Báo cáo đồ án Data Fitting

Đoàn Đặng Phương Nam - 22127280 - 22CLC02

I. Đánh giá mức độ hoàn thành

Chức năng	Tỷ lệ hoàn thành
Đọc và phân tách dữ liệu thành 2 tập features và labels	100%
Xây dựng các hàm hỗ trợ tính toán trên vector và ma trận	100%
Xây dựng mô hình sử dụng toàn bộ 11 đặc trưng đề bài cung cấp	100%
Xây dựng mô hình sử dụng duy nhất 1 đặc trưng cho kết quả tốt nhất	100%
Xuất kết quả mô hình và trực quan mô hình (nếu có thể)	100%
Thực nghiệm để tìm mô hình tốt nhất có thể	100%

II. Ý tưởng thực hiện và các hàm chức năng tương ứng

Bước 1: Đọc và phân tách dữ liệu

Ở bước này, em sử dụng thư viện **Pandas**, cụ thể là hàm **read_csv** để đọc dữ liệu từ file *wine.csv*. Từ dữ liệu vừa đọc được, em tiến hành loại bỏ tên các cột dữ liệu, ép kiểu về kiểu **list** trong **Python** rồi mới phân tách dữ liệu thành 2 thành phần:

- **features**: Có được bằng cách bỏ đi cột cuối cùng của bộ dữ liệu, tạm gọi là ma trận X, căn cứ vào dữ liệu đề bài, $x \in \mathbb{R}^{1199\times 11}$
- **labels**: Cột cuối cùng của bộ dữ liệu, tạm gọi là vector y, căn cứ vào dữ liệu đề bài, $y \in \mathbb{R}^{1199}$

Bước 2: Xây dựng các hàm hỗ trợ

Ở bước này, em sẽ tự xây dựng các hàm hỗ trợ, chủ yếu là dành cho các thao tác tính toán đối với vector và ma trận, cụ thể, trong bài làm của em có những hàm hỗ trợ sau:

- **checkMatrix(matrix)**: Hàm này nhằm kiểm tra xem ma trận đầu vào có hợp lệ hay không (chiều dài của mỗi hàng là như nhau)
- **transposeMatrix(matrix)**: Hàm dùng để chuyển vị ma trận
- **L2_Norm(vector)**: Hàm dùng để tính Euclid norm của 1 vector bất kỳ
- **multiplyTwoMatrices(matrix1, matrix2)**: Hàm dùng để tính tích của 2 ma trận
- **addTwoMatrices(matrix1, matrix2)**: Hàm dùng để tính tổng của 2 ma trận
- **multiplyMatrixByScalar(matrix, scalar)**: Hàm dùng để nhân ma trận với 1 hằng số nào đó
- **Inverse(squareMatrix)**: Hàm dùng để tính nghịch đảo của 1 ma trận vuông bất kỳ.

Ngoài ra còn có các hàm con khác nằm lồng trong những hàm trên nhưng em sẽ không đề cập đến ở trong báo cáo vì chúng không được sử dụng trong các chức năng (bước) tiếp theo.

Bước 3: Xây dựng mô hình sử dụng cả 11 đặc trưng của dữ liệu

Dựa trên **giải thuật OLS** đã được học trên lớp Lý thuyết, ý tưởng của em cho bước này sẽ là tìm bộ tham số $\theta = (\theta_0, \theta_1, \theta_2, ..., \theta_{11})$ cho mô hình

$$\hat{y} = \theta_0 + \sum_{i=1}^{11} \theta_i x_i$$

sao cho $\text{RSS} = \sum_x (y - \hat{y})^2$ nhỏ nhất

Để làm được điều này, em đã xây dựng 1 hàm tên là **model_build_full_features**, nhận vào 2 tham số là **features** và **labels**, ở đây, em sẽ:

- Thành lập ma trận X bằng cách lấy tham số **features** rồi gán thêm số 1 vào đầu mỗi hàng của ma trận **features** *(số 1 tương trựng cho tham số θ_0 trong mô hình)*
- Tính vector tham số $\theta = (X^T X)^{-1} X^T y$
- Tính RSS $= \|y - X\theta\|^2$

Bước 4: Xây dựng mô hình sử dụng duy nhất 1 đặc trưng cho kết quả tốt nhất

Cũng tương tự như **Bước 3**, ý tưởng của em ở bước này là sẽ dựa trên **thuật toán OLS**, duyệt qua lần lượt 11 đặc trưng rút trích từ dữ liệu. Với mỗi đặc trưng thứ i *(ứng với biến x_i)*, em sẽ tìm tham số $\theta_i = (a, b)$ của mô hình

$$\hat{y_i} = a + bx_i$$

để khớp với dữ liệu đề bài nhất.

Dựa trên ý tưởng đó, em xây dựng 1 hàm **model_build_each_feature** nhận vào 2 tham số **features** và **labels**, ở đây, em thiết lập 2 **list** lần lượt phục vụ cho việc lưu **tham số** và lưu **giá trị RSS** của từng mô hình, rồi mới duyệt qua từng đặc trưng một. Ở đặc trưng thứ i:

- Thành lập ma trận X_i bằng cách thành lập 1 cột gồm 1199 số 1 rồi gộp với cột thứ i của **features**
- Tính vector tham số $\theta_i = (X_i^T X_i)^{-1} X_i^T y$ rồi lưu vào danh sách **tham số**
- Tính RSS_i $= \|y - X_i\theta_i\|^2$ rồi lưu vào danh sách **giá trị RSS**

Bước 5: Xuất kết quả mô hình và trực quan hoá mô hình (nếu có thể)

Sau khi đã có kết quả mô hình từ **hai bước 3 và 4**

- Với mô hình ở **Bước 3**, em sẽ xuất ra kết quả dưới dạng $y = \theta_0 + \theta_1 x_1 + ... + \theta_{11} x_{11}$, gán các tham số $\theta_0, \theta_1, ..., \theta_{11}$ bởi kết quả trả về từ **Bước 3**. Cùng với đó, em sẽ xuất ra **giá trị RSS** của mô hình, tạo cơ sở để so sánh sự "tốt hơn" giữa mô hình so với mô hình ở **yêu cầu c**. Em chỉ có thể in ra kết quả như vậy vì đây là mô hình nhiều chiều, em không có giải pháp nào để trực quan nó.
- Với kết quả có được từ **Bước 4**, em sẽ duyệt qua kết quả tham số và giá trị RSS ở mô hình ứng với từng đặc trưng, rồi sử dụng thư viện **Matplotlib** để **plot** các điểm cùng với đường thẳng khớp dữ liệu nhất lên bức ảnh 2D kích thước 15 x 6

Bước 6: Thực nghiệm để tìm ra mô hình tốt nhất có thể

Ở bước này, ý tưởng của em như sau:

- Đầu tiên, em xét tất cả mô hình có dạng

$$y = \theta_0 + \sum_i \theta_i x_i$$

trong đó các x_i tạo thành 1 tập là tập con khác rỗng của $X = \{x_1, x_2, ..., x_{11}\}$, như vậy, sẽ có tất cả 2047 mô hình khác nhau được em xem xét, mục tiêu là tìm ra mô hình cho ra **RSS** nhỏ nhất sau khi đã khớp dữ liệu. Kết quả sau khi em thực nghiệm cho thấy mô hình sử dụng toàn bộ 11 đặc trưng sẽ cho ra **RSS** nhỏ nhất.

- Tiếp theo, em sử dụng toàn bộ 11 đặc trưng và dựa trên đó bắt đầu thêm vào các hàm cơ bản như $\frac{1}{x^a}$, x^a, a^x, $\sin ax$, $\cos ax$, cùng với đó là các hạng tử trong biểu diễn $\left(\sum_{i=1}^{11} x_i^a\right)^2$, $\left(\sum_{i=1}^{11} x_i^a\right)^3$, $\left(\sum_{i=1}^{11} x_i^a\right)^4$, lúc này em tạo được 1 hàm có dạng tổng quát như sau:

$$y = \theta_0 + \frac{\theta_1}{x_i^a} + \sum_{i=3}^{11} \frac{\theta_i}{x_i^a} + \sum_{i=1}^{11} \varphi_i x_i^b + \sum_{i=1}^{11} \gamma_i c^{x_i} + \sum_{i=1}^{11} \omega_i \sin(dx_i) + \sum_{i=1}^{11} \lambda_i \cos(ex_i) + \sum_{i=1}^{11} \sum_{j=i}^{11} \mu_{ij} (x_i x_j)^f + \sum_{i=1}^{11} \sum_{j=i+1}^{11} \sum_{k=j+1}^{11} \mu_{ijk} (x_i x_j x_k)^g + \sum_{i=1}^{11} \sum_{j=i+1}^{11} \sum_{k=j+1}^{11} \sum_{r=k+1}^{11} \mu_{ijkr} (x_i x_j x_k x_r)^h$$

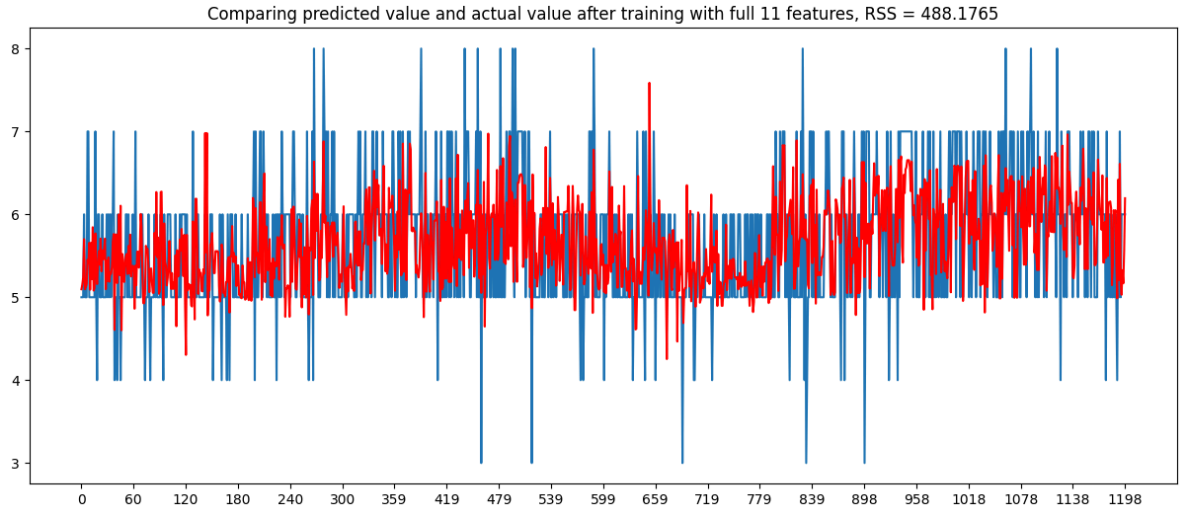
- Sau khi thử với các bộ giá trị khác nhau của a, b, c, d, e, f, trong đó em đã giới hạn khoảng thử nghiệm:
  - $a, b, c, f, g, h \in (0, 3]$ và là các số có 2 chữ số thập phân
  - $d, e \in (0, 10000]$ và là các số tự nhiên

  Cuối cùng em đã tìm ra bộ số (a, b, c, d, e, f, g, h) cho ra **giá trị RSS** tốt nhất, đó là:

$$\boxed{a = 0.53, b = 0.05, c = 2.75, d = 8741, e = 4370, f = 1.76, g = 1.23, h = 1}$$

III. Hình ảnh kết quả

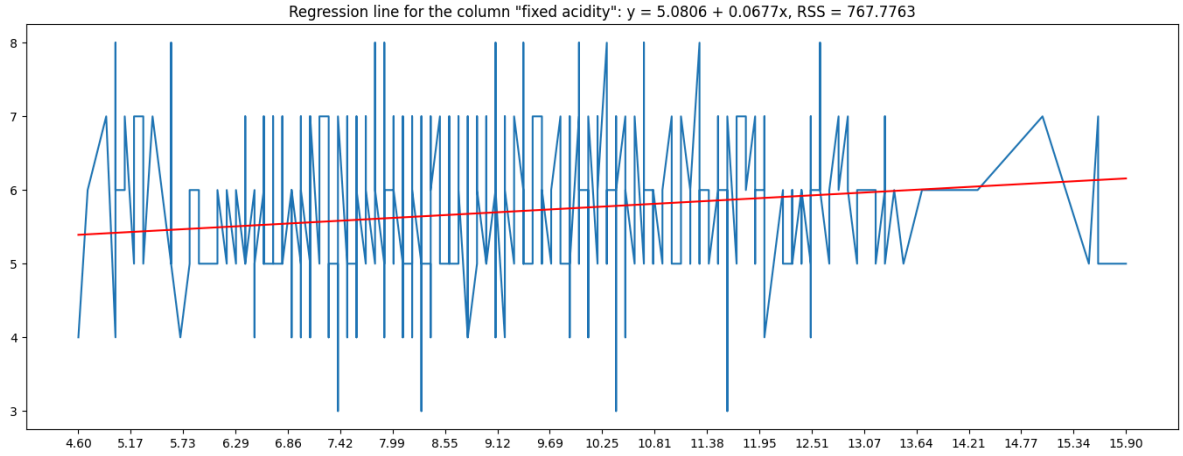
1. Mô hình sử dụng cả 11 đặc trưng của dữ liệu



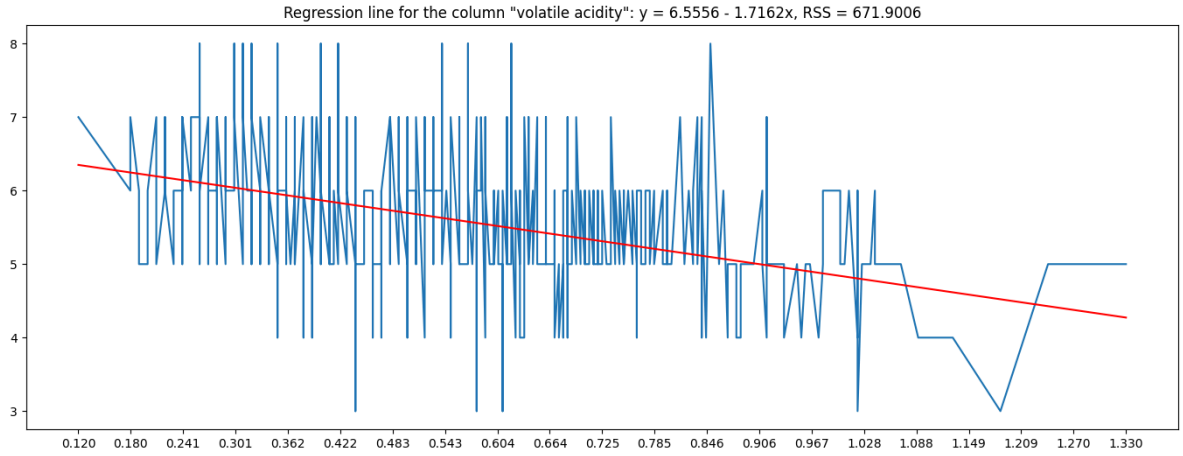
Ảnh 1: Kết quả mô hình sử dụng cả 11 đặc trưng của dữ liệu

2. Các mô hình sử dụng chỉ 1 đặc trưng của dữ liệu

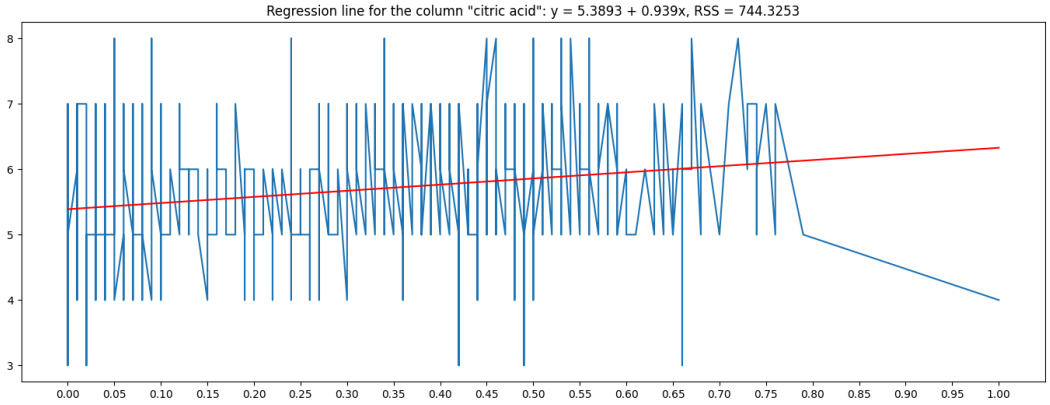
Sau khi tính **giá trị RSS** của cả 11 mô hình, em nhận thấy mô hình sử dụng thuộc tính *alcohol* cho giá trị RSS thấp nhất, đồng nghĩa với việc mô hình đó là **tốt nhất** trong 11 mô hình được xét



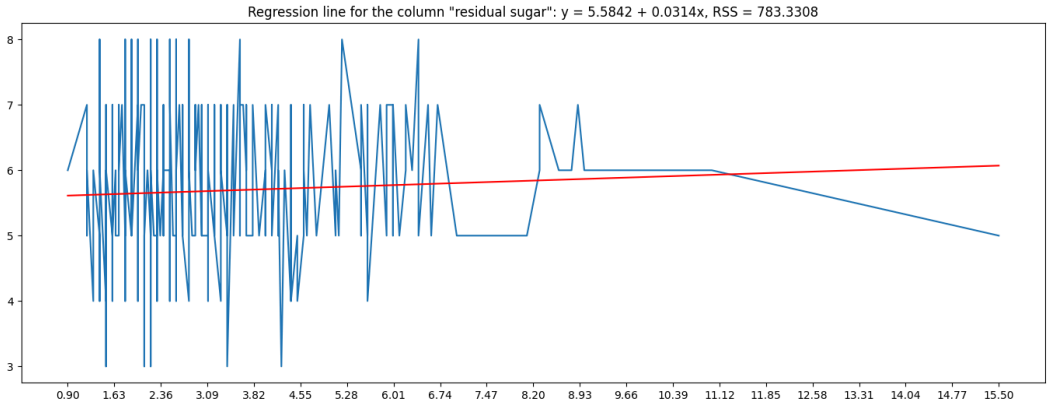
Ảnh 2: Kết quả mô hình sử dụng thuộc tính *fixed acidity*



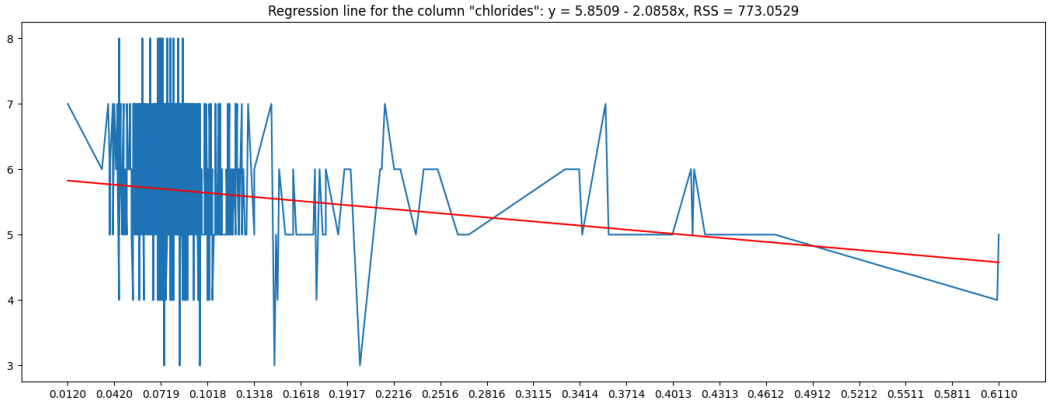
Ảnh 3: Kết quả mô hình sử dụng thuộc tính *volatile acidity*



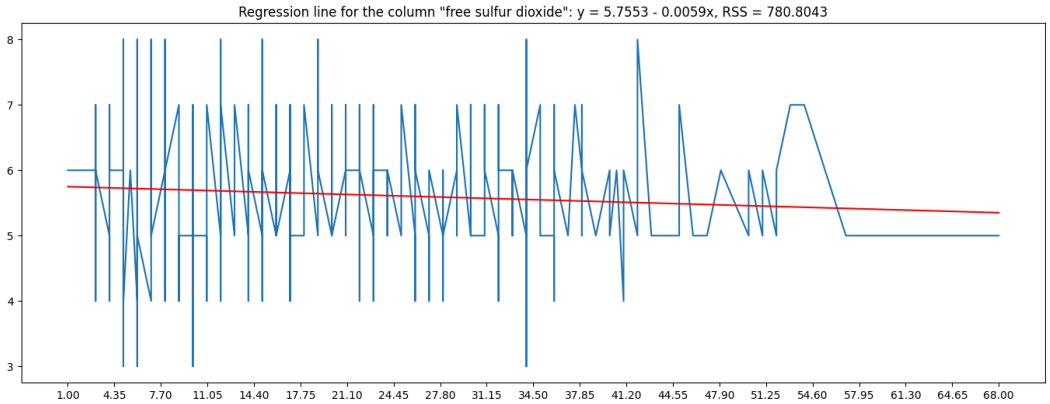
Ảnh 4: Kết quả mô hình sử dụng thuộc tính *citric acid*



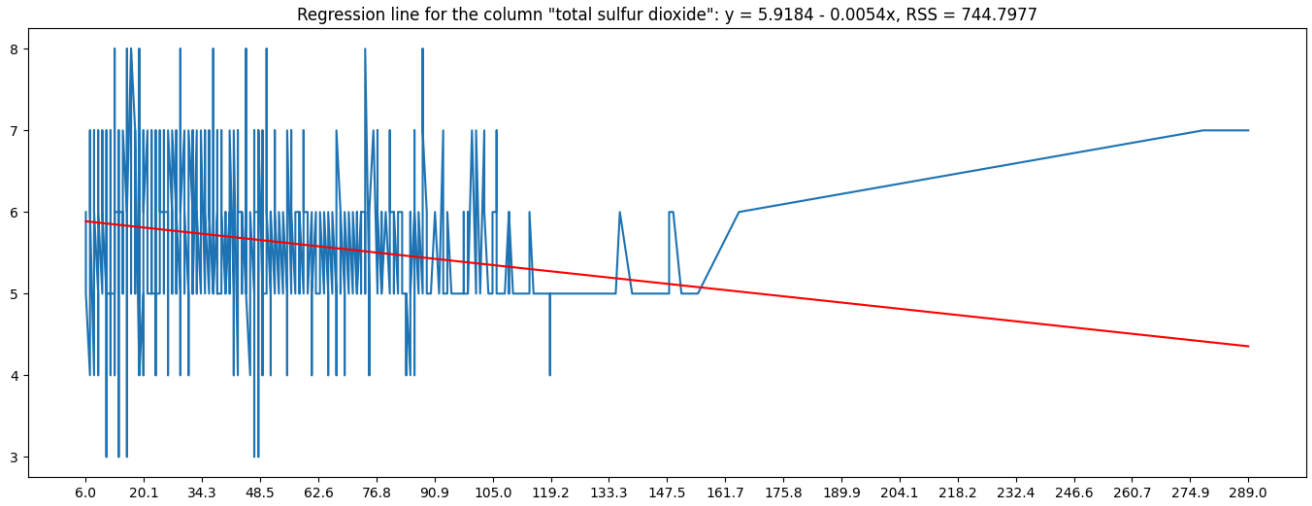
Ảnh 5: Kết quả mô hình sử dụng thuộc tính *residual sugar*



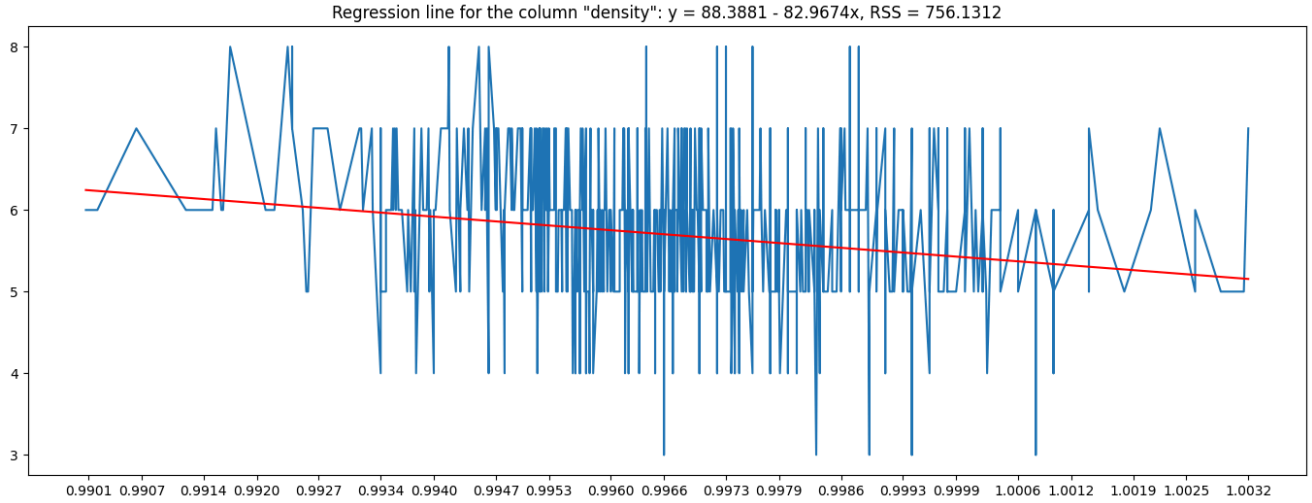
Ảnh 6: Kết quả mô hình sử dụng thuộc tính *chlorides*



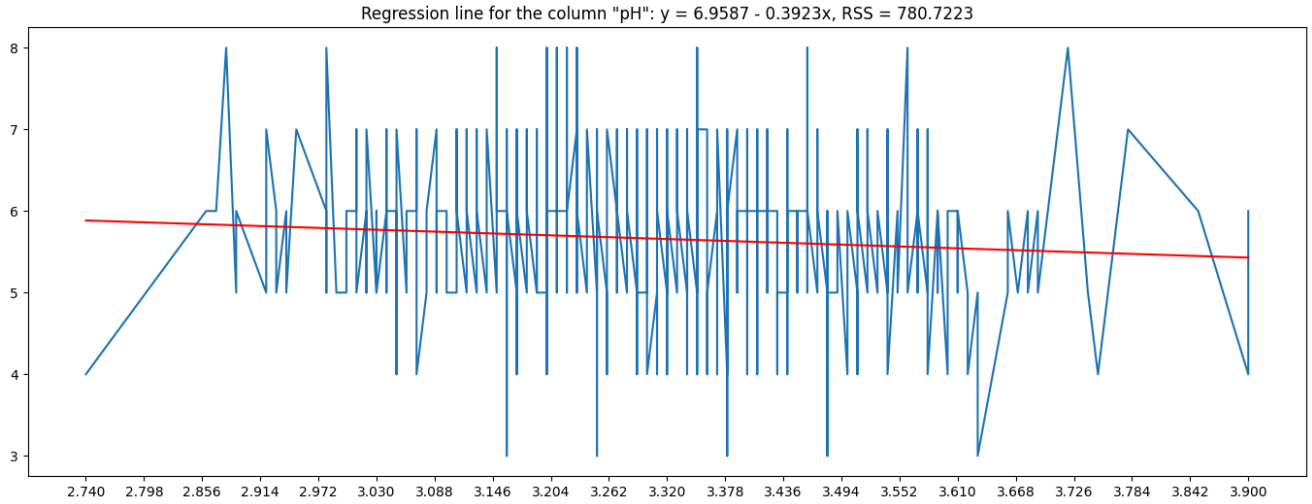
Ảnh 7: Kết quả mô hình sử dụng thuộc tính *free sulfur dioxide*



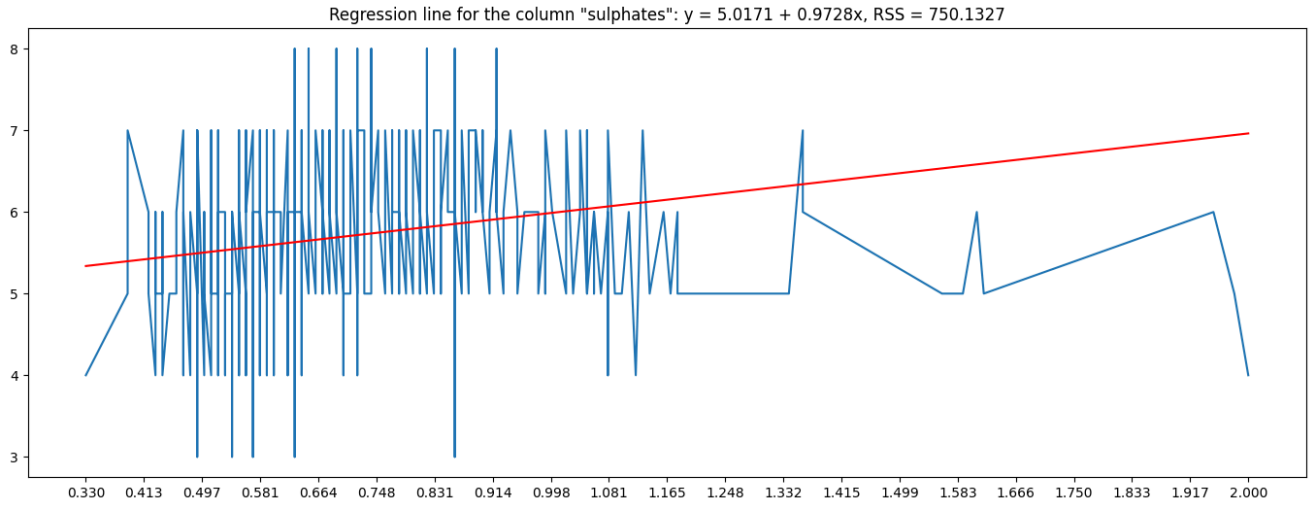
Ảnh 8: Kết quả mô hình sử dụng thuộc tính *total sulfur dioxide*



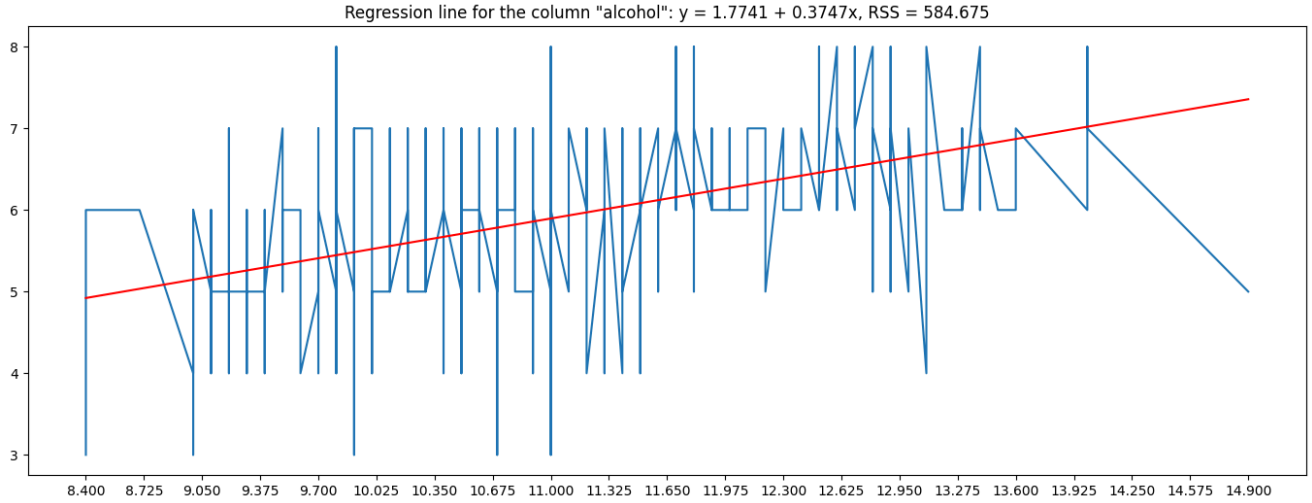
Ảnh 9: Kết quả mô hình sử dụng thuộc tính *density*



Ảnh 10: Kết quả mô hình sử dụng thuộc tính *pH*



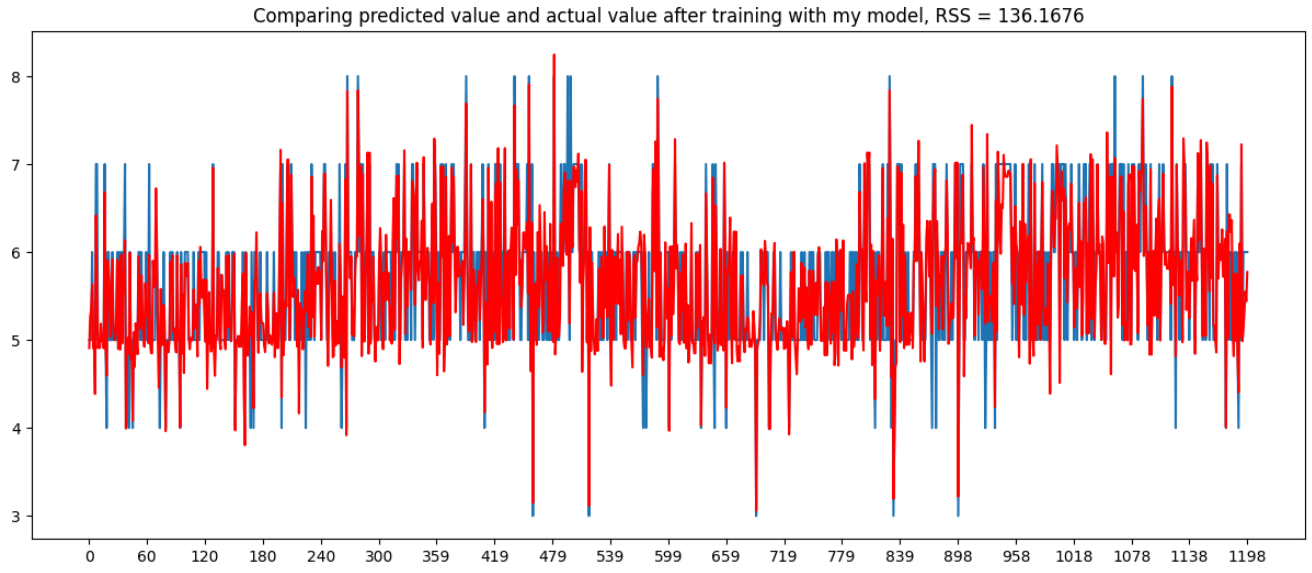
Ảnh 11: Kết quả mô hình sử dụng thuộc tính *sulphates*



Ảnh 12: Kết quả mô hình sử dụng thuộc tính *alcohol*

3. Mô hình tự xây dựng

Sau khi đo đạc cẩn thận, giá trị RSS cuối cùng mà em đạt được là RSS = 136.1676, thấp hơn tương đối nhiều so với các mô hình ở 2 yêu cầu a và b



Ảnh 13: Kết quả mô hình tự xây