TRƯỜNG ĐẠI HỌC THỦY LỢI

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

HỌC PHẦN: HỌC MÁY

ĐỀ TÀI: Sử dụng một số phương pháp Hồi quy trong Học có giám sát (Linear Regression, Ridge Regression, Neural network) và mô hình Học kết hợp (Stacking) để dự đoán giá điện thoại di động

Giáo viên hướng dẫn: TS. Nguyễn Thị Kim Ngân

Sinh viên/nhóm sinh viên thực hiện:

1. Vũ Anh Xuân, lớp 63TTNT
2. Đào Thị Trang, lớp 63TTNT
3. Khổng Quốc Trung, lớp 63TTNT
4. Nguyễn Quang Việt, lớp 63TTNT

Hà Nội, tháng 1 năm 2024

MỤC LỤC

Phần 1. Tổng quan

1. Giới thiệu về học máy

- Lịch sử và vai trò của Machine Learning.

+) Machine Learning đã có lịch sử hơn 70 năm và đã trải qua nhiều giai đoạn phát triển. Điều này bắt đầu từ những năm 1940 khi các nhà khoa học đã phát triển các thuật toán học máy sơ khai. Sau đó, trong những năm 1950 và 1960, Machine Learning được phát triển mạnh mẽ với sự ra đời của các thuật toán học máy như Perceptron, Neural Network và Decision Tree.

Học có giám sát	**Học không có giám sát**
- Là học có tiêu chí để đánh giá, tức là dữ liệu được gán nhãn yi - Là bài toán nhằm dự đoán 1 giá trị thực chính xác - Các bài toán: Hồi quy và Phân lớp	- Là học không có tiêu chí để đánh giá, tức là dữ liệu không có gán nhãn yi - Các bài toán: Phân cụm

Hồi quy	**Phân lớp**	**Phân cụm**
- Là bài toán học có giám sát (tức có chỉ tiêu để đánh giá, dữ liệu được gán nhãn yi) nhằm dự đoán 1 giá trị thực, liên tục, chính xác - Các phương pháp đã học: Hồi quy tuyến tính, Hồi quy Ridge, Hồi quy Lasso	- Là bài toán học có giám sát (tức có chỉ tiêu để đánh giá, dữ liệu được gán nhãn yi) nhằm dự đoán 1 giá trị thực, rời rạc, chính xác - Các phương pháp đã học: Perceptron, SVM, ID3, CART, Neural network, Hồi quy Logistic	- Là bài toán học không giám sát (tức là không có chỉ tiêu để đánh giá) - Các phương pháp đã học: K-means

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

I.Linear regression

1.Định nghĩa: Phân tích hồi quy tuyến tính được sử dụng để dự đoán giá trị của một biến dựa trên giá trị của biến khác. Biến bạn muốn dự đoán được gọi là biến phụ thuộc. Biến bạn đang sử dụng để dự đoán giá trị của biến khác được gọi là biến độc lập.

2.Input: Cho tập dữ liệu huấn luyện gồm N mẫu. Mỗi mẫu là một cặp (x_i,y_i):

- x_i: vector đặc trưng
- y_i: giá trị của vector đặc trưng x

3.Output: Hàm tuyến tính có dạng $\mathbf{f(x_i) = wx_i + w_0}$

4.Phương pháp: Cần tìm hệ số w của hàm $f(x_i)$ sao cho trung bình sai số giữa y_i và $f(x_i)$ là nhỏ nhất. Nghĩa là, tìm w để hàm số sau đạt giá trị nhỏ nhất

+) Dạng của linear regression

- Trong phương trình $\mathbf{f(x)=w_1x_1+w_2x_2+w_3x_3+w_0}$, nếu chúng ta đặt:

  +) $\mathbf{w=[w_0,w_1,w_2,w_3]^T}$ = là vector (cột) hệ số cần phải tối ưu

  +) $\bar{x}$ **=[1,x_1,x_2,x_3])** là vector (hàng) dữ liệu đầu vào mở rộng. Số 1 ở đầu được thêm vào để phép tính đơn giản hơn và thuận tiện cho việc tính toán.

- Khi đó, phương trình $\mathbf{f(x)=w_1x_1+w_2x_2+w_3x_3+w_0}$ có thể được viết lại dưới dạng:

$$\mathbf{y} \approx \bar{x}\mathbf{w} = \hat{y}$$

- Chú ý rằng $\bar{x}$ là một vector hàng

+) Sai số dự đoán

- Chúng ta mong muốn rằng sự sai khác e giữa giá trị thực y và giá trị dự đoán $\hat{y}$ là nhỏ nhất. Nói cách khác, chúng ta muốn giá trị sau đây càng nhỏ càng tốt:

$$\frac{1}{2}e^2 = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}(y - \bar{x}w)^2$$

- Trong đó hệ số 1/2 là để thuận tiện cho việc tính toán (khi tính đạo hàm thì số 1/2 sẽ bị triệt tiêu). Chúng ta cần vì e^2 vì $e = y - \hat{y}$ có thể là một số âm,

việc nói e nhỏ nhất sẽ không đúng vì khi e = $-\infty$ là rất nhỏ nhưng sự sai lệch là rất lớn.

+) Hàm mất mát

- Điều tương tự xảy ra với tất cả các cặp (input, outcome) (x_i,y_i), i=1,2,…,N, với N là số lượng dữ liệu quan sát được. Điều chúng ta muốn, tổng sai số là nhỏ nhất, tương đương với việc tìm w để hàm số sau đạt giá trị nhỏ nhất:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \bar{\mathbf{x}}_i\mathbf{w})^2 \quad (2)$$

- Hàm số L(w) được gọi là **hàm mất mát** (loss function) của bài toán Linear Regression. Chúng ta luôn mong muốn rằng sự mất mát (sai số) là nhỏ nhất, điều đó đồng nghĩa với việc tìm vector hệ số w sao cho giá trị của hàm mất mát này càng nhỏ càng tốt. Giá trị của w làm cho hàm mất mát đạt giá trị nhỏ nhất được gọi là **điểm tối ưu** (optimal point), ký hiệu:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w})$$

- Trước khi đi tìm lời giải, chúng ta đơn giản hóa phép toán trong phương trình hàm mất mát (2).

  +) Đặt $y=[y_1;y_2;\ldots;y_N]$ là một vector cột chứa tất cả các output của training data;

  +) $\bar{X}=[\bar{x}_1;\bar{x}_2;\ldots;\bar{x}_N]$ là ma trận dữ liệu đầu vào (mở rộng) mà mỗi hàng của nó là một điểm dữ liệu.

- Khi đó hàm số mất mát L(w) được viết dưới dạng ma trận đơn giản hơn:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \bar{\mathbf{x}}_i\mathbf{w})^2$$

$$= \frac{1}{2}\|\mathbf{y} - \bar{\mathbf{X}}\mathbf{w}\|_2^2 \quad (3)$$

với $\|z\|_2$ là Euclidean norm (chuẩn Euclid, hay khoảng cách Euclid), nói cách khác $\|z\|_2^2$ là tổng của bình phương mỗi phần tử của vector z. Tới đây, ta đã có một dạng đơn giản của hàm mất mát được viết như phương trình (3).

+) Nghiệm của bài toán Linear Regression

- Cách phổ biến nhất để tìm nghiệm cho một bài toán tối ưu (chúng ta đã biết từ khi học cấp 3) là giải phương trình đạo hàm (gradient) bằng 0! Tất nhiên đó là khi việc tính đạo hàm và việc giải phương trình đạo hàm bằng 0 không quá phức tạp. Thật may mắn, với các mô hình tuyến tính, hai việc này là khả thi.

- Đạo hàm theo w của hàm mất mát là:

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} = \bar{\mathbf{X}}^T(\bar{\mathbf{X}}\mathbf{w} - \mathbf{y})$$

- Phương trình đạo hàm bằng 0 tương đương với:

$$\bar{\mathbf{X}}^T\bar{\mathbf{X}}\mathbf{w} = \bar{\mathbf{X}}^T\mathbf{y} \triangleq \mathbf{b} \quad (4)$$

- Nếu ma trận vuông $\mathbf{A} \triangleq \bar{\mathbf{X}}^T\bar{\mathbf{X}}$ khả nghịch (non-singular hay invertible) thì phương trình (4) có nghiệm duy nhất:

II.Hồi quy Ridge

1.Định nghĩa: là một kỹ thuật để phân tích dữ liệu hồi quy nhiều lần. Khi xảy ra đa cộng tuyến, các ước lượng bình phương nhỏ nhất là không chệch. Một mức độ chệch được thêm vào các ước tính hồi quy và kết quả là hồi quy sườn núi làm giảm các sai số tiêu chuẩn.

2.Input: Cho tập dữ liệu huấn luyện gồm N mẫu. Mỗi mẫu là một cặp (x_i,y_i):

- x_i: vector đặc trưng
- y_i: giá trị của vector đặc trưng x

3.Output: Hàm tuyến tính có dạng $\mathbf{f(x_i) = wx_i + w_0}$

4. Phương pháp:

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:

$$\mathbf{y = Xw + b}$$

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

$$\mathbf{L(w) = \frac{1}{2} * ||y - Xw - b||^2 + \alpha * ||w||_2^2}$$

Trong đó:

· ||.|| là norm L2 của một vector.

· $||w||\_2^2$ là bình phương của norm L2 của vector trọng số w.

· α là tham số điều chỉnh sự đóng góp của regularization.

Để tìm giá trị của w mà làm giảm giá trị của hàm mất mát, chúng ta cần tìm giá trị w mà làm cho đạo hàm riêng của hàm mất mát theo w bằng 0:

$$\mathbf{\partial L(w)/\partial w = 0}$$

Đạo hàm riêng này có thể tính được dễ dàng bằng phép tính đạo hàm thông thường. Khi giải phương trình ∂L(w)/∂w = 0, ta thu được:

$$\mathbf{(X^TX + \alpha I)w = X^Ty}$$

Trong đó:

· X^T là ma trận chuyển vị của X.

· I là ma trận đơn vị.

Giải phương trình trên, chúng ta có:

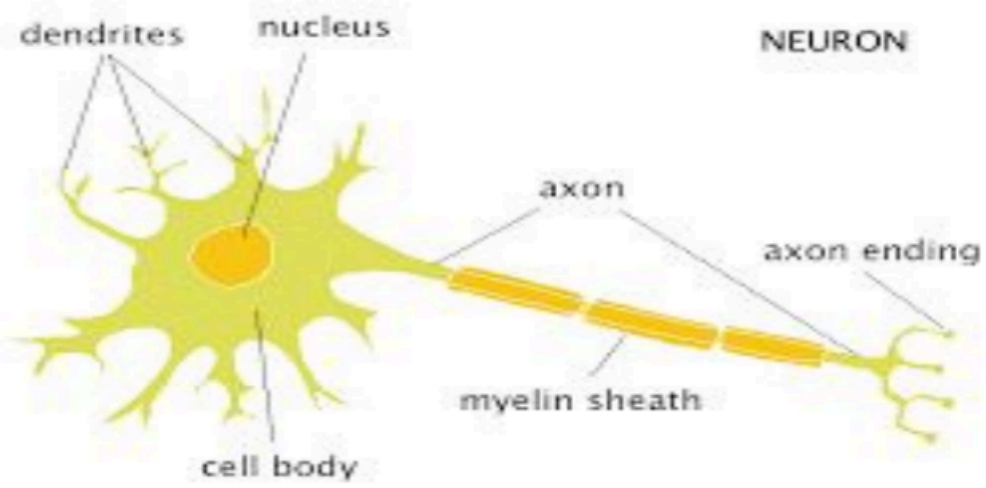
$$\mathbf{w = (X^TX + \alpha I)^{-1} X^Ty}$$

Quá trình ước lượng các trọng số w trong phương pháp Ridge được thực hiện bằng cách tính toán ma trận $\mathbf{(X^TX + \alpha I)^{-1}}$ và áp dụng nó vào phép nhân ma trận để tính toán w.

Khi giá trị của alpha tăng lên, phương pháp Ridge sẽ có xu hướng giảm kích thước của các trọng số, từ đó giảm khả năng overfitting và cải thiện tính tổng quát hóa của mô hình.

III.Phương pháp Neural Network

a) Định nghĩa



- Soma: thân nơron, tiếp nhận hoặc phát ra các xung động thần kinh
- Dendrites: dây thần kinh vào, đưa tín hiệu tới nơron
- Axon: đầu dây thần kinh ra, nối với dây thần kinh vào hoặc tới nhân tế bào của nơron khác thông qua khớp nối
- Synapse: khớp để kích hoạt hoặc kích thích thông tin

b) Đầu vào

Cho tập dữ liệu huấn luyện gồm N mẫu. Mỗi mẫu là một cặp (x_i,y_i):

- x_i: vector đặc trưng
- y_i: giá trị của vector đặc trưng x

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)

c) Đầu ra

$$y = f(Net) = f(\sum wi\ xi\ )$$

f được gọi là hàm truyền (transfer function) hay kích hoạt (activation function)

Bộ vector trọng số của các liên kết giữa các nơron (W) để hàm mất mát đạt giá trị tối ưu

d) Phương pháp

- 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, và vector bias b

IV, Mô hình học kết hợp Stacking

- **Input**

  + Tập Dữ Liệu Huấn Luyện

  + Bộ Mô Hình Cơ Bản (Base Models): Một tập hợp các mô hình học máy cơ bản được chọn để tạo thành các "đầu ra" cho mô hình stacking.

- **Output**

  + Một mô hình tổng hợp được tạo ra từ việc kết hợp đầu ra của các mô hình cơ bản để cải thiện hiệu suất dự đoán

- **Bài toán tối ưu của phương pháp**

  + Bài toán tối ưu trong Stacking thường liên quan đến việc tối thiểu hóa hàm mất mát trên tập dữ liệu huấn luyện, giống như các phương pháp học máy khác.

- **Cách giải bài toán tối ưu hoặc các bước thực hiện của thuật toán**

  + Xây dựng một số model (thường là khác loại) và một meta model (supervisor model)

  + Train những model này độc lập

+ Sau đó meta model sẽ học cách kết hợp kết quả dự báo của một số mô hình một cách tốt nhất

- **Các độ đo để đánh giá chất lượng mô hình dự đoán**

+ Với bài toán hồi quy:

NSE(ytest, ypred)

R2(ytest, ypred)

MAE(ytest, ypred)

RMSE(ytest, ypred)

Phần 2. Thực nghiệm

1. Mô tả bài toán

- Tên bài toán: Sử dụng một số phương pháp Hồi quy trong Học có giám sát (Linear Regression, Ridge Regression, Neural network) và mô hình Học kết hợp (Stacking) để dự đoán giá điện thoại di động

- Mục đích của bài toán: Giá di động phụ thuộc vào nhiều yếu tố khác nhau như độ phân giải, thương hiệu, kích thước, trọng lượng, chất lượng hình ảnh, RAM, pin và sức mạnh cpu. Trong đề tài này, chúng tôi muốn ước tính giá điện thoại di động bằng cách sử dụng các tính năng trên.

- Input: Giá cả, doanh thu, trọng lượng, độ phân giải, mật độ điểm ảnh, thông số kỹ thuật của điện thoại

- Output: Dự đoán giá cả điện thoại dựa trên thông số và tính năng

- Tóm tắt công việc thực hiện của bài toán: Từ dữ liệu đầu vào, dự đoán giá cả.

2. Mô tả tập dữ liệu của bài toán

- Số lượng mẫu dữ liệu: 161

- Dữ liệu gồm các trường thông tin:

+ **Thickness:** Độ dày

+ **Sale**: Số lượng bán ra

+ **Weight:** Trọng lượng

+ **Resoloution:** Độ phân giải

+ **PPI (Phone Pixel Density)**: Số điểm ảnh trên màn hình

+ **Cpu_core**: Vi xử lý

+ **Cpu_freq**: Tốc độ xung nhịp CPU

+ **Internal_Mem**: Bộ nhớ trong

+ **Ram:** bộ nhớ đệm

+ **RearCam:** Cam sau

+ **Front_Cam:** Cam trước

+ **Battery:** Pin

- Ma trận dữ liệu X [Thickness, Sale, Weight, Resoloution, PPI, Cpu_core, Cpu_freq, Internal_mem, Ram, RearCam, Front_Cam, battery]

- Ma trận dữ liệu Y [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

- Cách tiền xử lý dữ liệu: Dữ liệu không cần tiền xử lý

- Đọc file dữ liệu:

```
data = pd.read_csv(r"Cellphone.csv")

data.head()
```

- Chia tập dữ liệu với tỉ lệ 70:30

```
dtTrain, dtTest = train_test_split(data, test_size = 0.3, shuffle = False)

X_Train = dtTrain.iloc[:,2:]

y_Train = dtTrain.iloc[:,1]

X_Test = dtTest.iloc[:,2:]
```

```
y_Test = dtTest.iloc[:,1]
```

3.1 Linear Regression

- Cách lựa chọn các tham số của mỗi phương pháp để xây dựng mô hình:

- Nhóm code tay model Hồi quy tuyến tính:

class LinearRegression:

+ Hàm khởi tạo:

```
def __init__(self):
    self.w = None
    self.bias = None
```

+Hàm huấn luyện

```
def fit(self, X, Y):
    X_bar = np.concatenate((X, np.ones((X.shape[0], 1))), axis=1)
    params = np.linalg.inv(X_bar.T @ X_bar) @ X_bar.T @ Y
    self.w = params[:-1]
    self.bias = params[-1]
```

+Hàm dự đoán:

```
def predict(self, X):
    X_bar = np.concatenate((X, np.ones((X.shape[0], 1))), axis=1)
    predicted_Y = X_bar @ np.append(self.w, self.bias)
    return predicted_Y
```

- Gọi model Hồi quy tuyến tính để huấn luyện trên tập X_train, y_train:

```
from Model.LinearRegression import LinearRegression
model = LinearRegression()
```

```
model.fit(X_Train.values, y_Train.values)

print("Các nghiệm của bài toán là: ")

print("=> Weight: ", model.w)

print("=> bias: ", model.bias)
```

+ Kết quả tập nghiệm và bias thu được:

```
Các nghiệm của bài toán là:

=> Weight:  [-8.04664034e-02  1.02064106e+00
-1.02585905e+02  1.23160866e+00

  5.33949185e+01  7.73332862e+01  7.08879412e+00
1.00411155e+02

  1.66115553e+01 -6.40980207e+00  5.81574899e-02
-8.95090952e+01]

=> bias:  1895.3694205631527
```

- Dự đoán trên tập X_test và in ra độ chênh lệch:

```
print("Giá trị thực tế\t Giá trị dự đoán\t Độ lệch")

y_pred = np.array(model.predict(X_Test))

for i in range (0, len(y_pred)):

   print(y_Test.values[i],"\t\t",y_pred[i], "\t", y_Test.values[i] - y_pred[i])
```

3.2 Ridge

- Chọn alpha tốt nhất cho mô hình:

```
Ridge_model = Ridge()



# Thiết lập các giá trị alpha cần thử nghiệm

param_grid = {'alpha': [0.1, 0.5, 1.0, 2.0]}
```

```
# Sử dụng GridSearchCV để tìm giá trị alpha tối ưu

grid_search = GridSearchCV(Ridge_model, param_grid, cv=5)

grid_search.fit(X_Train, y_Train)

# Lấy giá trị alpha tối ưu

best_alpha = grid_search.best_params_['alpha']

print(best_alpha)
```

+Kết quả thu được:

best_alpha = 2.0

- Dùng best_alpha vừa tìm được để huấn luyện mô hình và dự đoán trên tập X_test:

```
model = Ridge(alpha = best_alpha)

model.fit(X_Train, y_Train)

y_pred = model.predict(X_Test)
```

3.3 Neural Network

-Sau khi thử từng tham số nhóm chúng em chọn được mô hình với những tham số tốt nhất cho tỉ lệ dự đoán đúng cao:

```
model = MLPRegressor(hidden_layer_sizes=(170, 200,)
,max_iter=5000, activation="identity", alpha=0.001)

model.fit(X_Train, y_Train)

y_pred = model.predict(X_Test)
```

3.4 Các độ đo

```
from sklearn.metrics import r2_score,mean_absolute_error,
mean_squared_error

y_mean = np.mean(y_Test)

# Tính NSE

nse = 1 - (np.sum((y_Test - y_pred) ** 2) / np.sum((y_Test - y_mean) ** 2))

#Tinh R2

r2 = r2_score(y_Test, y_pred)

# Tính MAE

mae = mean_absolute_error(y_Test, y_pred)

# Tính RMSE

rmse = np.sqrt(mean_squared_error(y_Test, y_pred))
```

3.5 Mô hình học kết hợp Stacking

-Nhóm chúng em sử dụng mô hình học kết hợp Stacking

- Code tay mô hình Stacking:

```
class StackingRegressor:

    def __init__(self, estimators = []):

        self.est = estimators

    def fit(self, X_Train, y_Train):

        for model in self.est:

    model.fit(X_Train.values, y_Train.values)

    def predict(self, X_Test):
```

```
y_pred = 0

for model in self.est:

 y_pred += model.predict(X_Test)

return y_pred/len(self.est)
```

- Gọi model StackingRegressor để huấn luyện trên tập X_train, y_train:

+Mảng arr gồm các model đã chọn tham số:

arr = [LinearRegression(), Ridge(alpha = 2.0), MLPRegressor(hidden_layer_sizes=(170, 200,) ,max_iter=5000, activation="identity", alpha=0.001)]

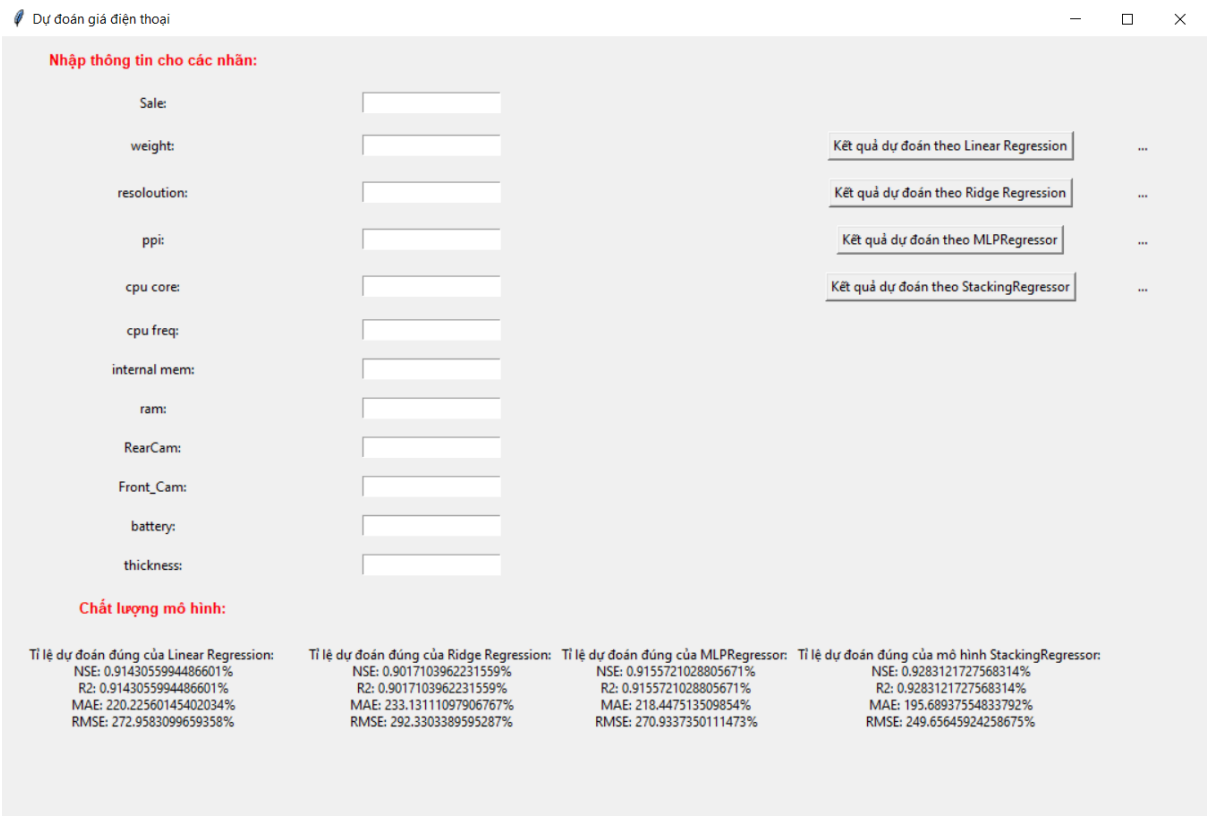
+ Huấn luyện và dự đoán:

```
model = StackingRegressor(estimators=arr)

model.fit(X_Train, y_Train)

y_pred = model.predict(X_Test)
```

3.6 Giao diện người dùng.

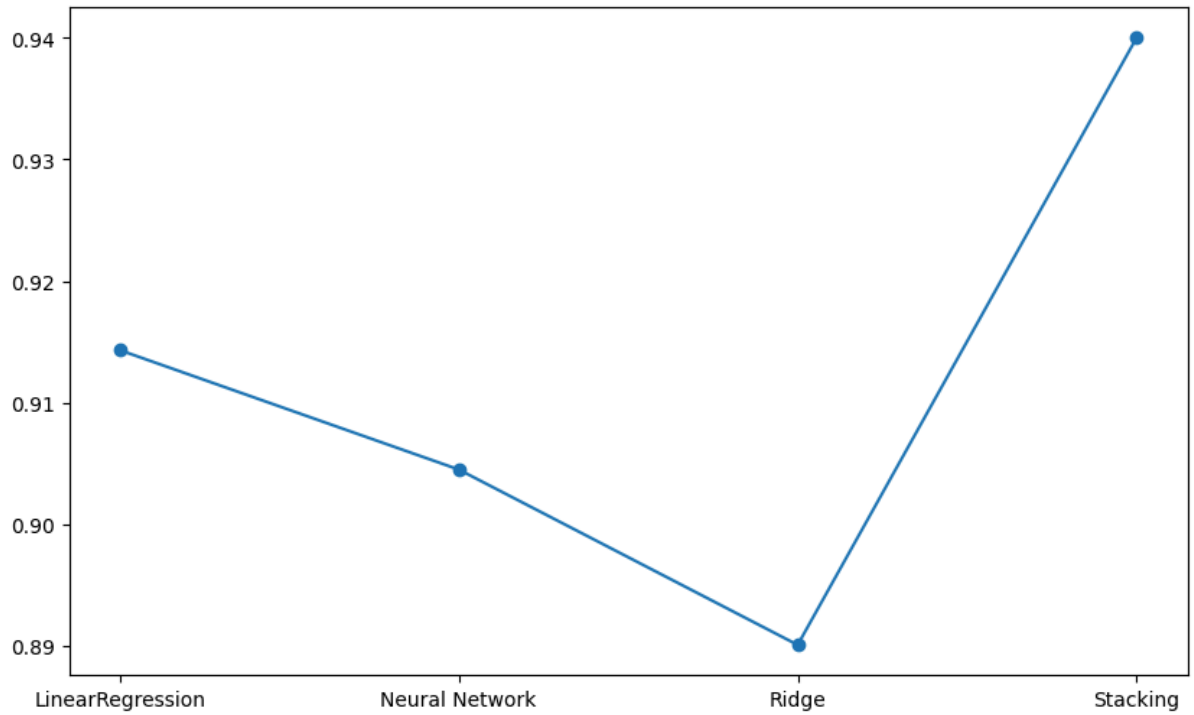


4. Phân tích kết quả của chương trình

Dùng các độ đo đánh giá chất lượng dự đoán của các mô hình. Từ đó, lựa chọn mô hình tốt nhất cho bài toán.

	Linear Regression	Ridge Regression	Neural network	Stacking Regressor
NSE	**0.9143055994489925**	**0.8900443813580798**	**0.9044778659976963**	**0.9400209053432642**
R2	**0.9143055994489925**	**0.8900443813580798**	**0.9044778659976963**	**0.9400209053432642**
MAE	**220.22560145360535**	**243.44457241259792**	**156.89679450882846**	**181.4194424078014**
RMSE	**272.95830996540644**	**309.19240056544993**	**211.93353419399216**	**228.36000876932087**

Biểu đồ so sánh độ đo đánh giá R2 của các mô hình:



Phần 3. Kết luận

- Sử dụng một số phương pháp Hồi quy trong Học có giám sát (Linear Regression, Ridge Regression, Neural network) và mô hình Học kết hợp (Stacking) để dự đoán giá điện thoại di động.
- Đánh giá chất lượng mô hình bằng các độ đo R2, NSE, MAE, RMSE.

Tài liệu tham khảo

- Bài giảng Machine Learning, TS. Nguyễn Thị Kim Ngân

- Sách Machine Learning cơ bản, Vũ Hữu Tiệp