TRƯỜNG ĐẠI HỌC THỦY LỢI

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN**

HỌC PHẦN: HỌC MÁY

**ĐỀ TÀI:** Thực hiện phương pháp

***Hồi quy tuyến tính, Lasso, Neural Network*** (trong thư viện sklearn, chọn

MLPRegressor) dự đoán giá vàng

Giáo viên hướng dẫn: Trần Anh Đạt

Sinh viên/nhóm sinh viên thực hiện:nhóm 8-64KTPM5

Thành viên: Đỗ Hữu Minh Tuấn-2251172544(

Nguyễn Khắc Nam Phương-2251172459

Nguyễn Minh Huy-2251172372

Phạm Việt Hưng-2251172363

**Hà Nội, tháng 10 năm 2024**

[Phần 1. Tổng quan…………………………………………………………………... 3](#_Toc155778833)

[1. Giới thiệu về học máy…………………………………………………………………... 3](#_Toc155778834)

[2. Trình bày các phương pháp học máy được sử dụng trong đề tài mà nhóm chọn….. 5](#_Toc155778835)

[Phần 2. Thực nghiệm……………………………………………………………….. 7](#_Toc155778836)

[1. Mô tả bài toán…………………………………………………………………………... 7](#_Toc155778837)

[***1.1. Phương pháp hồi quy tuyến tính……………………………………………………* 7**](#_Toc155778838)

[***1.2. Phương pháp Lasso…………………………………………………………………* 9**](#_Toc155778839)

[***1.3. Phương pháp neural network……………………………………………………..* 10**](#_Toc155778840)

[2. Mô tả tập dữ liệu của bài toán………………………………………………………... 11](#_Toc155778841)

[3. Viết ứng dụng………………………………………………………………………….. 12](#_Toc155778842)

[***3.1. Cách lựa chọn các* tham *số của mỗi phương pháp để xây dựng mô hình………* 12**](#_Toc155778843)

[**3.1.1. Linear Regression (Tự Code)………………………………………………... 12**](#_Toc155778844)

[**3.1.2. Linear Regression (Thư viện có sẵn)………………………………………... 13**](#_Toc155778845)

[**3.1.3. Lasso…………………………………………………………………………... 14**](#_Toc155778846)

[**3.1.4. Neutural Network……………………………………………………………. 14**](#_Toc155778847)

[**3.1.5. Phân tích quá trình hội tụ……………………………………………………………. 14**](#_Toc155778847)

3.1.6.**phân tích tham số đánh giá của mô hình**

[***3.2. CÁCH XÂY DỰNG MÔ HÌNH HỌC KẾT HỢP…………………………………* 15**](#_Toc155778848)

[***3.3. GIAO DIỆN NGƯỜI DÙNG……………………………………………………...* 19**](#_Toc155778849)

[4. Phân tích kết quả của chương trình…………………………………………………. 20](#_Toc155778850)

[**4.1. Biểu đồ so sánh độ đo R2………………………………………………………… 20**](#_Toc155778851)

[**4.2. Biểu đồ so sánh độ đo MAE……………………………………………………… 22**](#_Toc155778853)

[**4.3 Biểu đồ so sánh độ đo RMSE……………………………………………………... 23**](#_Toc155778854)

[Phần 3. Kết luận……………………………………………………………………. 24](#_Toc155778855)

[Tài liệu tham khảo………………………………………………………………. 25](#_Toc155778856)

**Phần 1. Tổng quan**

## 1. Giới thiệu về học máy

***-Lịch sử của Machine Learning.***

Machine Learning nổi lên như một bằng chứng của cuộc cách mạng công nghiệp lần thứ tư. Machine learning là thuật ngữ được đặt bởi Arthur Samuel vào năm 1959. Samuel là một IBMer người Mỹ kiêm nhà tiên phong trong lĩnh vực trí tuệ nhân tạo và máy tính chơi game. Năm 1960, thuật ngữ học máy phổ biến hơn thông qua cuốn sách của Nilsson, nội dung đề cập đến việc phân loại máy học.

Machine learning hiện đại bao gồm hai mục tiêu chính: phân loại dữ liệu thông qua mô hình đã được phát triển và đưa ra dự đoán về kết quả trong tương lai dựa trên mô hình này.

***-Vai trò của Machine Learning.***

Học máy (Machine Learning): nghiên cứu và xây dựng các kĩ thuật cho phép các hệ thống “học” tự động từ dữ liệu để giải quyết những vấn đề cụ thể

Một chương trình máy tính được gọi là **học** từ *kinh nghiệm E* để hoàn thành *nhiệm vụ T*, với hiệu quả được đo bằng *phép đánh giá P*, nếu hiệu quả của nó khi thực hiện nhiệm vụ *T*, khi được đánh giá bởi *P*, cải thiện theo kinh nghiệm *E*.

|  |  |  |
| --- | --- | --- |
| Bài toán | Tập dữ liệu | Các phương pháp |
| Hồi quy | Tập dữ liệu đã được gán nhãn (Xtrain, ytrain). Miền giá trị của tập nhãn ytrain là liên tục và có vô số giá trị | Linear Regression, Lasso, Ridgre, K-Fold |
| Phân lớp | Tập dữ liệu đã được gán nhãn (Xtrain, ytrain). Miền giá trị của tập nhãn ytrain là rời rạc và hữu hạn | SVM, perceptron, cây quyết định, Neural Network, Logistic Regression |
| Phân Cụm | Tập dữ liệu huấn luyện không được gán nhãn (Xtrain) | K-means |

**-Trình bày :**

|  |  |  |  |
| --- | --- | --- | --- |
| Kiểu học | Tập dữ liệu | Bài toán | Phương pháp |
| Học có giám sát | Tập dữ liệu huấn luyện được gán nhãn (Xtrain, ytrain) | Hồi quy, phân lớp | Linear Regression, Lasso, Ridgre, K-Fold, SVM, perceptron, cây quyết định, Neural Network, Logistic Regression |
| Học không có giám sát | Tập dữ liệu huấn luyện không được gán nhãn (Xtrain) | Phân cụm | K-means |

## 2. Trình bày các phương pháp học máy được sử dụng trong đề tài mà nhóm chọn

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Phương pháp | Input | Output | Bài toán ưu (hoặc hàm mất mát) | Cách thực hiện |
| Linear Regression (Hồi quy tuyến tính) | Tập dữ liệu huấn luyện đã được gán nhãn (Xtrain,ytrain) | Hàm tuyến tính có dạng f(xi)=wxi + w0 |  | Tìm giá trị tối ưu của w giải phương trình đạo hàm của hàm mất mát theo w bằng 0  1 |
| Neural Network | Tập dữ liệu huấn luyện đã được gán nhãn (Xtrain,ytrain), kiến trúc mạng nơ ron (số lớp ẩn, số nơron của mỗi lớp ẩn), hàm kích hoạt (activation function), hàm mất mát (loss function) | Bộ vector trọng số của các liên kết giữa các nơ ron (W) để hàm mất mát đạt giá trị tối ưu | Hàm mất mát tại tầng ouput của mạng nơ-ron tùy thuộc vào bài toán thực hiện. Nếu bài toán Hồi quy, hàm mất mát có thể sử hàm mất mát của Hồi quy tuyến tính. Nếu bài toán phân lớp, có thể sử dụng hàm mất mát của Perceptron hoặc Logistic Regression, … | Phương pháp phổ biến nhất để tối ưu hàm mất mát của mạng nơ-ron là Gradient Descent (GD)  Để áp dụng GD, chúng ta cần tính được đạo hàm của hàm mất mát theo từng trọng số trong mạng nơ-ron bằng phương pháp lan truyền ngược (backpropagation) |
| Lasso | Tập dữ liệu huấn luyện đã được gán nhãn (Xtrain,Ytrain) | Đánh giá thông qua các độ đo như MSE, R-squared hoặc các độ đo khác | IMG_256 | Sử dụng các phương pháp tối ưu gradient descent Thuật toán tối ưu hóa sẽ cố gắng điều chỉnh giá trị của các hệ số anpha để đạt được giá trị nhỏ nhất của hàm mất mát |

**Phần 2. Thực nghiệm**

## 1. Mô tả bài toán

***1.1. Phương pháp hồi quy tuyến tính***

***- Mục đích thực hiện bài toán:*** xấy dựng một mô hình dự đoán giá trị của một biến phụ thuộc (đối tượng cần dự đoán) dựa trên các biến độc lập (đặc trưng)

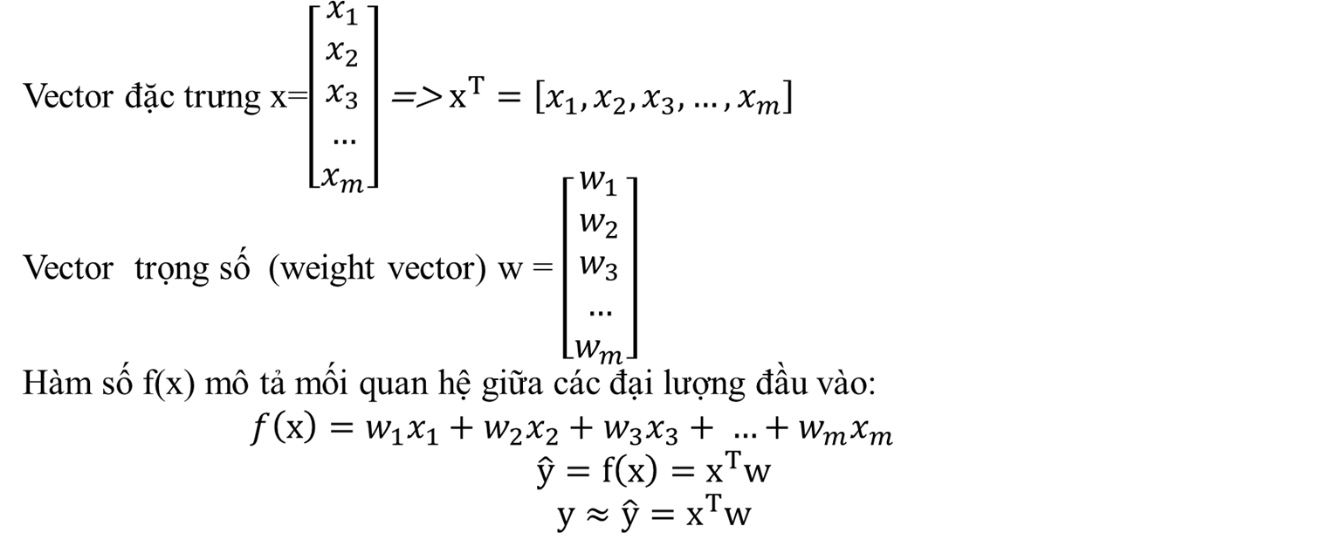
***- Input:*** Cho tập dữ liệu huấn luyện gồm N mẫu. Mỗi mẫu là một cặp (xi ,yi ):

• xi : vector đặc trưng

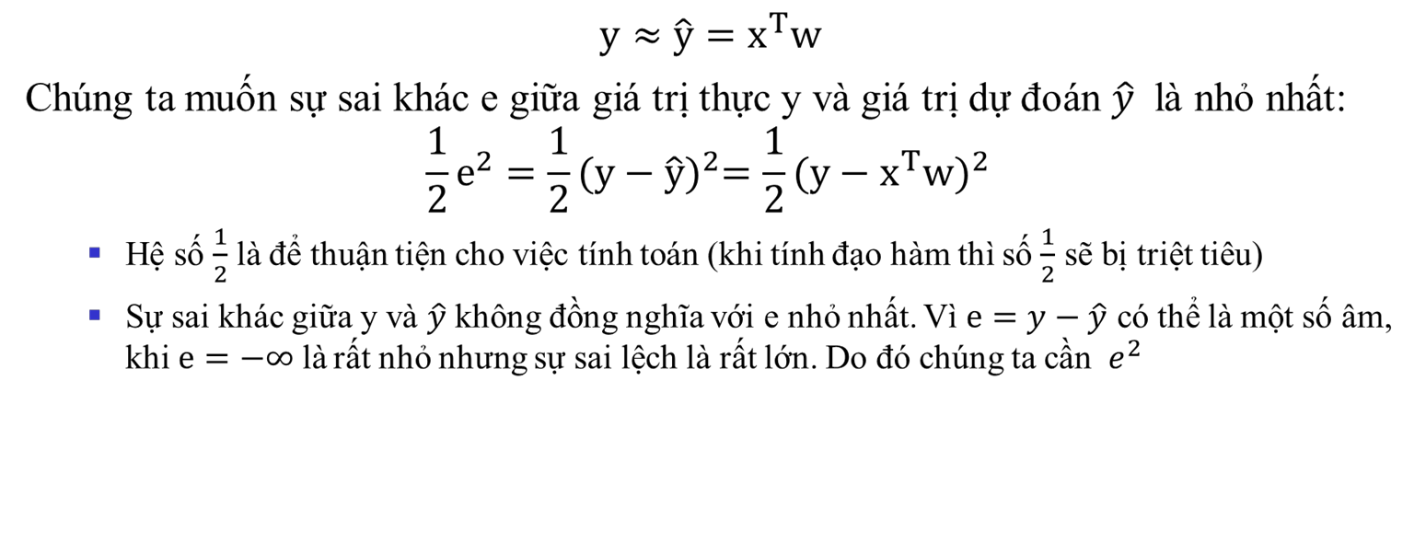
• yi : giá trị của vector đặc trưng xi

***- Output:*** Hàm tuyến tính có dạng f(xi ) = wxi + w0

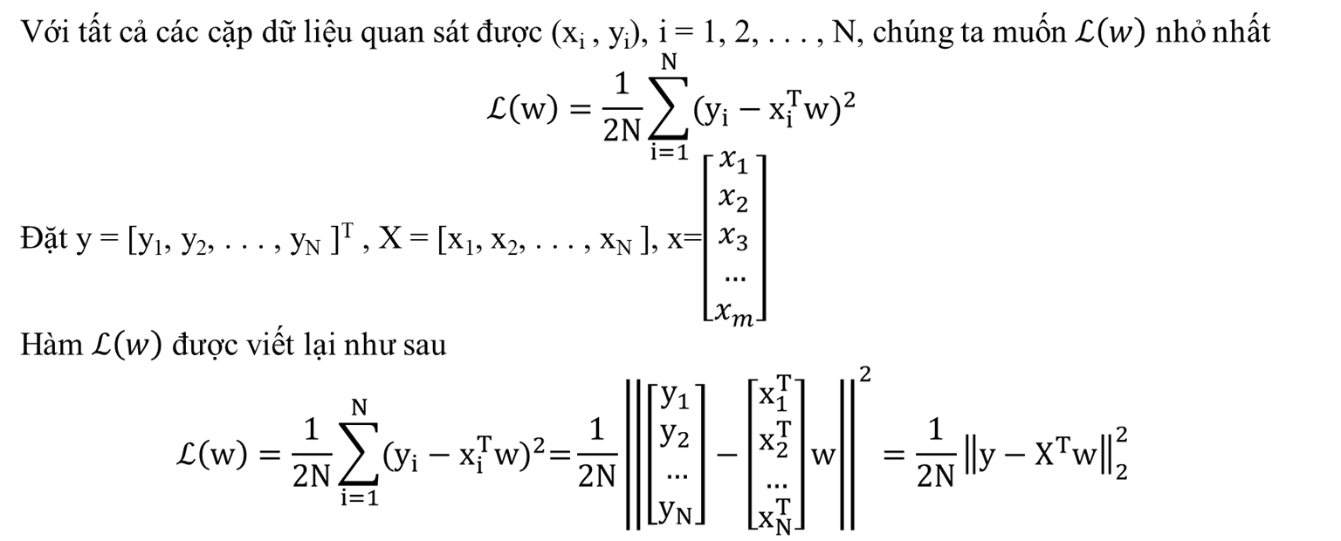
***- Tóm tắt công việc bài toán :***



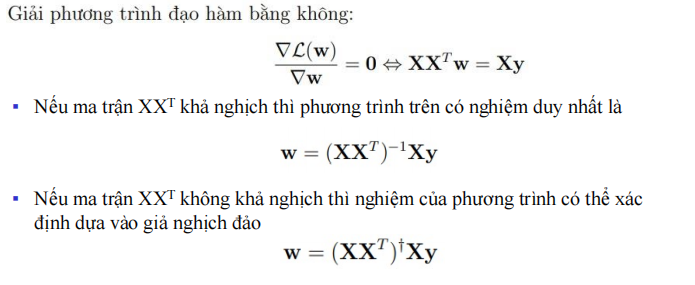
**+ Sai số dự đoán**



**+ Hàm mất mát**



**+Nghiệm cho bài toán Hồi quy tuyến tính**



***1.2. Phương pháp Lasso***

***- Mục đích thực hiện bài toán :*** giúp giảm overfitting và giúp chọn lựa chọn đặc trưng hữu ích cho mô hình

***- Input:*** Tập dữ liệu huấn luyện đã được gán nhãn (Xtrain,Ytrain).

***- Output:*** Đánh giá thông qua các độ đo như MSE, R-squared hoặc các độ đo khác.

- ***Tóm tắt công việc bài toán :***

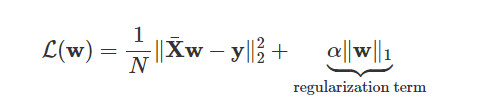
Giả sử chúng ta có một tập dữ liệu huấn luyện gồm mẫu dữ liệu (instances) và nhãn tương ứng (labels). Chúng ta muốn tìm một mô hình hồi quy tuyến tính có dạng:

y = Xw + b

Trong đó:

* y là vector nhãn (labels).
* X là ma trận mẫu dữ liệu (instances), với mỗi hàng là một mẫu và mỗi cột là một đặc trưng.
* w là vector trọng số (weights) tương ứng với các đặc trưng.
* b là sai số chệch (bias).

Phương pháp LASSO thực hiện tối ưu hóa hàm mất mát (loss function) theo công thức:



Trường hợp anpha=0, thành phần điều chuẩn bị tiêu giảm và chúng ta quay trở về bài toán hồi qui tuyến tính.

Trường hợp anpha nhỏ thì vai trò của thành phần điều chuẩn trở nên ít quan trọng. Mức độ kiểm soát quá khớp của mô hình sẽ trở nên kém hơn.

Trường hợp anpha lớn chúng ta muốn gia tăng mức độ kiểm soát lên độ lớn của các hệ số ước lượng.

***1.3. Phương pháp neural network***

***-Mục đích thực hiện bài toán :*** giúp thực hiện các nhiệm vụ như phân loại, dự đoán và nhận dạng thông tin từ dữ liệu bằng cách sử dụng các lớp và trọng số được tự động điều chỉnh trong quá trình huấn luyện

***-Input*:** Tập dữ liệu (Xtrain, ytrain), kiến trúc mạng nơ ron (số lớp ẩn, số nơ ron của mỗi lớp ẩn), hàm kích hoạt (activation function), hàm mất mát (loss function)

**-Output:** Bộ vector trọng số của các liên kết giữa các nơ ron (W) để hàm mất mát đạt giá trị tối ưu

**-*Tóm tắt phương pháp bài toán*:** Phương pháp phổ biến nhất để tối ưu MLP vẫn là Gradient Descent (GD)

+Để áp dụng GD, chúng ta cần tính được gradient của hàm mất mát theo từng ma trận trọng số W^*l ,* và vector bias b^*l*

***Phương pháp giảm gradient***

+Hàm lỗi (tổng chênh lệch giũa đầu ra thu được và đầu ra mong muốn) là một hàm f(w) của các trọng số liên kết

+Cần tìm ra một tọng số w mà tại đó hàm lỗi là nhỏ nhất

+Hàm lỗi sẽ giảm dần mỗi khi học 1 dữ liệu mẫu, túc là đầu ra thu được của mang sẽ tiến sát dần đến đầu ra mong muốn

Wi+1=Wi -f’(Wi , x)

## 2. Mô tả tập dữ liệu của bài toán

* Dữ liệu gồm những chiều thông tin gì (mỗi mẫu (vertor) dữ liệu có những thông tin gì), có bao nhiêu mẫu dữ liệu (ít nhất là 100 vector dữ liệu). Mô tả nhãn lớp của dữ liệu.

+) Dữ liệu gồm 2228 mẫu.

+) Các chiều thông tin của dữ liệu:

* Ngày giao dịch (date): ngày mà dữ liệu được ghi nhận
* Giá (price): giá vàng của ngày hôm đó
* Giá mở cửa (open): giá mở cửa của vàng vào ngày giao dịch đó
* Giá cao nhất (high): giá cao nhất mà vàng đạt được trong ngày giao dịch
* Giá thấp nhất (low): giá thấp nhất mà vàng đạt được trong ngày giao dịch
* Khối lượng giao dịch (vol.): số lượng vàng được giao dịch trong ngày đó, thường được tính bằng đơn vị K (nghìn) hoặc K (triệu)
* Phần trăm thay đổi (change): Phần trăm thay đổi giá trị từ ngày trước đó đến ngày giao dịch đó

+) Features: Open, High, Low, vol, change

+) Nhãn lớp của dữ liệu là: Price (giá vàng của ngày hôm đó)

Mô tả ma trận dữ liệu (X), nhãn lớp (Y).

+) Ma trận X bao gồm: 5 cột và 2228 dòng (Open, High, Low, Change, Volume, Ch%)

+) Ma trận Y bao gồm: 1 cột và 2228 dòng (Price)

* Chia tập dữ liệu thành 2 phần: 70% dùng để huấn luyện mô hình, 30% dùng để kiểm tra sự phù hợp của mô hình.

## 3. Viết ứng dụng

* ***3.1. Cách lựa chọn các tham số của mỗi phương pháp để xây dựng mô hình.***

#### **3.1.1. Linear Regression (Tự Code)**

class LinearRegressionTuCode:

def \_\_init\_\_(self, has\_bias=True):

self.has\_bias = has\_bias

self.w = None

self.coef\_ = None

self.intercept\_ = None

def add\_bias(self, X):

if self.has\_bias:

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

X\_with\_bias = np.concatenate((ones, X), axis=1)

return X\_with\_bias

return X

def fit(self, X, y):

X\_with\_bias = self.add\_bias(X)

self.w = np.linalg.pinv(X\_with\_bias.T @ X\_with\_bias) @ X\_with\_bias.T @ y # w = np.linalg.pinv(X\_train@X\_train.T)@X\_train@y\_train

self.intercept\_ = self.w[0] if self.has\_bias else 0

self.coef\_ = self.w[1:] if self.has\_bias else self.w

def predict(self, X):

X\_with\_bias = self.add\_bias(X)

return X\_with\_bias @ self.w #f(x)=X\_T@W

* Giải thích:

+) Tham số được nhóm em lựa chọn ở đây là has\_bias = True

+) Khi has\_bias được thiết lập là True trong phương thức khởi tạo của class LinearRegressionTuCode, điều này có tác dụng bao gồm hệ số chệch (intercept) vào mô hình hồi quy tuyến tính. Hệ số chệch là một giá trị cộng thêm vào đầu ra dự đoán, không phụ thuộc vào bất kỳ đặc trưng nào của dữ liệu đầu vào. Có hệ số chệch giúp mô hình có thể biểu diễn được sự thay đổi trong đầu ra mà không cần phải thông qua điểm gốc (0,0) của các đặc trưng.

+) Điều này giúp mô hình hồi quy tuyến tính có thể học được hệ số chệch và làm cho mô hình linh hoạt hơn trong việc biểu diễn các dữ liệu.

#### **3.1.2. Linear Regression (Thư viện có sẵn)**

reg = LinearRegression(fit\_intercept=False).fit(X\_train, y\_train)

- Giải thích:

+) fit\_intercept=False: Tham số này kiểm soát xem mô hình hồi quy tuyến tính có sử dụng hệ số chệch (intercept) hay không. Khi fit\_intercept là False, mô hình sẽ không sử dụng hệ số chệch và chỉ sử dụng các đặc trưng của dữ liệu để dự đoán đầu ra. Trong trường hợp này, mô hình được tạo không có hệ số chệch và chỉ tập trung vào các đặc trưng để làm dự đoán.

#### **3.1.3. Lasso**

lasso = Lasso(alpha=1.0,max\_iter=1000,tol=0.01).fit(X\_train,y\_train)

- Giải thích:

+) alpha: Tham số alpha trong mô hình Lasso là tham số chính kiểm soát mức độ regularization (điều chuẩn) được áp dụng trong quá trình huấn luyện. Regularization giúp kiểm soát việc overfitting (quá mức khớp) trong mô hình bằng cách giảm giá trị của các hệ số (weights) của đặc trưng.

+) max\_iter: Là số lần lặp tối đa (iterations) mà thuật toán sẽ thực hiện để tìm giải pháp. Nếu thuật toán không hội tụ trong số lượng lặp này, nó sẽ dừng và trả về kết quả hiện tại.

+) tol: Là ngưỡng hội tụ. Nếu thay đổi tối thiểu giữa hai lần lặp liên tiếp của thuật toán là dưới ngưỡng này, thuật toán sẽ dừng. tol được sử dụng để kiểm soát độ chính xác của kết quả.

#### **3.1.4. Neutural Network**

mlp = MLPRegressor(hidden\_layer\_sizes=(100, 100, 100), activation='relu', solver='adam', max\_iter=1000).fit(X\_train, y\_train)

- Giải thích:  
 +) hidden\_layer\_sizes=(100, 100, 100): đây là một trong những tham số quan trọng nhất. Nó xác định kiến trúc của mạng nơ-ron, đặc biệt là số lượng và kích thước của các tầng ẩn. Trong trường hợp này, mô hình có ba tầng ẩn, mỗi tầng có 100 nơ-ron.

+) activation='relu': Đây là hàm kích thích được sử dụng trong các nơ-ron ẩn. Trong trường hợp này, bạn sử dụng Rectified Linear Unit (ReLU) làm hàm kích thích. ReLU là một hàm phi tuyến tính thường được sử dụng để giảm vấn đề biến mất gradient và giúp mô hình học các đặc trưng phi tuyến tính hiệu quả.

+) solver='adam': Là thuật toán tối ưu hóa được sử dụng để điều chỉnh trọng số của mô hình trong quá trình huấn luyện. Trong trường hợp này, bạn sử dụng thuật toán Adam. Adam là một phương pháp tối ưu hóa thông dụng và thích hợp cho nhiều loại dữ liệu và bài toán.

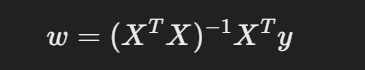
+) max\_iter=1000: Là số lượng lần lặp tối đa mà thuật toán sẽ thực hiện trong quá trình huấn luyện. Nếu mô hình không hội tụ sau số lần lặp này, quá trình huấn luyện sẽ dừng lại.

3.1.3. Phân tích quá trình hội tụ:

3.1.3.1. Linear Regression tự code (3.1.1)

Quá trình hội tụ:

* Công thức chuẩn: Trong hồi quy tuyến tính, trọng số www được tính dựa trên công thức bình phương nhỏ nhất thông qua phép tính ma trận:



 Sử dụng np.linalg.pinv: Sử dụng hàm pinv() để tính ma trận nghịch đảo giả, nhằm tìm nghiệm cho mô hình hồi quy tuyến tính.

 Tham số has\_bias=True: Khi has\_bias=True, ta thêm một cột các giá trị 1 vào ma trận đầu vào để đại diện cho hệ số chệch, giúp mô hình linh hoạt hơn.

Phân tích hội tụ:

* Mô hình hồi quy tuyến tính không yêu cầu thuật toán tối ưu hóa lặp lại (như gradient descent) vì công thức này cho ra nghiệm chính xác ngay lập tức (chỉ cần tính toán nghịch đảo ma trận).
* Điểm hội tụ là ngay tại khi tính xong ma trận nghịch đảo và ra kết quả hệ số www. Do đó, không có "quá trình hội tụ" lặp đi lặp lại như trong các thuật toán dựa trên gradient.

3.1.3.2. Linear Regression từ thư viện (3.1.2)

Quá trình hội tụ:

* Với thư viện scikit-learn, nếu sử dụng fit\_intercept=True, quá trình tính toán hồi quy tuyến tính sẽ bao gồm cả hệ số chệch.
* Hàm tối ưu: Mô hình sẽ sử dụng công thức bình phương nhỏ nhất (tương tự như phiên bản tự code) nhưng được tối ưu hóa mạnh mẽ hơn về mặt tính toán, đảm bảo hội tụ nhanh và chính xác.

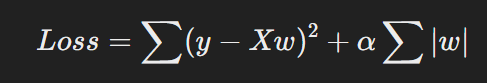
Phân tích hội tụ:

* Tương tự như phiên bản tự code, mô hình hội tụ ngay sau khi tính toán nghịch đảo ma trận. Không có thuật toán lặp lại cần thiết.
* Nếu fit\_intercept=False, hệ số chệch sẽ không được tính đến, và mô hình sẽ chỉ học các trọng số www dựa trên đặc trưng đầu vào.

3.1.3.3. Lasso Regression (3.1.3)

Quá trình hội tụ:

* Lasso là một dạng hồi quy tuyến tính có thêm điều chuẩn L1. Điều này có nghĩa là mô hình không chỉ tối ưu hóa hàm mất mát, mà còn thêm một điều khoản phạt dựa trên tổng giá trị tuyệt đối của các trọng số:



Thuật toán tối ưu hóa: Lasso thường sử dụng các thuật toán tối ưu như Coordinate Descent để tìm ra trọng số tối ưu dần dần qua các vòng lặp.

Phân tích hội tụ:

* max\_iter=1000: Số lần lặp tối đa cho thuật toán. Nếu thuật toán chưa hội tụ sau 1000 vòng lặp, quá trình tối ưu hóa sẽ dừng lại.
* tol=0.01: Đây là ngưỡng hội tụ. Khi sự thay đổi giữa hai lần lặp liên tiếp nhỏ hơn giá trị này, thuật toán sẽ dừng lại, coi như đã hội tụ.

Trong quá trình hội tụ của Lasso, mô hình sẽ giảm dần giá trị của hàm mất mát qua từng vòng lặp, đồng thời điều chỉnh các trọng số theo hướng làm giảm các trọng số không cần thiết (do điều chuẩn L1).

3.1.3.4. Neural Network (MLP) (3.1.4)

Quá trình hội tụ:

* Neural Network (MLP) sử dụng các tầng ẩn với số lượng nơ-ron xác định trong mỗi tầng (100 nơ-ron trong ví dụ trên). Mạng nơ-ron học thông qua việc điều chỉnh trọng số trong quá trình huấn luyện.
* Hàm kích hoạt ReLU: Sau khi mỗi nơ-ron nhận đầu vào, nó áp dụng hàm ReLU để tạo ra đầu ra phi tuyến tính.
* Thuật toán tối ưu hóa Adam: Adam là một biến thể của Stochastic Gradient Descent (SGD), sử dụng động lượng và phương pháp điều chỉnh tốc độ học dựa trên gradient trước đó để tối ưu hóa nhanh hơn.

Phân tích hội tụ:

* max\_iter=1000: Mô hình sẽ thực hiện tối đa 1000 vòng lặp để tối ưu hóa trọng số. Nếu sau 1000 vòng lặp mà mô hình chưa hội tụ, quá trình sẽ dừng lại.
* Hội tụ trong mạng nơ-ron: Hội tụ xảy ra khi giá trị hàm mất mát dừng giảm hoặc giảm rất chậm. Adam giúp mô hình đạt được điều này nhờ điều chỉnh tốc độ học phù hợp.
* Vấn đề gradient: ReLU giúp giảm vấn đề gradient biến mất, giúp quá trình hội tụ nhanh hơn so với các hàm kích hoạt khác như sigmoid.

3.1.6.**phân tích tham số đánh giá của mô hình:**

R2:

sử dụng trong code:

st.write(f"R2: {r2\_score(y\_test, y\_predict\_lr):.6f}")

đây là hàm hiển thị giá trị R2 cho mô hình Linear Regression

hàm r2\_score sẽ so sánh giá trị thực tế với giá trị dự đoán để tính R2

MAE:

def MAE(y\_test, y\_predict): # càng nhỏ càng tốt

return mean\_absolute\_error(y\_test, y\_predict)

MAE là sai số tuyệt đối trung bình. Hàm này đo mức độ sai lệch giữa giá trị dự đoán (y\_predict) và giá trị thực tế (y\_test). MAE càng nhỏ thì càng tốt

RMSE:

Hàm math.sqrt(mean\_squared\_error(y\_test, y\_predict\_lasso)) được sử dụng để tính Sai số bình phương trung bình (Mean Squared Error - MSE) và sau đó lấy căn bậc hai của giá trị đó.

Khi gọi hàm mean\_squared\_error(y\_test, y\_predict\_lasso), hàm này sẽ trả về giá trị MSE giữa các giá trị thực tế và giá trị dự đoán.

Sau đó, hàm math.sqrt(...) sẽ tính căn bậc hai của giá trị MSE để cho ra RMSE.

### ***3.2. CÁCH XÂY DỰNG MÔ HÌNH HỌC KẾT HỢP***

Bước 1: Chuẩn bị dữ liệu

* Chia dữ liệu: Phân chia dữ liệu thành bộ huấn luyện và bộ kiểm thử để đánh giá hiệu suất của mô hình stacking.
* Chọn mô hình cơ bản: Chọn một loạt các mô hình đơn lẻ khác nhau. Các mô hình có thể thuộc các họ khác nhau để đảm bảo đa dạng, trong bài của nhóm em sử dụng các mô hình: Linear Regression, Lasso và Neural Network

Bước 2: Huấn luyện các mô hình đơn lẻ

* Huấn luyện mô hình cơ bản: Huấn luyện từng mô hình trên bộ huấn luyện.
* Dự đoán trên bộ kiểm thử: Sử dụng mỗi mô hình để dự đoán trên bộ kiểm thử.

Bước 3: Xây dựng mô hình stacking

* Xây dựng bộ dự đoán đa mô hình: Tạo một ma trận hoặc tập hợp các dự đoán từ các mô hình đơn lẻ trên bộ kiểm thử.
* Chọn mô hình meta-learner: Chọn một mô hình meta-learner để học từ dự đoán của các mô hình đơn lẻ. Một mô hình đơn lẻ khác sẽ được sử dụng để xây dựng mô hình cuối cùng.
* Huấn luyện meta-learner: Sử dụng dự đoán từ các mô hình đơn lẻ làm đặc điểm đầu vào và nhãn thực tế làm đầu ra để huấn luyện mô hình meta-learner.

Bước 4: Dự đoán với mô hình stacking

* Dự đoán cuối cùng: Sử dụng mô hình stacking để dự đoán trên bộ kiểm thử. Đầu vào cho mô hình stacking là kết quả dự đoán từ các mô hình đơn lẻ.

Bước 5: Đánh giá và điều chỉnh

* Đánh giá hiệu suất: Đánh giá hiệu suất của mô hình stacking trên bộ kiểm thử.
* Tinh chỉnh tham số: Có thể cần điều chỉnh các tham số của các mô hình đơn lẻ và meta-learner để cải thiện hiệu suất.

- Lặp lại nếu cần: Lặp lại quá trình trên nếu cần thiết để tối ưu hóa mô hình stacking.

**CODE STACKING:**

from sklearn.ensemble import StackingRegressor

from sklearn.linear\_model import LinearRegression, Lasso

from sklearn.neural\_network import MLPRegressor

import numpy as np

class StackingRegressorCustom:

def \_\_init\_\_(self, model1=LinearRegression(fit\_intercept=False), model2=Lasso(alpha=1.0, max\_iter=1000, tol=0.01),

model3=MLPRegressor(hidden\_layer\_sizes=(100, 100, 100), activation='relu', solver='adam', max\_iter=1000),

final\_estimator=LinearRegression(), cv=5, passthrough=True, n\_jobs=-1):

self.model1 = model1

self.model2 = model2

self.model3 = model3

self.final\_estimator = final\_estimator

self.cv = cv

self.passthrough = passthrough

self.n\_jobs = n\_jobs

self.stacked\_model = StackingRegressor(

estimators=[('model1', model1), ('model2', model2), ('model3', model3)],

final\_estimator=final\_estimator,

cv=cv,

passthrough=passthrough,

n\_jobs=n\_jobs

)

def fit(self, X\_train, y\_train):

self.model1.fit(X\_train, y\_train)

self.model2.fit(X\_train, y\_train)

self.model3.fit(X\_train, y\_train)

self.stacked\_model.fit(X\_train, y\_train)

def predict(self, X\_test):

y\_pred1 = self.model1.predict(X\_test)

y\_pred2 = self.model2.predict(X\_test)

y\_pred3 = self.model3.predict(X\_test)

y\_pred\_stacked = self.stacked\_model.predict(X\_test)

return y\_pred\_stacked

# Sử dụng class StackingRegressorCustom

stacked\_classifier = StackingRegressorCustom()

stacked\_classifier.fit(X\_train, y\_train)

y\_predict\_stacked = stacked\_classifier.predict(X\_test)

print("r2\_score of StackingRegressorCustom :",r2\_score(y\_test,y\_predict\_stacked))

print("MAE of StackingRegressorCustom :",mean\_absolute\_error(y\_test,y\_predict\_stacked))

print("RMSE of StackingRegressorCustom :",np.sqrt(mean\_squared\_error(y\_test,y\_predict\_stacked)))

**CÁC THAM SỐ TRONG MÔ HÌNH**

- model1, model2, model3:

* Các mô hình hồi quy cơ bản được sử dụng trong bộ tổ hợp stacking.
* Giá trị mặc định:
* model1: Hồi quy tuyến tính với fit\_intercept được đặt là False.
* model2: Hồi quy Lasso với alpha=1.0, max\_iter=1000 và tol=0.01.
* model3: Mô hình Hồi quy đa tầng Perceptron (MLP) với các tham số cụ thể (hidden\_layer\_sizes=(100, 100, 100), activation='relu', solver='adam', max\_iter=1000).

- estimators:

* Một danh sách các bộ đôi trong đó mỗi bộ đôi chứa một chuỗi (tên) và một mô hình hồi quy.
* Xác định các mô hình cơ bản được sử dụng trong bộ tổ hợp stacking.

- final\_estimator:

* Mô hình siêu cuối cùng kết hợp các dự đoán của các mô hình cơ bản.

- cv:

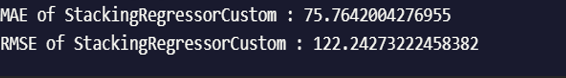
* Số lượng fold trong quá trình cross-validation để dự đoán.

- passthrough:

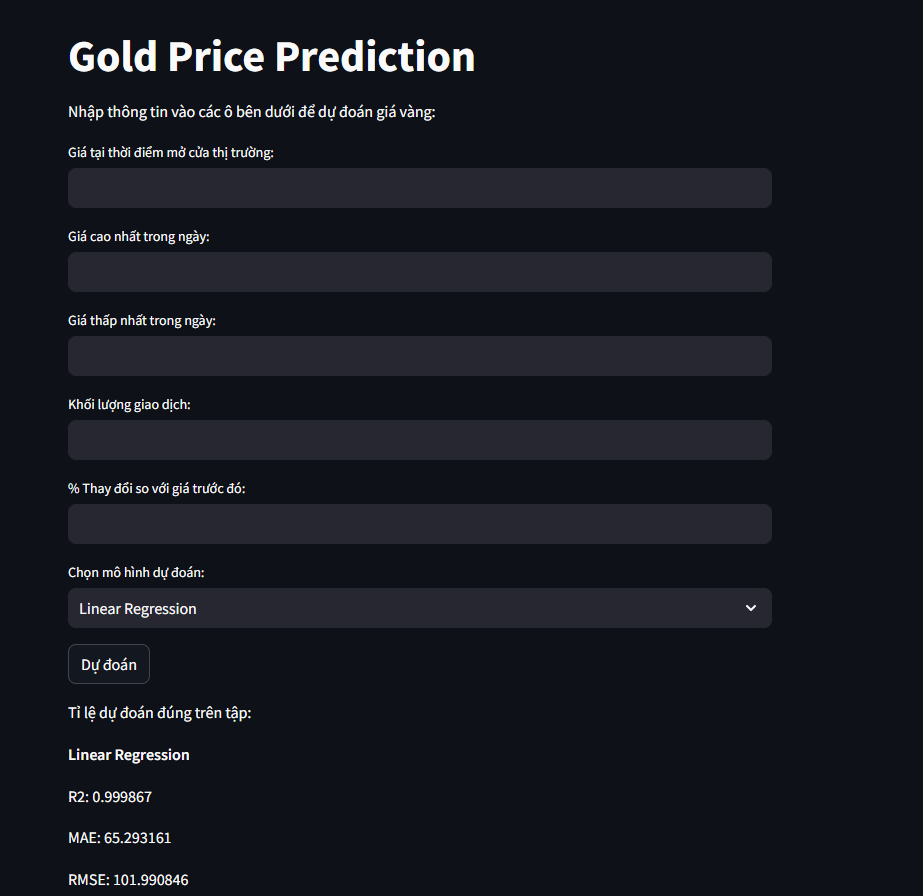
* Xác định liệu có chuyển tiếp các đặc trưng gốc đến mô hình cuối cùng hay không.

- n\_jobs:

* Số lượng công việc chạy song song trong quá trình cross-validation và fitting.

**ĐỘ ĐO ĐÁNH GIÁ CỦA MÔ HÌNH**

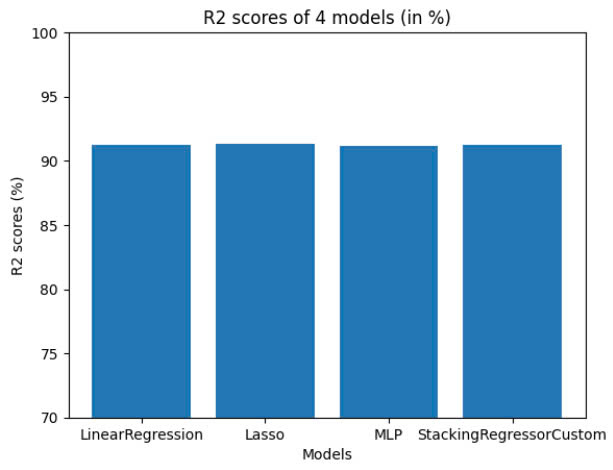
### ***3.3. GIAO DIỆN NGƯỜI DÙNG***

******

## 

## 4. Phân tích kết quả của chương trình

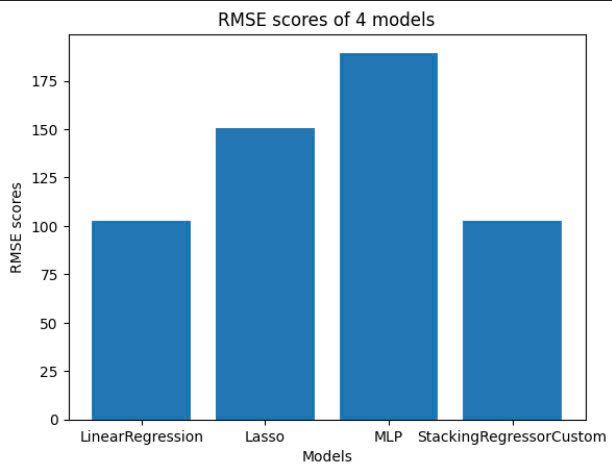
### **4.1. Biểu đồ so sánh độ đo R2:**



### 

### **4.2. Biểu đồ so sánh độ đo MAE:**

### **4.3 Biểu đồ so sánh độ đo RMSE**



* Ở đây chúng ta sẽ sử dụng độ đo R2 là độ đo chính để đánh giá. Chúng ta thấy rằng model Linear Regression sẽ là mô hình phù hợp nhất với bài toán “Dự đoán giá vàng”

**Phần 3. Kết luận**

* Sử dụng thành công các mô hình học máy: Linear Regression, Lasso, Neural Network để dự đoán giá vàng
* Sử dụng các độ đo: R2, MAE, RMSE để đánh giá mô hình học máy và lựa chọn ra được mô hình tốt nhất cho bài toán đó là Linear Regression
* Xây dựng giao diện người dùng có chức năng nhập vào các thông tin và dự đoán được giá vàng dựa vào các model học máy

**Tài liệu tham khảo**

* <https://www.kaggle.com/code/farzadnekouei/gold-price-prediction-lstm-96-accuracy/input>
* <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html>
* <https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html>
* https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LinearRegression.html
* Slide bài giảng GV.Trần Anh Đạt