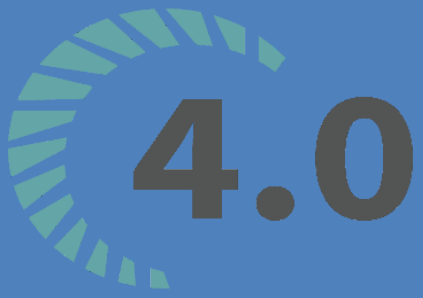
KHOA CÔNG NGHỆ THÔNG TIN

ĐẠI HỌC KHOA HỌC TỰ NHIÊN THÀNH PHỐ HỒ CHÍ MINH, ĐẠI HỌC QUỐC GIA TP HCM

TRỰC QUAN HÓA DỮ LIỆU



Nhóm thực hiện: Nhóm 11

19127476 - Trần Thị Huế Minh

19127486 - Nguyễn Lê Nguyên

19127125 - Lê Duy Dũng

MỤC LỤC

I. Phân chia công việc

Họ và tên	MSSV	Công việc			Đánh giá công việc
Trần Thị Huế Minh	19127476	Exploratory analysis of Car MPG data [5,6,7,8,9]	Electric power consumption data	Chỉnh sửa báo cáo	100%
Lê Duy Dũng	19127125	Tìm hiểu thư viện Matplotlib	Exploratory analysis of Car MPG data [1,2]	Viết Báo Cáo	100%
Nguyễn Lê Nguyên	19127486	Tìm hiểu thư viện Numpy, Pandas	Exploratory analysis of Car MPG data [3,4]	Viết Báo Cáo	100%

II. Tìm hiểu các thư viện

A. Numpy

1. Numpy là gì?

- NumPy là một thư viện quan trọng cho tính toán khoa học trong ngôn ngữ lập trình Python. Nó cung cấp một đối tượng mảng đa chiều mạnh mẽ, cho phép làm việc hiệu quả với ma trận và mảng, đặc biệt là dữ liệu ma trận và mảng lớn với tốc độ xử lý nhanh, các phép toán trên mảng chẳng hạn như tính toán toán học, thống kê, biến đổi hình dạng và sắp xếp. Thêm vào đó, NumPy cũng cung cấp nhiều công cụ hữu ích khác như đại số tuyến tính, biến đổi Fourier rời rạc, mô phỏng ngẫu nhiên và nhiều hơn nữa.
- Với các tính năng trên, NumPy là một công cụ quan trọng cho các nhà khoa học dữ liệu, kỹ sư máy tính và các nhà nghiên cứu khoa học.

2. Cài Đặt

- NumPy không đi kèm với Python. Chúng tôi phải cài đặt nó bằng trình cài đặt python pip. Thực hiện lệnh sau:

```
Python

pip install numpy
```

3. Cách sử dụng thư viện Numpy

- Khai báo thư viện trong Python, ta dùng lệnh sau: `import numpy as np`
- Cách sử dụng mảng trong Numpy:

  Mảng là cấu trúc dữ liệu trung tâm của thư viện NumPy, đó là một tập hợp các giá trị được tổ chức theo dạng lưới. Mảng chứa thông tin về dữ liệu thô, các chỉ số và phương thức để thực hiện các phép toán trên mảng. Mảng có thể chứa các kiểu dữ liệu khác nhau, nhưng tất cả các phần tử trong mảng cùng có kiểu dữ liệu giống nhau. Điều này làm cho mảng trở thành một công cụ hữu ích để lưu trữ và xử lý dữ liệu cho các ứng dụng khoa học dữ liệu và tính toán

❖ **Khởi tạo mảng:**

Mảng 1 chiều

```
arr = np.array([1,3,4,5,6], dtype = int)
arr = np.array([1,3,4,5,6])
```

```
3   # Mảng 1 chiều
4   ##Khởi tạo mảng một chiều với kiểu dữ liệu các phần tử là Integer
5   arr = np.array([1,3,4,5,6], dtype = int)
6   print("Mảng 1 chiều với kiểu dữ liệu là Integer: ", arr)
7   ##Khởi tạo mảng một chiều với kiểu dữ liệu mặc định
8   arr1 = np.array([1,3,4,5,6])
9   print("Mảng 1 chiều kiểu dữ liệu mặc định: ", arr1)
10
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\Minh> & C:/Users/Minh/AppData/Local/Microsoft/WindowsApps/python3.10.exe
Mảng 1 chiều với kiểu dữ liệu là Integer:  [1 3 4 5 6]
Mảng 1 chiều kiểu dữ liệu mặc định:  [1 3 4 5 6]
PS C:\Users\Minh>
```

Mảng 2 chiều

```
arr2 = np.array([(2,4,0,6), (4,7,5,6)],dtype = int)
```

```
11  #Mảng 2 chiều
12  arr2 = np.array([(2,4,0,6), (4,7,5,6)],dtype = int)
13  print("Mảng 2 chiều: ", arr2)
14
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\Minh> & C:/Users/Minh/AppData/Local/Microsoft/Windows
Mảng 2 chiều:  [[2 4 0 6]
 [4 7 5 6]]
PS C:\Users\Minh>
```

Mảng 3 chiều

```
arr3 = np.array(([(2,4,0,6), (4,7,5,6)],
                 [(0,3,2,1), (9,4,5,6)],
                 [(5,8,6,4), (1,4,6,8)]), dtype = int)
```

```
15   #Mảng 3 chiều
16   arr3 = np.array(([(2,4,0,6), (4,7,5,6)],
17                    [(0,3,2,1), (9,4,5,6)],
18                    [(5,8,6,4), (1,4,6,8)]), dtype = int)
19
20   print("Mảng 3 chiều: ", arr3)
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

PS C:\Users\Minh> & C:/Users/Minh/AppData/Local/Microsoft/Windows
Mảng 3 chiều:  [[[2 4 0 6]
  [4 7 5 6]]

 [[0 3 2 1]
  [9 4 5 6]]

 [[5 8 6 4]
  [1 4 6 8]]]
PS C:\Users\Minh>
```

Có thể khởi tạo mảng bằng các hàm có sẵn:

- **np.zeros((3,4), dtype = int)**: Tạo mảng hai chiều các phần tử 0 với kích thước 3x4.

```
22   arr4 = np.zeros((3,4), dtype = int)
23   print(arr4)
24
```

```
ROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

S C:\Users\Minh> & C:/Users/Minh/AppData/Local/M
[[0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]
S C:\Users\Minh>
```

- **np.ones((2,3,4), dtype = int)**: Tạo mảng 3 chiều các phần tử 1 với kích thước 2x3x4.

```
26   arr5 = np.ones((2,3,4), dtype = int)
27   print(arr5)
28
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

PS C:\Users\Minh> & C:/Users/Minh/AppData/Local/M
[[[1 1 1 1]
  [1 1 1 1]
  [1 1 1 1]]

 [[1 1 1 1]
  [1 1 1 1]
  [1 1 1 1]]]
PS C:\Users\Minh>
```

- **np.arange(1,10,3)**: Tạo mảng với các phần tử từ 1 - 9 với bước nhảy là 3

```
27   arr6 = np.arange(1,9,3)
28   print(arr6)
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TE

PS C:\Users\Minh> & C:/Users/Minh/Ap
[1 4 7]
PS C:\Users\Minh>
```

- **np.full((2,3),9)**: Tạo mảng 2 chiều các phần tử 9 với kích thước 2x3.

```
31   arr7 = np.full((2,3),9)
32   print(arr7)
33
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE

PS C:\Users\Minh> & C:/Users/Minh/
[[9 9 9]
 [9 9 9]]
PS C:\Users\Minh>
```

- **np.eye(4, dtype=int)**: Tạo ma trận đơn vị với kích thước là 4x4.

```
35    arr8 = np.eye(4, dtype=int)
36    print(arr8)
37
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINA

PS C:\Users\Minh> & C:/Users/Minh/AppData
[[1 0 0 0]
 [0 1 0 0]
 [0 0 1 0]
 [0 0 0 1]]
PS C:\Users\Minh>
```

- **np.random.random((2,3))**: Tạo ma trận các phần tử ngẫu nhiên với kích thước 2x3.

```
38    arr9 = np.random.random((2,3))
39    print(arr9)
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\Users\Minh> & C:/Users/Minh/AppData/Lo
[[0.1201472  0.63570639 0.01217133]
 [0.18609854 0.60953098 0.01655028]]
PS C:\Users\Minh>
```

❖ Thao tác trên mảng

Kiểm tra thuộc tính mảng:

- dtype: Kiểu dữ liệu của phần tử trong mảng.
- shape: Kích thước của mảng.
- size: Số phần tử trong mảng.
- ndim: Số chiều của mảng.

Ví dụ:

```
38  arr9 = np.random.random((2,3))
39  print(arr9)
40  print("Kiểu dữ liệu của phần tử trong mảng:", arr9.dtype)
41  print("Kích thước của mảng:", arr9.shape)
42  print("Số phần tử trong mảng:", arr9.size)
43  print("Số chiều của mảng:", arr9.ndim)
44
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\Minh> & C:/Users/Minh/AppData/Local/Microsoft/WindowsApps/py
[[0.7990501  0.1566578  0.19188758]
 [0.65012706 0.99663941 0.3540699 ]]
Kiểu dữ liệu của phần tử trong mảng: float64
Kích thước của mảng: (2, 3)
Số phần tử trong mảng: 6
Số chiều của mảng: 2
PS C:\Users\Minh>
```

Truy cập các phần tử trong mảng

- **arr[i]**: Truy cập tới phần tử thứ i của mảng 1 chiều.
- **arr1[i,j]**: Truy cập tới phần tử hàng i, cột j của mảng 2 chiều.
- **arr2[n,i,j]**: Truy cập tới phần tử chiều n, hàng i, cột j của mảng 3 chiều.
- **arr[a:b]**: Truy cập tới các phần tử từ a đến b-1 trong mảng 1 chiều.
- **arr1[:,:i]**: Truy cập tới phần tử từ cột 0 đến cột i-1, của tất cả các hàng trong mảng 2 chiều.

Các hàm thống kê

- **arr.max() hoặc np.max(arr)**: Lấy giá trị lớn nhất của mảng arr.
- **arr.min() hoặc np.min(arr)**: Lấy giá trị nhỏ nhất của mảng arr.
- **arr.sum() hoặc np.sum(arr)**: Tổng tất cả các phần tử trong mảng arr.
- **arr.mean() hoặc np.mean(arr)**: Trung bình cộng của tất cả các phần tử trong mảng arr.
- **np.median(arr)**: Trả về giá trị trung vị của mảng arr.

Ví dụ:

```
11  #Mảng 2 chiều
12  arr2 = np.array([(2,4,0,6), (4,7,5,6)],dtype = int)
13  print("Mảng 2 chiều: ", arr2)
14  print("max = ",arr2.max())
15  print("min = ",arr2.min())
16  print("sum = ",arr2.sum())
17  print("mean = ",arr2.mean())
18
19  median = np.median(arr2)
20  print("median = ",median)
21
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\Minh> & C:/Users/Minh/AppData/Local/Microsoft/WindowsAp
Mảng 2 chiều:  [[2 4 0 6]
 [4 7 5 6]]
max =  7
min =  0
sum =  34
mean =  4.25
median =  4.5
PS C:\Users\Minh>
```

❖ **NumPy dtype**

Đối tượng kiểu dữ liệu 'dtype' là một thực thể (instance) của lớp numpy.dtype.

Basic Type	Available Numpy types	Comments
Boolean	bool	Các phần tử có kích thước 1 byte.
Integer	int8, int16, int32, int64, int128, int	Kiểu int mặc định có kích thước là kích thước của kiểu int trong C trên nền tảng đang sử dụng
Unsigned Integer	uint8, uint16, uint32, uint64, uint128, uint	Kiểu uint mặc định có kích thước là kích thước của unsigned int trong C trên nền tảng đang sử dụng.
Float	float32, float64, float, longfloat	Kiểu Float luôn là một giá trị số thực độ chính xác kép (64 bit). longfloat đại diện cho số thực độ chính xác lớn và kích thước của nó phụ thuộc vào nền tảng.

Complex	complex64, complex128, complex	Phần thực và ảo của complex64 được biểu diễn bởi một giá trị số thực đơn (32 bit) mỗi phần, cho tổng kích thước là 64 bit.
Strings	str, unicode	Unicode luôn là UTF32 (UCS4).
Object	object	đại diện cho các phần tử trong một mảng là các đối tượng Python.
Records	void	được sử dụng cho các cấu trúc dữ liệu tùy ý trong các mảng ghi nhớ.

❖ **Các phép toán với mảng**

■ **Cộng/ trừ hai mảng**

Ví dụ:

```
A = np.array([[1, 3, 4], [-2, 6, 0], [-5, 7, 2]])
B = np.array([[2, 3, 4], [-1, -2, -3], [0, 4, -4]])
print("A + B = \n", A + B)
print("A - B = \n", A - B)
```

Kết quả:

```
A + B =
 [[ 3  6  8]
 [-3  4 -3]
 [-5 11 -2]]
A - B =
 [[-1  0  0]
 [-1  8  3]
 [-5  3  6]]
```

■ Phép nhân/ chia ma trận

Ví dụ:

```
A = np.array([[1, 3, 4], [-2, 6, 0], [-5, 7, 2]])
print("A * 3 = \n", A * 3)
print("A  / 4 = \n", A / 4)
```

Kết quả:

```
A * 3 =
 [[  3   9  12]
 [ -6  18   0]
 [-15  21   6]]
A  / 4 =
 [[ 0.25  0.75  1.  ]
 [-0.5   1.5   0.  ]
 [-1.25  1.75  0.5 ]]
```

- Phép nhân hai ma trận

Ví dụ:

```
A = np.array([[1, 3, 4], [-2, 6, 0], [-5, 7, 2]])
B = np.array([[2, 3, 4], [-1, -2, -3], [0, 4, -4]])
print("A * B = \n", A @ B)
print("B * A = \n", B @ A)
```

Kết quả

```
A * B =
 [[ -1  13 -21]
 [-10 -18 -26]
 [-17 -21 -49]]
B * A =
 [[-24  52  16]
 [ 18 -36 -10]
 [ 12  -4  -8]]
```

Tuy vậy, Python vẫn hỗ trợ nếu như ta muốn lấy phần tử của ma trận này nhân với phần tử tương ứng của ma trận kia qua toán tử * thông thường

```
A = np.array([[1, 3, 4], [-2, 6, 0], [-5, 7, 2]])
B = np.array([[2, 3, 4], [-1, -2, -3], [0, 4, -4]])

print(A * B)
```

Kết quả:

```
[[  2   9  16]
```

```
 [  2 -12   0]
 [  0  28  -8]]
```

- Ma trận đơn vị

Ví dụ:

```
arr = np.eye(3)
print(arr)
```

Kết quả:

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

- Ma trận đường chéo

Ví dụ:

```
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(np.diag(A))
```

Kết quả:

```
[1 5 9]
```

- Ma trận chuyển vị

Ví dụ:

```
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(A.T)
print(np.transpose(A))
```

Kết quả:

```
[[1 4 7]
 [2 5 8]
 [3 6 9]]
```

- Định thức của ma trận vuông

  Ví dụ:

```
A = np.array([[2, 1, 3], [5, 3, 2], [1, 4, 3]])

print(np.linalg.det(A))
```

  Kết quả:

```
40.000000000000014
```

- Ma trận nghịch đảo

  Ví dụ:

```
A = np.array([[1, 2, 3], [2, 5, 3], [1, 0, 8]])

print(np.linalg.inv(A))
```

  Kết quả:

```
[[-40.  16.   9.]
 [ 13.  -5.  -3.]
 [  5.  -2.  -1.]]
```

- Một số ví dụ điển hình

```
import numpy as np
a = np.array([1,2,3,4])
b = np.zeros((3,5))
c = np.random.rand(4,6)
c[[2,3], 3] = 0
colmean = np.mean(c, axis = 0)
rowmean = np.mean(c, axis=1)
nonzeromean = np.mean(c[nonzero(c[:,3]), 3])
```

Giải thích

```
import numpy as np
```

- Nhập thư viện NumPy vào chương trình và đặt tên viết tắt cho thư viện là "np" để có thể sử dụng các hàm và phương thức của thư viện.

```
a = np.array ([1 ,2 ,3 ,4])
```

- Tạo một mảng một chiều có giá trị là 1, 2, 3 và 4 sử dụng hàm array() của thư viện NumPy và lưu trữ nó vào biến a.

```
b = np.zeros ((3 ,5))
```

- Tạo một mảng hai chiều có kích thước 3 hàng và 5 cột, tất cả các giá trị trong mảng đều bằng 0, sử dụng hàm zeros() của thư viện NumPy và lưu trữ nó vào biến b.

```
c = np.random.rand (4 ,6)
```

- Tạo một mảng hai chiều có kích thước 4 hàng và 6 cột với các giá trị ngẫu nhiên trong phạm vi từ 0 đến 1 sử dụng hàm rand() của thư viện NumPy và lưu trữ nó vào biến c.

```
c[[2 ,3] , 3] = 0
```

- Gán giá trị 0 cho tất cả các phần tử của hàng thứ 2 và 3 và cột thứ 3 của mảng c.

```
colmean = np.mean(c, axis = 0)
```

- Tính giá trị trung bình của mỗi cột trong mảng c bằng cách sử dụng hàm mean() của thư viện NumPy với trục xác định bởi tham số axis=0 và lưu trữ nó vào biến colmean.

```
rowmean = np.mean(c, axis =1)
```

- Tính giá trị trung bình của mỗi hàng trong mảng c bằng cách sử dụng hàm mean() của thư viện NumPy với trục xác định bởi tham số axis=1 và lưu trữ nó vào biến rowmean.

```
nonzeromean = np.mean(c[nonzero(c[: ,3]) , 3])
```

- Tính giá trị trung bình của tất cả các phần tử của cột thứ 3 trong mảng c mà có giá trị khác 0 bằng cách sử dụng hàm mean() của thư viện NumPy với mảng được chọn là các phần tử của cột thứ 3 của c mà có giá trị khác 0 sử dụng hàm nonzero().

Kết quả từng câu lệnh

- Result 1

```
import numpy as np
#Tạo mảng 1 chiều a
a = np.array ([1 ,2 ,3 ,4])
print('Array A: \n',a)
```

```
Array A:
 [1 2 3 4]
```

- Result 2

```
#Tạo mảng 2 chiều chứa phần tử 0 với kích thước 3 dòng 5 cột
b = np.zeros ((3 ,5))
print('Array B with All Values "0": \n',b)
```

```
Array B with All Values "0":
  [[0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0.]]
```

- Result 3

```
#Tạo mảng 2 chiều chứa các phần tử ngẫu nhiên với kích thước 4 dòng 6 cột
c = np.random.rand (4 ,6)
print('Array C with Random Values: \n',c)
```

```
Array C with Random Values:
  [[0.31361512 0.27746476 0.39942345 0.56226652 0.5682639  0.96317209]
  [0.2125145  0.70960171 0.71511806 0.95363657 0.80882661 0.89990526]
  [0.695881   0.85126321 0.42015549 0.75164481 0.19245366 0.37418395]
  [0.82901794 0.20849761 0.91507902 0.65880201 0.77240042 0.42426225]]
```

- Result 4

```
# Gán giá trị 0 cho tất cả các phần tử của hàng thứ 2 và 3 và cột thứ 3 của mảng c.
c[[2 ,3] , 3] = 0
print('Array C with Random Values after update value 0 in C[2,3], C[3,3]: \n',c)
```

```
Array C with Random Values after update value 0 in C[2,3], C[3,3]:
 [[0.05073291 0.53481732 0.10156206 0.02907581 0.50738416 0.84683804]
 [0.98376347 0.05396486 0.21118166 0.78611616 0.24460127 0.81946803]
 [0.37495861 0.64210952 0.37004421 0.         0.93639052 0.46347092]
 [0.45918448 0.13787878 0.25297257 0.         0.61604654 0.56510724]]
```

- Result 5

```
# #Tính giá trị trung bình của mỗi cột trong mảng c
colmean = np.mean(c, axis = 0)
print('Average Values each Columns in Array C: \n',colmean)
```

```
Average Values each Columns in Array C:
 [0.56819774 0.54507621 0.32714491 0.35368489 0.64761381 0.6701418 ]
```

- Result 6

```
# #Tính giá trị trung bình của mỗi hàng trong mảng c
rowmean = np.mean(c, axis =1)
print('Average Values each Rows in Array C: \n',rowmean)
```

```
Average Values each Rows in Array C:
 [0.41714443 0.63489727 0.26112625 0.29195011]
```

- Result 7

```
# #Tính giá trị trung bình của tất cả các phần tử của cột thứ 3 trong mảng c mà có giá trị khác 0
nonzeromean = np.mean(c[np.nonzero(c[: ,3]) , 3])
print('Average Values Column 3 with values different from "0" in Array C: \n',nonzeromean)
```

```
Average Values Column 3 with values different from "0" in Array C:
 0.6427926376352446
```

B. Pandas

1. Pandas là gì?

- Pandas là một thư viện Python mã nguồn mở, cung cấp các cấu trúc dữ liệu nhanh, mạnh mẽ, linh hoạt và mang hàm ý. Tên thư viện được bắt nguồn từ panel data (bảng dữ liệu). Pandas được thiết kế để làm việc dễ dàng và trực quan với dữ liệu có cấu trúc (dạng bảng, đa chiều, có tiềm năng không đồng nhất) và dữ liệu chuỗi thời gian.

- Mục đích của pandas là trở thành khối xây dựng cấp cao cơ bản để thực hiện phân tích dữ liệu trong thế giới thực, thực tế bằng Python, mà mục tiêu rộng hơn còn là trở thành công cụ thao tác/phân tích dữ liệu nguồn mở mạnh mẽ và linh hoạt nhất hiện có bằng bất kỳ ngôn ngữ nào.

2. Cài Đặt

- Sử dụng pip và gõ lệnh: pip install pandas

- Hoặc bằng Anaconda, dùng lệnh: conda install pandas

3. Cách sử dụng thư viện Pandas

- Khai báo dữ liệu:sử dụng câu lệnh `import pandas as pd`
- Thao tác với cấu trúc dữ liệu cơ bản: Pandas có 2 cấu trúc dữ liệu cơ bản

❖ Series (1 chiều)

Series là mảng một chiều giống như mảng Numpy, hay như một cột của một bảng, nhưng nó bao gồm thêm một bảng đánh label. Series có thể được khởi tạo như sau: Series([data, index, dtype, name, copy, . . .})

Tạo Series

- Không truyền index

```
import pandas as pd
s = pd.Series([0,1,2,3])
print(s)
```

Output:

```
0    0
1    1
2    2
3    3
dtype: int64
```

- Truyền index

```
s = pd.Series([0,1,2,3], index=["a","b","c","d"])

print(s)
```

Output:

```
a    0
b    1
c    2
d    3
dtype: int64
```

Tạo Series từ dict

```
data = {'a' : -1.3, 'b' : 11.7, 'd' : 2.0, 'f': 10, 'g': 5}
ser = pd.Series(data,index=['a','c','b','d','e','f'])
print(ser)
```

- Output

```
a    -1.3
c     NaN
b    11.7
d     2.0
e     NaN
f    10.0
dtype: float64
```

Tạo Series từ Scalar

```
ser = pd.Series(5, index=[1, 2, 3, 4, 5])

print(ser)
```

- Output

```
1    5
2    5
3    5
4    5
5    5
dtype: int64
```

Truy cập dữ liệu từ Series với index và vị trí

- Ví dụ:

```
data = {'a' : -1.3, 'b' : 11.7, 'd' : 2.0, 'f': 10, 'g': 5}

ser = pd.Series(data,index=['a','c','b','d','e','f'])
```

Lấy dữ liệu tại index cụ thể

```
print(ser['d'])
```

- Output

```
2.0
```

Lấy dữ liệu từ đầu đến vị trí index cụ thể

```
print(ser[:'d'])
```

- Output

```
a    -1.3
c     NaN
b    11.7
d     2.0
dtype: float64
```

Lấy dữ liệu theo vị trí: 2 dữ liệu đầu

```
print(ser[:2])
```

- Output:

```
a   -1.3
c    NaN
dtype: float64
```

Lấy dữ liệu theo vị trí: 3 dữ liệu cuối

```
print(ser[-3:])
```

Output:

```
d     2.0
e     NaN
f    10.0
dtype: float64
```

Chuyển đổi sang dạng khác

- Lấy dạng array của Series bằng numpy.asarray từ ví dụ trên

```
import pandas as pd
import numpy as np


data = {'a' : -1.3, 'b' : 11.7, 'd' : 2.0, 'f': 10, 'g': 5}
```

```
ser = pd.Series(data,index=['a','c','b','d','e','f'])

a = np.asarray(ser)

print(a)
```

- Output:

```
[-1.3  nan 11.7  2.   nan 10. ]
```

❖ **DataFrame (2 chiều)**

Cấu trúc dữ liệu được gắn nhãn hai chiều với các cột và hàng như bảng tính (spreadsheet) hoặc bảng (table). Giống như Series, DataFrame có thể chứa bất kỳ loại dữ liệu nào. Một điều quan trọng cần làm nổi bật là tất cả các cột trong khung dữ liệu là series Pandas.

■ Tạo DataFrame từ dict các Series

```
import pandas as pd

# tạo dict từ các series

s = {'một': pd.Series([1., 2., 3., 5.], index=['a', 'b', 'c','e']),

     'hai': pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c','d'])}



# tại DataFrame từ dict

df = pd.DataFrame(s)

print(df)
```

Output:

```
  một  hai
a  1.0  1.0
b  2.0  2.0
c  3.0  3.0
d  NaN  4.0
e  5.0  NaN
```

■ Các thao tác chọn, thêm, xóa cột

Chọn Cột

```
import pandas as pd



s = {'một': pd.Series([1., 2., 3., 5.], index=['a', 'b', 'c', 'e']),

     'hai': pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd']),

       'ba': pd.Series([9., -1.3, 3.5, 41.1], index=['a', 'b', 'c',
'd'])}

df = pd.DataFrame(s)

# chọn cột hai

df_hai = df['hai']

print(df_hai)
```

Output:

```
a    1.0
b    2.0
c    3.0
d    4.0
e    NaN
Name: hai, dtype: float64
```

Thêm Cột

```
# thêm cột bốn với giá trị mỗi ô theo công thức

df['bốn'] = df['hai'] - df['ba']

# thêm cột với giá trị vô hướng (scalar value)

df['Chuẩn'] = 'OK'

# thêm cột không cùng số lượng index với DataFrame

df['Khác'] = df['hai'][:3]

# thêm cột True/False theo điều kiện

df['KT'] = df['một'] == 3.0
```

```
# dùng hàm insert. Cột "chèn" bên dưới sẽ ở vị trí 2 (tính từ
0), có giá trị bằng cột một

df.insert(2, 'chèn', df['một'])
```

Output:

```
   một  hai  chèn    ba   bốn Chuẩn  Khác     KT
a  1.0  1.0   1.0   9.0  -8.0    OK   1.0  False
b  2.0  2.0   2.0  -1.3   3.3    OK   2.0  False
c  3.0  3.0   3.0   3.5  -0.5    OK   3.0   True
d  NaN  4.0   NaN  41.1 -37.1    OK   NaN  False
e  5.0  NaN   5.0   NaN   NaN    OK   NaN  False
```

Thêm Cột: Có thể xóa cột bằng lệnh def hoặc hàm pop

```
# xóa cột hai

del df['hai']

# pop cột ba với dict tv_ba

tv_ba = df.pop('ba')

print( df)
```

Output:

```
   một
a  1.0
b  2.0
c  3.0
d  NaN
e  5.0
```

- Lập chỉ mục/ lựa chọn

Cho ví dụ sau:

```
import pandas as pd

s = {'một': pd.Series([1., 2., 3., 5.], index=['a', 'b', 'c', 'e']),

     'hai': pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd']),

        'ba':  pd.Series([9.,  -1.3,  3.5,  41.1],  index=['a',  'b',  'c',
'd'])}

df = pd.DataFrame(s)
```

Chọn dòng theo label

```
# chọn dòng theo label

d = df.loc['a']
```

Output:

```
một    1.0

hai    1.0

ba     9.0

Name: a, dtype: float64
```

Chọn dòng theo vị trí nguyên

```
# chọn dòng theo vị trí nguyên

d = df.iloc[4]
```

Output:

```
một    5.0

hai    NaN

ba     NaN

Name: e, dtype: float64
```

Cắt (slice) các dòng

```
df = pd.DataFrame(s)
```

```
# cắt lấy ra từ dòng 3 đến dòng 4

d = df[2:4]
```

Output:

```
   một  hai    ba

c  3.0  3.0   3.5

d  NaN  4.0  41.1
```

4. Đọc file csv sử dụng thư viện pandas

- Có thể dễ dàng đọc vào một file .csv bằng cách sử dụng hàm read_csv và được trả về 1 dataframe.
- Bạn có thể in ra n bản ghi đầu tiên của dataframe sử dụng hàm head. Ngược lại của hàm head là hàm tail

Tuy nhiên, bạn cũng sẽ phải lưu ý một vài tham số của hàm read_csv như:

- ***encoding***: chỉ định encoding của file đọc vào. Mặc định là utf-8.
- ***sep***: thay đổi dấu ngăn cách giữa các cột. Mặc định là dấu phẩy (‘,’)
- ***header***: chỉ định file đọc vào có header(tiêu đề của các cột) hay không. Mặc định là infer.
- ***index_col***: chỉ định chỉ số cột nào là cột chỉ số(số thứ tự). Mặc định là None.
- ***n_rows***: chỉ định số bản ghi sẽ đọc vào. Mặc định là None – đọc toàn bộ.

- Ví dụ:

```
peoples_df = pd.read_csv('./people.csv', encoding='utf-8',header=None,
sep=',')

peoples_df.head(5)
```

- Kết quả trả về sẽ có dạng như thế này:

	0	1	2	3	4	5	6	7	8
0	person_ID	name	first	last	middle	email	phone	fax	title
1	3130	Burks, Rosella	Rosella	Burks	NaN	BurksR@univ.edu	963.555.1253	963.777.4065	Professor
2	3297	Avila, Damien	Damien	Avila	NaN	AvilaD@univ.edu	963.555.1352	963.777.7914	Professor
3	3547	Olsen, Robin	Robin	Olsen	NaN	OlsenR@univ.edu	963.555.1378	963.777.9262	Assistant Professor
4	1538	Moises, Edgar Estes	Edgar	Moises	Estes	MoisesE@univ.edu	963.555.2731x3565	963.777.8264	Professor

C. Matplotlib

1. Matplotib là gì?

- Tổng quan: Matplotlib là một thư viện Python được sử dụng để tạo các trực quan hóa tĩnh, có tính tương tác trong Python. Thư viện này được phát triển bởi John Hunter vào năm 2003.
- Các tính năng của Matplotlib: Matplotlib cung cấp nhiều tùy chọn để vẽ đồ thị, bao gồm đồ thị đường, đồ thị điểm phân tán, đồ thị cột, histograms và còn nhiều nữa. Nó cũng cung cấp các tùy chọn tùy chỉnh mở rộng để trực quan hóa dữ liệu.
- Lợi ích của Matplotlib: Matplotlib rất dễ sử dụng, linh hoạt và được sử dụng rộng rãi trong cộng đồng khoa học. Nó cũng tương thích với nhiều thư viện Python khác, bao gồm NumPy và Pandas.

2. Cài đặt

- Matplotlib có thể được import vào một script Python bằng cách sử dụng câu lệnh

```
import matplotlib.pyplot as plt
```

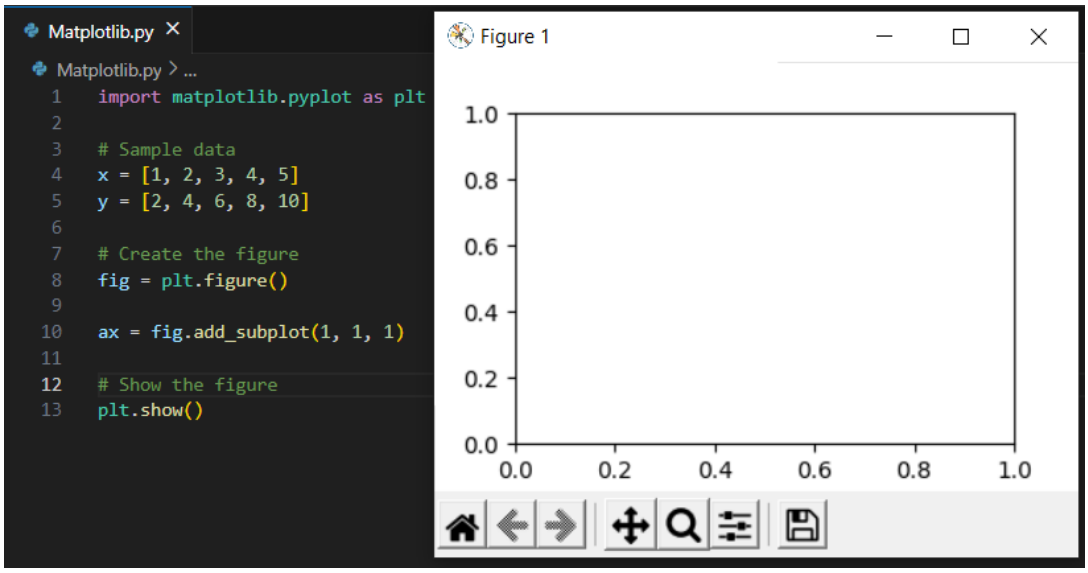
3. Cách sử dụng thư viện Matplotlib

a. Vẽ đồ thị cơ bản với Matplotlib:

- Tạo một figure: Một figure là khung chứa ngoài cùng của một đồ thị Matplotlib. Nó có thể được tạo bằng cách sử dụng lệnh “plt.figure()”.

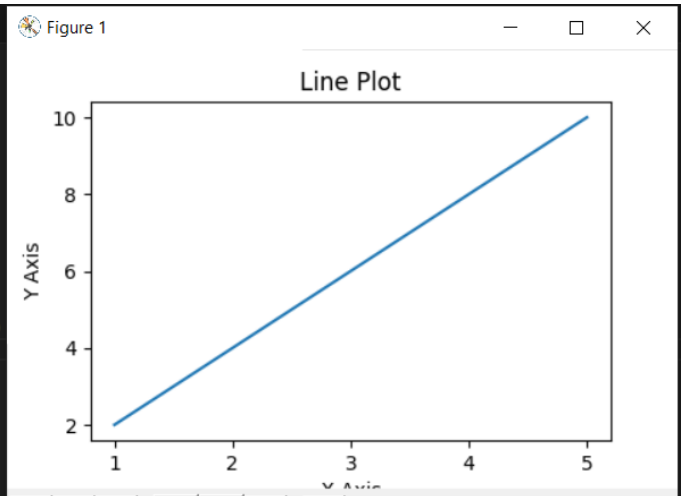
Ví dụ code python tạo figure sử dụng Matplotlib.

- Thêm các trục: Trục là các khu vực bên trong một figure nơi mà dữ liệu được dùng để lập đồ thị. Chúng có thể được thêm bằng cách sử dụng dòng lệnh "fig.add_subplot()".



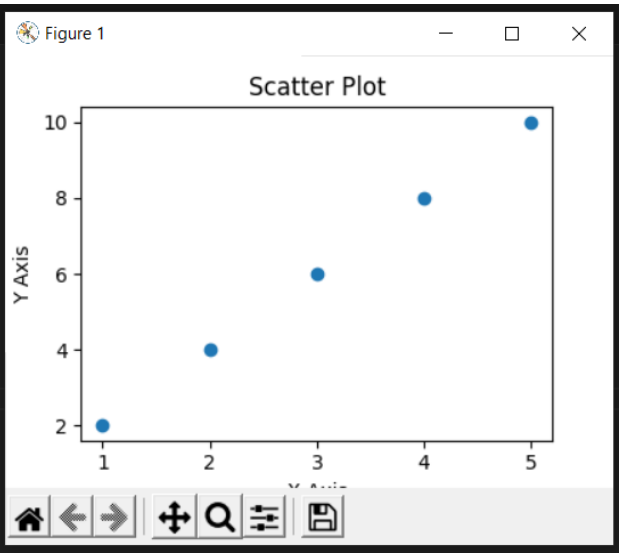
- Vẽ đồ thị dữ liệu: Dữ liệu có thể được biểu diễn thành đồ thị bằng cách sử dụng các câu lệnh Matplotlib, bao gồm "plt.plot((x-axis values, y-axis values)", "plt.scatter(x-axis values, y-axis values)", "plt.bar(categorical variables, values, color)" và "plt.hist((values, number of bins)"

```python
import matplotlib.pyplot as plt

# Sample data
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Create the figure
fig = plt.figure()

ax = fig.add_subplot(1, 1, 1)

# Create a line plot using plt.plot()
ax.plot(x, y)

# Set the title and axis labels
ax.set_title("Line Plot")
ax.set_xlabel("X Axis")
ax.set_ylabel("Y Axis")

# Show the figure
plt.show()
```

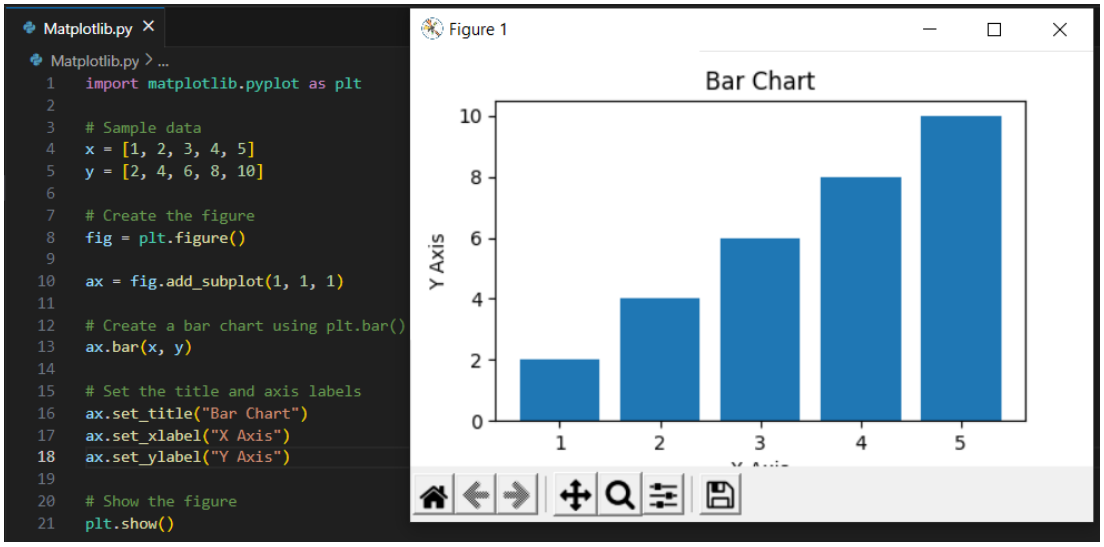


Sử dụng plt.plot()

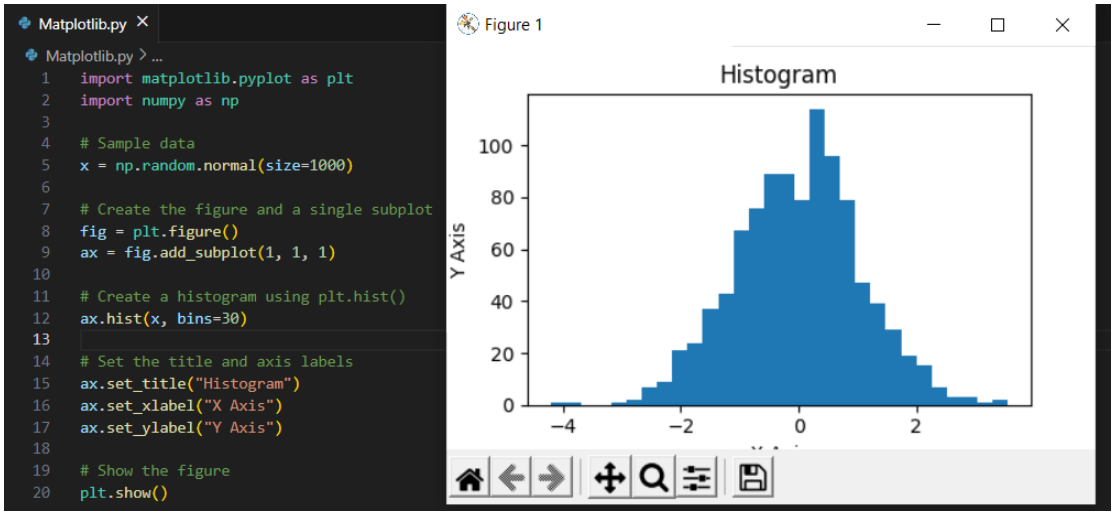
```python
import matplotlib.pyplot as plt

# Sample data
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Create the figure
fig = plt.figure()

ax = fig.add_subplot(1, 1, 1)

# Create a scatter plot using plt.scatter()
ax.scatter(x, y)

# Set the title and axis labels
ax.set_title("Scatter Plot")
ax.set_xlabel("X Axis")
ax.set_ylabel("Y Axis")

# Show the figure
plt.show()
```



Sử dụng plt.scatter() để vẽ scatter plot.



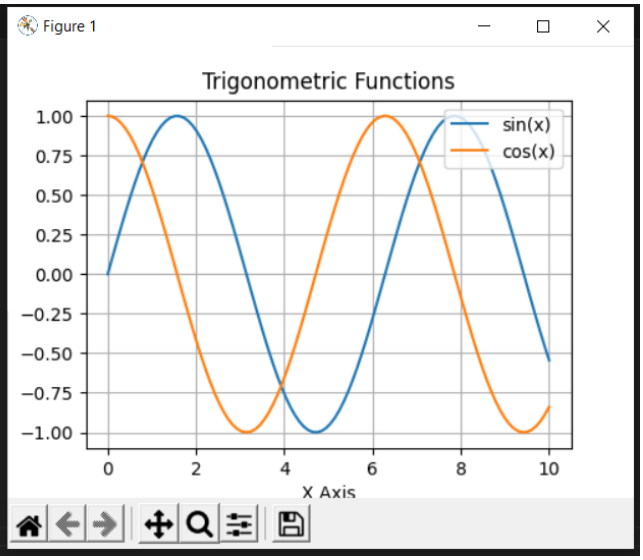
Sử dụng plt.bar()



Sử dụng plt.hist()

- Tùy chỉnh giao diện: Giao diện của một đồ thị có thể được tùy chỉnh bằng cách sử dụng các câu lệnh như là “plt.xlabel(“string”)”, “plt.title(“string”), “plt.legend(loc)”, “plt.grid()”.

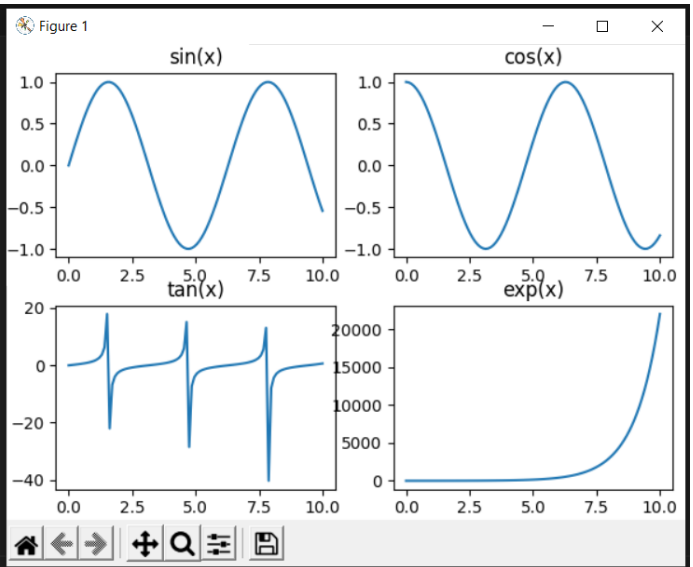
```python
import matplotlib.pyplot as plt
import numpy as np

# Sample data
x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)

# Create the figure and a single subplot
fig, ax = plt.subplots()

# Create line plots using plt.plot()
ax.plot(x, y1, label='sin(x)')
ax.plot(x, y2, label='cos(x)')

# Set the title, axis labels, legend, and grid
ax.set_title("Trigonometric Functions")
ax.set_xlabel("X Axis")
ax.set_ylabel("Y Axis")
ax.legend(loc='upper right')
ax.grid()

# Show the figure
plt.show()
```



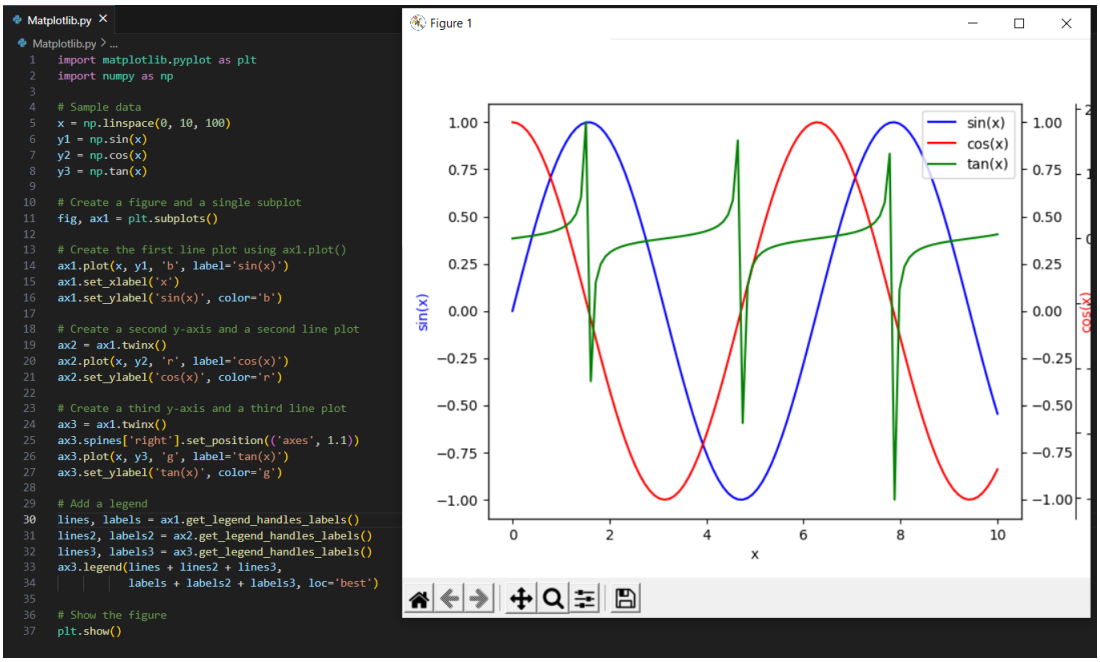
b. Vẽ đồ thị nâng cao với Matplotlib:

- Tạo các đồ thị con: Ta có thể tạo nhiều đồ thị bên trong một figure bằng cách sử dụng câu lệnh "plt.subplots(nrows, ncols, figsize)".

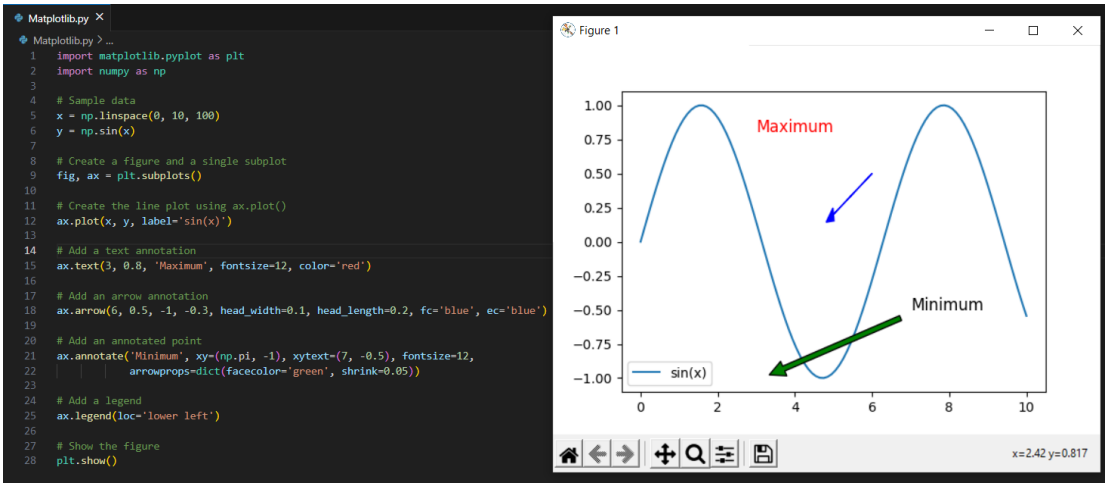
```python
import matplotlib.pyplot as plt
import numpy as np

# Sample data
x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.tan(x)
y4 = np.exp(x)

# Create a figure with 4 subplots
fig, axs = plt.subplots(2, 2, figsize=(8, 6))

# Create line plots for each subplot
axs[0, 0].plot(x, y1)
axs[0, 0].set_title('sin(x)')
axs[0, 1].plot(x, y2)
axs[0, 1].set_title('cos(x)')
axs[1, 0].plot(x, y3)
axs[1, 0].set_title('tan(x)')
axs[1, 1].plot(x, y4)
axs[1, 1].set_title('exp(x)')

# Adjust the spacing between subplots
fig.tight_layout()

# Show the figure
plt.show()
```



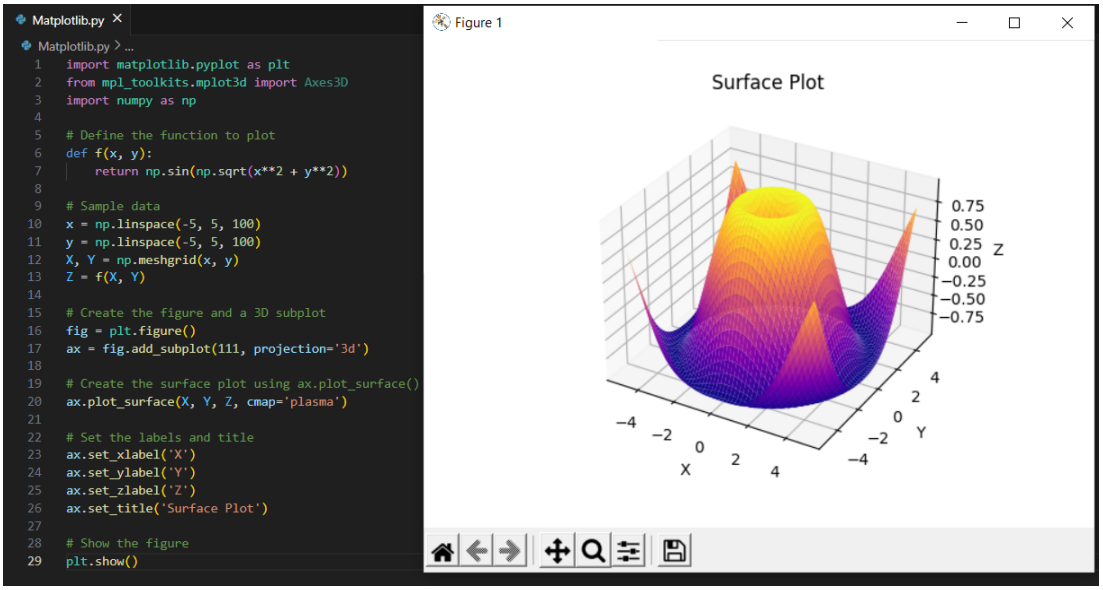
- Nhiều trục trong một đồ thị: Ta có thể thêm nhiều trục vào một đồ thị bằng cách sử dụng các câu lệnh như "ax.twinx()" and "ax.twiny()".



- Các chú thích: Văn bản, các mũi tên và các khối hình có thể được thêm vào một đồ thị thông qua việc sử dụng các câu lệnh như "plt.text()", "plt.arrow()", and "plt.annotate()".



- Các loại đồ thị khác nhau: Matplotlib hỗ trợ nhiều loại đồ thị, bao gồm đồ thị 3D, các đồ thị viền và các heatmaps.



c. Lưu trữ và chia sẻ các đồ thị với Matplotlib:

- Lưu trữ các đồ thị: Các đồ thị có thể được lưu lại thông qua câu lệnh "plt.savefig()". Định dạng tệp có thể được chỉ định bằng phần mở rộng tệp chẳng hạn như .png, .pdf hoặc .svg.
- Nhúng các đồ thị: Các đồ thị có thể được nhúng vào một website thông qua câu lệnh "plt.savefig()" và bao gồm cả file hình ảnh trong code HTML.

Matplotlib.py

```python
import matplotlib.pyplot as plt
import numpy as np

# Sample data
x = np.linspace(0, 10, 100)
y = np.sin(x)

# Create the line plot using plt.plot()
plt.plot(x, y)

# Set the labels and title
plt.xlabel('x')
plt.ylabel('sin(x)')
plt.title('Line Plot')

# Save the figure to an image file
plt.savefig('line_plot.png')

# Create the HTML code to embed the image
html = '<img src="line_plot.png" alt="Line Plot">'

# Print the HTML code
print(html)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  COMMENTS
[Running] python -u "c:\Users\Admin\Desktop\BTTH\Matplotlib.py"
<img src="line_plot.png" alt="Line Plot">

[Done] exited with code=0 in 1.16 seconds
```

- Chia sẻ các đồ thị: Các đồ thị có thể được chia sẻ với người khác dưới dạng các file hình ảnh, hoặc chia sẻ script Python được dùng để tạo ra đồ thị đó.

d. Các câu lệnh khác trong Matplotlib:

- **show()** : hiển thị biểu đồ
- **scatter3D(x-axis values, y-axis values)** : Vẽ sơ đồ phân tán ba chiều với các giá trị trục x so với giá trị trục y
- **plot3D(x-axis values, y-axis values)** : Vẽ đồ thị đường ba chiều với các giá trị trục x so với giá trị trục y
- **xticks(index, categorical variables)** : Get hoặc set vị trí đánh dấu hiện tại và nhãn của trục x
- **barh(categorical variables, values, color)** : Dùng để tạo biểu đồ thanh ngang
- **xlim(start value, end value)** : Được sử dụng để đặt giới hạn giá trị của trục x
- **ylim(start value, end value)** : Được sử dụng để đặt giới hạn giá trị của trục y
- **set_xlabel("string")** : Được sử dụng để set nhãn x cho plot được chỉ định

- **set_ylabel("string")** : Được sử dụng để set nhãn y cho plot được chỉ định

4. So sánh Matplotlib với Pandas:

- Tạo các đồ thị từ các cấu trúc dữ liệu Pandas: Dữ liệu trong một Pandas Series hay DataFrame có thể được dùng để tạo đồ thị thông qua lệnh "df.plot()".
- Tùy chỉnh giao diện với Matplotlib: Giao diện của một đồ thị được tạo với Pandas có thể được tùy chỉnh thông qua các câu lệnh như "plt.xlabel()" và "plt.title()".
- Tạo đồ thị cho các loại dữ liệu khác nhau: Pandas hỗ trợ nhiều loại đồ thị, bao gồm đồ thị đường, đồ thị phân tán và đồ thị cột.

D. Reference

1. https://codelearn.io/sharing/tim-hieu-thu-vien-numpy-trong-python
2. https://vn.got-it.ai/blog/tong-quan-thu-vien-numpy-trong-python
3. https://codelearn.io/sharing/xu-ly-du-lieu-voi-pandas-trong-python
4. https://viblo.asia/p/pandas-python-tutorial-XL6lAxaDZek

III. Exploratory analysis of Car MPG data

Mô tả thuộc tính của bảng dữ liệu

- ***mpg (Miles per Gallon):*** Số dặm một chiếc xe có thể đi được với một gallon nhiên liệu (dặm/galon). Thuộc tính này đo lường hiệu suất tiêu thụ nhiên liệu của xe.
- ***cylinders (Các xi lanh):*** Số lượng các xi lanh trong động cơ của xe. Thuộc tính này cho biết động cơ của xe có bao nhiêu xi lanh.
- ***displacement (Dung tích):*** Dung tích động cơ tính bằng inch khối (in3). Thuộc tính này đo lường kích thước của động cơ.
- ***horsepower (Công suất):*** Công suất động cơ tính bằng mã lực (hp). Thuộc tính này cho biết động cơ của xe có công suất bao nhiêu.
- ***weight (Trọng lượng):*** Trọng lượng của xe tính bằng pound (lbs). Thuộc tính này đo lường khối lượng của xe.
- ***acceleration (Tăng tốc)***: Tăng tốc của xe từ 0 đến 60 dặm/giờ (giây). Thuộc tính này đo lường khả năng tăng tốc của xe.
- ***model (Năm sản xuất):*** Năm sản xuất của xe (chỉ ghi cuối của năm, ví dụ năm 1978 được ghi là 78). Thuộc tính này cho biết năm sản xuất của xe.

- origin (Xuất xứ): Mã số xác định nguồn gốc sản xuất của xe, có thể là 1, 2 hoặc 3, tương ứng với Mỹ, Châu Âu và Nhật Bản. Thuộc tính này cho biết xuất xứ của xe.
- car name (Tên xe): Tên của xe (dưới dạng chuỗi ký tự). Thuộc tính này cho biết tên của xe.

1. Có bao nhiêu xe ô tô và bao nhiêu thuộc tính trong tập dữ liệu.

```
#1. How many cars and how many attributes are in the data set.
print("Number of cars: ", data.shape[0])
print("Number of attributes: ", data.shape[1])
```

Kết quả

```
Number of cars:  406
Number of attributes:  9
```

Có 406 xe ô tô và 9 thuộc tính trong tập dữ liệu

2. Có bao nhiêu công ty ô tô riêng biệt được đại diện trong bộ dữ liệu? Tên của chiếc xe với MPG lớn nhất là gì? Hãng xe nào sản xuất nhiều xe 8 xi-lanh nhất? Tên của những chiếc xe 3 xi-lanh là gì?

Thêm cột hãng xe vào data

```
#Lấy dữ liệu tên hãng xe
data['car_company'] = data["car_name"].str.split().str[0]
print(data)
```

Kết quả

	mpg	cylinders	displacement	horsepower	...	model	origin	car_name	car_company
0	18.0	8.0	307.0	130.0	...	70.0	1.0	chevrolet chevelle malibu	chevrolet
1	15.0	8.0	350.0	165.0	...	70.0	1.0	buick skylark 320	buick
2	18.0	8.0	318.0	150.0	...	70.0	1.0	plymouth satellite	plymouth
3	16.0	8.0	304.0	150.0	...	70.0	1.0	amc rebel sst	amc
4	17.0	8.0	302.0	140.0	...	70.0	1.0	ford torino	ford
..	...	...	...	...	...	...	...	...	...
401	27.0	4.0	140.0	86.0	...	82.0	1.0	ford mustang gl	ford
402	44.0	4.0	97.0	52.0	...	82.0	2.0	vw pickup	vw
403	32.0	4.0	135.0	84.0	...	82.0	1.0	dodge rampage	dodge
404	28.0	4.0	120.0	79.0	...	82.0	1.0	ford ranger	ford
405	31.0	4.0	119.0	82.0	...	82.0	1.0	chevy s-10	chevy

- Có bao nhiêu công ty ô tô riêng biệt được đại diện trong bộ dữ liệu?

```
#How many distinct car companies are represented in the data set?
companies_dis = np.unique(data['car_company'] )
print("Number distinct car companies are represented: ", len(companies_dis),'\n')
```

Kết quả

```
Number distinct car companies are represented:  38
```

Vậy có 38 công ty ô tô riêng biệt

- Tên của chiếc xe với MPG lớn nhất là gì?

```
# What is the name of the car with the best MPG?
index_MPG_max = np.argmax(data['mpg'])
best_MPG = data.iloc[index_MPG_max]['car_name']
print("The name of the car with the best MPG: ", best_MPG,'\n')
```

Kết quả

```
The name of the car with the best MPG:  mazda glc
```

Vậy xe mazda glc là xe có MPG lớn nhất

- Hãng xe nào sản xuất nhiều xe 8 xi-lanh nhất?

```
#Car company produced the most 8-cylinder cars
temp = data.loc[data['cylinders']==8.0]
car_counts = temp['car_company'].value_counts()
print("Car company produced the most 8-cylinder cars: ", car_counts.idxmax())
print("The company's number of 8-cylinder vehicles : ", car_counts.max()),'\n'
```

Kết quả

```
Car company produced the most 8-cylinder cars:  ford
The company's number of 8-cylinder vehicles :  22
```

Hãng xe ford sản xuất nhiều xe có 8 xi-lanh nhất, với số lượng là 22 chiếc

- Tên của những chiếc xe 3 xi-lanh?

```
#The names of 3-cylinder cars
tempt = data.loc[data['cylinders']==3.0]
name_car = tempt['car_name'].to_list()
print(name_car,'\n')
```

Kết quả

```
['mazda rx2 coupe', 'maxda rx3', 'mazda rx-4', 'mazda rx-7 gs']
```

Các xe có 3 xi-lanh là : 'mazda rx2 coupe', 'maxda rx3', 'mazda rx-4', 'mazda rx-7 gs'

3. Phạm vi, giá trị trung bình và độ lệch chuẩn của mỗi thuộc tính là gì?

Kiểm tra missing value

```
# Check missing value
print(data.isnull().sum())
```

```
mpg             8
cylinders       0
displacement    0
horsepower      6
weight          0
acceleration    0
model           0
origin          0
car_name        0
car_company     0
dtype: int64
```

- Nhận xét: Vì số lượng missing value nhỏ so với kích thước dữ liệu, nhóm quyết định giữ nguyên missing value để tránh mất mát dòng dữ liệu quan trọng

Phạm vi, giá trị trung bình và độ lệch chuẩn của mỗi thuộc tính

```
# What is the range, mean, and standard deviation of each attribute
print(data.describe().loc[['mean','std','min','max']])
```

Kết quả

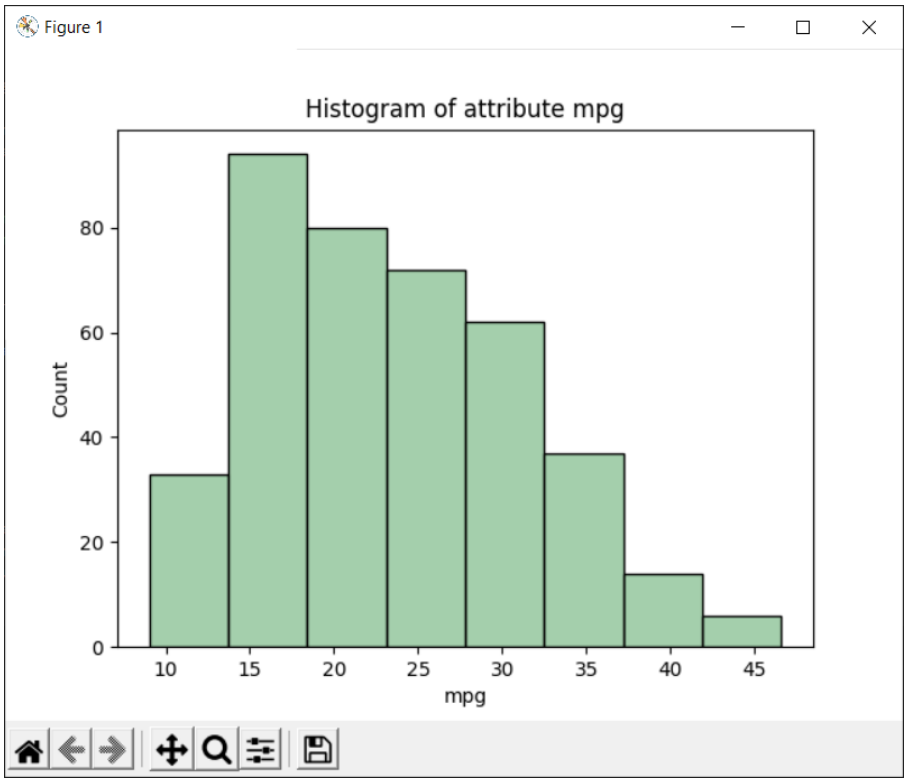
	mpg	cylinders	displacement	...	acceleration	model	origin
mean	23.514573	5.475369	194.779557	...	15.519704	75.921182	1.568966
std	7.815984	1.712160	104.922458	...	2.803359	3.748737	0.797479
min	9.000000	3.000000	68.000000	...	8.000000	70.000000	1.000000
max	46.600000	8.000000	455.000000	...	24.800000	82.000000	3.000000

[4 rows x 8 columns]

4. Vẽ biểu đồ cho từng thuộc tính. Hãy chú ý đến việc lựa chọn số lượng thùng phù hợp. Viết 2-3 câu tóm tắt một số khía cạnh thú vị của dữ liệu bằng cách nhìn vào biểu đồ.

```
names = ['mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'model', 'origin']
for col in names:
    sns.histplot(data=data, x=col, bins=8, kde=False, color='#86bf91', fill=True)
    plt.title("Histogram of attribute " + col)
    plt.show()
```

Kết quả biểu đồ



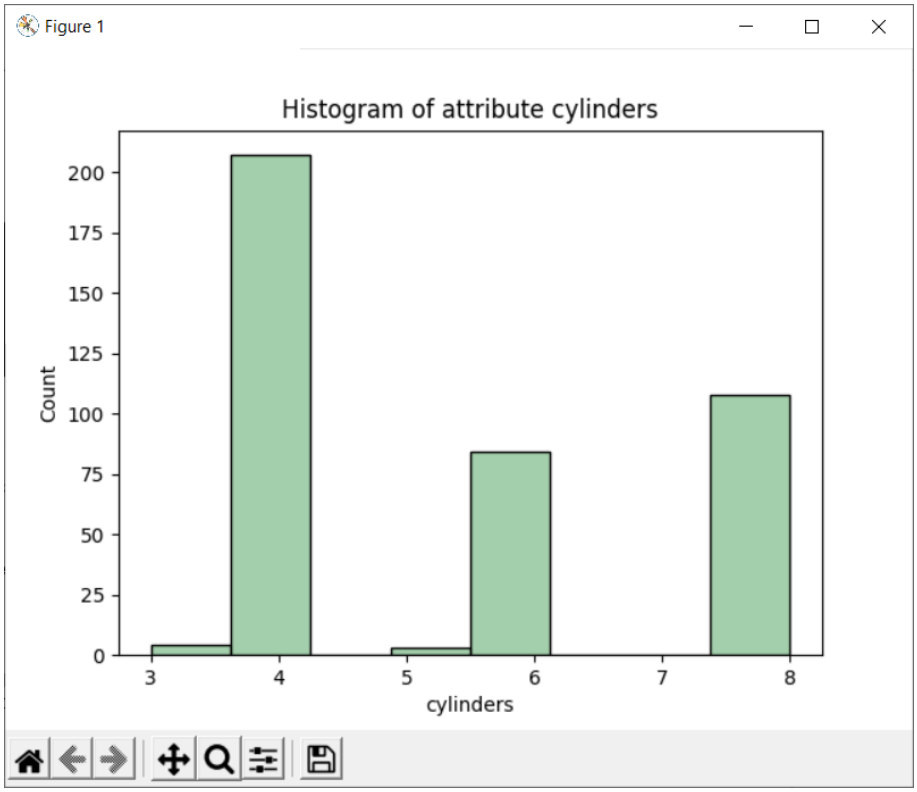


Figure 1
Histogram of attribute cylinders
Count
cylinders
0
25
50
75
100
125
150
175
200
3
4
5
6
7
8

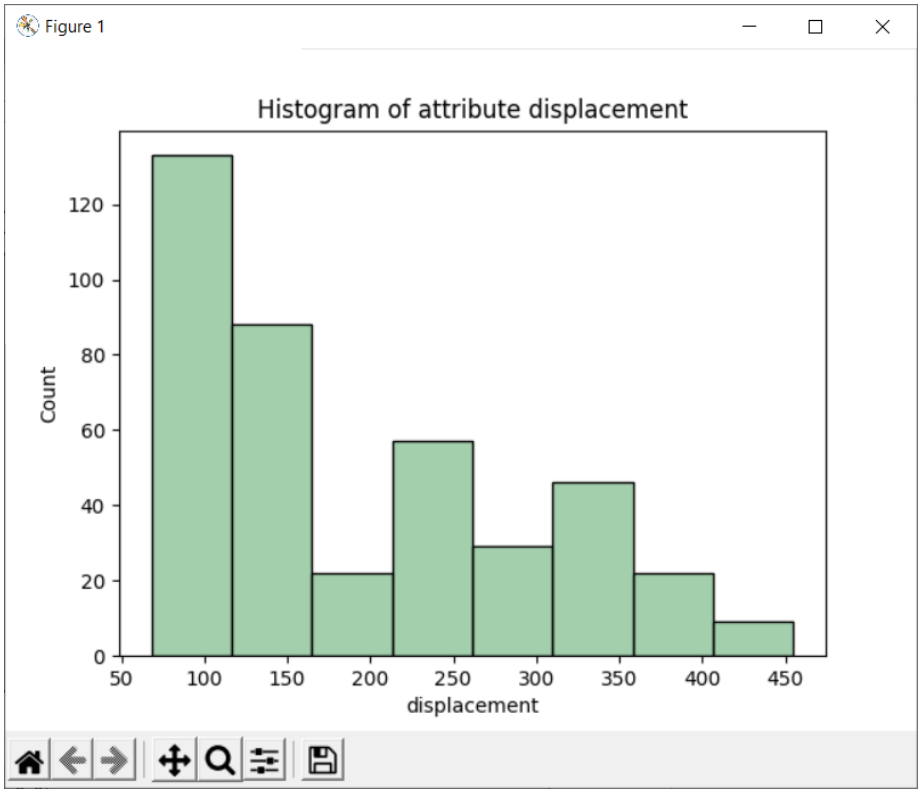


Figure 1
Histogram of attribute displacement
Count
displacement
0
20
40
60
80
100
120
50
100
150
200
250
300
350
400
450

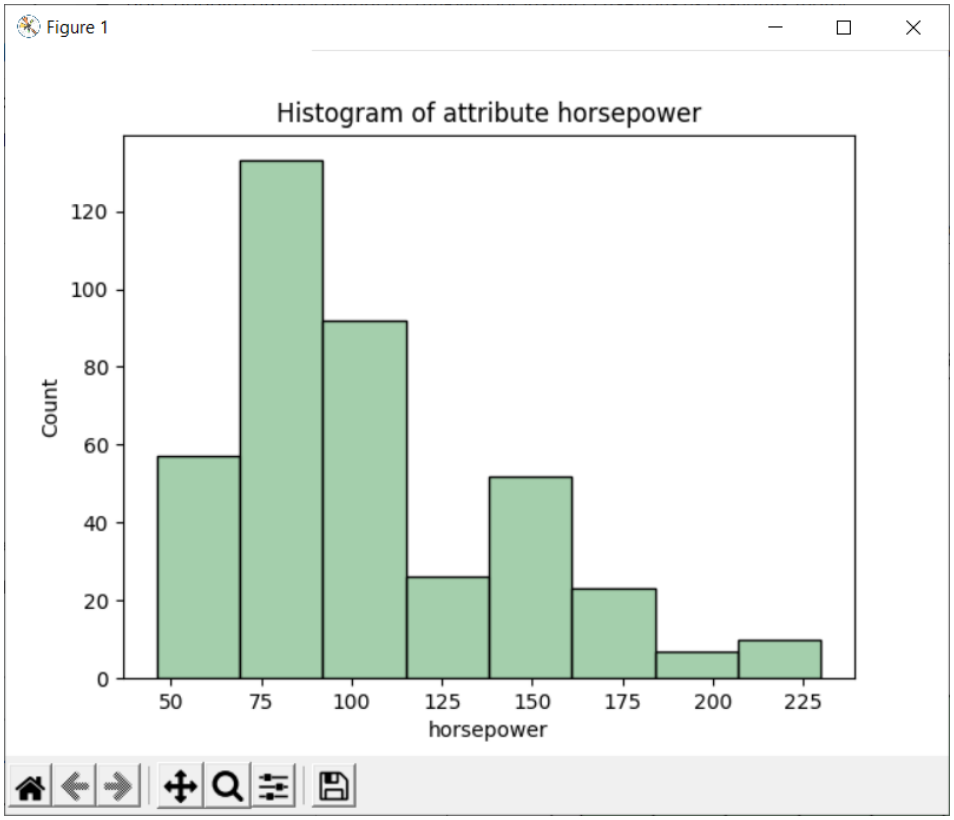


Figure 1
Histogram of attribute horsepower
Count
0
20
40
60
80
100
120
50
75
100
125
150
175
200
225
horsepower

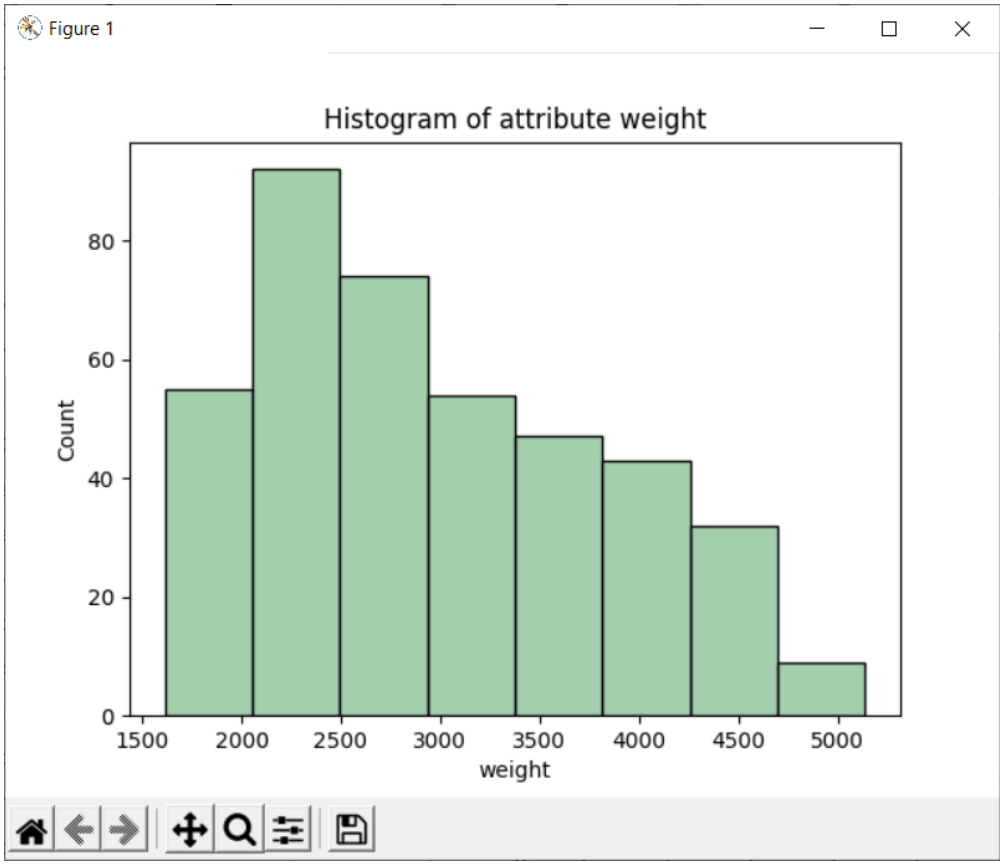


Figure 1
Histogram of attribute weight
Count
0
20
40
60
80
1500
2000
2500
3000
3500
4000
4500
5000
weight

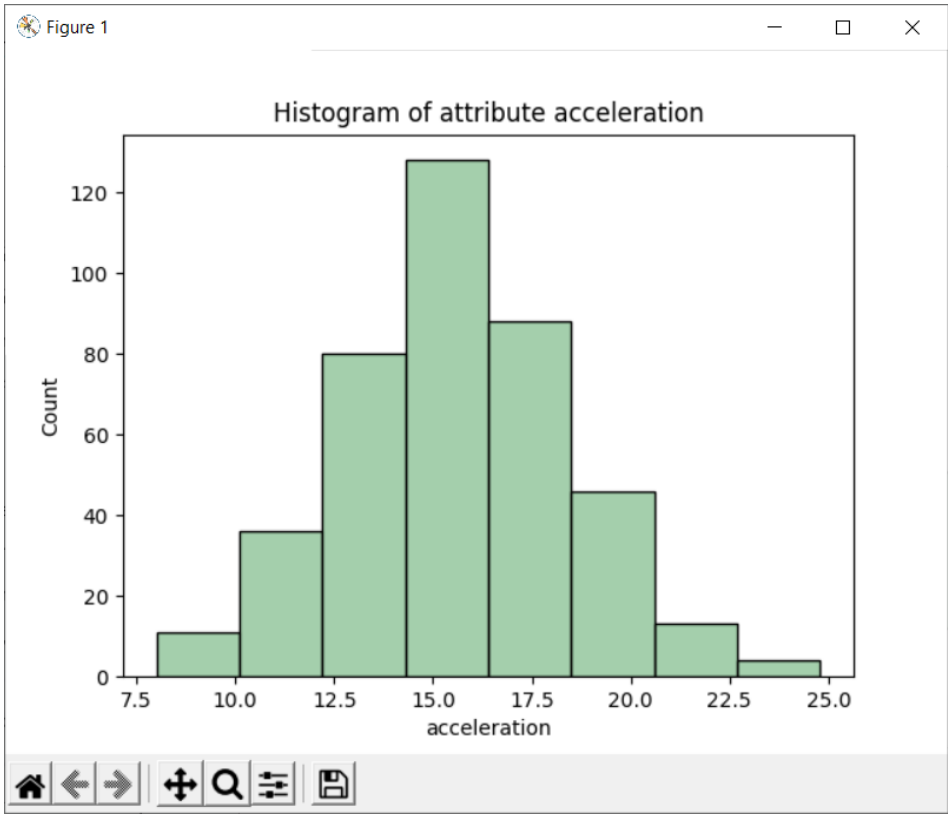


Figure 1
Histogram of attribute acceleration
Count
acceleration
0
20
40
60
80
100
120
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0

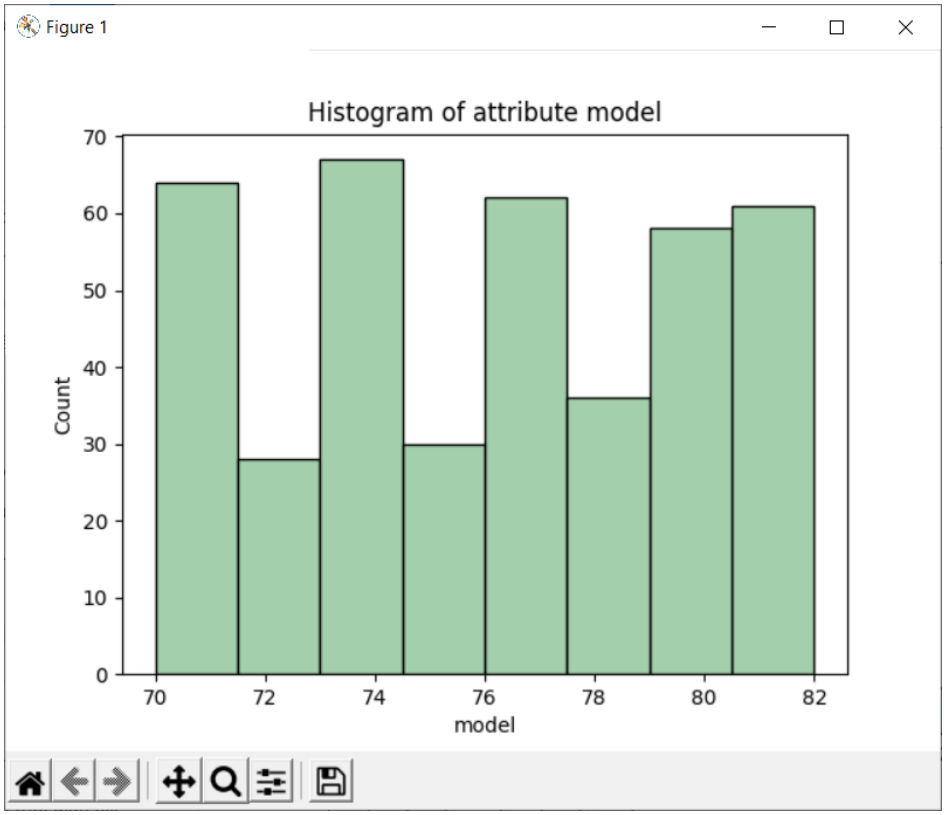
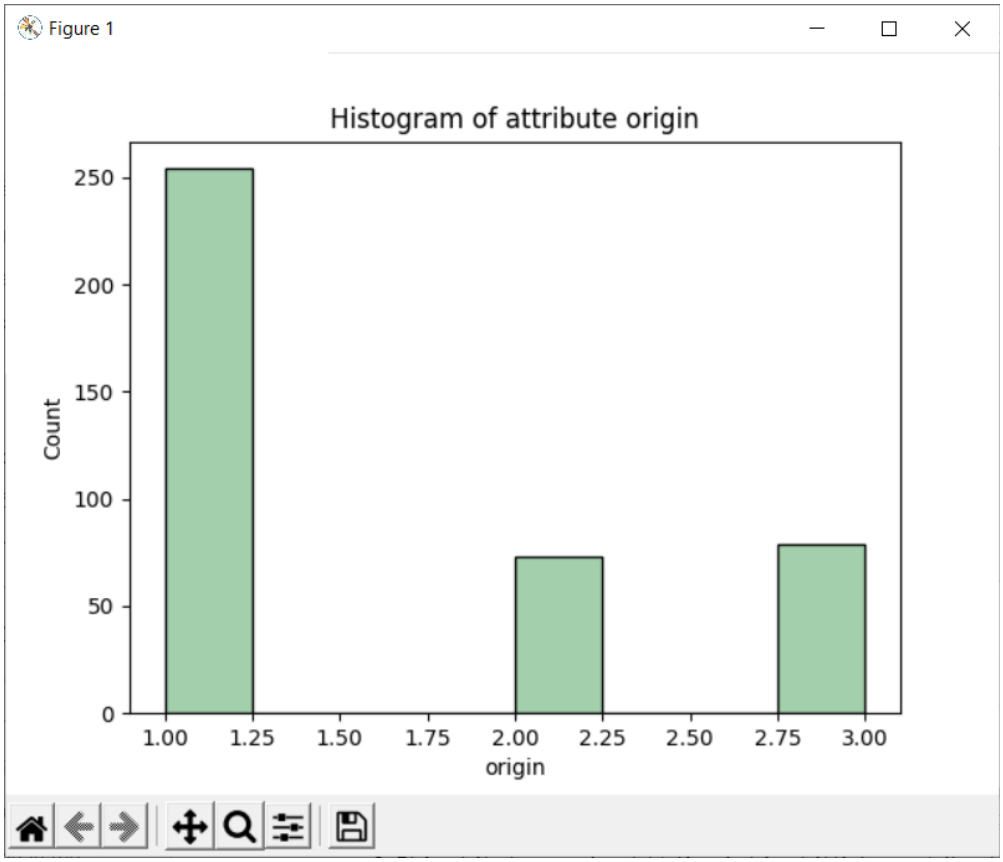


Figure 1
Histogram of attribute model
Count
model
0
10
20
30
40
50
60
70
70
72
74
76
78
80
82



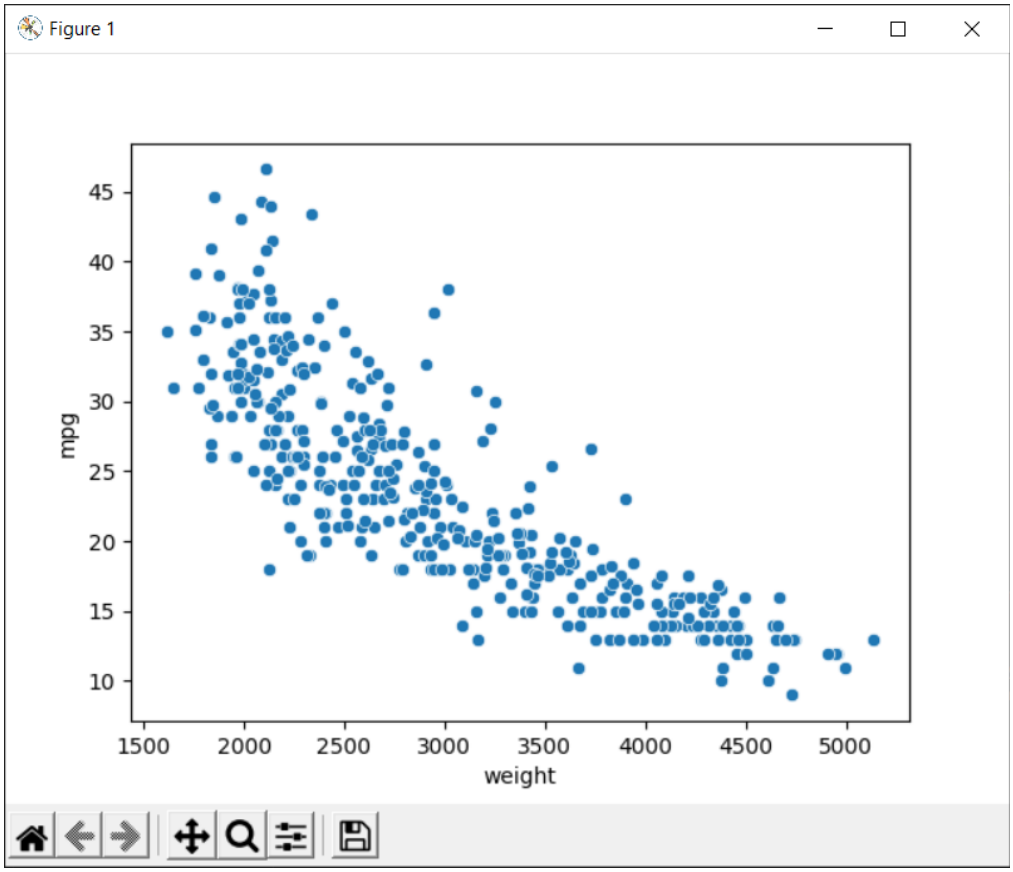
Nhận xét về biểu đồ:

1. Dữ liệu về tốc độ tăng tốc của xe - acceleration có phân phối chuẩn, điều này cũng có nghĩa là các loại xe tập trung vào thiết kế các xe có acceleration trung bình
2. Phân phối của mpg và weight đều có phân phối lệch sang trái, nghĩa là số lượng xe được tạo với mpg và weight càng lớn thì càng ít dần
3. Trong phân phối của cyclines, dữ liệu tập trung nhất ở giá trị 4, nghĩa là các loại xe chủ yếu sẽ là 4 xi lanh

5. Vẽ một biểu đồ phân tán của các thuộc tính trọng lượng so với MPG. Bạn kết luận gì về mối quan hệ giữa các thuộc tính? Hệ số tương quan giữa 2 thuộc tính là gì?

```
sns.scatterplot(x="weight", y="mpg", data=data)
plt.show()
```

Kết quả biểu đồ



Nhận xét biểu đồ: Hai dữ liệu có sự tương quan cao: Khi dữ liệu của weight càng tăng thì mpg càng giảm. Điều này có nghĩa weight và mpg tương quan âm.

```
corr = data["weight"].corr(data["mpg"])

print("Hệ số tương quan giữa weight vs. MPG  : ", round(corr,2))
```

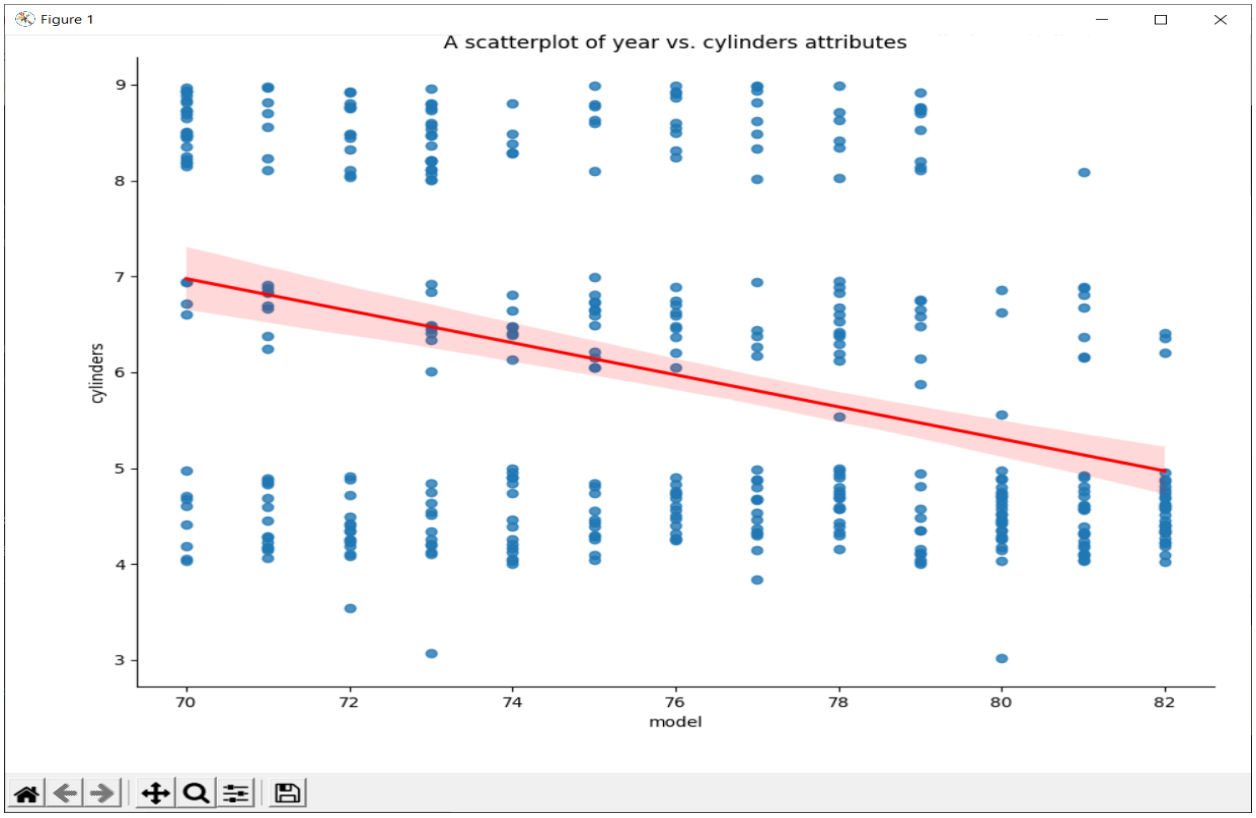
Kết quả

```
Hệ số tương quan giữa weight vs. MPG  :  -0.83
```

6. Vẽ biểu đồ phân tán của năm so với các thuộc tính xi lanh. Thêm một nhiễu ngẫu nhiên nhỏ vào các giá trị để làm cho biểu đồ phân tán trông đẹp hơn. Bạn có thể kết luận điều gì?

```
tempt = data
tempt.cylinders = tempt.cylinders + np.random.random(len(tempt.cylinders))
sns.lmplot(x="model", y="cylinders", data=tempt, line_kws={'color': 'red'})
plt.title("A scatterplot of year vs. cylinders attributes")
plt.show()
```

Kết quả biểu đồ



Nhận xét biểu đồ:

- Dựa vào biểu đồ ta thấy được rằng số lượng cylinder giảm qua các năm
  - Ta có thể biết rằng khi số lượng cylinder càng lớn thì công suất và tốc độ xe cũng tăng theo. Đây có thể là nguyên do chính dẫn ra xu hướng thiết kế xe hơi có càng ít xi lanh càng tốt. Ngoài ra cũng có thể vì một số nhu cầu sau:
    - Trong những năm 70 và đầu những năm 80, chính phủ Hoa Kỳ đã đưa ra những quy định nghiêm ngặt về khí thải độc hại

của các phương tiện giao thông, nhằm bảo vệ môi trường và sức khỏe con người. Điều này đã đẩy các nhà sản xuất xe hơi phải thay đổi thiết kế và sử dụng công nghệ mới để giảm thiểu khí thải. Trong đó, giảm số lượng xi lanh trên mỗi xe là một trong những giải pháp được áp dụng. Việc giảm số lượng xi lanh trên mỗi xe giúp giảm lượng nhiên liệu tiêu thụ và do đó giảm khí thải.

- Ngoài ra, cũng có sự tiến bộ trong công nghệ sản xuất động cơ, giúp tăng hiệu suất và sử dụng nhiên liệu một cách hiệu quả hơn.
- uy nhiên, trong thời kỳ này, khi nền kinh tế Mỹ bị suy thoái, sự cạnh tranh về giá thành ô tô làm giảm doanh số bán xe của Mỹ, vì thế Mỹ đã tập trung sản xuất xe nhỏ gọn tiết kiệm nhiên liệu và giá thành thấp hơn để cải thiện doanh số

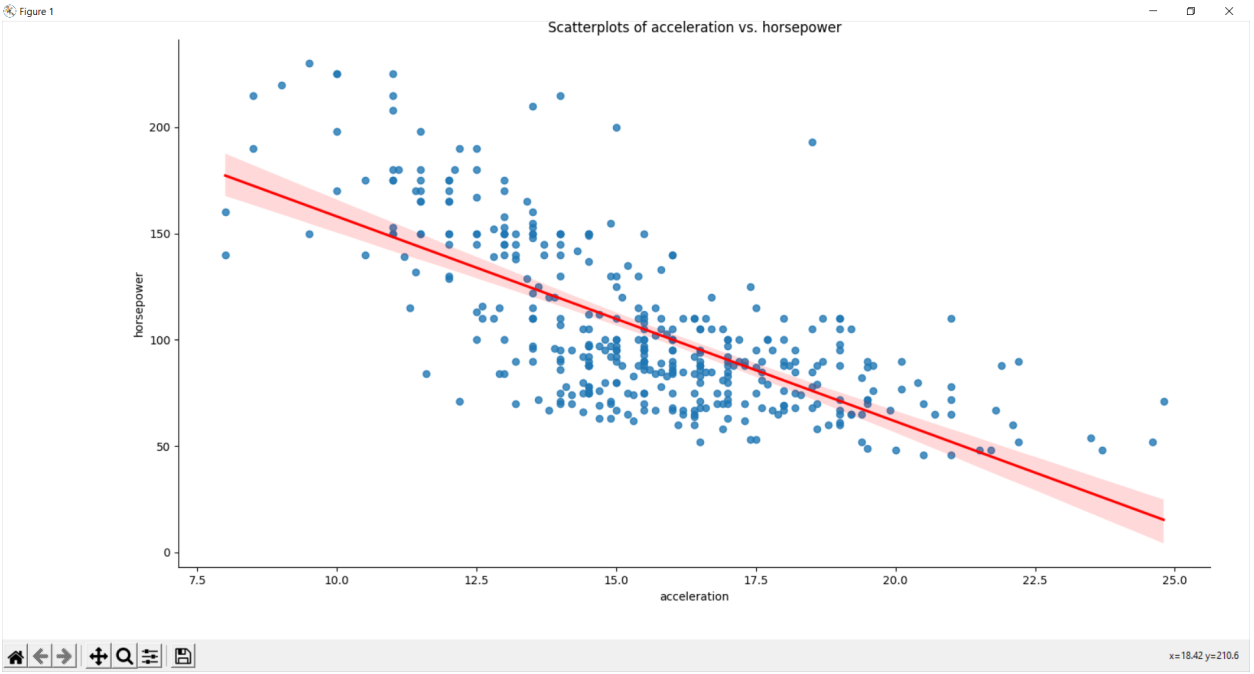
⇒ Vì vậy số lượng Cylinder của xe hơi có xu hướng giảm.

7. Hiển thị thêm 2 biểu đồ phân tán thú vị. Thảo luận về những gì bạn nhìn thấy.

Scatterplots of acceleration vs. horsepower

```
sns.lmplot(x="acceleration", y="horsepower", data=data, line_kws={'color': 'red'})
plt.title("Scatterplots of acceleration vs. horsepower")
plt.show()
```

Kết quả biểu đồ



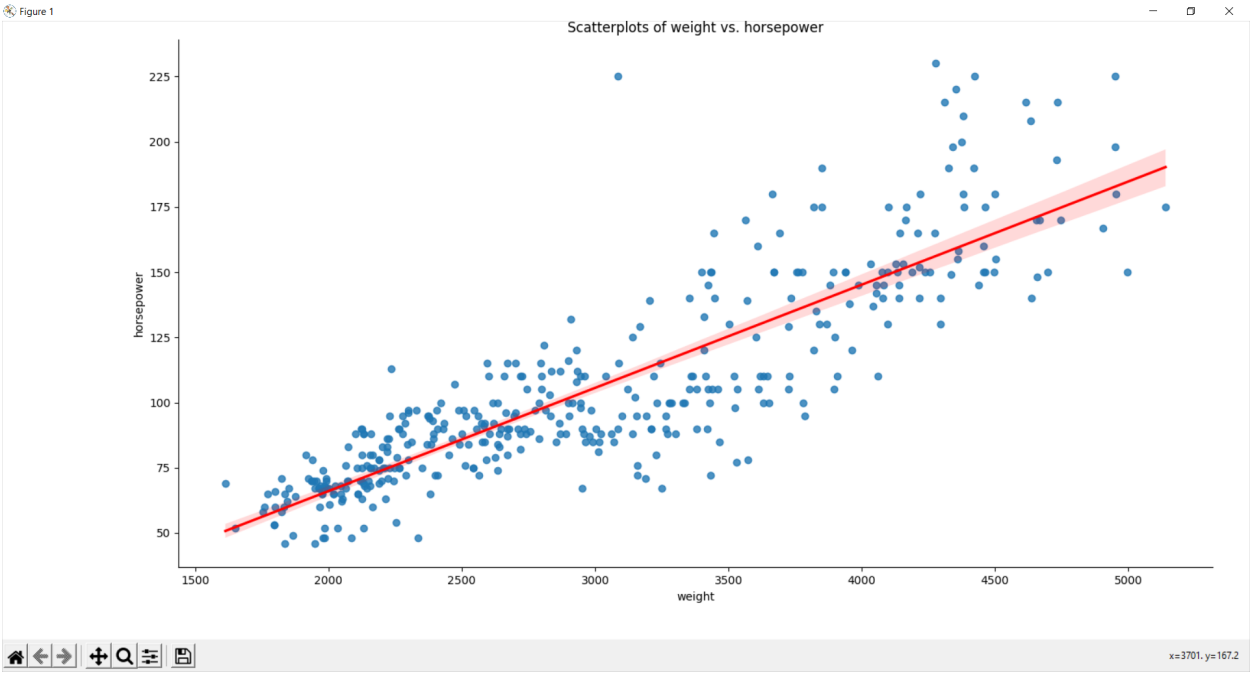
Nhận xét:

- Horsepower và acceleration có mối tương quan âm, vì vậy ta có thể hiểu:
  - Tốc độ tăng tốc của động cơ càng giảm thì công suất của động cơ càng lớn, xe càng mạnh.
  - Điều này liên quan đến cách mà công suất được tính toán. Trong một động cơ, công suất được tính bằng công việc được thực hiện trong một đơn vị thời gian. Tốc độ tăng tốc của động cơ càng chậm, thì động cơ càng dễ dàng đạt được mức công suất cao hơn.

Scatterplots of weight vs. horsepower

```
sns.lmplot(x="weight", y="horsepower", data=data, line_kws={'color': 'red'})
plt.title("Scatterplots of weight vs. horsepower")
plt.show()
```

Kết quả biểu đồ



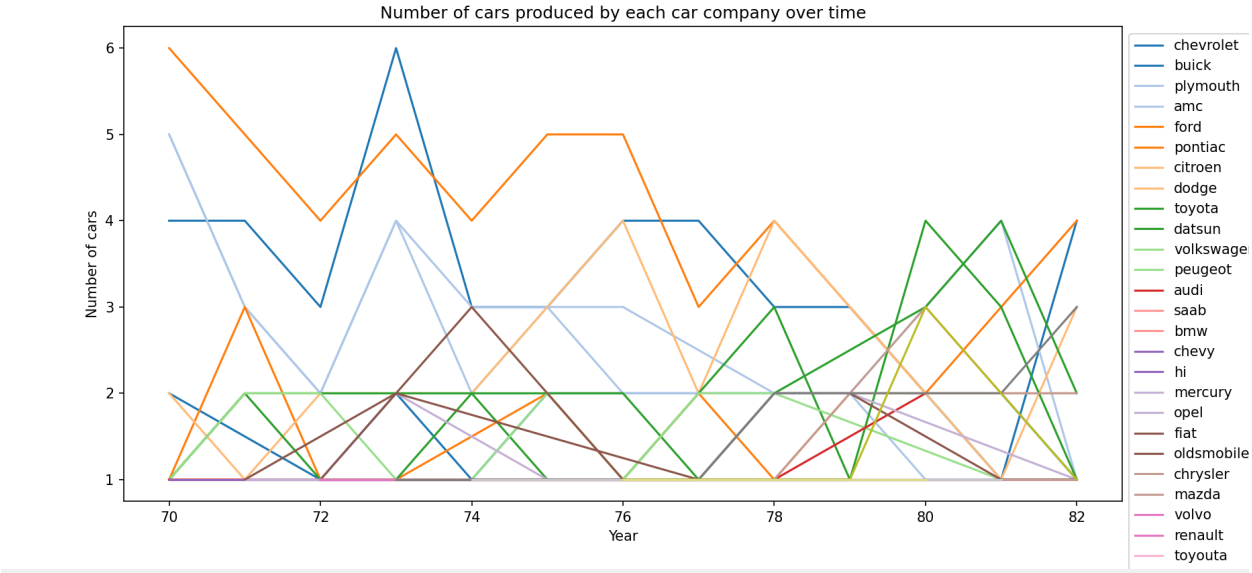
Nhận xét:

- Weight và horsepower có mối tương quan dương, vì vậy ta có thể hiểu:
- Khi trọng lượng của xe càng tăng thì công suất của động cơ càng lớn. CÓ lẽ vì thường thì những chiếc xe hơi có động cơ công suất lớn sẽ có trọng lượng tương đối nặng để đảm bảo khả năng vận hành ổn định và an toàn

8. Vẽ một chuỗi thời gian cho tất cả các công ty cho biết họ giới thiệu bao nhiêu xe mới trong mỗi năm. Bạn có thấy một số xu hướng thú vị?

```
df_grouped = data.groupby(['car_company', 'model']).count().reset_index()

fig,ax = plt.subplots(figsize=(25, 10))
colors = cm.tab20(np.linspace(0, 1, len(data['car_company'].unique())))
for name, color in zip(data['car_company'].unique(), colors):
    df = df_grouped[df_grouped['car_company'] == name]
    ax.plot(df['model'], df['car_name'], label=name, color=color)

ax.legend(bbox_to_anchor=(1, 1))
ax.set_xlabel('Year')
ax.set_ylabel('Number of cars')
ax.set_title('Number of cars produced by each car company over time')
plt.show()
```

Kết quả biểu đồ



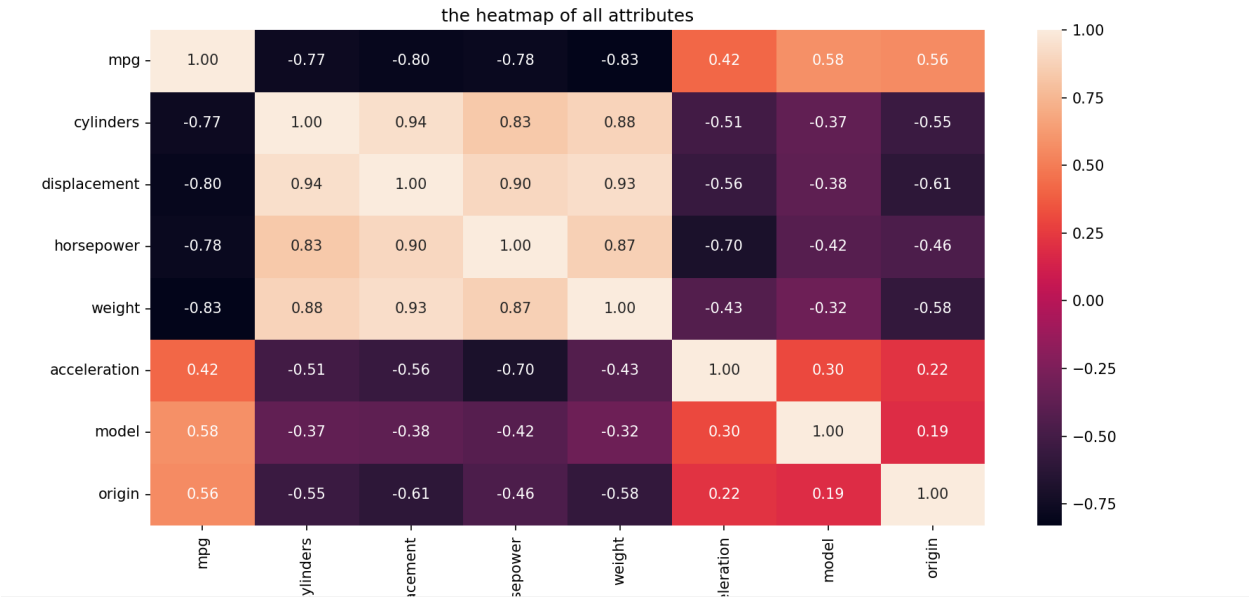
Nhận xét biểu đồ:

- Hãng xe nổi tiếng như BMW, Ford có xu hướng giảm số lượng xe ra mắt mỗi năm
- Cuối thập niên 70 đầu thập niên 80 thị trường ngày càng xôi nổi khi có các thương hiệu mới nổi đến từ Hàn, Nhật như Huyndai, Nissan, Kia,...

9. Tính toán tương quan theo cặp và vẽ bản đồ nhiệt bằng Matplotlib. Bạn có thấy một số tương quan thú vị? (Gợi ý: data.iloc[:,0:8].corr(), plt.pcolor() vẽ bản đồ nhiệt.)

```
temp = data
temp.drop(['car_name', 'car_company'], axis=1)
corr = temp.corr().round(2)
sns.heatmap(corr, annot=True, fmt=".2f")
plt.title('the heatmap of all attributes')
plt.show()
```

Kết quả biểu đồ



Nhận xét:

- Hệ số tương quan giữa các thuộc tính cylinders, displacement, horsepower và weight có sự tương quan dương với nhau.Trong đó, displacement và cylinders có sự tương quan chặt chẽ với Hệ số tương quan là 0.95
- Riêng mpg là tương quan âm với các thuộc tính cylinders, displacement, horsepower, và weight.Trong đó hệ tương quan của mpg và weight là -0.84

IV. Electric power consumption data

1. Tiền xử lý dữ liệu

Lấy dữ liệu và chuyển đổi dữ liệu Date và Time theo cấu trúc của Python

```
#Đọc file dữ liệu
df = pd.read_csv('household_power_consumption.txt', sep=';', low_memory=False)

#Chuyển kiểu dữ liệu Date
df['Date'] = pd.to_datetime(df['Date'], dayfirst= True)

#Lọc dữ liệu để lấy các bản ghi từ ngày 2007-02-01 đến 2007-02-02.
data = df.loc[(df['Date'] >= '2007-02-01') & (df['Date'] <= '2007-02-02')].reset_index(drop=True)

#Chuyển kiểu dữ liệu Time
data['Time'] = pd.to_datetime(data['Time'], format='%H:%M:%S').dt.time
data['Time'] = pd.to_datetime(data['Date'].astype(str) + ' ' + data['Time'].astype(str))
```

Tiền xử lý dữ liệu

1. Mô tả thuộc tính
    - ***Date (Ngày)*:** Ngày ghi nhận dữ liệu tiêu thụ điện năng
    - ***Time (Thời gian):*** Thời gian ghi nhận dữ liệu tiêu thụ điện năng trong ngày
    - ***Global_active_power (Công suất toàn cầu hoạt động):*** Công suất toàn bộ các thiết bị điện trong hộ gia đình đang hoạt động tính bằng đơn vị kilowatt (kW) tại một thời điểm cụ thể.
    - ***Global_reactive_power (Công suất toàn cầu phản kháng):*** Công suất toàn cầu phản kháng tính bằng kilowatt (kW) tại thời điểm đó. Công suất này là do các thiết bị điện trong hộ gia đình tạo ra để hỗ trợ quá trình chuyển đổi điện năng.
    - ***Voltage (Điện áp)*:** Điện áp tính bằng volt (V) tại thời điểm đó
    - ***Global_intensity (Dòng điện toàn cầu):*** Dòng điện của toàn bộ các thiết bị điện trong hộ gia đình đang tiêu thụ tính bằng đơn vị ampere (A) tại một thời điểm cụ thể. Nó thể hiện mức độ tiêu thụ điện hiện tại của hộ gia đình
    - ***Sub_metering_1 (Điện năng tiêu thụ con 1):*** Điện năng tiêu thụ của phòng 1 tính bằng watt-hour (Wh) tại thời điểm đó
    - ***Sub_metering_2 (Điện năng tiêu thụ con 2):*** Điện năng tiêu thụ của phòng 2 tính bằng watt-hour (Wh) tại thời điểm đó
    - ***Sub_metering_3 (Điện năng tiêu thụ con 3):*** Điện năng tiêu thụ của phòng 3 tính bằng watt-hour (Wh) tại thời điểm đó

2. Kiểm tra kiểu dữ liệu của các thuộc tính

```
print(data.info())
```

Kết quả

```
Data columns (total 9 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Date                   2880 non-null   datetime64[ns]
 1   Time                   2880 non-null   datetime64[ns]
 2   Global_active_power    2880 non-null   object
 3   Global_reactive_power  2880 non-null   object
 4   Voltage                2880 non-null   object
 5   Global_intensity       2880 non-null   object
 6   Sub_metering_1         2880 non-null   object
 7   Sub_metering_2         2880 non-null   object
 8   Sub_metering_3         2880 non-null   float64
dtypes: datetime64[ns](2), float64(1), object(6)
memory usage: 202.6+ KB
None
```

3. Chuyển kiểu dữ liệu

```
columns = ['Global_active_power','Global_reactive_power','Voltage','Global_intensity','Sub_metering_1','Sub_metering_2','Sub_metering_3']
for col in columns:
    data[col] = data[col].astype('float64')
```

Kiểm tra lại kiểu dữ liệu

```
Data columns (total 9 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Date                   2880 non-null   datetime64[ns]
 1   Time                   2880 non-null   datetime64[ns]
 2   Global_active_power    2880 non-null   float64
 3   Global_reactive_power  2880 non-null   float64
 4   Voltage                2880 non-null   float64
 5   Global_intensity       2880 non-null   float64
 6   Sub_metering_1         2880 non-null   float64
 7   Sub_metering_2         2880 non-null   float64
 8   Sub_metering_3         2880 non-null   float64
dtypes: datetime64[ns](2), float64(7)
memory usage: 202.6 KB
None
```

4. Kiểm tra missing value

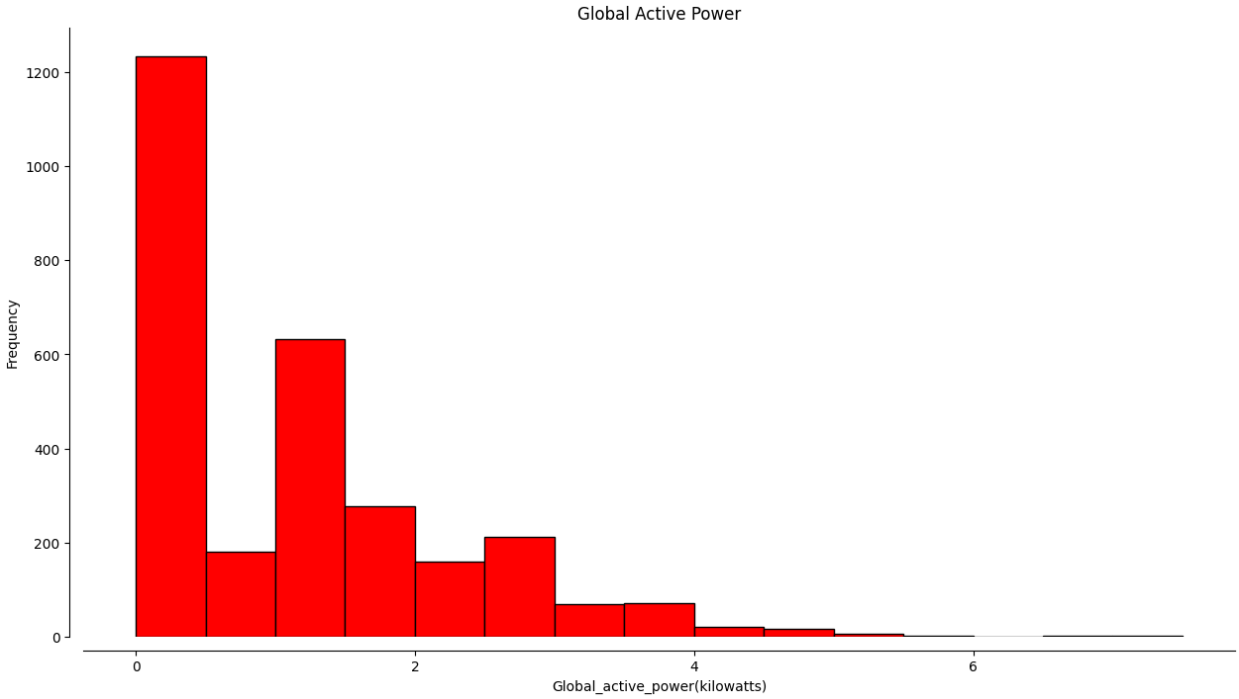
```
#    Thay thế các giá trị ? trong dataframe bằng null
data = data.replace('?', np.nan)
print(data.isna().sum())
```

Kết quả:

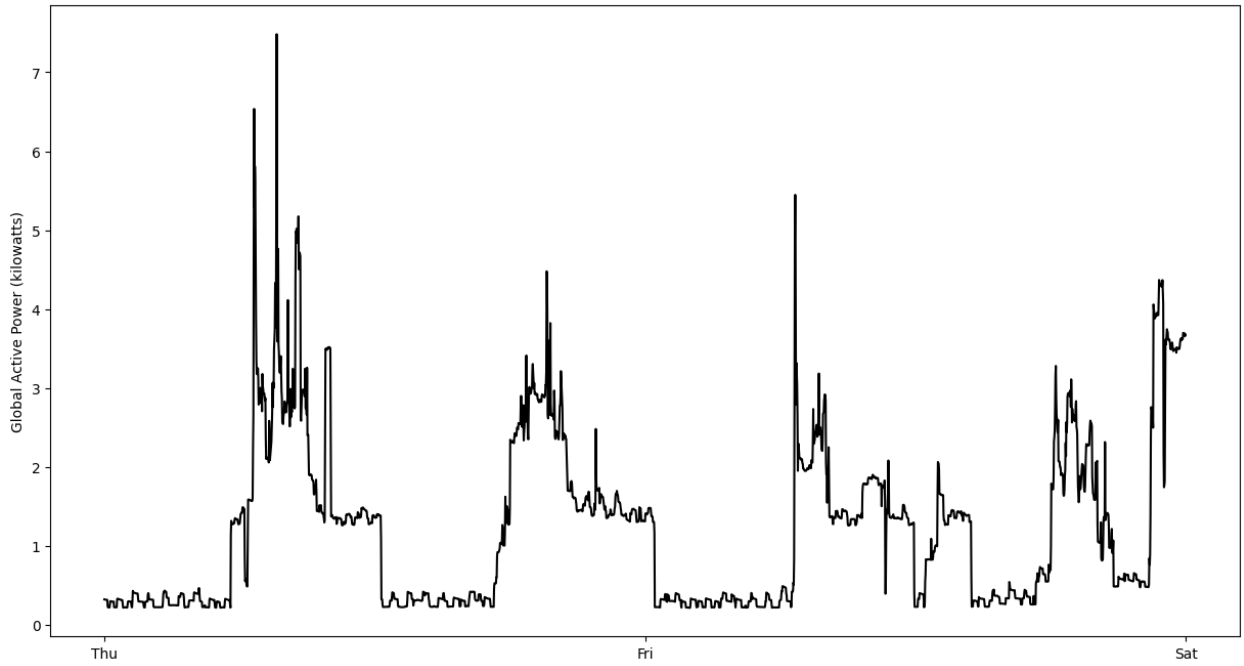
```
Date                   0
Time                   0
Global_active_power    0
Global_reactive_power  0
Voltage                0
Global_intensity       0
Sub_metering_1         0
Sub_metering_2         0
Sub_metering_3         0
dtype: int64
```

Nhận xét: Không có giá trị null trong dữ liệu này

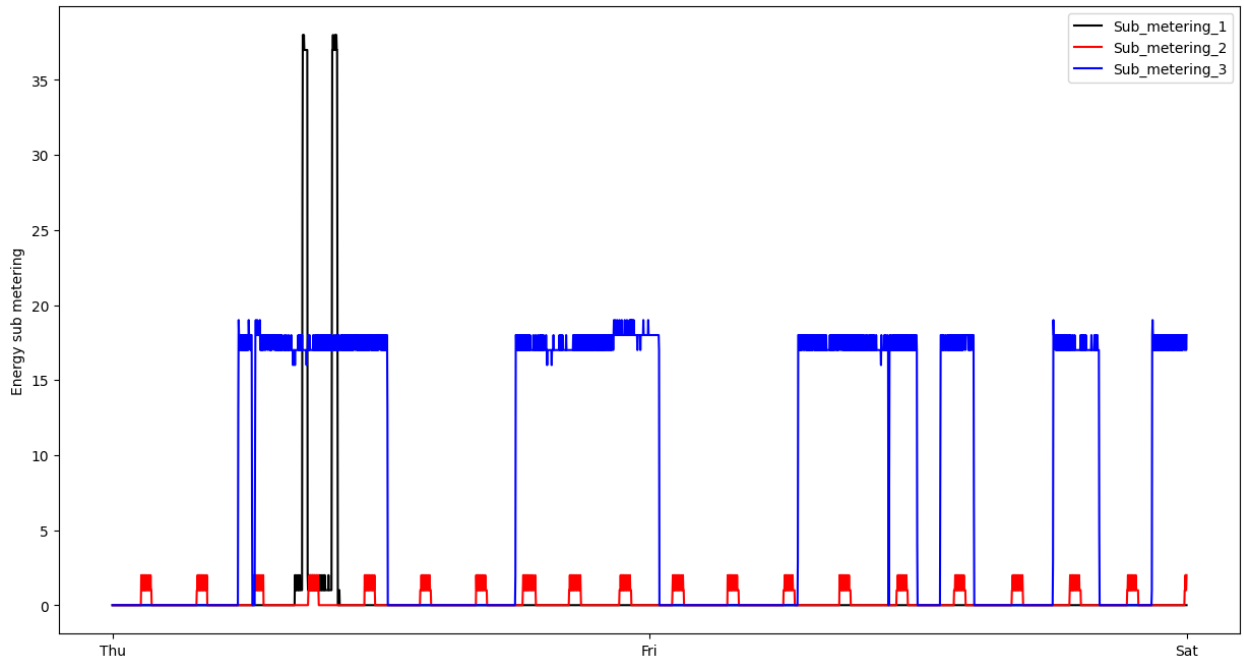
Plot1. Histogram of Global Active Power



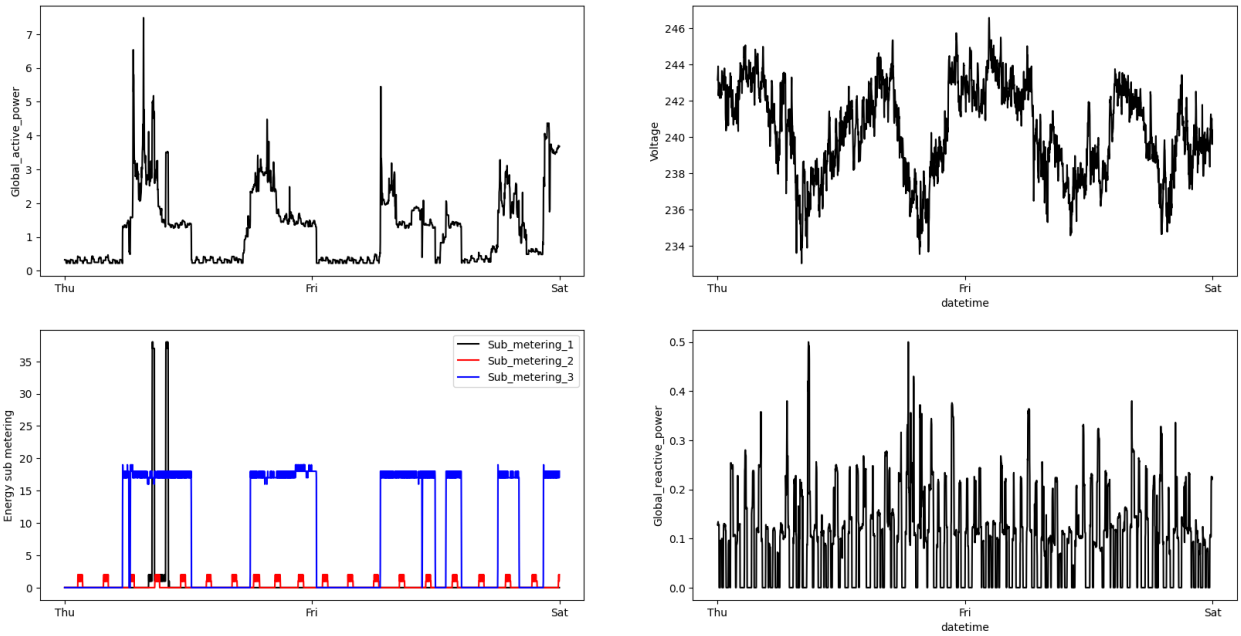
Plot2. Chart Line showing Global Active Power over time



Plot3. Chart Line showing Energy sub metering over time



Plot4. A collection of Chart representing data over time



2. StoryTelling

Dựa vào các biểu đồ trên, biểu diễn năng lượng được tiêu thụ của hộ gia đình theo thời gian được lấy bằng phút. Trong đó:

- Histogram Plot1 thể hiện Tần số tiêu thụ năng lượng toàn cầu của hộ gia đình: Có thể thấy rằng phần lớn năng lượng tiêu thụ của hộ gia đình này có xu hướng rất thấp, có thể do gia đình sử dụng ít thiết bị điện hoặc các thiết bị sử dụng điện có công suất thấp.

- Biểu đồ đường Plot2 cho biết về năng lượng tiêu thụ toàn cầu của tất cả các thiết bị trong hộ gia đình theo từng phút:
    - Có thể thấy đường biểu diễn của ngày 01/02 (Là khoảng từ Thu đến Fri) và ngày 02/02 (từ Fri đến Sat) có sự tương đồng về hình dạng, điều đó nói rằng có thể liên quan đến thói quen sinh hoạt hàng ngày của các hộ gia đình.

    - Có thể thấy rằng các đỉnh xảy ra vào khoảng thời gian ban ngày (trưa tới chiều) và ban đêm (tối tới khuya). Có thể đưa giả thiết các đỉnh xảy ra là các hộ gia đình đa phần tiêu thụ năng lượng khi họ sinh hoạt các hoạt đồng thường ngày và ngược lại là năng lượng tiêu thụ thấp khi họ nghỉ ngơi.

- Biểu đồ đường Plot3 cho biết thông tin giá trị về điện năng tiêu thụ của 3 khu vực khác nhau trong nhà:
    - Đường biểu diễn điện năng của khu vực 1 (màu đen) cho thấy khu vực này chỉ sử dụng điện năng vào khoảng một thời gian duy nhất của ngày 01/02, có lẽ khu vực này không được sử dụng hằng ngày, tuy nhiên điện năng tiêu thụ cho khu vực này là khá cao, thiết bị được sử dụng ở khu vực này là các thiết bị tiêu tốn nhiều điện năng, như máy giặt, máy sấy
    - Đường biểu diễn điện năng của khu vực 2 (màu đỏ) cho thấy khu vực này tiêu thụ lượng điện năng thấp nhưng đều đặn mỗi giờ theo thời gian. Điều đó thể hiện rằng hộ gia đình có thể sử dụng các thiết bị luôn luôn hoạt động nhưng tiêu tốn ít điện năng, chẳng hạn như đồng hồ điện.
    - Đường biểu diễn điện năng của khu vực 3 (màu xanh) cho thấy khu vực này tiêu thụ điện năng ở mức trung bình và có tần suất cố định, các đỉnh xảy ra vào khung giờ buổi trưa và buổi tối. Điều này có sự tương đồng với biểu đồ Plot2, vì vậy có thể biết rằng các thiết bị sử dụng trong khu vực 3 đều phục vụ cho hoạt động sinh hoạt thường ngày trong gia đình như nấu ăn, sử dụng thiết bị điện tử, Tivi,... Và khu vực này cũng chính là khu vực ảnh hưởng đến tổng điện năng tiêu thụ lớn nhất.
- Trong các mô hình của Plot4, các biểu đồ này cho ta thấy rằng hộ gia đình đang sử dụng năng lượng một cách hiệu quả và theo thói quen sinh hoạt hằng ngày.
    - Phần lớn tiêu thụ năng lượng nằm trong các khoảng thấp hơn. Tuy nhiên, có những đỉnh trong mô hình Global Active Power over time (Hình 1 - Plot2) và biểu đồ Energy sub metering over time (Hình 2 - Plot3) có một số thiết bị hoặc khu vực trong nhà đang sử dụng năng lượng quá mức cần thiết.
    - Hình 4 trong mô hình của Plot 4 cho biết công suất của các thiết bị điện tạo ra để hỗ trợ quá trình chuyển đổi điện năng. Ở đây ta thấy có sự tương đồng với Hình 2 là tổng điện năng tiêu thụ của các thiết bị điện, điều này dễ hiểu vì điện năng

tiêu thụ càng nhiều thì quá trình chuyển đổi điện năng có công suất càng lớn.

- Hình 3 trong mô hình của Plot4 cho biến điện áp được tính theo phút, ta có thể thấy sự tương phản của Hình 1 và Hình 3. Đúng như vậy bởi vì điện áp và công suất tiêu thụ điện có mối quan hệ nghịch biến, khi điện áp càng lớn thì công suất tiêu thụ càng thấp và ngược lại

Tổng thể, các biểu đồ trên có thể giúp các hộ gia đình về việc giảm tiêu thụ năng lượng ở những khu vực này có thể giúp hộ gia đình tiết kiệm năng lượng và giảm chi phí điện.