bảo mật**: Render hỗ trợ các tính năng như SSL tự động, auto-scaling, giám sát logs và hiệu suất ứng dụng, giúp giảm thiểu nguy cơ về bảo mật và tối ưu hóa hiệu suất mà không cần cấu hình thêm. Các tính năng này tạo nên một môi trường vận hành ổn định và an toàn.
* **Chi phí hợp lý**: Render cung cấp các gói dịch vụ với chi phí phù hợp cho nhiều quy mô dự án, bao gồm cả các gói miễn phí và trả phí linh hoạt. Điều này đặc biệt có lợi cho các dự án mới, cá nhân, hoặc startup muốn tối ưu chi phí.
* **Hỗ trợ đa dạng công nghệ**: Render có thể triển khai nhiều ngôn ngữ lập trình và framework phổ biến như Python, Node.js, Ruby, Docker..., nên dễ dàng tích hợp với nhiều loại ứng dụng mà không cần thay đổi cấu trúc mã nguồn ban đầu.
* **Đảm bảo tính linh hoạt và mở rộng**: Với Render, bạn có thể dễ dàng mở rộng tài nguyên máy chủ khi lưu lượng truy cập tăng hoặc khi cần cải thiện hiệu suất. Tính năng này đảm bảo ứng dụng có thể phát triển theo nhu cầu sử dụng thực tế mà không bị gián đoạn.

A screenshot of a computer

Description automatically generated

*Hình 1: Giao diện triển khai trên Render*

Việc lựa chọn Render không chỉ giúp đơn giản hóa quy trình triển khai và quản lý, mà còn đảm bảo tính ổn định và an toàn cho người dùng. Đây là yếu tố then chốt để cung cấp dịch vụ dự đoán giá vàng một cách chính xác, hiệu quả và duy trì trải nghiệm mượt mà cho người dùng trong thời gian dài.

### **2. Chuẩn bị mã nguồn**

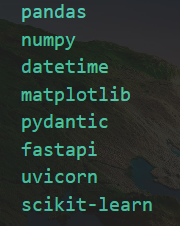
#### **2.1 Cấu trúc mã nguồn**

Xác định cấu trúc thư mục và các file chính cần thiết để đảm bảo mã nguồn gọn gàng và dễ dàng triển khai. Ví dụ: Thư mục src (source code) chứa mã nguồn chính, bao gồm tệp khởi động ứng dụng (app.py hoặc main.py), các tệp xử lý điều hướng (routes/), mô hình dữ liệu (models/), logic nghiệp vụ (services/), và các hàm tiện ích (utils/). Thư mục config chứa các file cấu hình như config.py cho thông số chung và database\_config.py cho thông tin kết nối cơ sở dữ liệu. Thư mục static lưu trữ các tài nguyên tĩnh như hình ảnh (images/), tệp CSS (css/), và JavaScript (js/)…

#### **2.2 Cài đặt các dependencies**

Để đảm bảo ứng dụng hoạt động chính xác trên nền tảng Render, việc liệt kê và cài đặt các thư viện cùng công cụ cần thiết là rất quan trọng. Bạn có thể sử dụng các file quản lý dependencies như requirements.txt cho ứng dụng Python hoặc package.json cho ứng dụng Node.js.

* **Đối với Python**: Tạo file requirements.txt chứa danh sách các thư viện mà ứng dụng sử dụng. Ví dụ



*Hình 2: Các thư viện trong requirements.txt*

Sau khi tạo xong file này, bạn chỉ cần chạy lệnh sau để cài đặt tất cả các dependencies:



*Hình 3: Terminal hiển thị vị trí thư mục và lệnh pip install -r requirements.txt.*

* **Đối với Node.js**: Sử dụng file package.json, nơi liệt kê tất cả các gói mà dự án cần. Khi bạn chạy lệnh npm install, Node.js sẽ tự động cài đặt tất cả các gói được chỉ định trong package.json.

A screen shot of a computer program

Description automatically generated

*Hình 4: Hình minh họa sau khi chạy npm install.*

Điều này sẽ giúp đảm bảo rằng môi trường chạy trên Render có đầy đủ các thư viện cần thiết để ứng dụng hoạt động chính xác.

#### **Tạo file Dockerfile hoặc Procfile (nếu cần)**

Hướng dẫn cách tạo Dockerfile để đóng gói ứng dụng nếu ứng dụng sử dụng Docker, hoặc tạo Procfile nếu sử dụng các ngôn ngữ như Python, Node.js để chỉ định cách ứng dụng sẽ được khởi động trên Render.

#### **2.4 Cấu hình biến môi trường (.env)**

Cung cấp các biến môi trường cần thiết như thông tin database, API key hoặc các thông tin bảo mật khác. Chúng có thể được cấu hình trong file .env để dễ dàng quản lý và bảo mật thông tin nhạy cảm.

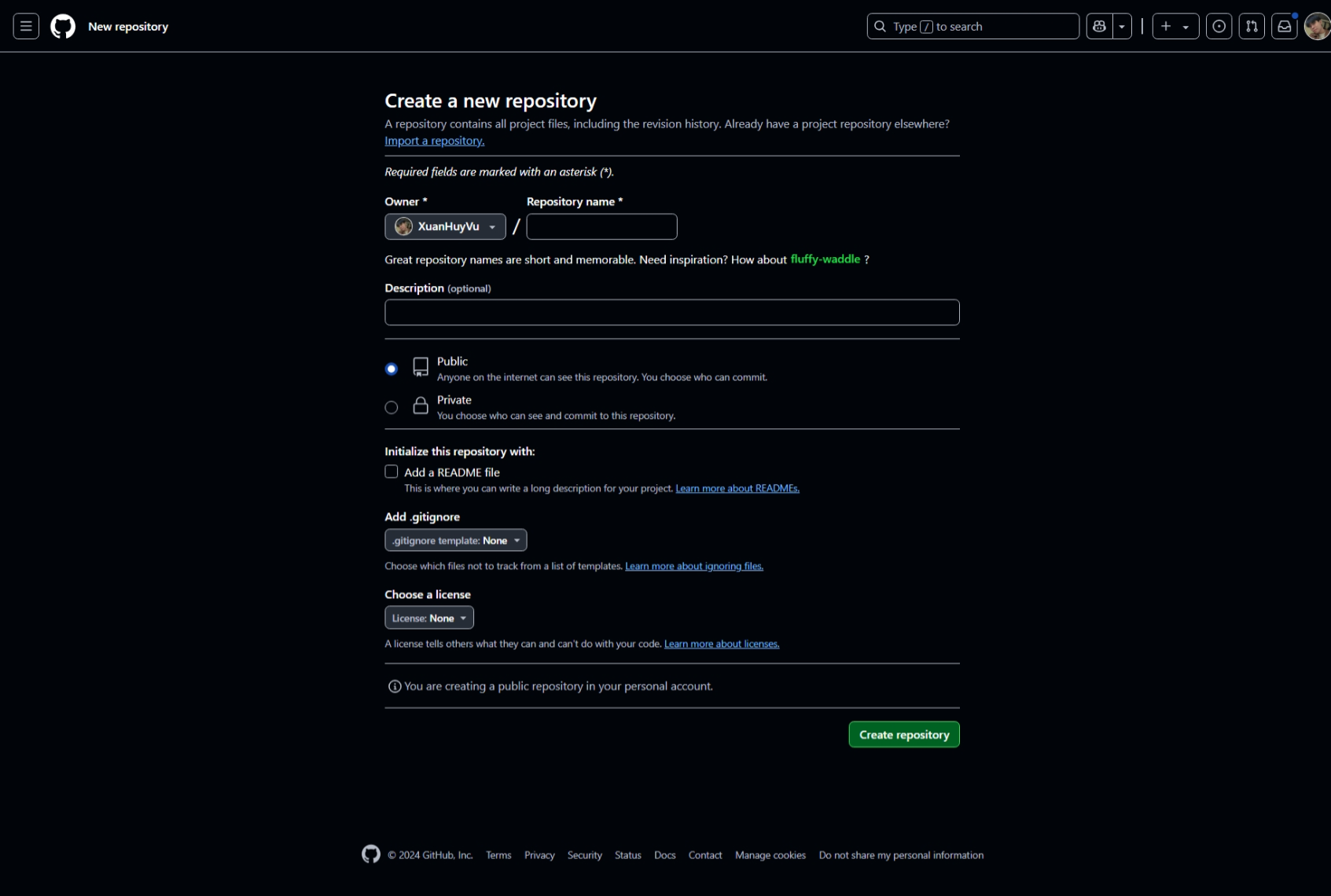
#### **2.5 Kiểm tra mã nguồn trước khi triển khai**

Để đảm bảo mã nguồn không có lỗi cú pháp và lỗi logic trước khi triển khai ứng dụng lên Render, cần thực hiện kiểm tra mã nguồn cẩn thận. Trước hết, sử dụng ESLint để kiểm tra cú pháp và phong cách mã. Sau đó, viết các bài kiểm thử với Mocha hoặc Jest để đảm bảo các chức năng hoạt động như mong đợi. Cuối cùng, chạy ứng dụng trên môi trường phát triển để kiểm tra các chức năng thủ công và đảm bảo mọi biến môi trường trong tệp .env được cấu hình chính xác. Việc này giúp giảm thiểu rủi ro và đảm bảo ứng dụng hoạt động ổn định sau khi triển khai.

### **Đẩy mã nguồn lên GitHub**

*Bước 1:* Tạo Repository trên GitHub

1. Đăng nhập vào GitHub: Truy cập [GitHub](https://github.com) và đăng nhập vào tài khoản của bạn.
2. Tạo Repository mới: Nhấp vào nút "+" ở góc trên bên phải và chọn "New repository".
3. Nhập thông tin: Điền tên cho repository, chọn chế độ công khai (public) hoặc riêng tư (private), và nhấn "Create repository".

****

*Hình 5: Tạo Repository trên GitHub*

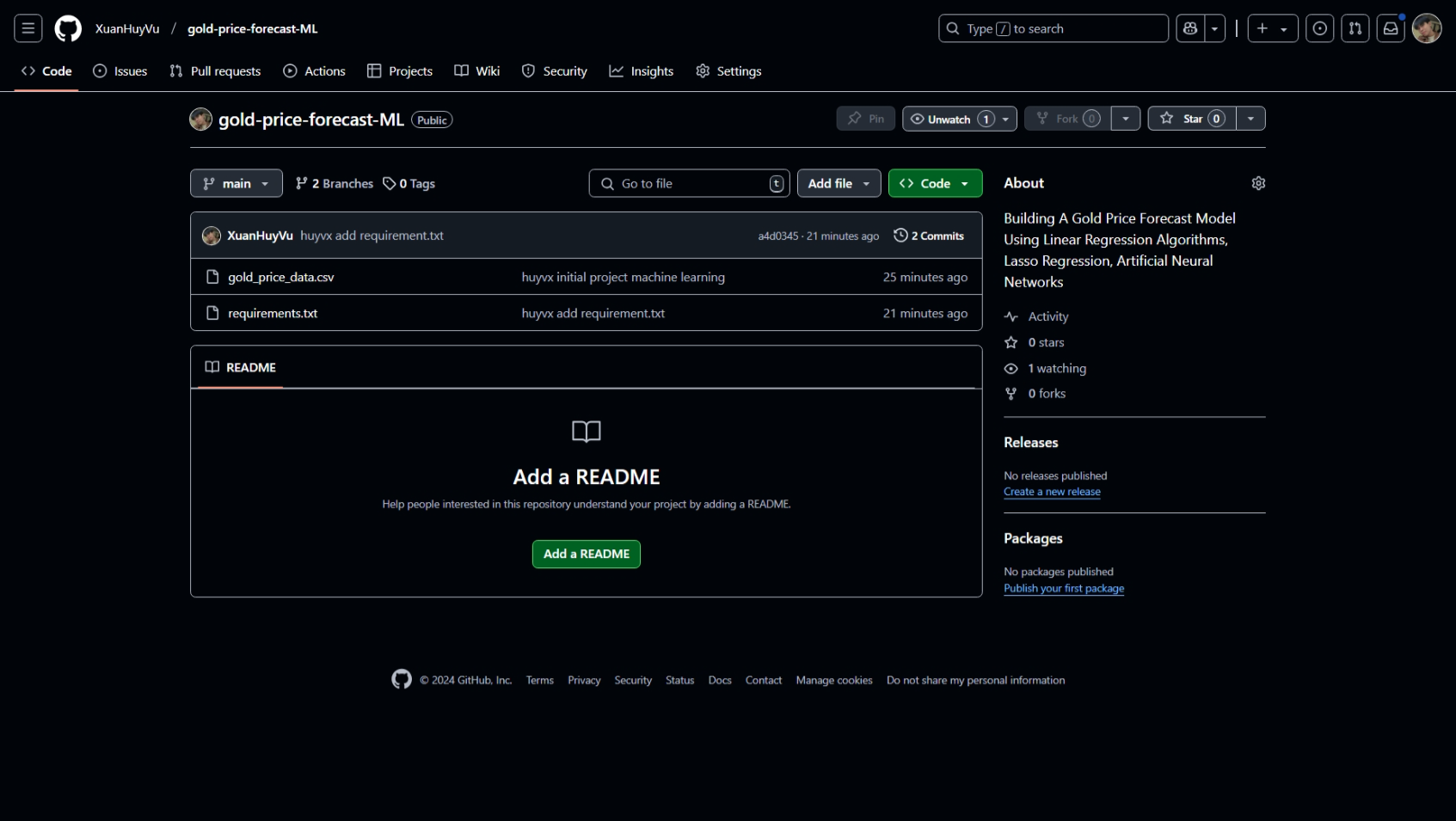
*Bước 2:* Đẩy mã nguồn lên GitHub

1. Mở Terminal (hoặc Command Prompt): Truy cập vào thư mục dự án của bạn.
2. Khởi tạo Git (nếu chưa): git init

1. Thêm remote repository: Liên kết thư mục cục bộ với repository vừa tạo trên GitHub. git remote add origin https://github.com/XuanHuyVu/gold-price-forecast-ML.git
2. Thêm các tệp vào staging: git add .
3. Commit mã nguồn: git commit -m 'huyvx initial project machine learning'
4. Tạo nhánh main: git branch -M main
5. Đẩy mã nguồn lên GitHub: git push -u origin main

***Kết thúc***

Sau khi hoàn thành các bước trên, mã nguồn sẽ được đẩy lên GitHub. Có thể kiểm tra lại repository trên GitHub để xác nhận rằng các tệp đã được tải lên thành công.



*Hình 12: Đẩy mã nguồn lên GitHub thành công*

### **4. Cấu hình trên Render**

* 1. **Truy cập Render:** Vào trang [Render](https://render.com/) và đăng ký hoặc đăng nhập tài khoản của bạn.

#### **4.2 Tạo Web Service mới:**

* Tại dashboard của Render, chọn **"New"** -> **"Web Service"**.
* Chọn repo từ GitHub mà bạn đã đẩy mã lên trong bước trước và cấp quyền truy cập.

#### **4.3 Cấu hình Web Service:**

* **Name**: Đặt tên cho ứng dụng của bạn. Tên này không chỉ giúp bạn dễ dàng nhận diện ứng dụng trên dashboard của Render mà còn hiển thị trên URL nếu bạn không cấu hình tên miền riêng. Hãy chọn một cái tên ngắn gọn, dễ nhớ và phản ánh chức năng của ứng dụng. Ví dụ: nếu ứng dụng của bạn dự đoán giá vàng, bạn có thể đặt tên là “GoldPrice”.
* **Region**: Chọn khu vực triển khai ứng dụng. Bạn có thể lựa chọn giữa **US** (Hoa Kỳ) hoặc **EU** (Châu Âu).
* **Branch**: Chọn nhánh Git mà bạn muốn triển khai. Thông thường, nhánh chính sẽ là **main** hoặc **master**.
* **Build Command**: Đây là lệnh được thực thi để xây dựng ứng dụng. Trong nhiều trường hợp, bạn có thể để trống trường này, đặc biệt đối với ứng dụng FastAPI, vì chúng không yêu cầu bước build đặc biệt.
* **Start Command**: Nhập lệnh để khởi chạy ứng dụng của bạn. Đây là lệnh mà Render sẽ sử dụng để bắt đầu ứng dụng sau khi nó được triển khai. Ví dụ, nếu bạn sử dụng Uvicorn để chạy ứng dụng FastAPI

uvicorn main:app --host 0.0.0.0 --port 8000

Trong đó:

* main là tên của tệp Python mà bạn đã viết mã nguồn, không bao gồm phần mở rộng .py.
* app là tên của đối tượng FastAPI mà bạn đã tạo trong mã. Đảm bảo rằng tên này chính xác để ứng dụng có thể khởi động mà không gặp phải lỗi.

*A screenshot of a computer

Description automatically generated*

*Hình 14: Minh họa cấu hình web service*

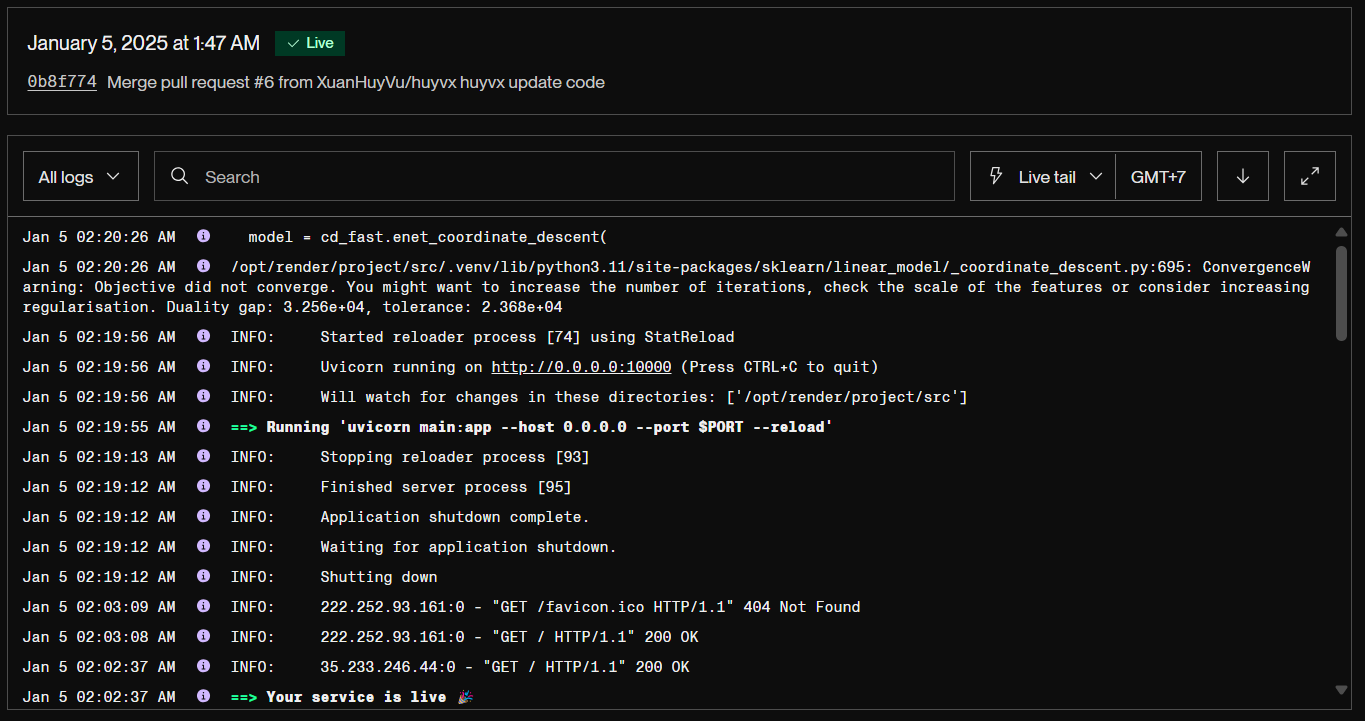
#### **4.4 Cấu hình biến môi trường:**

Trong phần **Environment Variables**, thêm các biến môi trường mà ứng dụng cần, như DATABASE\_URL, API\_KEY, v.v.

### **5. Triển khai ứng dụng trên Render**

#### **5.1 Kiểm tra quá trình triển khai:**

* Sau khi nhấn nút **"Create Web Service"**, Render sẽ tự động lấy mã từ GitHub và triển khai.
* Theo dõi log để xem quá trình build và khởi chạy ứng dụng có thành công hay không.



*Hình 15: Minh họa log quá trình build và khởi chạy ứng dụng trên Render.*

* Log này bao gồm từng bước trong quy trình build, giúp người dùng xác định xem ứng dụng có khởi chạy thành công hay không. Khi build hoàn tất, log sẽ hiển thị các thông báo xác nhận như “Build Successful” hoặc “Application Started” để báo hiệu rằng ứng dụng đã sẵn sàng hoạt động. Bất kỳ lỗi nào phát sinh trong quá trình này cũng sẽ được hiển thị, giúp dễ dàng phát hiện và sửa chữa trước khi ứng dụng đi vào hoạt động.

#### **5.2 Kiểm tra ứng dụng:**

Sau khi hoàn tất việc triển khai trên Render, bạn sẽ nhận được một URL để truy cập trực tiếp vào ứng dụng. URL này có dạng https://<tên-app>.onrender.com, ví dụ như <https://gold-price-forecast-ml.onrender.com/>.

**Bước kiểm tra ứng dụng**:

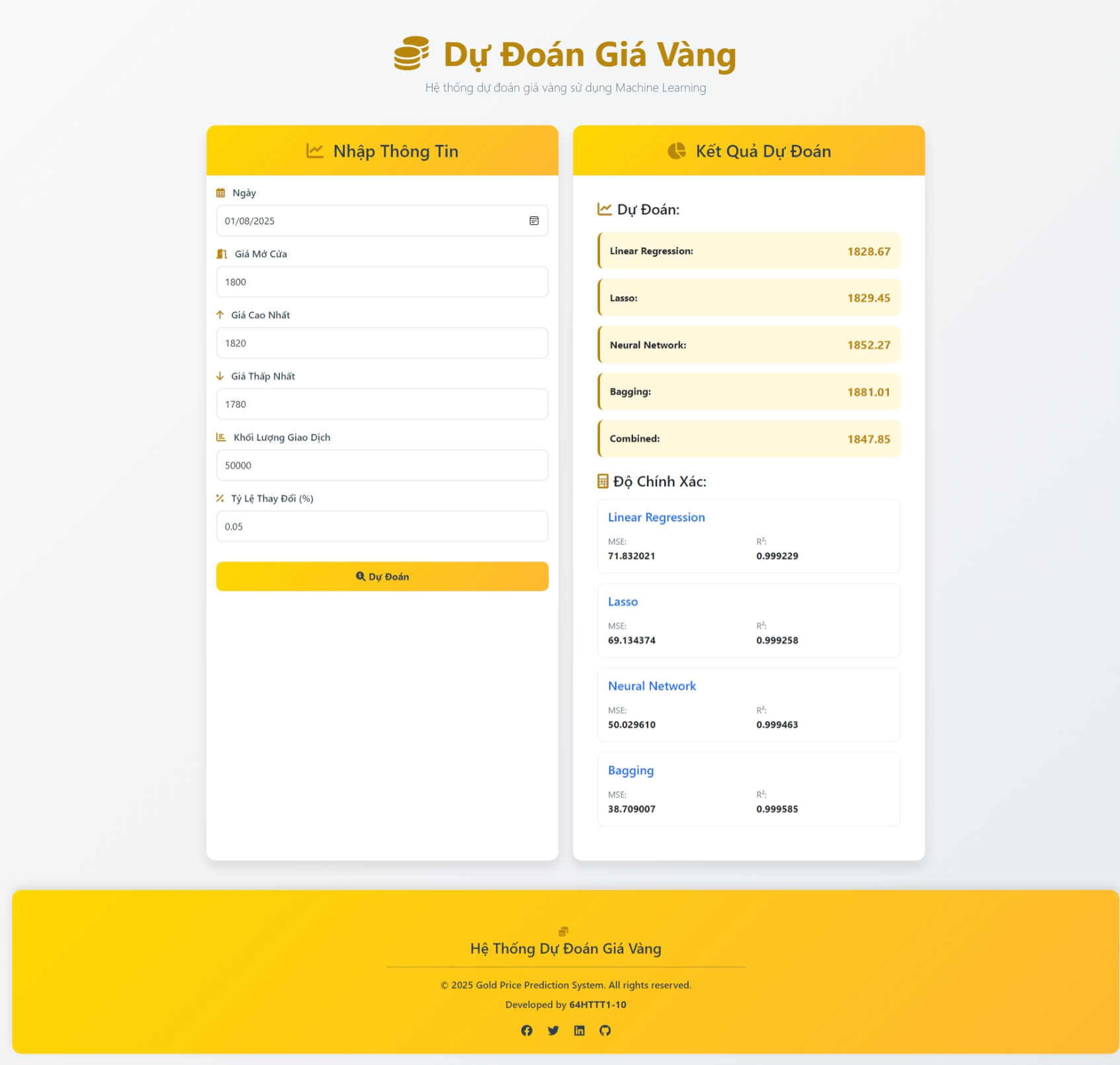
* + 1. Truy cập vào URL đã được Render cung cấp, ví dụ

<https://gold-price-forecast-ml.onrender.com/>.

**5.2.2** Kiểm tra toàn bộ các tính năng, bao gồm:

* Chức năng nhập và hiển thị dữ liệu.
* Các nút hoặc tùy chọn tương tác.
* Kết quả hiển thị và bất kỳ thông tin phản hồi nào từ ứng dụng.

Nếu có vấn đề xảy ra, bạn có thể xem lại phần logs trong Render để xem chi tiết lỗi và khắc phục.

**

*Hình 16: Minh họa ứng dụng dự đoán giá vàng chạy ứng dụng trên Render.*

### **6. Cấu hình tên miền tùy chỉnh (nếu có)**

#### **6.1. Thêm tên miền vào Render:**

* Truy cập vào **trang quản lý dịch vụ của Render** và tìm ứng dụng muốn cấu hình tên miền tùy chỉnh.
* Trên giao diện quản lý của ứng dụng, chọn **Settings** (Cài đặt).
* Trong mục **Custom Domains** (Tên miền tùy chỉnh), chọn **Add Custom Domain** (Thêm tên miền tùy chỉnh).
* Nhập tên miền muốn sử dụng (ví dụ: <https://goldpricenhom4.onrender.com>), sau đó nhấn **Save** (Lưu) để xác nhận.

#### **6.2. Cấu hình DNS trên trang quản lý tên miền:**

* Truy cập vào **trang quản lý tên miền** (chẳng hạn như Namecheap, GoDaddy, Cloudflare, v.v.).
* Tìm phần **DNS Settings** (Cài đặt DNS) hoặc **Domain Management** (Quản lý tên miền).
* Thêm một **bản ghi CNAME** (Canonical Name) trong DNS. Thông tin bản ghi CNAME cần thiết lập như sau:

**Tên (Name)**: Nhập tên miền muốn dùng (ví dụ: www).

**Giá trị (Value)**: Trỏ bản ghi CNAME tới tên miền mà Render cung cấp, thường là dạng <tên-app>.onrender.com.

* Lưu lại các thay đổi. Lưu ý rằng việc cập nhật DNS có thể mất vài phút hoặc lâu hơn tùy thuộc vào nhà cung cấp dịch vụ tên miền.

#### **6.3. Cài đặt chứng chỉ SSL (Secure Socket Layer):**

* Render sẽ **tự động cài đặt chứng chỉ SSL** cho tên miền tùy chỉnh của bạn sau khi xác nhận tên miền và DNS được cấu hình chính xác. Điều này đảm bảo rằng kết nối giữa người dùng và ứng dụng sẽ được mã hóa và bảo mật.
* Không cần thực hiện thêm bất kỳ thao tác nào cho việc cài đặt SSL. Sau khi quá trình cập nhật DNS hoàn tất, trang web của bạn sẽ tự động sử dụng **https** với chứng chỉ SSL đã được Render cung cấp.

### **7. Theo dõi và bảo trì ứng dụng trên Render**

#### **7.1. Kiểm tra log ứng dụng:**

* Trong **dashboard của Render**, tìm và chọn ứng dụng.
* Vào mục **Logs** để xem các thông tin ghi lại từ quá trình chạy ứng dụng. Logs sẽ hiển thị chi tiết về các lỗi, cảnh báo và thông tin khác, giúp bạn theo dõi tình trạng hoạt động của ứng dụng.
* Thường xuyên kiểm tra log để phát hiện sớm các lỗi hoặc sự cố xảy ra trong ứng dụng và xử lý kịp thời.

#### **7.2. Cập nhật ứng dụng:**

* Khi thực hiện thay đổi mã nguồn trên GitHub, hãy **commit và push** các thay đổi đó lên GitHub.
* Render sẽ tự động phát hiện các thay đổi mới này, sau đó tiến hành **build và triển khai lại** phiên bản mới của ứng dụng mà không cần can thiệp thủ công.
* Đảm bảo rằng các bản cập nhật không gây gián đoạn cho người dùng, đặc biệt với các bản cập nhật quan trọng. Cân nhắc lên lịch bảo trì hoặc thông báo trước cho người dùng nếu cần thiết.

## **PHẦN 3: KẾT LUẬN**

Trong báo cáo kỹ thuật này, đã trình bày quy trình triển khai ứng dụng trên nền tảng Render, từ việc cấu hình và triển khai ban đầu cho đến các bước bảo trì và theo dõi. Quy trình triển khai bao gồm:

1. **Chuẩn bị mã nguồn:** Kiểm tra mã nguồn và cấu hình các biến môi trường trong file .env để đảm bảo bảo mật.
2. **Đẩy mã nguồn lên GitHub:** Khởi tạo git, kết nối với kho lưu trữ, và đẩy mã nguồn lên GitHub.
3. **Cấu hình trên Render:** Tạo web service mới và cấu hình các thông tin cần thiết cho ứng dụng.
4. **Triển khai ứng dụng:** Theo dõi quá trình triển khai và kiểm tra log để phát hiện lỗi.
5. **Cấu hình tên miền tùy chỉnh:** Thêm tên miền riêng và cấu hình DNS, với SSL tự động cho bảo mật.
6. **Theo dõi và bảo trì:** Kiểm tra log thường xuyên và cập nhật mã nguồn tự động để duy trì hoạt động ổn định

Bằng cách tuân thủ các bước này, ứng dụng không chỉ được triển khai thành công mà còn có thể duy trì sự ổn định và an toàn qua thời gian. Việc theo dõi liên tục và cập nhật kịp thời sẽ đảm bảo rằng ứng dụng luôn đáp ứng được nhu cầu của người dùng và giữ vững hiệu suất hoạt động.

Nhận thấy rằng việc triển khai trên Render cung cấp nhiều tiện ích như tự động hóa trong quá trình xây dựng và triển khai, giúp giảm thiểu công sức cần thiết cho quản lý ứng dụng. Điều này góp phần vào việc tối ưu hóa quy trình phát triển phần mềm, cho phép đội ngũ phát triển tập trung vào việc cải tiến và mở rộng các tính năng của ứng dụng hơn là quản lý hạ tầng.

Tóm lại, quá trình triển khai ứng dụng trên Render không chỉ dễ dàng mà còn hiệu quả, mở ra nhiều cơ hội để cải thiện và phát triển ứng dụng trong tương lai.

# **CHƯƠNG 2: LÝ THUYẾT CÁC THUẬT TOÁN**

## **PHẦN 1: GIỚI THIỆU VỀ CÁC THUẬT TOÁN HỌC MÁY**

### **1. Định nghĩa và ứng dụng**

Học máy (Machine Learning) là một nhánh quan trọng của trí tuệ nhân tạo (Artificial Intelligence), nghiên cứu và phát triển các thuật toán cho phép máy tính tự động học hỏi và cải thiện hiệu suất thông qua trải nghiệm với dữ liệu mà không cần sự can thiệp của con người. Bằng cách áp dụng các phương pháp thống kê và phân tích, học máy mở ra cánh cửa cho những khả năng tiên tiến trong việc giải quyết các bài toán phức tạp.

**Ứng dụng của học máy hiện nay rất phong phú và đa dạng, bao gồm:**

* **Nhận diện hình ảnh**: Công nghệ nhận diện khuôn mặt, phân loại hình ảnh và phân tích video đang trở thành xu hướng trong nhiều ứng dụng thực tiễn, từ an ninh đến giải trí.
* **Xử lý ngôn ngữ tự nhiên:** Học máy giúp cải thiện các hệ thống chatbot, dịch ngôn ngữ tự động, và phân tích cảm xúc, góp phần tạo ra trải nghiệm người dùng tốt hơn.
* **Dự đoán và phân tích dữ liệu**: Học máy đóng vai trò quan trọng trong việc dự đoán giá trị thị trường, phân tích hành vi khách hàng, và phát hiện các bất thường trong dữ liệu.
* **Y tế**: Ứng dụng trong chẩn đoán bệnh, phân tích gen, và phát triển thuốc, học máy đang giúp tăng cường hiệu quả và chính xác trong lĩnh vực y tế.

### **2. Phân loại các thuật toán học máy**

Các thuật toán học máy có thể được chia thành ba loại chính, mỗi loại có phương pháp và ứng dụng riêng biệt:

* **Học có giám sát (Supervised Learning)**: Trong phương pháp này, mô hình được huấn luyện trên một tập dữ liệu đã được gán nhãn. Mục tiêu là xây dựng một hàm dự đoán chính xác nhãn cho dữ liệu chưa thấy.
* **Học không giám sát (Unsupervised Learning)**: Ngược lại với học có giám sát, trong học không giám sát, mô hình làm việc với dữ liệu không có nhãn. Mục tiêu là khám phá cấu trúc và mẫu trong dữ liệu. Các thuật toán như phân cụm K-means và phân tích thành phần chính (PCA) giúp tìm ra những thông tin ẩn giấu trong tập dữ liệu.
* **Học tăng cường (Reinforcement Learning)**: Máy học thông qua tương tác với môi trường và đưa ra các hành động để tối ưu hóa phần thưởng tổng thể. Dựa trên cơ chế thưởng và phạt; thuật toán tự điều chỉnh hành vi để đạt được kết quả tốt nhất theo thời gian.Ứng dụng trong các hệ thống tự động hoá như robot, game AI, xe tự lái.

### **PHẦN 2: TIỀN XỬ LÝ DỮ LIỆU**

## **Định nghĩa hàm preprocess\_data:**

Hàm này nhận một DataFrame (df) làm tham số và thực hiện các bước tiền xử lý dữ liệu.

## **Loại bỏ các hàng chứa giá trị null:**



*Hình 17: Câu lệnh loại bỏ hàng chứa giá trị null – rỗng*

**Mô tả**: Dòng lệnh này loại bỏ tất cả các hàng trong DataFrame (df) mà chứa bất kỳ giá trị null (NaN) nào. Việc này giúp đảm bảo rằng dữ liệu không có khoảng trống và có thể phân tích chính xác hơn.

**Chú thích**:

* .copy() tạo một bản sao của Dataframe để tránh các warning về việc thay đổi dữ liệu gốc.

## **Xử lý cột Date:**

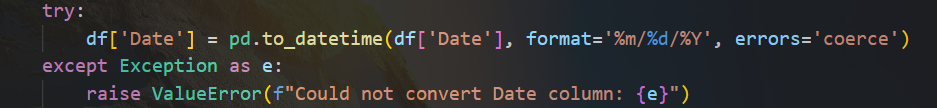


*Hình 18: Câu lệnh xử lý dữ liệu cột Date*

Xử lý cột Date:

* .str.replace('"', ''): Xóa bỏ các dấu ngoặc kép trong chuỗi ngày tháng
* .str.strip(): Xóa khoảng trắng thừa ở đầu và cuối chuỗi

1. **Chuyển cột Date sang định dạng datetime.**



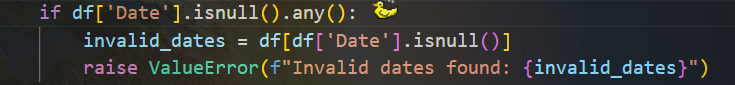
*Hình 19: Mã lệnh chuyển dữ liệu cột Date sang định dạng datetime.*

**Mô tả**: Chuyển đổi giá trị trong cột 'Date' của DataFrame df từ kiểu dữ liệu hiện tại (ví dụ: chuỗi hoặc số) sang kiểu datetime64[ns].Sau khi chuyển đổi, cột 'Date' sẽ lưu các giá trị ngày tháng dưới dạng datetime, cho phép thực hiện các thao tác phân tích dữ liệu thời gian, như trích xuất năm, tháng, ngày hoặc so sánh thời gian.

**Chú thích:**

* format='%m/%d/%Y': Định dạng ngày tháng đầu vào (tháng/ngày/năm)
* errors='coerce': Nếu có giá trị không chuyển đổi được sẽ chuyển thành NaT (Not a Time)
* Nếu có lỗi, raise một ValueError với thông báo lỗi.

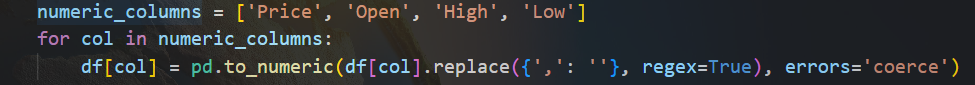
1. **Kiểm tra ngày tháng không hợp lệ**

****

*Hình 20: Mã lệnh kiểm tra ngày tháng không hợp lệ.*

**Mô tả:** Kiểm tra xem có giá trị null trong cột Date không. Nếu có, tìm các dòng có ngày không hợp lệ và raise ValueError.

1. **Chuyển đổi dữ liệu các cột số**



*Hình 21: Mã lệnh chuyển đổi các cột số.*

**Chú thích:**

* Định nghĩa list các cột cần chuyển đổi sang số.
* replace({',': ''}, regex=True):
* Xóa dấu phẩy trong số (ví dụ: "1,234.56" -> "1234.56")
* Sử dụng regex để tìm kiếm và thay thế
* pd.to\_numeric():
* Chuyển đổi chuỗi sang kiểu số.
* Tự động xác định kiểu số phù hợp (int hoặc float)
* errors='coerce': Đặt giá trị không hợp lệ thành NaN

1. **Xử lý dữ liệu cột Vol.**

****

*Hình 22: Câu lệnh xử lý dữ liệu cột Vol. (Volume)*

**Mô tả:** Xử lý đặc biệt cho cột Vol. vì có định dạng khác: Xóa cả dấu phẩy và ký tự 'K' (viết tắt của nghìn). Ví dụ: "1,234K" -> "1234" -> 1234000. Nhân với 1000 để chuyển đổi từ K (nghìn) sang đơn vị gốc.

1. **Xử lý dữ liệu cột Change %**



*Hình 23: Câu lệnh xử lý dữ liệu cột Change %*

**Mô tả:** Xử lý cột phần trăm:

* Xóa ký tự '%', chuyển sang số, chia cho 100 để chuyển từ phần trăm sang decimal.
* Ví dụ: "5.2%" -> 0.052

## **Trả về DataFrame đã được tiền xử lý:**



*Hình 24: Câu lệnh trả về DataFrame (df) được sắp xếp theo cột Date*

**Mô tả**: Dòng lệnh này trả về DataFrame (df) đã được làm sạch và loại bỏ các cột không cần thiết, giúp chuẩn bị dữ liệu cho các bước tiếp theo trong quy trình phân tích hoặc xây dựng mô hình.

**Lợi ích**: Việc trả về DataFrame đã được xử lý giúp đảm bảo rằng các bước tiếp theo trong quy trình phân tích dữ liệu sẽ hoạt động với dữ liệu chất lượng cao hơn.

## **Gọi hàm preprocess data:**



*Hình 25: Câu lệnh gọi lại hàm preprocess\_data*

Gọi hàm preprocess\_data với DataFrame (df) ban đầu và gán kết quả đã được tiền xử lý trở lại cho df

### **PHẦN 3: CÁC THUẬT TOÁN CỤ THỂ ĐƯỢC SỬ DỤNG**

## **Hồi quy tuyến tính.**

Là một kỹ thuật phân tích dữ liệu dự đoán giá trị của dữ liệu không xác định bằng cách sử dụng một giá trị dữ liệu liên quan và đã biết khác

Mô hình toán học biến không xác định hoặc phụ thuộc và biến đã biết hoặc độc lập như một phương trình tuyến tính

***Hồi quy tuyến tính có 2 dạng chính:***

*1.1. Hồi quy tuyến tính đơn:*

Hồi quy tuyến tính đơn mô tả mối quan hệ giữa **một** biến độc lập và **một** biến phụ thuộc. Công thức của mô hình này là: +

*Trong đó:*

* **y**: Biến phụ thuộc (giá trị cần dự đoán),
* **x**: Biến độc lập (biến giải thích),
* **:** Hằng số chặn (intercept), là giá trị của **y** khi **x** bằng 0,
* : Hệ số hồi quy (slope), thể hiện mức độ thay đổi của **y** khi **x** thay đổi một đơn vị,
* **ϵ**: Sai số (error term), đại diện cho sự khác biệt giữa giá trị thực và giá trị dự đoán.

*1.2. Hồi quy tuyến tính đa biến:*

Hồi quy tuyến tính đa biến mô tả mối quan hệ giữa **nhiều** biến độc lập và **một** biến phụ thuộc. Công thức của mô hình này là:

+

*Trong đó:*

* **y**: Biến phụ thuộc,
* **:** Các biến độc lập,
* : Hằng số chặn,
* **, , ..., :** Các hệ số hồi quy tương ứng với từng biến độc lập,
* **ϵ**: Sai số.

***Các bước thực hiện***

Bước 1: Chuẩn bị dữ liệu: thu thập dữ liệu, tiền xử lý, chia tập dữ liệu

Bước 2: Đặt tên các biến x (biến độc lập), y (biến phụ thuộc)

Bước 3: Xây dựng mô hình, hàm Loss Function

Bước 4: Huấn luyện mô hình

Bước 5: Đánh giá mô hình: kiểm tra chính xác trên tập kiểm tra

***Ưu điểm của Hồi quy tuyến tính***

* Đơn giản: Hồi quy tuyến tính dễ thực hiện và cho ra kết quả khả quan. Mô hình có thể hoạt động hiệu quả ngay cả trên hệ thống có sức mạnh tính toán thấp.
* Hiệu suất trên dữ liệu tuyến tính: Thích hợp cho các bộ dữ liệu có khả năng tách tuyến tính, giúp tìm hiểu mối quan hệ giữa các biến.
* Giảm Overfitting: Sử dụng kỹ thuật Regularization để giảm độ phức tạp của mô hình, từ đó hạn chế tình trạng overfitting.

***Nhược điểm của Hồi quy tuyến tính***

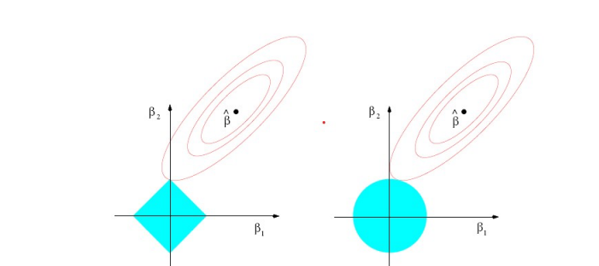
* Dễ Underfitting: Không thu thập dữ liệu đúng cách khi giả định mối quan hệ tuyến tính không phù hợp với dữ liệu phức tạp.
* Nhạy cảm với dữ liệu ngoại lai: Điểm dữ liệu bất thường có thể làm giảm hiệu suất và độ chính xác của mô hình. Cần xử lý dữ liệu ngoại lai trước khi áp dụng mô hình.

## **Hồi quy Lasso**

Phương pháp hồi quy LASSO (Least Absolute Shrinkage and Selection Operator: Toán tử chọn lọc và co rút tuyệt đối nhỏ nhất) là một kỹ thuật trong thống kê và học máy, được sử dụng để thực hiện cả việc lựa chọn biến và chính quy hóa (regularization) nhằm cải thiện độ chính xác dự đoán và khả năng diễn giải của mô hình hồi quy.

*2.1 Cơ sở toán học của LASSO:*

LASSO tìm các ước lượng tham số của mô hình hồi quy bằng *cách cực tiểu hóa hàm mục tiêu* là tổng bình phương các phần dư, nhưng với một ràng buộc bổ sung. Hàm mục tiêu của LASSO có dạng:



*Hình 26: Minh họa hình học hàm mục tiêu và miền giới hạn của Lasso (trái) và Ridge (phải) Nguồn: Hastie và cộng sự (2015).*

Trong đó:

là giá trị thực tế của biến phụ thuộc.

là giá trị của biến độc lập thứ ( j ) tại quan sát thứ ( i ).

là hệ số hồi quy cần ước lượng.

λ là tham số điều chỉnh, kiểm soát mức độ phạt áp dụng lên các hệ số hồi quy.

*2.2 Ý nghĩa và ứng dụng:*

* Lựa chọn biến tự động: LASSO có khả năng loại bỏ các biến không quan trọng bằng cách đặt các hệ số tương ứng của chúng về 0, giúp đơn giản hóa mô hình và tránh overfitting.
* Chính quy hóa: Bằng cách thêm ràng buộc vào tổng giá trị tuyệt đối của các hệ số, LASSO giúp giảm phương sai của các ước lượng, cải thiện độ ổn định và khả năng tổng quát hóa của mô hình.
* Kiểm soát overfitting: Nhờ vào việc chính quy hóa, LASSO giúp ngăn chặn hiện tượng overfitting, đặc biệt là khi số lượng biến độc lập lớn hơn số lượng quan sát.
* Hiệu quả tính toán: LASSO có thể được giải quyết bằng các thuật toán tối ưu hóa hiệu quả như phương pháp coordinate descent(1), giúp tiết kiệm thời gian tính toán.
* Ứng dụng thực tế: LASSO được sử dụng rộng rãi trong nhiều lĩnh vực như kinh tế, tài chính, y học, và khoa học dữ liệu. Ví dụ, trong phân tích dữ liệu ung thư vú, LASSO có thể giúp xác định các yếu tố quan trọng ảnh hưởng đến kết quả điều trị.
  1. *Cách Training Lasso:*

*Bước 1: Chuẩn bị dữ liệu*

* Thu thập dữ liệu: Đảm bảo rằng bạn có một tập dữ liệu đầy đủ với các biến độc lập (features) và biến phụ thuộc (target).
* Tiền xử lý dữ liệu: Xử lý các giá trị thiếu, chuẩn hóa hoặc chuẩn hóa các biến độc lập nếu cần thiết.

*Bước 2: Chia dữ liệu*

* Chia tập dữ liệu: Chia dữ liệu thành tập huấn luyện và tập kiểm tra (thường là 80% huấn luyện và 20% kiểm tra).

*Bước 3: Huấn luyện mô hình*

* Chọn mô hình LASSO: Sử dụng thư viện như *scikit-learn* trong Python để tạo mô hình LASSO.
* Đặt tham số điều chỉnh λ: Tham số λ kiểm soát mức độ phạt áp dụng lên các hệ số hồi quy. Bạn có thể sử dụng phương pháp cross-validation (2) để tìm giá trị λ tối ưu.

*Bước 4: Đánh giá mô hình*

* Đánh giá trên tập kiểm tra: Sử dụng tập kiểm tra để đánh giá hiệu suất của mô hình bằng các chỉ số như MSE (Mean Squared Error) (3) hoặc R² (4).

*Bước 5: Điều chỉnh và cải thiện*

* Điều chỉnh tham số: Nếu cần, điều chỉnh các tham số của mô hình và huấn luyện lại để cải thiện hiệu suất.
* Kiểm tra lại: Đánh giá lại mô hình trên tập kiểm tra sau khi điều chỉnh.

## **3. Neural Network (Mạng nơ-ron nhân tạo )**

- Là một mô hình toán học phức tạp được dùng để điều khiển máy tính dự đoán, nhận dạng và xử lý dữ liệu như một bộ não của con người.

- Biết đến với thuật ngữ quen thuộc là “[deep learning](https://vietnix.vn/deep-learning-la-gi/)”, nghĩa là việc vận dụng các nơ-ron hoặc các nút tạo sự liên kết với nhau trong cùng một cấu trúc phân lớp.

- Được phát triển dựa theo các mạng nơ-ron sinh học.

- Cụ thể hơn, Neural Network được xây dựng dựa theo mô hình hoạt động của các tế bào thần kinh của con người.

- Ở não người, các dây thần kinh kết nối các nút, gọi là tế bào thần kinh, lại với nhau. Còn ở Neural Network, các nút này được gọi là nơ-ron nhân tạo. Việc kết nối các điểm này lại sẽ tạo ra một hệ thống dây chằng chịt, khi đó nó được gọi là **mạng nơ-ron nhân tạo**.

**Đặc điểm của Neural Network**

* Mạng lưới nơ-ron nhân tạo hoạt động như nơ-ron trong não bộ con người. Trong đó, mỗi nơ-ron là một hàm toán học, có chức năng thu thập và phân loại dữ liệu, thông tin theo cấu trúc chi tiết.
* Neural Network tương đồng với những phương pháp thống kê theo đồ thị đường cong hoặc phân tích hồi quy.
* Mỗi nút là một tập hợp tri giác, cấu tạo tương tự hàm hồi quy đa tuyến tính, được sắp xếp liên kết với nhau. Các lớp này sẽ thu thập thông tin, sau đó phân loại và phát tín hiệu đầu ra tương ứng.

**Kiến trúc mạng Neural Network**

Mạng **Neural Network** được cấu thành từ những tầng perceptron, gồm 3 tầng chính:

* Tầng vào (input layer): Như trong hình ở dưới, tầng này nằm bên phía trái, thể hiện cho các dữ liệu đầu vào.
* Tầng ra (output layer): Ngược lại với tầng vào, tầng ra thể hiện cho đầu ra của mạng nơ-ron và nằm bên phía bên phải của hình.
* Tầng ẩn (hidden layer): Đây là tầng nằm ở giữa, thể hiện cho quá trình xử lý thông tin và suy luận của mạng. Nó sẽ nhận các thông tin đầu vào ở đầu vào và trả kết quả ở đầu ra thông qua chức năng kích hoạt.

**Lưu ý:** Mỗi Neural Network chỉ có một tầng input (tầng đầu vào) và output (tầng đầu ra) nhưng sẽ có nhiều hidden layer.

A diagram of a network

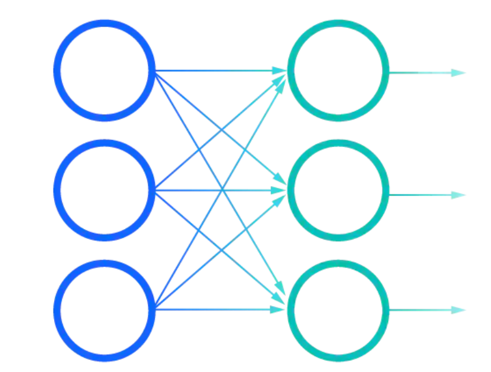
Description automatically generated

*Hình 27: Minh họa kiến trúc mang Neural Network*

Khi tiếp nhận thông tin từ bên ngoài thông qua Lớp đầu vào, các dữ liệu này sẽ đi qua một hoặc nhiều Lớp ẩn để xử lý. Thông thường, đây sẽ là những dữ liệu mà con người muốn tìm hiểu hoặc giải quyết. Tại đây các con số sẽ được xử lý bằng các thuật toán và kết quả sẽ được xuất ra tại Lớp đầu ra. Mỗi khi đi qua thêm một Lớp ẩn thì dữ liệu sẽ được phân tích sâu sắc hơn và được xử lý kỹ hơn.

**Các loại Neural Network:**

**Artificial Neural Network**(ANN) là một mô hình toán học được phát triển thông qua các nơ-ron sinh học. ANN được cấu tạo bởi nhiều điểm nối, nơi các dữ liệu sẽ được xử lý và phân tích.

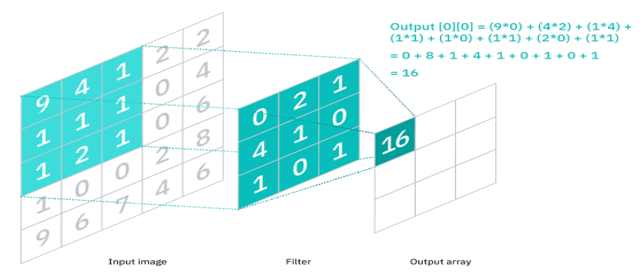


*Hình 28:* *Hình minh họa Convolutional Neural Network (ANN)*

ANN có thể được sử dụng để giải quyết các vấn đề liên quan đến:

* Dữ liệu dạng bảng
* Dữ liệu hình ảnh
* Dữ liệu văn bản

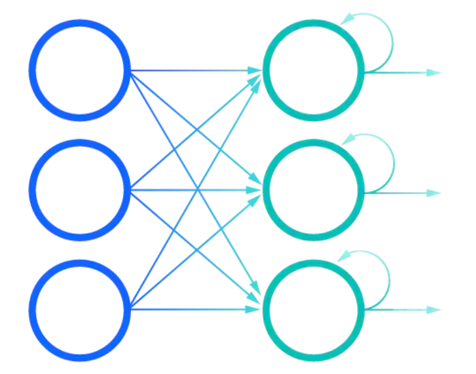
**Convolutional Neural Network** (CNN) hay còn gọi mà mạng nơ-ron tích chập. Đây là một trong những mô hình Deep Learning hiện đại, cho phép con người sử dụng nguồn thông tin với độ chính xác cực kỳ cao.



*Hình 29:* *Hình minh họa Convolutional Neural Network (CNN)*

Thuật toán này thường được sử dụng trong việc nhận dạng các đối tượng trong hình ảnh.

**Recurrent Neural Network** (RNN), hay còn được gọi là mạng nơ-ron tuần hoàn, là mạng lưới dùng để lưu lại thông tin từ quá trình xử lý dữ liệu, từ đó mà máy có thể đưa ra những dự đoán chính xác hơn.



*Hình 30:* *Hình minh họa Convolutional Neural Network (RNN)*

Chúng ta có thể sử dụng RNN tuần hoàn để giải quyết các vấn đề liên quan đến:

* Dữ liệu chuỗi thời gian
* Dữ liệu văn bản
* Dữ liệu âm thanh

***Cách Training Neural Network:***

Bước 1: Chuẩn bị dữ liệu (Chuẩn hóa, chia tập huấn luyện – kiểm tra)

Bước 2: Xây dựng kiến trúc mang (Xác định số lớp, số lượng nơ-ron của từng lớp, chọn hàm kích hoạt…).

Bước 3: Hàm mất mát (Hồi quy: MSE, MAE; Phân loại: Binary-Cross-Entropy)

Bước 4: Thuật toán tối ưu: Adam….

Bước 5: Huấn luyện Neural Network.

Bước 6: Đánh giá mô hình (MSE, MAE, F1-score, accuracy....).

Bước 7: Dự đoán kết quả.

# **CHƯƠNG 3: PHÂN TÍCH VÀ ĐÁNH GIÁ BIỂU ĐỒ PHÂN BỔ DỮ LIỆU VÀ MÔ HÌNH HỌC KẾT HỢP**

## **PHẦN 1: PHÂN TÍCH ĐÁNH GIÁ BIỂU ĐỒ PHÂN BỔ DỮ LIỆU PAIR PLOT**

- Pair plot: là một công cụ mạnh mẽ trong phân tích dữ liệu thăm dò (exploratory data analysis - EDA). Pair Plot giúp quan sát và hiểu rõ các mối quan hệ tiềm ẩn giữa các cặp biến trong bộ dữ liệu, cũng như phân phối của từng biến.

- Pair Plot là tập hợp của nhiều biểu đồ phân tán (scatter plot) và histogram:

Biểu đồ phân tán (scatter plot): Được dùng để thể hiện mối quan hệ giữa các cặp biến số. Mỗi ô trong Pair Plot là một scatter plot giữa hai biến cụ thể.

Biểu đồ phân phối (histogram): Thể hiện phân phối của từng biến trên đường chéo chính của ma trận.

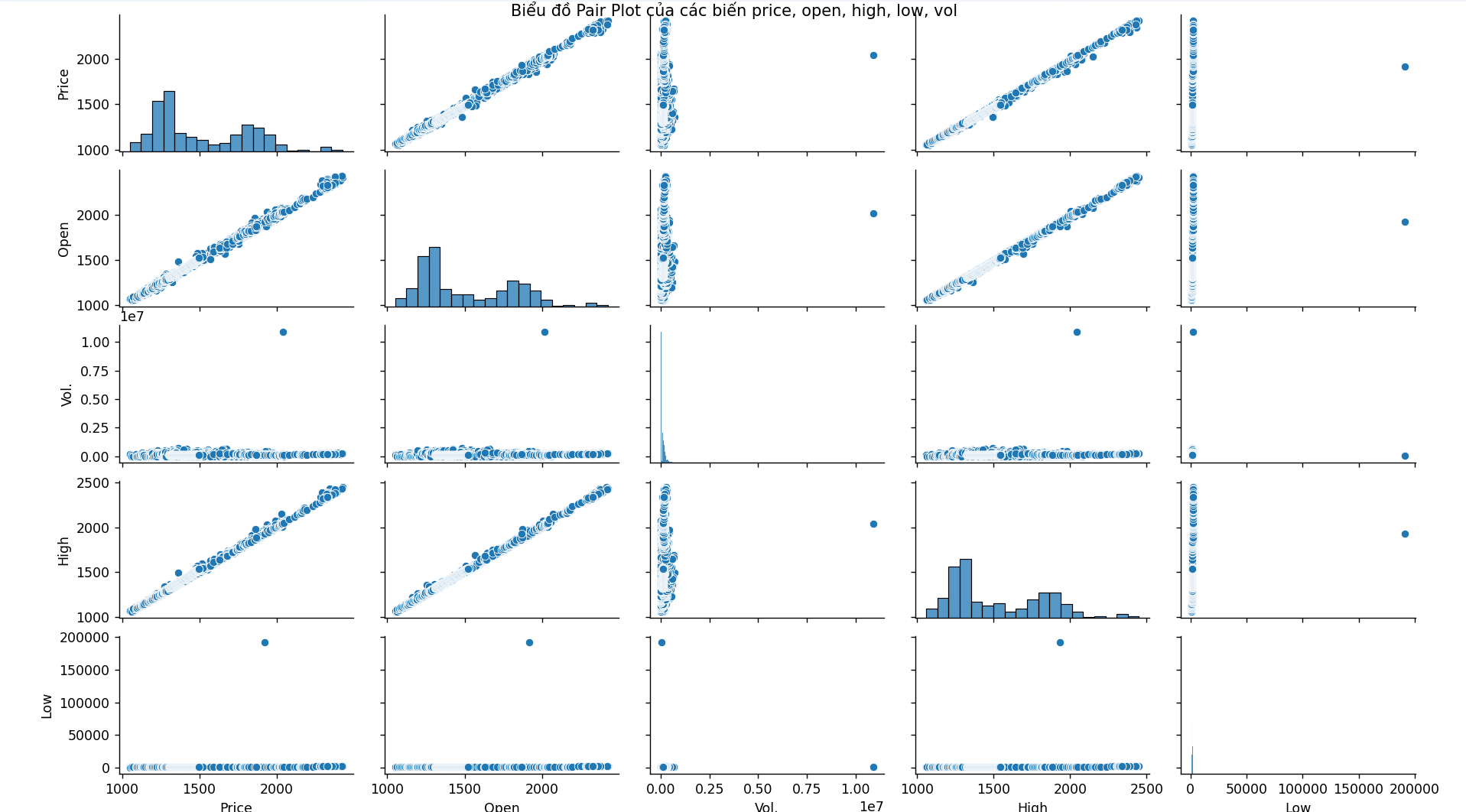
- Phân tích mối quan hệ giữa các cặp biến:

Biểu đồ phân tán giữa các biến: Các ô không nằm trên đường chéo của Pair Plot thể hiện mối quan hệ giữa các cặp biến khác nhau.

* Mối quan hệ tuyến tính: Nếu các điểm có xu hướng nằm trên một đường thẳng hoặc gần như thẳng, điều này cho thấy mối quan hệ tuyến tính giữa hai biến.
* Mối quan hệ phi tuyến tính: Nếu các điểm tạo thành các mẫu phức tạp hơn như hình parabol, hình chữ U, hoặc dạng đường cong khác, thì hai biến có thể có mối quan hệ phi tuyến tính.
* Không có mối quan hệ rõ ràng: Nếu các điểm phân tán ngẫu nhiên, không có mẫu nào rõ ràng, điều đó cho thấy có ít hoặc không có mối quan hệ giữa hai biến.
* Mật độ điểm: Khu vực có mật độ điểm cao hơn thể hiện các giá trị phổ biến hơn trong dữ liệu, giúp nhận diện các vùng tập trung hoặc các cụm (cluster) tự nhiên trong dữ liệu.

- Phân tích phân phối của từng biến:

* Histogram trên đường chéo: Trên đường chéo chính của Pair Plot, biểu đồ thể hiện phân phối của từng biến. Đây là histogram (hoặc đôi khi là KDE - Kernel Density Estimate nếu dùng thư viện seaborn) giúp phân tích sự phân bố của dữ liệu.
* Phân phối chuẩn (normal distribution): Nếu histogram có dạng chuông đối xứng, điều đó cho thấy biến có phân phối chuẩn.
* Phân phối lệch trái hoặc lệch phải: Nếu biểu đồ phân phối lệch về bên trái hoặc phải, điều này chỉ ra rằng biến có phân phối không đối xứng.
* Các giá trị ngoại lệ (outliers): Giá trị ngoại lệ có thể nhận biết qua các cột dữ liệu nằm cách xa phần lớn dữ liệu trên histogram. Những giá trị này có thể ảnh hưởng lớn đến phân tích và mô hình hóa dữ liệu.



*Hình 31: Biểu đồ phân bổ dữ liệu dự đoán giá vàng với 5 biến (Price, Open, Vol, High và Low).*

* **Ô số 1** - Biểu đồ cột của chính biến Price (Giá Vàng):
  + Phân phối của biến Price có hình dạng lệch một bên, thể hiện qua việc số lượng quan sát ở khoảng giữa cao hơn hẳn so với hai đầu của phân phối
  + Đặc biệt có một số cột cao tập trung ở vùng giữa của biểu đồ, cho thấy phần lớn giá đóng cửa của vàng nằm trong khoảng giá này
  + Sự phân bố này ngụ ý rằng giá vàng thường có xu hướng ổn định quanh một mức giá nhất định và ít khi có biến động mạnh
  + Có thể quan sát thấy một số cột thấp ở hai đầu biểu đồ, đại diện cho các giá trị ngoại lệ hoặc những thời điểm giá vàng biến động bất thường
  + Qua quan sát biểu đồ cột thể hiện phân phối của biến Price (Giá Vàng), có thể nhận thấy dữ liệu có dạng phân phối khá đặc biệt với độ lệch nhẹ về một phía. Cụ thể, các giá trị có xu hướng tập trung mạnh mẽ ở khoảng giá trung bình, thể hiện qua các cột cao ở vùng giữa biểu đồ. Điều này cho thấy giá đóng cửa của vàng thường có xu hướng dao động quanh một mức giá nhất định và duy trì tính ổn định tương đối. Tuy nhiên, vẫn tồn tại một số điểm ngoại lệ được thể hiện qua các cột thấp ở hai đầu biểu đồ, phản ánh những thời điểm giá vàng có biến động bất thường. Đặc điểm phân phối này gợi ý rằng giá vàng có tính chu kỳ và khá ổn định, tạo điều kiện thuận lợi cho việc áp dụng các mô hình thống kê truyền thống trong dự đoán, mặc dù cần lưu ý xử lý các giá trị ngoại lệ để đảm bảo độ chính xác của mô hình.

**Ô số 2** - Biểu đồ phân tán thể hiện mối tương quan giữa Price và Open:

* Các điểm dữ liệu tạo thành một đường thẳng chéo gần như hoàn hảo từ góc dưới bên trái lên góc trên bên phải
* Điều này thể hiện mối tương quan tuyến tính cực mạnh giữa giá mở cửa và giá đóng cửa
* Độ tập trung cao của các điểm quanh đường chéo cho thấy giá mở cửa có thể dự đoán rất tốt cho giá đóng cửa
* Hầu như không có điểm nào nằm xa đường chéo, nghĩa là rất hiếm khi có sự chênh lệch lớn giữa giá mở cửa và giá đóng cửa
* Điều này có nghĩa là giá mở cửa (Open) là một yếu tố dự đoán mạnh mẽ cho giá vàng (Price). Khi xây dựng mô hình dự đoán, việc sử dụng giá mở cửa làm biến độc lập sẽ giúp mô hình có độ chính xác cao. Do đó, hồi quy tuyến tính có thể là một lựa chọn hợp lý trong trường hợp này – Đây là biến độc lập duy nhất mà nhóm em chọn để đưa ra kết quả dự đoán giá Vàng.
* **Ô số 3** - Biểu đồ phân tán ngoài đường chéo thể hiện mối tương quan giữa biến Price (Giá Vàng) và biến Vol (Khối lượng giao dịch):
  + Các điểm dữ liệu phân tán rộng và không theo một mô hình hay xu hướng rõ ràng nào
  + Có sự tập trung điểm cao ở vùng khối lượng giao dịch thấp, và phân tán dần khi khối lượng tăng
  + Không thể nhận thấy mối quan hệ tuyến tính giữa giá đóng cửa và khối lượng giao dịch
  + Sự phân bố này gợi ý rằng khối lượng giao dịch không phải là yếu tố tốt để dự đoán giá vàng
* Khối lượng giao dịch (Vol) gần như không phải là một yếu tố quan trọng trong việc dự đoán giá vàng (Price). Mặc dù có thể thử nghiệm thêm trong các mô hình phức tạp như hồi quy phi tuyến tính hoặc cây quyết định để kiểm tra mối quan hệ tiềm ẩn, nhưng, nhóm em quyết định loại bỏ biến này ra khỏi mô hình dự đoán.
* **Ô số 4** - Biểu đồ phân tán thể hiện mối tương quan Price và Open (tương tự ô số 2):
  + Tạo thành đường chéo hoàn hảo từ góc dưới bên trái lên góc trên bên phải
  + Thể hiện mối tương quan tuyến tính rất mạnh giữa hai biến
  + Các điểm dữ liệu tập trung chặt chẽ quanh đường chéo
  + Xác nhận thêm rằng giá mở cửa là yếu tố dự báo tốt cho giá đóng cửa
* **Ô số 5** - Biểu đồ cột của biến Open:
  + Phân phối có dạng tương tự như biểu đồ của Price ở ô số 1
  + Có sự tập trung cao của các quan sát ở khoảng giá giữa
  + Biểu đồ cũng thể hiện độ lệch nhẹ về một phía
  + Sự tương đồng với phân phối của Price phản ánh mối quan hệ chặt chẽ giữa hai biến
* **Ô số 6** - Biểu đồ phân tán giữa Open và Vol:
  + Các điểm dữ liệu phân tán không theo quy luật
  + Không thể hiện mối quan hệ tuyến tính rõ ràng
  + Tập trung nhiều điểm ở vùng khối lượng giao dịch thấp
  + Tương tự như ô số 3, cho thấy Vol không có mối liên hệ mạnh với giá
* **Ô số 7 - Biểu đồ phân tán giữa Open và Open:**
  + Đây là biểu đồ tự lặp giữa cùng một biến nên tạo thành một đường chéo hoàn hảo.
  + Tất cả các điểm dữ liệu nằm trên đường chéo, thể hiện giá trị của biến Open là duy nhất tại mỗi thời điểm.
* **Ô số 8 - Biểu đồ phân tán giữa Open và Vol:**
  + Các điểm dữ liệu phân tán rộng và không tuân theo một xu hướng tuyến tính rõ ràng.
  + Có sự tập trung cao của điểm dữ liệu ở vùng khối lượng giao dịch thấp.
  + Mối quan hệ giữa Open và Vol không rõ ràng, cho thấy Vol không phải là yếu tố quan trọng để dự đoán Open.
* **Ô số 9 - Biểu đồ phân tán giữa Open và High:**
* Các điểm dữ liệu tập trung gần đường chéo, thể hiện mối tương quan tuyến tính rất mạnh giữa Open và High.
* Điều này cho thấy giá mở cửa (Open) và giá cao nhất (High) thường biến động đồng đều trong ngày.
* **Ô số 10 - Biểu đồ phân tán giữa Open và Low:**
* Tương tự biểu đồ giữa Open và High, các điểm dữ liệu tập trung quanh đường chéo.
* Mối tương quan tuyến tính mạnh mẽ, cho thấy giá mở cửa và giá thấp nhất (Low) cũng thường biến động cùng nhau.
* **Ô số 11 - Biểu đồ phân tán giữa Vol và Price:**
* Các điểm dữ liệu phân tán không theo một xu hướng rõ ràng, tập trung nhiều ở vùng khối lượng giao dịch thấp.
* Không có mối quan hệ tuyến tính rõ ràng, cho thấy Vol không phải là yếu tố dự báo tốt cho Price.
* **Ô số 12 - Biểu đồ phân tán giữa Vol và Open:**
* Tương tự như ô 11, không có xu hướng rõ ràng, và các điểm dữ liệu tập trung ở vùng khối lượng thấp.
* Điều này xác nhận rằng Vol không có mối tương quan chặt chẽ với Open.
* **Ô số 13 - Biểu đồ phân tán giữa Vol và High:**
* Các điểm dữ liệu phân tán rộng, không có mối quan hệ tuyến tính.
* Tập trung điểm dữ liệu ở vùng khối lượng giao dịch thấp, cho thấy Vol không phải là yếu tố dự báo tốt cho High.
* **Ô số 14 - Biểu đồ phân tán giữa Vol và Low:**
* Tương tự ô 13, không có mối tương quan rõ ràng giữa Vol và Low.
* Các điểm dữ liệu chủ yếu phân tán ngẫu nhiên, tập trung ở vùng khối lượng thấp.
* **Ô số 15 - Biểu đồ phân tán giữa High và Price:**
* Các điểm dữ liệu tập trung quanh đường chéo, cho thấy mối tương quan tuyến tính rất mạnh giữa High và Price.
* Giá cao nhất trong ngày có thể dự đoán tốt giá đóng cửa (Price).
* **Ô số 16 - Biểu đồ phân tán giữa High và Open:**
* Các điểm dữ liệu tạo thành một đường chéo gần như hoàn hảo.
* Điều này cho thấy giá cao nhất (High) và giá mở cửa (Open) có mối tương quan tuyến tính mạnh mẽ.
* **Ô số 17 - Biểu đồ phân tán giữa High và Vol:**
* Các điểm dữ liệu phân tán không theo quy luật, tập trung ở vùng khối lượng giao dịch thấp.
* Mối quan hệ giữa High và Vol không rõ ràng, cho thấy Vol không phải là yếu tố dự đoán mạnh cho High.
* **Ô số 18 - Biểu đồ phân tán giữa High và Low:**
* Các điểm dữ liệu nằm chặt chẽ trên một đường chéo, thể hiện mối tương quan tuyến tính cực mạnh giữa High và Low.
* Điều này cho thấy giá cao nhất và giá thấp nhất trong ngày biến động đồng đều, phản ánh sự ổn định tương đối.
* **Ô số 19 - Biểu đồ phân tán giữa Low và Price:**
* Các điểm dữ liệu tập trung quanh đường chéo, thể hiện mối tương quan tuyến tính rất mạnh giữa Low và Price.
* Giá thấp nhất trong ngày có thể dự đoán tốt giá đóng cửa (Price).
* **Ô số 20 - Biểu đồ phân tán giữa Low và High:**
* Các điểm dữ liệu tạo thành một đường chéo gần như hoàn hảo.
* Mối tương quan tuyến tính rất mạnh giữa Low và High, cho thấy cả hai biến động đồng đều.
* Phản ánh sự ổn định giá trong ngày và gợi ý rằng chỉ cần một trong hai biến làm đại diện trong mô hình dự đoán.

### **Ô số 21 - Biểu đồ phân tán giữa High và Low:**

* + Các điểm dữ liệu tạo thành một đường chéo gần như hoàn hảo từ góc dưới bên trái đến góc trên bên phải.
  + Thể hiện mối tương quan tuyến tính rất mạnh giữa giá cao nhất (High) và giá thấp nhất (Low).
  + Độ tập trung cao của các điểm quanh đường chéo cho thấy giá thấp nhất và cao nhất thường dao động trong một phạm vi hẹp.
  + Gần như không có điểm nằm xa đường chéo, cho thấy hiếm khi xảy ra sự chênh lệch lớn giữa High và Low.  
    High và Low có mối tương quan rất mạnh, và Low có thể được sử dụng làm biến độc lập để dự đoán High (hoặc ngược lại).

### **Ô số 22 - Biểu đồ phân tán giữa High và Vol:**

* + Các điểm dữ liệu phân tán rộng, không theo một xu hướng tuyến tính rõ ràng.
  + Sự tập trung cao của các điểm ở vùng khối lượng giao dịch thấp (Vol nhỏ), trong khi các điểm ở khối lượng lớn có sự phân tán ngẫu nhiên.
  + Không có mối liên hệ tuyến tính giữa giá cao nhất (High) và khối lượng giao dịch (Vol).
  + Vol không phải là yếu tố dự đoán tốt cho giá cao nhất trong ngày (High).

### **Ô số 23 - Biểu đồ phân tán giữa Low và Vol:**

* + Tương tự ô 22, các điểm dữ liệu không tuân theo bất kỳ xu hướng nào.
  + Tập trung nhiều ở vùng Vol thấp, cho thấy các giao dịch với khối lượng nhỏ chiếm phần lớn dữ liệu.
  + Không có mối tương quan tuyến tính giữa Low và Vol.
  + Vol không có giá trị dự báo đối với Low, và mối quan hệ giữa Low và Vol là không đáng kể.

### **Ô số 24 - Biểu đồ phân tán giữa High và Open:**

* + Các điểm dữ liệu tạo thành một đường chéo rõ ràng từ góc dưới bên trái lên góc trên bên phải.
  + Thể hiện mối tương quan tuyến tính rất mạnh giữa giá cao nhất (High) và giá mở cửa (Open).
  + Các điểm tập trung chặt chẽ quanh đường chéo, cho thấy sự tương đồng lớn giữa hai biến.
  + Open là một yếu tố quan trọng để dự đoán High, và mối tương quan tuyến tính giữa hai biến hỗ trợ việc áp dụng các mô hình hồi quy đơn giản.

### **Ô số 25 - Biểu đồ phân tán giữa Low và Open:**

* + Tương tự ô 24, các điểm dữ liệu tạo thành một đường chéo hoàn hảo, thể hiện mối tương quan tuyến tính rất mạnh giữa Low và Open.
  + Độ tập trung cao của các điểm quanh đường chéo chứng tỏ giá mở cửa (Open) có thể dự đoán tốt giá thấp nhất (Low).
  + Open là một yếu tố dự đoán mạnh mẽ cho Low, và mối quan hệ giữa hai biến hỗ trợ mạnh cho các phân tích dự đoán.
* **Nhận xét về biểu đồ cột thể hiện giá trị chính các biến**: Bốn biến giá (Price, Open, High, Low) có tính ổn định cao, mối quan hệ chặt chẽ và là các yếu tố chính để xây dựng mô hình dự đoán giá vàng. Trong khi đó, biến Vol cho thấy sự biến động lớn và thiếu tương quan với các biến giá, do đó có thể được loại bỏ hoặc chỉ đóng vai trò phụ trợ trong phân tích và dự đoán.
* **Biến Price (Giá đóng cửa):** Phân phối ổn định, tập trung ở mức giá trung bình, với ít biến động cực đoan. Biến này thể hiện mối tương quan tuyến tính mạnh với các biến Open, High và Low, chứng tỏ giá đóng cửa có thể dự đoán tốt dựa vào các biến này.
* **Biến Open (Giá mở cửa):** Phân phối tương tự như Price, phản ánh sự ổn định. Open là một biến độc lập quan trọng, có khả năng dự đoán cao đối với giá đóng cửa (Price). Ngoài ra, Open cũng có mối tương quan chặt chẽ với High và Low, hỗ trợ vai trò chính trong phân tích dữ liệu.
* **Biến High (Giá cao nhất trong ngày):** Phân phối ổn định với giá trị tập trung ở khoảng giữa. High có mối tương quan mạnh với Price, Open và Low, phản ánh xu hướng thị trường và đóng vai trò quan trọng trong việc phân tích giá.
* **Biến Low (Giá thấp nhất trong ngày):** Có đặc điểm phân phối và tương quan tương tự High, với mối liên hệ chặt chẽ với các biến còn lại. Điều này cho thấy Low cũng là một yếu tố phản ánh xu hướng giá trên thị trường.
* **Biến Vol (Khối lượng giao dịch):** Phân phối không đồng đều, với phần lớn giá trị nằm ở mức thấp, và một số ít đạt mức rất cao. Không có mối tương quan rõ ràng với các biến giá (Price, Open, High, Low), cho thấy Vol không phải là yếu tố quan trọng trong việc dự đoán giá vàng.

## **PHẦN 2: XÂY DỰNG MÔ HÌNH HỌC KẾT HỢP**

Có 2 loại mô hình học kết hợp: Bagging và Stacking

Bagging là một kỹ thuật ensemble tạo ra nhiều mô hình con (base models) bằng cách huấn luyện trên các tập dữ liệu con khác nhau (tạo ra từ dữ liệu gốc qua phương pháp lấy mẫu lại - bootstrapping) và kết hợp kết quả của chúng – với bài toán của nhóm em, em sẽ sử dụng bagging chỉ cho neural network:

Ưu điểm:

* **Giảm phương sai (variance):** Bagging giảm độ nhạy của mô hình đối với dữ liệu huấn luyện, giúp giảm phương sai, làm cho mô hình ổn định và ít bị overfitting.
* **Hiệu quả với các mô hình có độ phương sai cao:** Các mô hình dễ bị overfitting như cây quyết định (decision trees) sẽ hoạt động tốt khi kết hợp với Bagging.
* **Dễ triển khai và huấn luyện song song:** Các mô hình trong Bagging được huấn luyện độc lập nên có thể triển khai song song, giúp giảm thời gian huấn luyện.

Nhược điểm:

* **Không hiệu quả trong giảm sai số chệch (bias):** Nếu mô hình cơ bản có sai số chệch cao (như khi dữ liệu có các mẫu không tuyến tính phức tạp), Bagging không thể cải thiện nhiều do chỉ giảm phương sai mà không giảm sai số chệch.
* **Không phù hợp với các mô hình tuyến tính:** Các mô hình đơn giản với độ phương sai thấp (ví dụ: hồi quy tuyến tính) sẽ không được cải thiện đáng kể khi sử dụng Bagging.
* **Tăng chi phí tính toán và bộ nhớ:** Bagging yêu cầu huấn luyện và lưu trữ nhiều mô hình, điều này có thể làm tăng chi phí tính toán và yêu cầu bộ nhớ.

Các bước triển khai: Bagging có thể dễ dàng triển khai với base\_model sử dụng hàm MLPRegressor của Neural Network, sau đó triển khai bagging\_model dựa trên base\_model đã tạo ở trên với hàm BaggingRegressor:

* Bước 1: Chuẩn bị dữ liệu: Tạo base\_model dựa trên Neural Neural Network mà nhóm em đã triển khai, sau đó sử dụng BaggingRegressor để tạo bagging\_model dựa trên base\_model đã tạo
* Bước 2: Chia dữ liệu thành tập huấn luyện và tập kiểm tra: Chia dữ liệu thành tập huấn luyện và tập kiểm tra. Tập huấn luyện sẽ dùng để xây dựng mô hình, còn tập kiểm tra dùng để đánh giá độ chính xác của mô hình. Với tỷ lệ của nhóm em là 80% huấn luyện và 20% kiểm tra
* Bước 3: Huấn luyện mô hình Bagging
* Bước 4: Đánh giá mô hình: Sử dụng mô hình đã được huấn luyện để dự đoán trên tập kiểm tra, sau đó tính toán độ chính xác dựa trên số lượng dự đoán đúng và sai so với nhãn thực tế trong tập kiểm tra

Stacking là phương pháp học kết hợp mà ở đó, nhiều mô hình con (base models) được huấn luyện trên toàn bộ dữ liệu huấn luyện và kết quả của chúng được dùng làm đầu vào cho một mô hình thứ cấp (meta-model) để dự đoán đầu ra cuối cùng:

Ưu điểm:

* **Kết hợp các mô hình đa dạng:** Stacking cho phép kết hợp nhiều loại mô hình khác nhau (như hồi quy tuyến tính, SVM, cây quyết định) trong cùng một hệ thống, tận dụng điểm mạnh của từng loại mô hình.
* **Giảm cả sai số chệch và phương sai:** Bằng cách học được từ đầu ra của nhiều mô hình con, Stacking có thể giảm cả sai số chệch và phương sai, giúp tăng độ chính xác của mô hình tổng thể.
* **Tăng cường khả năng tổng quát hóa:** Meta-model học cách kết hợp thông tin từ các mô hình con khác nhau, giúp mô hình tổng thể hoạt động tốt hơn trên dữ liệu chưa thấy.

Nhược điểm:

* **Phức tạp trong triển khai và tối ưu hóa:** Stacking yêu cầu thiết lập một kiến trúc nhiều lớp (các mô hình con và meta-model), làm cho quá trình tối ưu hóa phức tạp và khó điều chỉnh.
* **Dễ bị overfitting ở meta-model:** Nếu meta-model quá phức tạp hoặc số lượng mô hình con quá lớn, Stacking có thể dẫn đến overfitting, đặc biệt nếu không có kỹ thuật điều chỉnh như cross-validation.
* **Tăng thời gian huấn luyện và yêu cầu tài nguyên:** Stacking yêu cầu huấn luyện nhiều mô hình con và sau đó huấn luyện thêm một meta-model, làm tăng chi phí tính toán và thời gian huấn luyện.

Các bước triển khai: Stacking yêu cầu các mô hình con (base models) và một mô hình meta (meta-model) của thư viện scikit-learn cung cấp StackingClassifier để dễ dàng triển khai:

* Bước 1: Chuẩn bị dữ liệu: Sử dụng tập dữ liệu của nhóm em đã lấy được từ Kaggle
* Bước 2: Chia dữ liệu thành tập huấn luyện và tập kiểm tra: Chia dữ liệu thành tập huấn luyện và tập kiểm tra. Tập huấn luyện sẽ dùng để xây dựng mô hình, còn tập kiểm tra dùng để đánh giá độ chính xác của mô hình. Với tỷ lệ của nhóm em là 80% huấn luyện và 20% kiểm tra
* Bước 3: Khởi tạo các mô hình con và mô hình meta: Trong Stacking, các mô hình con và mô hình meta có thể là bất kỳ mô hình nào (như hồi quy tuyến tính, cây quyết định, SVM).
* Bước 4: Huấn luyện mô hình Stacking.
* Bước 5: Đánh giá mô hình: Sử dụng mô hình Stacking đã huấn luyện để dự đoán trên tập kiểm tra và đo lường độ chính xác của mô hình dựa trên số lượng dự đoán đúng và sai trên tập kiểm tra.
* Lý do nhóm em chọn mô hình học kết hợp Bagging:

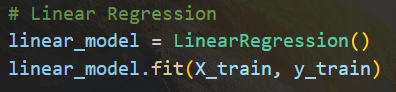
Thứ nhất, vì Bagging làm cho mô hình ổn định và ít bị overfitting (hiện tượng quá khớp)

Thứ hai, mô hình học kết hợp Bagging dễ triển khai và giảm thời gian huấn luyện

# **CHƯƠNG 4: PHÂN TÍCH BỘ THAM SỐ**

## **Tham số trong Hồi quy tuyến tính**

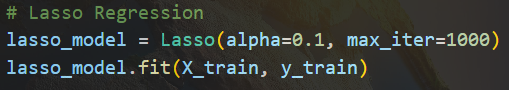
*Sử dụng hàm LinearRegression trong thư viện scikit-learn*



*Hình 26: Bộ tham số của thuật toán hồi quy tuyến tính trong dự đoán giá vàng*

## **Tham số trong Hồi quy Lasso**

*Sử dụng hàm Lasso trong thư viện scikit-learn*

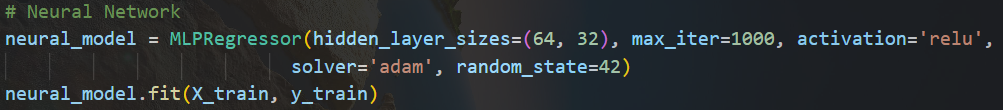
****

*Hình 29: Bộ tham số của thuật toán hồi quy lasso(sklearn) trong dự đoán giá vàng*

**max\_iter=1000**: Giới hạn số lần lặp trong thuật toán tối ưu hóa (gradient descent). Với tập dữ liệu lớn, việc giới hạn số lần lặp giúp tiết kiệm thời gian nhưng vẫn đạt độ hội tụ đủ tốt.

**α:** tương tự với cách tính toán tham số trong lasso.

## **Tham số trong Neural Network (MLPRegressor)**



*Hình 30: Bộ tham số của thuật toán neural network trong dự đoán giá vàng*

Trong mô hình mạng Neural Network, nhóm sử dụng phương pháp MLPRegressor, bao gồm các tham số sau:

**hidden\_layer\_sizes=(64, 32)** xác định kiến trúc mạng nơ-ron với hai lớp ẩn:

* Lớp ẩn thứ nhất có **64 nơ-ron**.
* Lớp ẩn thứ hai có **32 nơ-ron**.
* **Lý do lựa chọn**
* **Kiến trúc nhẹ và nhanh**: Với số lượng nơ-ron ít hơn so với các mạng phức tạp (như cấu trúc 500, 300), mô hình này có thể huấn luyện nhanh hơn, tiết kiệm tài nguyên bộ nhớ và xử lý.
* **Phù hợp với dữ liệu đơn giản**: Nếu dữ liệu không quá phức tạp hoặc mối quan hệ trong dữ liệu không yêu cầu biểu diễn quá sâu, kiến trúc này vẫn đủ khả năng nắm bắt các đặc trưng chính.
* **Giảm thiểu rủi ro quá khớp (overfitting)**: Với số lượng nơ-ron hạn chế, mô hình đơn giản hơn và ít có nguy cơ học quá mức từ dữ liệu huấn luyện (overfitting), đặc biệt khi kích thước dữ liệu không lớn.
* **Hiệu quả dự đoán**
* **Khả năng tổng quát hóa tốt hơn**: Mạng nhỏ hơn có thể tổng quát hóa tốt cho dữ liệu kiểm thử, đặc biệt khi dữ liệu không có độ phức tạp cao.
* **Độ chính xác dự đoán ổn định**: Dù có ít nơ-ron, mô hình vẫn có thể đưa ra dự đoán sát thực tế nếu các đặc trưng trong dữ liệu được biểu diễn tốt qua các lớp nhỏ hơn.
* **Thử nghiệm ban đầu và mô hình hóa nhanh chóng**: Kiến trúc này phù hợp để thử nghiệm nhanh ban đầu nhằm đánh giá độ phù hợp của dữ liệu và yêu cầu mô hình phức tạp hơn khi cần thiết.

|  |  |  |  |
| --- | --- | --- | --- |
| **Giá mở cửa** | **Giá vàng thực tế** | **Giá vàng dự đoán (64,32)** | **Giá vàng dự đoán (500,300)** |
| 1801 | 1804.2 | 1800.3102 | 1800.3102 |
| 1792.3 | 1825.5 | 1791.6165 | 1791.6145 |
| 1817 | 1809.6 | 1816.2980 | 1816.2987 |
| 1674.3 | 1676.1 | 1673.7019 | 1673.7029 |
| 1709.6 | 1717.7 | 1708.9763 | 1708.9783 |

*Bảng 1: So sánh kết quả dự đoán của(64,32) và (500,300)*

**So sánh dự đoán**:

* Kết quả của hai mô hình rất giống nhau, với chênh lệch không đáng kể, cho thấy kiến trúc nhẹ **64, 32** đủ để mô hình hóa dữ liệu hiện tại.
* Kiến trúc **500, 300** có thể hữu ích nếu yêu cầu phân tích dữ liệu có tính phức tạp cao hoặc khi mô hình cần có khả năng dự báo tốt hơn với dữ liệu mới

**Bảng 1** minh họa mô hình nhỏ gọn hơn như (64, 32) có thể đủ mạnh với dữ liệu không phức tạp, giúp đạt hiệu quả tính toán nhanh mà vẫn đảm bảo tính chính xác.

**activation='relu'**: ReLU giúp mô hình học nhanh hơn bằng cách chỉ giữ lại các giá trị dương. Tuy nhiên, nếu có quá nhiều giá trị âm trong dữ liệu, ReLU có thể gây ra vấn đề "vanishing gradient (Độ dốc biến mất)" ở các tầng sâu. ReLU là một lựa chọn phù hợp vì nó giúp mô hình hội tụ nhanh hơn trong quá trình huấn luyện so với các hàm kích hoạt khác như sigmoid hoặc tanh. Khi kích hoạt ReLU, các giá trị âm được đặt thành 0, giúp giảm bớt vấn đề tiêu tốn tài nguyên tính toán và tạo độ tuyến tính theo từng đoạn trong mô hình, nhờ đó mô hình học được những đặc trưng phi tuyến phức tạp.

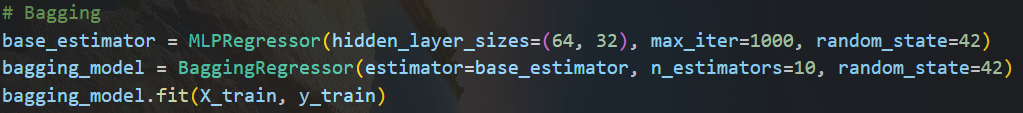
**solver='adam'**: Adam là bộ tối ưu hóa mạnh mẽ và hiệu quả, tự động điều chỉnh tốc độ học (learning rate) trong quá trình huấn luyện. Adam điều chỉnh tốc độ học thích ứng cho từng trọng số riêng biệt, nhờ vậy rất phù hợp cho các bài toán hồi quy phi tuyến tính.

**max\_iter=1000**: Số vòng lặp có thể ảnh hưởng đến khả năng hội tụ của mô hình. Nếu số vòng lặp quá ít, mô hình có thể chưa kịp hội tụ; ngược lại, quá nhiều vòng lặp có thể gây mất thời gian tính toán. Số lần 1000 cũng không quá lớn, giúp cân bằng giữa độ chính xác và thời gian tính toán.

**random\_state=0**: Khi sử dụng một giá trị cố định như 0, mỗi lần chạy mô hình sẽ cho cùng một kết quả. Điều này giúp đảm bảo tính nhất quán khi đánh giá hiệu quả của mô hình.

## **Hàm bagging**

**Mô hình cơ sở:**



*Hình 31: Bộ tham số của thuật toán bagging trong dự đoán giá vàng.*

**hidden\_layer\_sizes=(64, 32)** xác định kiến trúc hàm bagging với hai lớp ẩn:

* Lớp ẩn thứ nhất có **64 nơ-ron**.
* Lớp ẩn thứ hai có **32 nơ-ron**.

**Lý do lựa chọn**

* **Kiến trúc nhẹ và nhanh**: Với số lượng nơ-ron ít hơn so với các mạng phức tạp (như cấu trúc 500, 300), mô hình này có thể huấn luyện nhanh hơn, tiết kiệm tài nguyên bộ nhớ và xử lý.
* **Phù hợp với dữ liệu đơn giản**: Nếu dữ liệu không quá phức tạp hoặc mối quan hệ trong dữ liệu không yêu cầu biểu diễn quá sâu, kiến trúc này vẫn đủ khả năng nắm bắt các đặc trưng chính.
* **Giảm thiểu rủi ro quá khớp (overfitting)**: Với số lượng nơ-ron hạn chế, mô hình đơn giản hơn và ít có nguy cơ học quá mức từ dữ liệu huấn luyện (overfitting), đặc biệt khi kích thước dữ liệu không lớn.

**Hiệu quả dự đoán**

* **Khả năng tổng quát hóa tốt hơn**: Mạng nhỏ hơn có thể tổng quát hóa tốt cho dữ liệu kiểm thử, đặc biệt khi dữ liệu không có độ phức tạp cao.
* **Độ chính xác dự đoán ổn định**: Dù có ít nơ-ron, mô hình vẫn có thể đưa ra dự đoán sát thực tế nếu các đặc trưng trong dữ liệu được biểu diễn tốt qua các lớp nhỏ hơn.
* **Thử nghiệm ban đầu và mô hình hóa nhanh chóng**: Kiến trúc này phù hợp để thử nghiệm nhanh ban đầu nhằm đánh giá độ phù hợp của dữ liệu và yêu cầu mô hình phức tạp hơn khi cần thiết.

|  |  |  |  |
| --- | --- | --- | --- |
| **Giá mở cửa** | **Giá vàng thực tế** | **Giá vàng dự đoán (64,32)** | **Giá vàng dự đoán (500,300)** |
| 1752 | 1739.6 | 1751.5383 | 1751.2943 |
| 1792.3 | 1825.5 | 1791.8186 | 1791.5757 |
| 1817 | 1809.6 | 1816.5066 | 1816.2644 |
| 1674.3 | 1676.1 | 1673.8762 | 1673.6300 |
| 1709.6 | 1717.7 | 1709.1590 | 1708.9137 |

*Bảng 2: So sánh kết quả dự đoán của (64,32) và (500,300)*

**So sánh dự đoán**:

* Kết quả của hai mô hình rất giống nhau, với chênh lệch không đáng kể, cho thấy kiến trúc nhẹ **64, 32** đủ để mô hình hóa dữ liệu hiện tại.
* Kiến trúc **500, 300** có thể hữu ích nếu yêu cầu phân tích dữ liệu có tính phức tạp cao hoặc khi mô hình cần có khả năng dự báo tốt hơn với dữ liệu mới

**Bảng 2** minh họa mô hình nhỏ gọn hơn như (64, 32) có thể đủ mạnh với dữ liệu không phức tạp, giúp đạt hiệu quả tính toán nhanh mà vẫn đảm bảo tính chính xác.

**max\_iter=1000**: Số lần lặp tối đa là 1000, để đảm bảo rằng mỗi mạng con trong bagging có đủ thời gian để hội tụ.

Giá trị này đảm bảo các mô hình con có thời gian học đầy đủ mà không ảnh hưởng lớn đến thời gian huấn luyện chung của mô hình bagging.

**random\_state=42**: random\_state=42 đảm bảo tính ngẫu nhiên có thể lặp lại khi khởi tạo trọng số trong mạng nơ-ron.

Điều này tạo ra sự ổn định và nhất quán trong quá trình huấn luyện để dễ dàng tái lập kết quả.

****

*Hình 32: Bộ tham số của thuật toán bagging trong dự đoán giá vàng.*

**estimator=base\_model**: là mô hình cơ sở dùng trong bagging, ở đây là MLPRegressor.

Các phiên bản khác nhau của base\_model sẽ được huấn luyện trên các mẫu dữ liệu ngẫu nhiên khác nhau để tạo nên một tập hợp các dự đoán đa dạng, giúp tăng độ chính xác của mô hình.

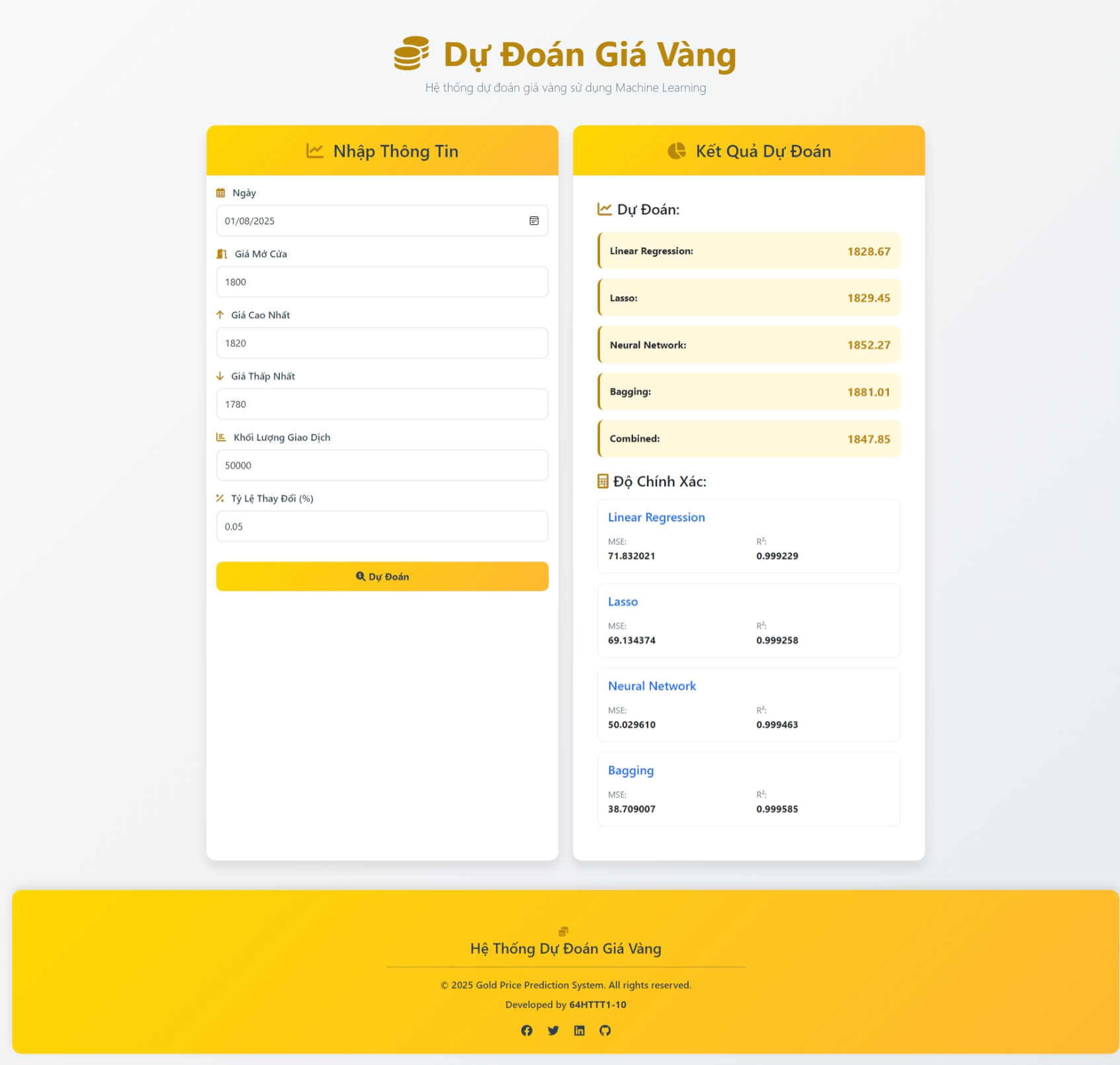
**n\_estimators=10**: tạo ra 10 mô hình con (bộ học) từ mô hình mạng nơ-ron cơ bản. Số lượng mô hình con không quá lớn để tránh tăng chi phí tính toán nhưng cũng đủ để cải thiện độ ổn định và chính xác của dự đoán qua phương pháp bagging.

Việc này giúp giảm nhiễu và giảm sai số do quá khớp của từng mô hình con, nên mô hình tổng hợp sẽ ổn định và chính xác hơn.

**random\_state=42**:Tương tự như với random\_state của MLPRegressor, giá trị này giúp quá trình lấy mẫu ngẫu nhiên trong bagging có thể lặp lại, đảm bảo tính ổn định và dễ dàng so sánh khi thử nghiệm với các bộ tham số khác.

# **CHƯƠNG 5: PHÂN TÍCH KẾT QUẢ KHI CHẠY ĐƯỢC MÔ HÌNH**

## **Kết quả sau khi đã nhập giá mở cửa:**

**

*Hình 33: Kết quả dự đoán sau khi người dùng đã nhập*

*Công thức Hồi quy tuyến tính:*

*Công thức Hồi quy Lasso:*

*Công thức Neural Network:*

*Công thức Bagging:*

### **1.1 Hồi Quy Tuyến Tính và Hồi Quy Lasso:**

* **Giá dự đoán từ Hồi Quy Tuyến Tính (sklearn)**: Kết quả từ mô hình hồi quy tuyến tính sử dụng thư viện sklearn cho ra giá dự đoán là **1828.67**.
* **Giá dự đoán từ Hồi Quy Lasso (sklearn)**: Kết quả từ mô hình Lasso sử dụng thư viện sklearn cho giá dự đoán là **1829.45**, cũng rất gần với kết quả của mô hình trên.
* **Phân tích**: Sự đồng nhất giữa các giá dự đoán này cho thấy rằng cả hai mô hình hồi quy tuyến tính và Lasso có khả năng nắm bắt được mối quan hệ giữa các biến độc lập và giá vàng một cách chính xác và ổn định. Điều này có thể xuất phát từ việc cả hai mô hình đều dựa trên giả định tuyến tính, giúp chúng hoạt động hiệu quả trong trường hợp dữ liệu không quá phức tạp.

### **1.2 Mô Hình Neural Network:**

* **Giá dự đoán**: Mô hình Neural Network cho ra giá dự đoán là **1852.27**, cao hơn so với hồi quy tuyến tính và hồi quy Lasso.
* **Phân tích**: Việc giá dự đoán của mô hình Neural Network cao hơn cho thấy nó có thể đã nắm bắt được các mối quan hệ phi tuyến tính giữa các biến trong dữ liệu, nhưng cũng có thể dẫn đến việc dự đoán không chính xác trong một số trường hợp.

### **1.3 Mô Hình Bagging:**

* **Giá dự đoán**: Mô hình Bagging cung cấp giá dự đoán là **1881.01**, cao hơn so với hồi quy tuyến tính, hồi quy Lasso và Mạng Nơ-ron nhân tạo.
* **Phân tích**: Bagging, bằng cách kết hợp nhiều mô hình yếu, có thể cải thiện độ chính xác của dự đoán. Giá dự đoán vẫn vượt qua được mô hình Neural Network trong trường hợp này, cho thấy rằng Bagging có thể tận dụng tối đa khả năng nắm bắt các mối quan hệ phức tạp như Neural Network.

**Tóm Tắt**

* Hồi quy tuyến tính và Lasso cho giá dự đoán rất gần nhau và nhất quán, cho thấy tính ổn định trong việc dự đoán.
* Neural Network có khả năng nắm bắt các mối quan hệ phức tạp hơn, nhưng cũng đưa ra giá dự đoán cao hơn, điều này có thể dẫn đến một số rủi ro trong việc dự đoán giá vàng.
* Mô hình Bagging, mặc dù cung cấp giá dự đoán tốt, nhưng vẫn không đạt được mức độ chính xác như Neural Network.

## **2. Đánh giá MSE và R²**

### **2.1 Mean Squared Error (MSE)**

**Hồi Quy Tuyến Tính và Hồi Quy Lasso:**

* Giá trị MSE: Mô hình hồi quy tuyến tính có giá trị MSE là **71.832021**
* Giá trị MSE: Mô hình hồi quy Lasso có giá trị MSE là **69.134374**
* Phân tích: Giá trị MSE thấp cho thấy rằng sai số bình quân của các dự đoán là lớn, nghĩa là mô hình này có độ chính xác thấp trong việc dự đoán giá vàng. Tuy nhiên, việc có giá trị MSE cao hơn có thể chỉ ra rằng các mô hình này có một số khó khăn trong việc nắm bắt chính xác các mối quan hệ trong dữ liệu, dẫn đến sai số lớn hơn.

**Neural Network và Bagging:**

* Giá trị MSE: Mô hình Neural Network có giá trị MSE là **50.029610**, trong khi mô hình Bagging có giá trị MSE là **38.709007**
* Phân tích: MSE của các mô hình này thấp hơn so với hồi quy tuyến tính và Lasso, chúng vẫn cho thấy hiệu suất tốt hơn trong việc dự đoán.

### **2.2 R² (Coefficient of Determination)**

**Hồi Quy Tuyến Tính và Hồi Quy Lasso**:

* **Giá trị R²**: Mô hình hồi quy tuyến tính có R² là **0.999229**
* **Giá trị R²**: Cả hai mô hình đều có R² là **0.999258**
* **Phân tích**: Giá trị R² cao này cho thấy rằng khoảng **99.92%** biến thể của giá vàng được giải thích bởi các biến độc lập trong mô hình. Điều này chỉ ra rằng mô hình có khả năng giải thích rất tốt sự biến động của giá vàng, và hầu hết thông tin trong dữ liệu đã được mô hình nắm bắt. Đây là một dấu hiệu cho thấy cả hai mô hình hoạt động rất hiệu quả.

**Neural Network và Bagging**:

* **Giá trị R²**: Mô hình Neural Network có giá trị R² là **0.999463**
* **Giá trị R²**: Mô hình Bagging có giá trị R² là **0.999585**
* **Phân tích**: Giá trị R² chỉ cao hơn một chút so với hồi quy tuyến tính và Lasso, nhưng cho thấy rằng cả hai mô hình cũng giải thích được một phần lớn sự biến động của giá vàng. Giá trị R² này cho thấy mô hình vẫn hoạt động tốt và đạt được hiệu quả cao hơn hồi quy tuyến tính và hồi quy Lasso.

# 

# **KẾT LUẬN**

## ***1. Kết quả đạt được***

Trong quá trình nghiên cứu và triển khai đề tài, nhóm chúng em đã hoàn thành báo cáo kỹ thuật về việc triển khai và xây dựng mô hình dự đoán giá vàng sử dụng ba thuật toán học máy: hồi quy tuyến tính, hồi quy Lasso, và mạng Nơ-ron nhân tạo. Dưới đây là những kết quả cụ thể nhóm đạt được:

* **Hiểu về quy trình triển khai mô hình dự đoán và deploy ứng dụng**: Qua việc thực hiện đề tài, nhóm chúng em đã nắm được các bước quan trọng trong quy trình xây dựng một mô hình dự đoán từ khâu thu thập và xử lý dữ liệu, xây dựng và tinh chỉnh mô hình, đến triển khai ứng dụng trên môi trường đám mây như Render. Điều này giúp nhóm hiểu rõ và thực hành quy trình một cách toàn diện từ ý tưởng đến sản phẩm cuối cùng.
* **Nền tảng cho các môn học và dự án tương lai**: Thông qua quá trình nghiên cứu, nhóm chúng em đã tích lũy được kiến thức chuyên sâu về các thuật toán hồi quy và mạng nơ-ron, cùng với kỹ năng triển khai ứng dụng trên nền tảng đám mây. Những kiến thức và kỹ năng này sẽ là cơ sở cho các môn học tiếp theo và cho đồ án tốt nghiệp sau này.
* **Củng cố kiến thức học máy và tiếp thu kiến thức mới**: Bài nghiên cứu không chỉ giúp nhóm vận dụng kiến thức đã học mà còn đòi hỏi việc tìm hiểu thêm các khái niệm nâng cao về học máy và kỹ thuật tối ưu hóa mô hình. Việc tìm kiếm và đọc tài liệu bổ sung đã giúp nhóm nắm vững hơn về các phương pháp học máy, các cách triển khai mô hình và tối ưu hóa chúng trong thực tế.
* **Phát triển kỹ năng tư duy và làm việc nhóm**: Quá trình nghiên cứu giúp nhóm cải thiện kỹ năng tư duy độc lập và sáng tạo trong việc lựa chọn giải pháp cho từng bước xử lý và triển khai mô hình. Làm việc nhóm còn giúp chúng em phát triển kỹ năng giao tiếp, phối hợp và phân công công việc phù hợp với thế mạnh của từng thành viên.
* **Nâng cao kỹ năng mềm và thuyết trình**: Việc thực hiện và bảo vệ đề tài không chỉ giúp chúng em phát triển kỹ năng thuyết trình, mà còn rèn luyện sự tự tin khi trình bày kết quả nghiên cứu trước hội đồng. Đây là một trải nghiệm quý báu, giúp chúng em tự tin hơn trong việc diễn đạt và bảo vệ lập trường khoa học của mình.
* **Kết quả mô hình dự đoán giá vàng**: Cuối cùng, nhóm đã triển khai thành công mô hình dự đoán giá vàng với ba thuật toán hồi quy tuyến tính, hồi quy Lasso và mạng Nơ-ron nhân tạo. Qua đó, chúng em đánh giá được hiệu suất từng mô hình, từ đó có thể đề xuất giải pháp phù hợp cho các bài toán dự đoán giá vàng trong thực tế.

## **2. Hướng phát triển**

Sau quá trình nghiên cứu và thực hiện đề tài về dự đoán giá vàng, nhóm nhận thấy cần có những cải tiến để phát triển và hoàn thiện hơn nữa. Cụ thể, các hướng phát triển mà nhóm đề xuất gồm:

**Mở rộng phạm vi ứng dụng và nâng cao khả năng triển khai trên nhiều nền tảng**  
Nhóm có thể mở rộng phạm vi dự án để hỗ trợ các nền tảng và thiết bị khác nhau như máy tính để bàn, thiết bị di động, hoặc trang web trực tuyến. Điều này giúp mở rộng tính ứng dụng và sự thuận tiện cho người dùng.

**Nghiên cứu và cập nhật các thuật toán mới.**

Nhóm dự định nghiên cứu các mô hình và thuật toán dự đoán khác ngoài hồi quy tuyến tính, hồi quy Lasso, và neural network, như các mô hình học sâu hoặc các phương pháp ensemble (như Random Forest, Gradient Boosting) để tăng độ chính xác dự đoán.

**Tối ưu hóa và cải thiện hiệu suất hệ thống**

Để tăng hiệu suất và tốc độ, nhóm sẽ tập trung vào tối ưu hóa mã nguồn và dữ liệu đầu vào, đặc biệt là các phương pháp xử lý và chuẩn hóa dữ liệu. Việc này giúp mô hình chạy nhanh hơn và đạt hiệu suất tốt hơn khi triển khai trên các nền tảng đám mây như Render.

**Tích hợp công nghệ hiện đại và tăng cường bảo mật**

Nhóm sẽ xem xét tích hợp các tính năng bảo mật cao hơn, như xác thực hai yếu tố hoặc đăng nhập bằng vân tay/khuôn mặt. Việc này giúp bảo vệ dữ liệu người dùng và nâng cao trải nghiệm người dùng khi sử dụng ứng dụng.

**Phát triển hệ thống phân tích và dự đoán trong thời gian thực**  
Nhóm cũng có thể nghiên cứu phát triển khả năng dự đoán và phân tích giá vàng trong thời gian thực, kết hợp với các dữ liệu từ nhiều nguồn khác nhau, như dữ liệu kinh tế và chính trị toàn cầu, để cải thiện độ chính xác và cập nhật của dự đoán.

# 

# **LỜI CẢM ƠN**

Lời đầu tiên, chúng em xin gửi lời cảm ơn sâu sắc đến thầy *PGS.TS Nguyễn Quang Hoan*, người đã không chỉ truyền đạt kiến thức quý báu mà còn luôn tận tâm hướng dẫn, giúp đỡ chúng em trong suốt quá trình tìm hiểu và học tập môn học máy. Nhờ vào sự tận tình của thầy, chúng em đã có cơ hội tiếp thu những kiến thức mới, hiểu rõ hơn về các thuật toán cũng như các phương pháp phân tích và ứng dụng chúng vào thực tiễn, giúp mở ra một cái nhìn mới mẻ về lĩnh vực học máy, đầy hứa hẹn và thú vị.

Sự hỗ trợ và giảng dạy của thầy đã giúp chúng em tích lũy được rất nhiều kinh nghiệm và kỹ năng hữu ích trong quá trình học tập và nghiên cứu. Chúng em trân trọng những bài học sâu sắc, không chỉ về mặt chuyên môn mà còn cả về tinh thần học hỏi không ngừng và sự tận tâm trong nghề giáo mà thầy đã truyền đạt. Những kiến thức này chắc chắn sẽ là nền tảng quan trọng giúp chúng em phát triển trên con đường học vấn và nghề nghiệp tương lai.

Mặc dù đã cố gắng hết mình, nhưng với sự mới mẻ và độ phức tạp của môn học, chúng em vẫn không thể tránh khỏi những thiếu sót trong quá trình thực hiện bài tập lớn. Chúng em hy vọng thầy có thể dành thời gian xem xét và góp ý để bài tập của nhóm ngày càng được hoàn thiện hơn, không ngừng nâng cao chất lượng và sự hiểu biết về học máy.

Cuối cùng, chúng em xin kính chúc các thầy luôn dồi dào sức khỏe, gặp nhiều niềm vui và thành công trong sự nghiệp “trồng người” đầy ý nghĩa. Chúng em chân thành cảm ơn thầy đã truyền động lực, niềm đam mê và giúp chúng em có cái nhìn sâu sắc hơn về những tiềm năng trong học máy.

Chúng em xin chân thành cảm ơn!

# **TÀI LIỆU THAM KHẢO**

*1.Học máy:*

Slide bài giảng môn Học Máy, trường Đại học Thủy Lợi

(Tài liệu cung cấp kiến thức nền tảng về học máy, các thuật toán và ứng dụng trong dự đoán giá.)

*2.Cập nhật giá vàng:*

Duy, K. (2024, 09 28). *Cập nhật giá vàng chốt phiên 28.9: Vàng nhẫn trụ vững trên đỉnh cao.* Retrieved 01 04, 2025, from [https://laodong.vn/tien-te-dau-](https://laodong.vn/tien-te-dau-tu/cap-nhat-gia-vang-chot-phien-289-vang-nhan-tru-vung-tren-dinh-cao-1400414.ldo) [tu/cap-nhat-gia-vang-chot-phien-289-vang-nhan-tru-vung-tren-dinh-cao-](https://laodong.vn/tien-te-dau-tu/cap-nhat-gia-vang-chot-phien-289-vang-nhan-tru-vung-tren-dinh-cao-1400414.ldo) [1400414.ldo](https://laodong.vn/tien-te-dau-tu/cap-nhat-gia-vang-chot-phien-289-vang-nhan-tru-vung-tren-dinh-cao-1400414.ldo)

(Bài viết cập nhật giá vàng và phân tích tình hình thị trường.)

*3.Giới thiệu về FastAPI:*

GMO-Z. (2023, 12 04). *FastAPI là gì, vì sao lại là nó?* Retrieved 01 04, 2025, from <https://blog.vietnamlab.vn/fastapi-la-gi-va-vi-sao-lai-chon-no/>

(Giới thiệu về FastAPI và lý do chọn nó cho phát triển ứng dụng web.)

*4.Hướng dẫn setup FastAPI:*

John. (2023, 07 05). *FastAPI (Phần 1): Giới thiệu và setup môi trường.*

Retrieved 01 04, 2025, from [https://viblo.asia/p/fastapi-phan-1-gioi-thieu-va-setup-](https://viblo.asia/p/fastapi-phan-1-gioi-thieu-va-setup-moi-truong-WR5JRxjQVGv) [moi-truong-WR5JRxjQVGv](https://viblo.asia/p/fastapi-phan-1-gioi-thieu-va-setup-moi-truong-WR5JRxjQVGv)

(Hướng dẫn cài đặt và cấu hình môi trường cho FastAPI)

*5.Dự đoán giá vàng:*

Kane, J. (2024, 09 23). *Dự báo và dự đoán giá vàng: Năm 2024 và xa hơn nữa.* Retrieved 01 04, 2025, from [https://litefinance.vn/blog/analysts-opinions/du-](https://litefinance.vn/blog/analysts-opinions/du-bao-gia-vang/) [bao-gia-vang/](https://litefinance.vn/blog/analysts-opinions/du-bao-gia-vang/)

(Phân tích và dự báo về giá vàng trong tương lai.)

*6.So sánh FastAPI và Flask:*

Moore, M. (2024, 08 07). *FastAPI vs Flask: Choosing the Right Web Development Framework.* Retrieved 01 04, 2025, from <https://shakuro.com/blog/fastapi-vs-flask>

Turing. (n.d.). *Python FastAPI vs Flask: A Detailed Comparison.* Retrieved from <https://www.turing.com/kb/fastapi-vs-flask-a-detailed-comparison>

(So sánh ưu nhược điểm của hai framework FastAPI và Flask.)

1. *Hồi quy Lasso:*

MỸ, B. T. (2000). *Phương pháp ước lượng LASSO: Cơ sở toán học và ứng dụng\*.* Việt Nam: Trường Đại học Ngân hàng TP. Hồ Chí Minh. <https://jabes.ueh.edu.vn/Content/ArticleFiles/a3162398-d635-40c9-b8e5-ca1918383c35/JABES-2020-5-V49.pdf>

(Giới thiệu phương pháp LASSO trong hồi quy và ứng dụng trong phân tích dữ liệu.)

1. *Neural Network (Mạng nơ-ron nhân tạo):*

Nguyễn, H. (2024, 05 29). *Neural Network là gì? Đặc điểm và ứng dụng của Neural Network.* Retrieved 01 04, 2025, from <https://vietnix.vn/neural-network/>

nttuan8. (2019, 09 09). *Bài 3: Neural network.* Retrieved 01 04, 2025, from <https://nttuan8.com/bai-3-neural-network/>

(Giới thiệu về mạng nơ-ron và ứng dụng của nó trong phân tích và dự đoán dữ liệu.)

1. *Dữ liệu về giá vàng:*

Nekouei, F. (2022). *Gold Price Prediction| LSTM | Độ chính xác 96%.* Retrieved from [https://www.kaggle.com/code/farzadnekouei/gold-price-prediction-](https://www.kaggle.com/code/farzadnekouei/gold-price-prediction-lstm-96-accuracy) [lstm-96-accuracy](https://www.kaggle.com/code/farzadnekouei/gold-price-prediction-lstm-96-accuracy)

(Các phương pháp xử lý dữ liệu và cách thức triển khai mô hình cũng được trình bày chi tiết, giúp hiểu rõ hơn về quá trình xây dựng mô hình dự đoán trong lĩnh vực tài chính.)

1. *Render*

Render. (n.d.). *Dịch vụ Web Lưu trữ các ứng dụng web động (Express, Django, v.v.) tại một URL công khai.* From https://docs.render.com/web-services

sapernation. (2024, 2 26). *Render is a cloud platform specifically designed for developers #code #cloud #render*. From <https://www.youtube.com/watch?v=7bOG2QXVMQo>

Shepherd, T. (2024, 8 13). *Render: Simplifying Cloud Hosting for the Next Generation.* From [*https://www.usatoday.com/story/special/contributor-content/2024/08/13/render-simplifying-cloud-hosting-for-the-next-generation/74784208007/*](https://www.usatoday.com/story/special/contributor-content/2024/08/13/render-simplifying-cloud-hosting-for-the-next-generation/74784208007/)

Thao, N. V. (2021, 07 21). *Trình duyệt render một trang web như thế nào.*

Retrieved from <https://viblo.asia/p/trinh-duyet-render-mot-trang-web-nhu-the-nao-RnB5prO2ZPG>

(Giải thích cách trình duyệt render một trang web và các yếu tố ảnh hưởng đến hiệu suất.)

1. *Hồi quy tuyến tính:*

viblo. (2020, 02 17). *"Học máy" trong mô hình hồi quy tuyến tính, phân loại tuyến tính.* Retrieved 09 12, 2024, from [https://viblo.asia/p/hoc-may-trong-mo-hinh- hoi-quy-tuyen-tinh-phan-loai-tuyen-tinh-ByEZkxAqlQ0](https://viblo.asia/p/hoc-may-trong-mo-hinh-%20hoi-quy-tuyen-tinh-phan-loai-tuyen-tinh-ByEZkxAqlQ0)

1. *Dữ liệu giá vàng:*

Kitco. Live Charts/ Gold. Retrieved from [Gold Price Today | Price of Gold Per Ounce | 24 Hour Spot Chart | KITCO](https://www.kitco.com/charts/gold)