TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

----------oOo----------

TOÁN ỨNG DỤNG THỐNG KÊ

Lab 3: Linear Regression

Họ tên : Lê Nguyên Bình Nam
MSSV: 20127567

I. Mục tiêu và yêu cầu đồ án:

- Xây dựng mô hình hồi quy tuyến tính:
  a. Sử dụng toàn bộ 11 đặc trưng của tập dữ liệu.
  b. Sử dụng duy nhất 1 đặc trưng trong 11 đặc trưng và xuất ra thuộc tính tốt nhất.
  c. Xây dựng mô hình của riêng bản thân .

II. Các bước thực hiện:

1. Mô tả dữ liệu:

- Trước khi thực hiện yêu cầu ta cần biết dữ liệu hiện có gì cũng như dữ liệu có bị thiết xót hay sai gì không. Đây là kết quả mô tả dữ liệu:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1194	7.0	0.745	0.12	1.8	0.114	15.0	64	0.99588	3.22	0.59	9.5	6
1195	6.2	0.430	0.22	1.8	0.078	21.0	56	0.99633	3.52	0.60	9.5	6
1196	7.9	0.580	0.23	2.3	0.076	23.0	94	0.99686	3.21	0.58	9.5	6
1197	7.7	0.570	0.21	1.5	0.069	4.0	9	0.99458	3.16	0.54	9.8	6
1198	7.7	0.260	0.26	2.0	0.052	19.0	77	0.99510	3.15	0.79	10.9	6

1199 rows × 12 columns

- Dữ liệu gồn có 1198 điểm dữ liệu và 11 thuộc tính và 1 kết quả (quality)
- Thông tin dữ liệu của từng cột(count, mean, std, min, …):

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	1199.000000	1199.000000	1199.000000	1199.000000	1199.000000	1199.000000	1199.000000	1199.000000	1199.000000	1199.000000	1199.000000	1199.000000
mean	8.625271	0.519133	0.293353	2.564470	0.089266	15.242702	46.884070	0.997059	3.298582	0.665738	10.383069	5.664721
std	1.781795	0.179208	0.196751	1.264441	0.048310	10.210406	33.949177	0.001878	0.156161	0.175921	1.091891	0.809593
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000
25%	7.300000	0.390000	0.120000	1.900000	0.071000	7.000000	21.000000	0.996000	3.195000	0.560000	9.500000	5.000000
50%	8.300000	0.500000	0.290000	2.200000	0.080000	13.000000	38.000000	0.997020	3.300000	0.620000	10.000000	6.000000
75%	9.600000	0.630000	0.450000	2.700000	0.092000	21.000000	63.000000	0.998175	3.390000	0.735000	11.000000	6.000000
max	15.900000	1.330000	1.000000	15.500000	0.611000	68.000000	289.000000	1.003200	3.900000	2.000000	14.900000	8.000000

- Thông tin kiểu dữ liệu của từng cột và cột có chứa giá trị null hay không:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1199 entries, 0 to 1198
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1199 non-null   float64
 1   volatile acidity      1199 non-null   float64
 2   citric acid           1199 non-null   float64
 3   residual sugar        1199 non-null   float64
 4   chlorides             1199 non-null   float64
 5   free sulfur dioxide   1199 non-null   float64
 6   total sulfur dioxide  1199 non-null   int64
 7   density               1199 non-null   float64
 8   pH                    1199 non-null   float64
 9   sulphates             1199 non-null   float64
 10  alcohol               1199 non-null   float64
 11  quality               1199 non-null   int64
dtypes: float64(10), int64(2)
memory usage: 112.5 KB
```

- Tất cả các cột đều mang giá trị số và không có cột nào mang giá trị null

2. Câu a: Xây dựng mô hình với 11 thuộc tính:

```
y=theta_1x_1+theta_2x_2+...+theta_11x_11
```

   - Mô tả thuật toán:
     Sử dụng phương pháp bình phương nhỏ nhất (OLS):
     - Ta cần xây dựng các ma trận X và Y:

$$\mathbf{X} = \begin{bmatrix} f_1(x_1) & f_2(x_1) & \cdots & f_k(x_1) \\ f_1(x_2) & f_2(x_2) & \cdots & f_k(x_2) \\ \vdots & \vdots & & \vdots \\ f_1(x_N) & f_2(x_N) & \cdots & f_k(x_N) \end{bmatrix} \in \mathbb{R}^{N\times k}, \mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^{N\times 1}$$

     - Ta sẽ được

$$\mathbf{Y}-\mathbf{Xb}=\begin{bmatrix} y_1-(b_1f_1(x_1)+\cdots+b_kf_k(x_1)) \\ \vdots \\ y_i-(b_1f_1(x_i)+\cdots+b_kf_k(x_i)) \\ \vdots \\ y_N-(b_1f_1(x_N)+\cdots+b_kf_k(x_N)) \end{bmatrix}$$

- Với kết quả trên thì mô hình dự đoán

  $$f(x)=b_1f_1(x)+b_2f_2(x)+\cdots+b_kf_k(x),$$

  Sẽ trở thành

  $$RSS(\mathbf{b})=\|\mathbf{Y}-\mathbf{Xb}\|^2.$$

- Vì vậy ta cần tìm min của RSS, qua các cách chứng mình thì ta được:

  $$(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}=\underset{\mathbf{b}\in\mathbb{R}^k}{\arg\min}RSS(\mathbf{b}).$$

- Áp dụng vào bài toán câu a, ta xây dựng hàm để có thể lấy được các ma trận X và Y.
  - Ta xây dựng ma trận X là ma trận các thuộc tính trích ra từ data, ma trận Y là ma trận kết quả chất lượng.
  - Áp dụng mô hình thuật toán trên, tính được ma trận theta là ma trận các hệ số của model tuyến tính cần tìm.

```
Model is:
y = [0.00592516]x1 + [-1.10803754]x2 + [-0.26304628]x3 + [0.01532228]x4 + [-1.73050274]x5 +
[0.00380142]x6 + [-0.003899]x7 + [4.33858768]x8 + [-0.45853548]x9 + [0.72971866]x10 +
[0.30885865]x11
```

- Sau đấy ta tính sai số của model:

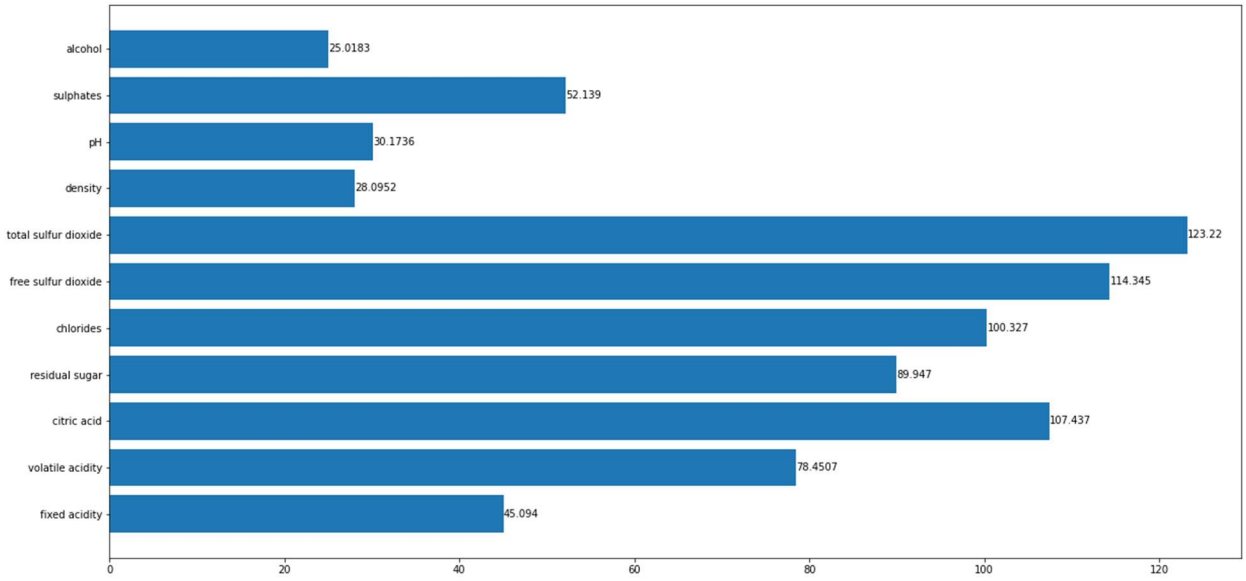
```
r = np.linalg.norm(x@thetafull - y)
r
✓ 0.1s
22.12434596534916
```

➔Với mô hình này cho ra sai số khá cao.

3. Câu b: Xây dựng mô hình tuyến tính với từng thuộc tính, đưa ra mô hình tuyến tính sử dụng 1 thuộc tính tốt nhất.

```
y=theta_ix_i (dùng mô hình lần lượt cho từng đặc trưng).
```

- Cách làm: Ta cũng sẽ áp dụng model như ở câu a, ma trận Y thì vẫn sẽ là cột chất lượng của data, nhưng ma trận X thì chỉ có 1 thuộc tính, ta sử dụng vòng lặp, ứng với mỗi lần lặp sẽ chạy model ứng với từng thuộc tính. Sau đó ta tính giá trị vector sai số và ghi nhận lại. Ta biểu diễn các giá trị đó lên biểu đồ.



➔Với kết quả của biểu đồ thì hàm tuyến tính sử dụng thuộc tính alcohol là cho ra kết quả sai số thấp nhất

4. Câu c: Xây dựng mô hình tuyến tính tốt nhất:

Bước 1 : Trước khi thực hiện ta sẽ chia bộ dữ liệu ra làm 2 phần là phần train và test với bộ test chiếm 20%, tham số random_state = 40 để tránh trường hợp dữ liệu train với test không có tính ngẫu nhiên.

```
X = df.drop("quality", axis=1)
y = df["quality"]
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=.2, random_state=40)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

- Kích thước của bộ train và test.

```
((959, 11), (240, 11), (959,), (240,))
```

Bước 2 : Ta tiến hành loại bỏ những cột mang giá trị hằng, tức là những cột có phương sai bằng 0.

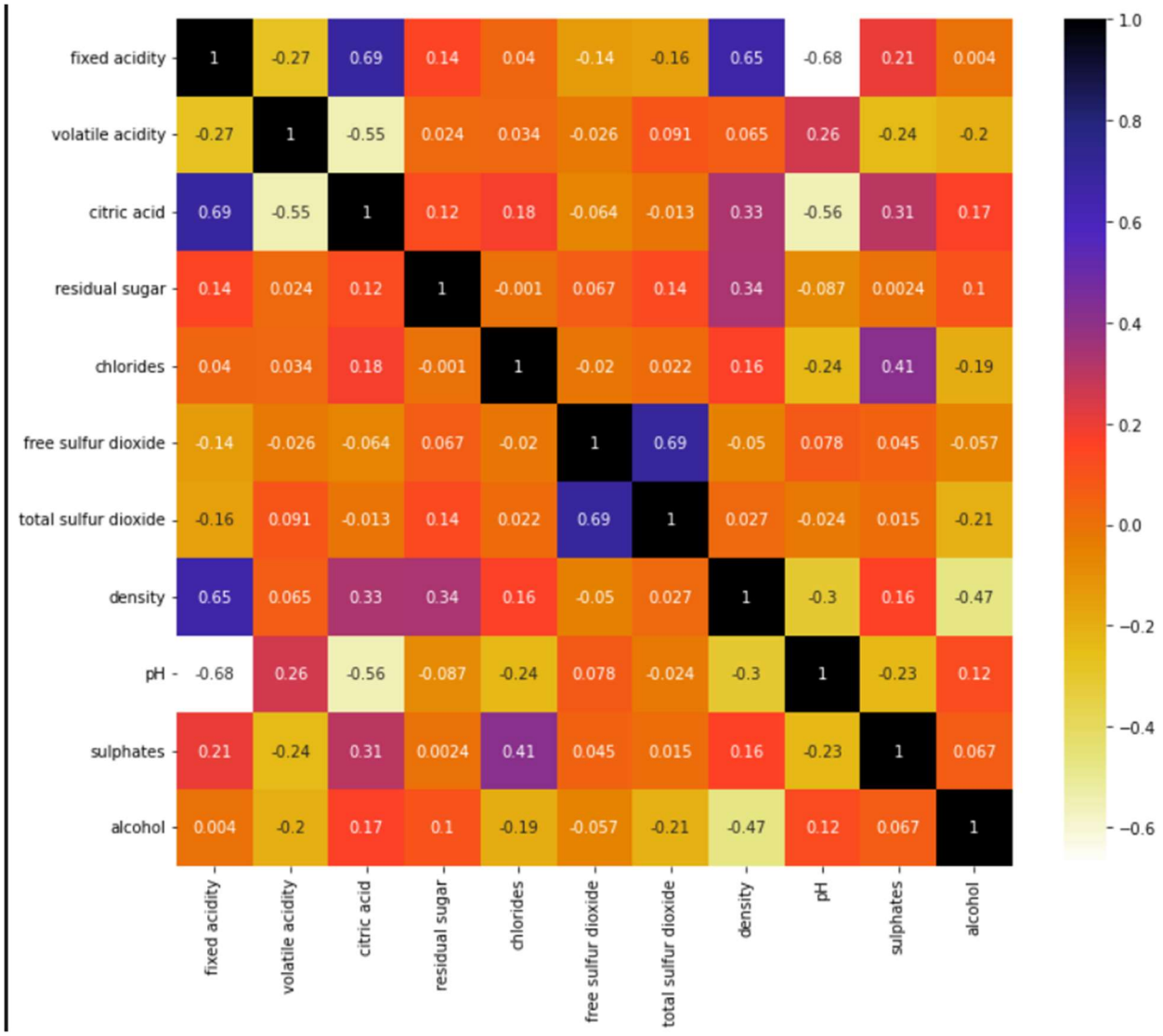
```
var_there = VarianceThreshold(threshold = 0)
var_there.fit(X_train)
var_there.get_support()
```

```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True])
```

➔Kết quả cho ra tất cả các cột đều không mang giá trị hằng.

Bước 3: Loại bỏ các cột mang tính tương quan cao:

- Ở bước này ta sẽ hiển thị mối tương qun của từng thuộc tính trong bộ dữ liệu test so với tất cả các thuộc tính còn lại. Ta kiến hành loại bỏ những thuộc tính mà có giá trị tuyệt đối lớn hơn 0.8 (nghĩa là 80% dữ liệu của thuộc tính này sẽ tỉ lệ với thuộc tính kia) bằng cách sử dụng hệ số tương quan (Pearson Correlation) và biểu diễn chúng bằng biểu đồ.



- Ta xây dựng hàm để loại bỏ những cột mang tính tương qua cao. Hàm nhận vào dataset và hệ số(trong TH này là 0.8). Hàm trả về các cột cần loại bỏ.

```python
def correlation(dataset, threshold):
    col_corr = set()
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold:
                colname = corr_matrix.columns[i]
                col_corr.add(colname)
    return col_corr


corr_feature = correlation(X_train, 0.8)
len(set(corr_feature))
```

✓ 0.4s Python

```
0
```

➔Output hàm là dãy các tính chất nhưng chiều dài bằng 0, Suy ra không có tính chất nào phụ thuộc lẫn nhau.

Bước 4: Feature Selection:

- Trong câu này sử dụng Lasso Regresssion(hồi quy Lasso):

- Trong hồi quy tuyến tính thì norm chuẩn bậc 1 đo lường sai số tuyệt đối giữa giá trị dự báo và giá trị thực tế.

$$L_1 = \|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|$$

Norm chuẩn bậc 1

- Tuy nhiên nó ít được sử dụng hơn so với chuẩn bậc 2 như là một loss function vì giá trị của nó có đạo hàm không liên tục. Điều này dẫn tới việc huấn luyện mô hình không ổn định.

$$L_p = \|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

Norm chuẩn bậc p với p>=1

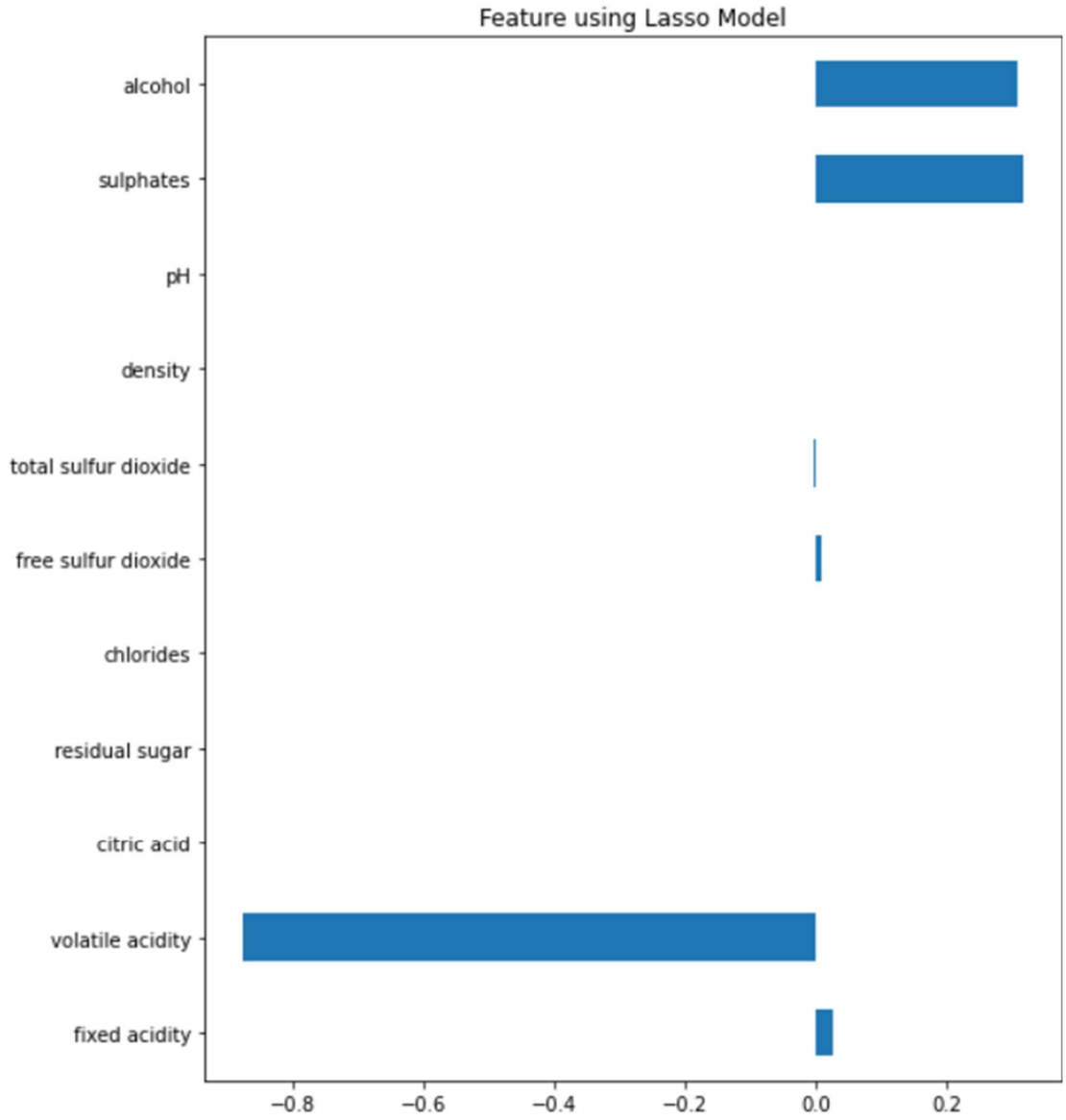
- Trong hồi quy Lasso, thay vì sử dụng norm chuẩn bậc 2 như đa số thì ở đây ta sử dụng norm chuẩn bậc 1 với sự thay đổi:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{N}\|\bar{\mathbf{X}}\mathbf{w} - \mathbf{y}\|_2^2 + \underbrace{\alpha\|\mathbf{w}\|_1}_{\text{regularization term}}$$

- Khi tiến hank hồi qui mô hình Lasso trên một bộ dữ liệu mà có các biến đầu vào đa cộng tuyến (multicollinear) thì mô hình hồi qui Lasso sẽ có xu hướng lựa chọn ra một biến trong nhóm các biến đa cộng tuyến và bỏ qua những biến còn lại. Trong khi ở mô hình hồi qui tuyến tính thông thường thì có xu hướng lựa chọn tất cả các biến.
- Việc thêm vào hệ số alpha dể gia tăng sự kiểm soát:
    - Với alpha = 0, thành phần điều chuẩn bị giảm và đưa về hồi quy tuyến tính như thông thường.
    - Với alpha 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 alpha lớn, chúng ta muốn gia tăng mức độ kiểm soát lên độ lớn của các hệ số ước lượng.
- Áp dụng mô hình này, nếu chúng ta giảm thiểu hàm chi phí, hồi quy Lasso sẽ tự động chọn những tính năng hữu ích và loại bỏ đi những tính năng dư thừa.

```
reg = LassoCV(cv=5, random_state=40)
reg.fit(X_train, y_train)
coef = pd.Series(reg.coef_, index = X.columns)
```
✓ 0.1s

- Đầu tiên khởi tạo mô hình, truyền vào tham số cv =5 (cross-validation generator or iterable, và sử dụng hệ số mặc định bằng 5) và random state = 40 (bước nhảy ngẫu nhiên với từng điểm dữ liệu)
- Lưu trị số của từng thuộc tính và biểu diễn chúng.



- Nhìn vào biểu đồ thì thấy được hồi Lasso chọn 6 thuộc tính có trị số khác 0 và loại bỏ đi 5 thuộc tính còn lại.

```
print("Lasso choosed " + str(sum(coef != 0)) + " variables and eliminated the other " +  str(sum(coef == 0)) + " variables")
✓ 0.6s
Lasso choosed 6 variables and eliminated the other 5 variables
```

- Từ những thuộc tính ta chọn được thì ta sẽ sử dụng phương pháp OLS để tìm ra hàm tuyến tính, dây là hệ số của hàm theta.

```
array([[ 0.31900264],
       [ 1.58685998],
       [ 0.02996615],
       [-0.00641309],
       [ 2.79325332]])
```

- Kiểm tra trên tập dữ liệu thì cho kết quả tốt hơn

```
15.97672173851104
```