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

TRỰC QUAN HÓA DỮ LIỆU

Sinh viên thực hiện:

18120408 - Trần Ngọc Lan Khanh

18120401 – Mai Khánh Huyền

18120379 – Võ Thị Hiếu

18120378 – Trần Văn Hiếu

GV phụ trách: Lê Ngọc Thành

HỌC KỲ II – NĂM HỌC 2020-2021

BẢNG THÔNG TIN CHI TIẾT NHÓM

MSSV	Họ tên	Email
18120408	Trần Ngọc Lan Khanh	18120408@student.hcmus.edu.vn
18120401	Mai Khánh Huyền	18120401@student.hcmus.edu.vn
18120379	Võ Thị Hiếu	18120379@student.hcmus.edu.vn
18120378	Trần Văn Hiếu	18120378@student.hcmus.edu.vn

MỨC ĐỘ HOÀN THÀNH TỔNG THỂ CỦA MỖI YÊU CẦU

Yêu cầu	Tự đánh giá
1. Mô tả các thư viện liên quan đến chức năng trực quan hóa dữ liệu	100%
2. Thực hiện theo yêu cầu và giải thích chi tiết kiến thức liên quan	100%

MỨC ĐỘ HOÀN THÀNH CỦA MỖI THÀNH VIÊN TRONG NHÓM

Thành viên	Công việc	Mức độ hoàn thành
Trần Ngọc Lan Khanh	Trình bày báo cáo, phân tích và trực quan Car MPG data	100%
Võ Thị Hiếu	Phân tích Electric power consumption data	100%
Mai Khánh Huyền	Phân tích và trực quan Car MPG data	100%
Trần Văn Hiếu	Trình bày báo cáo, phân tích và trực quan Electric power consumption data	100%

MỤC LỤC

I. Mô tả các thư viện liên quan đến chức năng trực quan hóa dữ liệu

1 Giới thiệu về thư viện Numpy

Numpy là một thư viên được sử dụng phổ biến trong ngành khoa học máy tính của Python. Numpy cung cấp nhiều hàm tối ưu cho việc hỗ trợ tính toán trên các mảng nhiều chiều và có kích thước lớn. Numpy đặc biệt hữu ích khi thực hiện các tính toán liên quan đêns đại số tuyến tính. Sau đây là một số hàm phổ biến trong thư viện Numpy

1.1 Tạo mảng:

- array() là một hàm được sử dụng rất phổ biến trong python với tác dụng tạo ra một numpy array.

```
In [4]: array = np.array([1, 2, 4, 5])
        array

Out[4]: array([1, 2, 4, 5])
```

- zeros()/ones(): tạo ra một numpy array với tất cả giá trị là 0/1

```
In [10]: zeros = np.zeros((5, 3))
         zeros

Out[10]: array([[0., 0., 0.],
                [0., 0., 0.],
                [0., 0., 0.],
                [0., 0., 0.],
                [0., 0., 0.]])
```

- full(): là một hàm tạo ra một numpy array với một giá trị định trước

```
In [12]: fives = np.full((3,2), 5)
         fives

Out[12]: array([[5, 5],
                [5, 5],
                [5, 5]])
```

- shape(): Hàm lấy ra số dòng và số cột của mảng theo công thức (số hàng, số cột)

-

```
In [6]: array.shape

Out[6]: (4,)
```

1.2 Truy xuất phần tử

- Sử dụng cách truy xuất như một mảng thông thường**:**

```
In [15]: arr = np.array([[1, 2, 3], [4, 5, 6]])
         arr[1,1]

Out[15]: 5
```

- Sử dụng slice:

-

```
In [19]: arr1 = arr[0:1, 0:2]
         arr1

Out[19]: array([[1, 2]])
```

1.3 Phép toán

Các phép toán cơ bản:

```
In [28]: x = np.array([[1,2],[3,4]])
         y = np.array([[5,6],[7,8]])
         print("x: ",x);
         print("y: ", y);

         x:  [[1 2]
          [3 4]]
         y:  [[5 6]
          [7 8]]

In [29]: print("Tổng: ", np.add(x, y));

         Tổng:  [[ 6  8]
          [10 12]]

In [30]: print("Hiệu: ", np.subtract(x, y));

         Hiệu:  [[-4 -4]
          [-4 -4]]

In [31]: print("Tích: ", np.multiply(x, y));

         Tích:  [[ 5 12]
          [21 32]]

In [32]: print("Thương: ", np.divide(x, y));

         Thương:  [[0.2        0.33333333]
          [0.42857143 0.5         ]]
```

- Nhân vector:

```
In [33]: np.dot(x, y)

Out[33]: array([[19, 22],
                [43, 50]])
```

- Tổng các phần tử:

```
In [34]: array = np.array([1, 2, 4, 5])
         sum(array)

Out[34]: 12
```

2 Pandas

2.1 Giới thiệu

Pandas là một thư viện mã nguồn mở được sử dụng phổ biến trong phân tích dữ liệu của ngôn ngữ Python. Pandas cung cấp các cấu trúc dữ liệu và các phép toán đẻ thao tác với dữ liệu dạng bảng và time series.

2.2 Ưu điểm

- Có thể xử lý dữ liệu với nhiều định dạng khác nhau
- Hỗ trợ import dữ liệu từ nhiều nguồn: CSV, XLS
- Cung cấp nhiều phép toán xử ly trên bộ dữ liệu
- Các hàm phổ biến

2.1. Import dữ liệu

- read_csv(): đọc dữ liệu từ file .csv
- read_excel(): đọc dữ liệu từ file .xlsx hoặc .xls

```
data_df = pd.read_csv("../data/coronavirus/coronavirus0410.csv", index_col="Country,Other")
data_df.head()
```

Country,Other	TotalCases	NewCases	TotalDeaths	NewDeaths	TotalRecovered	NewRecovered	ActiveCases	Serious,Critical	Tot Cases/1M pop	Deaths/1M pop	TotalTests
USA	31,802,772	NaN	574,840	NaN	24,346,766	NaN	6,881,166	9,078	95,647	1,729	416,789,330
Brazil	13,375,414	NaN	348,934	NaN	11,791,885	NaN	1,234,595	8,318	62,582	1,633	28,600,000
India	13,205,926	+3,143	168,467	NaN	11,990,859	+2,919	1,046,600	8,944	9,498	121	255,214,803
France	4,980,501	NaN	98,395	NaN	303,639	NaN	4,578,467	5,729	76,172	1,505	68,007,540
Russia	4,623,984	NaN	102,247	NaN	4,248,700	NaN	273,037	2,300	31,675	700	123,000,000

2.2. Xử lý dữ liệu

- drop(): xóa hàng hoặc cột bằng cách chỉ định tên nhãn và trục tương ứng

```
>>> df = pd.DataFrame(np.arange(12).reshape(3, 4),
...                   columns=['A', 'B', 'C', 'D'])
>>> df
   A  B   C   D
0  0  1   2   3
1  4  5   6   7
2  8  9  10  11
```

Drop columns

```
>>> df.drop(['B', 'C'], axis=1)
   A   D
0  0   3
1  4   7
2  8  11
```

```
>>> df.drop(columns=['B', 'C'])
   A   D
0  0   3
1  4   7
2  8  11
```

- to_numeric(): chuyển kiểu dữ liệu thành kiểu số

```
>>> s = pd.Series(['1.0', '2', -3])
>>> pd.to_numeric(s)
0    1.0
1    2.0
2   -3.0
dtype: float64
>>> pd.to_numeric(s, downcast='float')
0    1.0
1    2.0
2   -3.0
dtype: float32
>>> pd.to_numeric(s, downcast='signed')
0    1
1    2
2   -3
dtype: int8
>>> s = pd.Series(['apple', '1.0', '2', -3])
>>> pd.to_numeric(s, errors='ignore')
0    apple
1      1.0
2        2
3       -3
dtype: object
>>> pd.to_numeric(s, errors='coerce')
0    NaN
1    1.0
2    2.0
3   -3.0
dtype: float64
```

- iloc(): hàm để chọn ra các cột và dòng dữ liệu cần thiết

```
In [15]: mydict = [{'a': 1, 'b': 2, 'c': 3, 'd': 4},
                   {'a': 100, 'b': 200, 'c': 300, 'd': 400},
                   {'a': 1000, 'b': 2000, 'c': 3000, 'd': 4000 }]
         df = pd.DataFrame(mydict)
         df
```

Out[15]:

	a	b	c	d
0	1	2	3	4
1	100	200	300	400
2	1000	2000	3000	4000

```
In [16]: df.iloc[0:2]
```

Out[16]:

	a	b	c	d
0	1	2	3	4
1	100	200	300	400

3 Matplotlib

Trực quan hóa dữ liệu là cách đơn giản nhất để phân tích và tiếp thu thông tin. Các biểu đồ giúp ta đơn giản hóa các vấn đề phức tạp và giải quyết chúng tốt hơn, nhanh chóng hơn. Chúng ta còn có thể xây dựng một câu chuyện có nghĩa từ các biểu đồ và từ đó xây dựng, phát triển các chiến lược kinh doanh. Và hơn nữa, việc trực quan rất cần thiết cho các phân tích ở mức độ sâu hơn như EDA và Machine Learning. Một trong những thư viện được sử dụng rộng rãi cho việc trực quan là Matplotlib

Matplotlib là một thư viện vẽ đồ thị 2-D giúp trực quan các số liệu. Matplotlib mô phỏng Matlab như đồ thị và hình ảnh hóa tuy nhiên Matlab không miễn phí và khó mở rộng quy mô. Vì vậy, matplotlib trong Python được sử dụng vì nó là một thư viện mạnh mẽ, miễn phí và dễ dàng để trực quan hóa dữ liệu.

Vẽ biểu đồ bằng thì Matplotlib khá dễ dàng. Nói chung, khi vẽ hầu như ta sẽ làm theo các bước giống nhau ở tất cả các biểu đồ. Matplotlib có một mô-đun gọi là pyplot hỗ trợ vẽ đồ thị. Jupyter Notebook được sử dụng để chạy các

cell vì môi trường này hỗ trợ khá tốt cho việc trực quan. Để sử dụng matplotlib, chúng ta cần nhập **from matplotlib.pyplot as plt**

Nhập các thư viện và tập dữ liệu bắt buộc để vẽ biểu đồ bằng Pandas **pd.read_csv()**

Lọc những phần quan trọng cho việc trực quan bằng cách sử dụng điều kiện trên Pandas Dataframes

- `plt.plot()` được sử dụng để vẽ biểu đồ đường cũng như các hàm vẽ biểu đồ khác. Tất cả các hàm trực quan đều đòi hỏi có dữ liệu và các dữ liệu này được cung cấp cho hàm dưới dạng các tham số.

- `plot.xlabel , plt.ylabel` lần lượt để dán nhãn cho trục x và trục y

- `plt.xticks` , `plt.yticks` lần lượt để dán nhãn các giá trị quan sát được trên trục x và trục y

- `plt.legend()` biểu thị chú thích cho các biến quan sát

- `plt.title()` đặt tựa đề cho biểu đồ

- `plot.show()` hiển thị biểu đồ

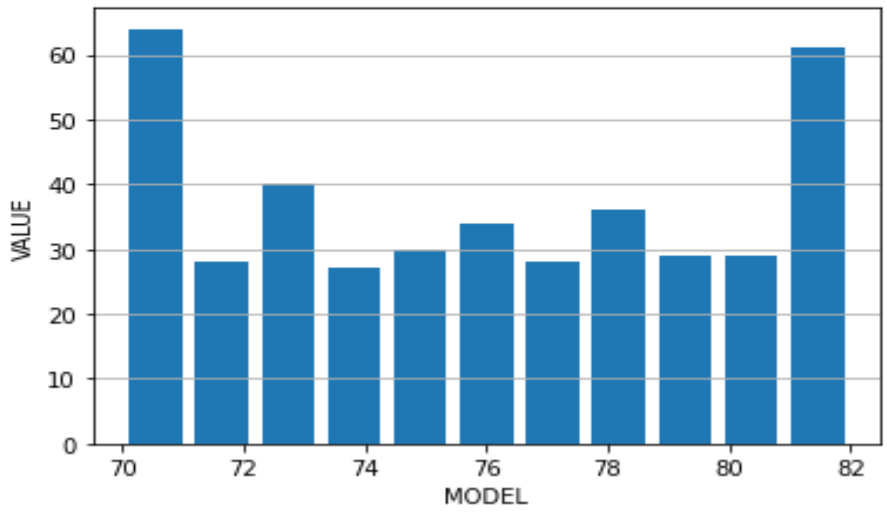
3.1 Histogram

Histogram là dạng biểu đồ lấy một chuỗi dữ liệu và chia dữ liệu thành một số ngăn. Sau đó, nó lập biểu đồ tần suất xuất hiện của các điểm dữ liệu trong mỗi thùng. Nó hữu ích trong việc hiểu số lượng các giá trị trong một phạm vi dữ liệu. Chúng ta nên sử dụng histogram khi chúng ta cần số biết số

lượng biến trong một biểu đồ. Ví dụ: Số lượng giá trị cụ thể của thuộc tính model

```
plt.hist(data.model, bins = 11, rwidth=0.8)
plt.grid(axis='y', alpha = 1)
plt.xlabel('MODEL')
plt.ylabel('VALUE')
```

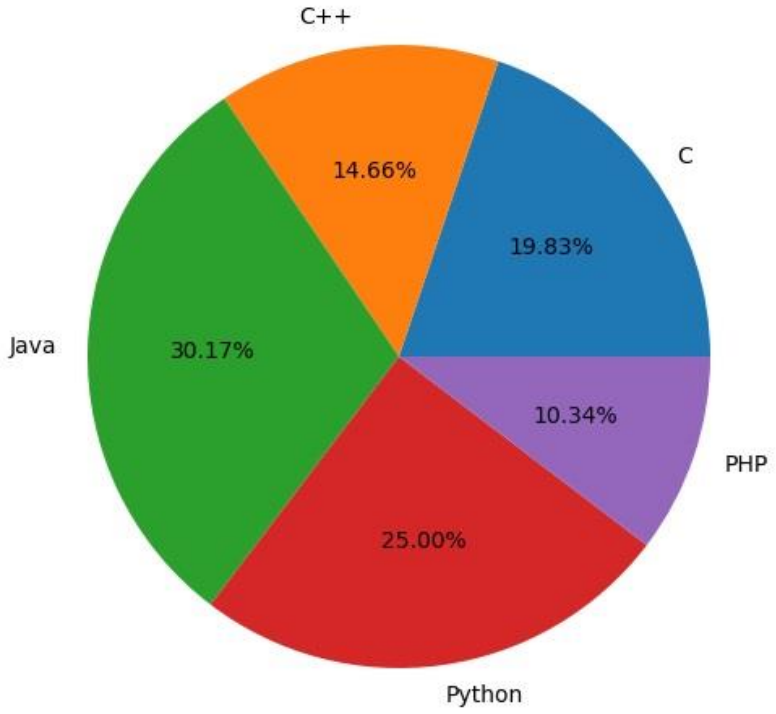
```
Text(0, 0.5, 'VALUE')
```



Plt.hist() lấy đối số đầu tiên làm dữ liệu số trong trục hoành, bins=11 sẽ tạo ra 11 ngăn giữa các giá trị model

3.2 Pie Chart

Biểu đồ tròn gồm một ô hình tròn được chia thành các lát để minh họa tỷ lệ số. Phần cắt của biểu đồ hình tròn hiển thị tỷ lệ các bộ phận trong tổng thể. Tuy nhiên, biểu đồ tròn ít khi được sử dụng vì rất khó so sánh các phần của biểu đồ. Biểu đồ cột được sử dụng thay thế vì dễ dàng so sánh các phần hơn.



Để vẽ biểu đồ tròn, ta sử dụng lệnh plt.pie()

3.3 Time Series

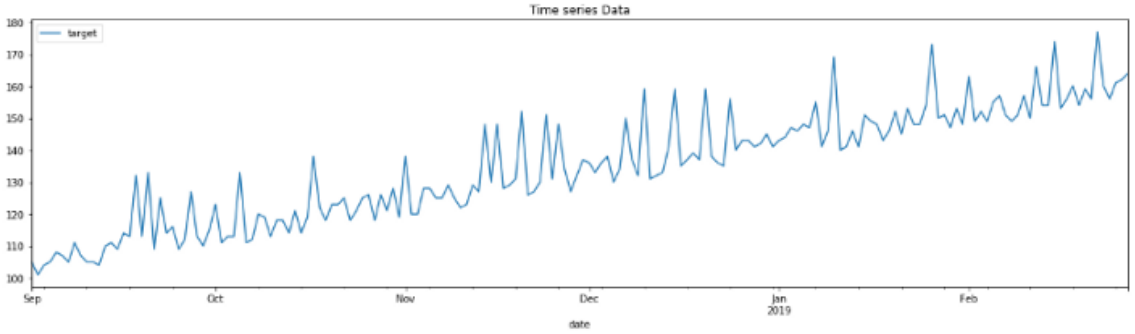
Time Series là một biểu đồ đường và về cơ bản thì nó kết nối các điểm dữ liệu bằng một đường thẳng. Nó được dùng để hiểu xu hướng cụ thể qua một khoảng thời gian và bằng cách đó ta có thể hiểu tương quan giữa các điểm dữ liệu. Xu hướng tăng là tương quan tích cực và xu hướng giảm là tương quan tiêu cực. Time Series được sử dụng chủ yếu cho các bài toán dự đoán.

Time Series được sử dụng khi một hoặc nhiều biến được biểu thị qua một khoảng thời gian, ví dụ như các bài toán dự đoán giá cổ phiếu, dự đoán thời tiết.

```
new_data
```

	chevrolet	buick	plymouth	amc	ford	pontiac	dodge	toyota	datsun	volkswagen	...	fiat	oldsmobile	chrysler	mazda	volvo	renault	honda	subi
70.0	5	2.0	5.0	5.0	6	1.0	2.0	1.0	1	1	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
71.0	4	NaN	3.0	3.0	5	3.0	1.0	2.0	2	2	...	1.0	NaN	NaN	NaN	NaN	NaN	NaN	N
72.0	3	1.0	2.0	2.0	4	1.0	2.0	3.0	1	2	...	NaN	1.0	1.0	1.0	1.0	1.0	NaN	N
73.0	6	2.0	4.0	4.0	5	1.0	2.0	2.0	1	1	...	2.0	2.0	1.0	1.0	1.0	NaN	NaN	N
74.0	3	1.0	2.0	3.0	4	NaN	2.0	2.0	2	1	...	3.0	NaN	NaN	NaN	NaN	NaN	1.0	
75.0	4	2.0	3.0	3.0	5	2.0	NaN	2.0	1	2	...	NaN	NaN	NaN	NaN	1.0	NaN	1.0	N
76.0	5	NaN	2.0	3.0	5	1.0	4.0	2.0	1	2	...	1.0	NaN	NaN	NaN	1.0	1.0	1.0	N
77.0	4	2.0	2.0	NaN	3	2.0	2.0	1.0	2	2	...	NaN	1.0	1.0	1.0	NaN	1.0	1.0	
78.0	3	2.0	2.0	2.0	4	1.0	4.0	2.0	3	2	...	NaN	2.0	NaN	1.0	1.0	NaN	2.0	N
79.0	3	2.0	2.0	2.0	3	2.0	3.0	NaN	1	1	...	1.0	2.0	1.0	1.0	NaN	NaN	NaN	N
80.0	2	NaN	NaN	1.0	2	NaN	2.0	3.0	4	4	...	NaN	NaN	NaN	3.0	NaN	1.0	2.0	
81.0	1	2.0	4.0	NaN	3	NaN	1.0	4.0	3	1	...	NaN	1.0	1.0	2.0	1.0	1.0	2.0	
82.0	5	1.0	1.0	1.0	4	2.0	3.0	2.0	1	2	...	NaN	1.0	1.0	2.0	NaN	NaN	3.0	N

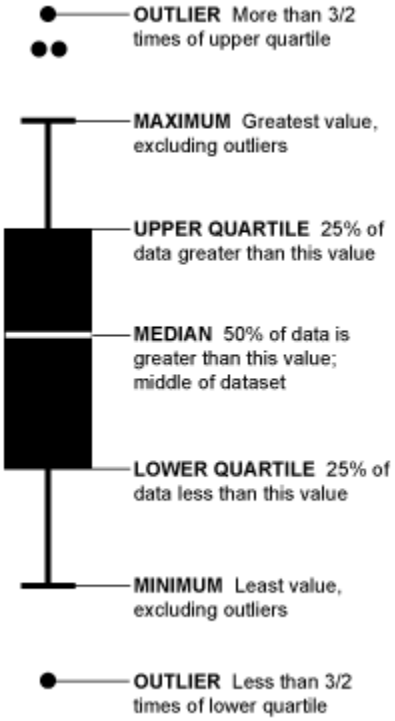
Trước khi trực quan, ta cần đặt cột mang giá trị Date làm index để thuận tiện cho việc trực quan.



3.4 Boxplot and Violinplot

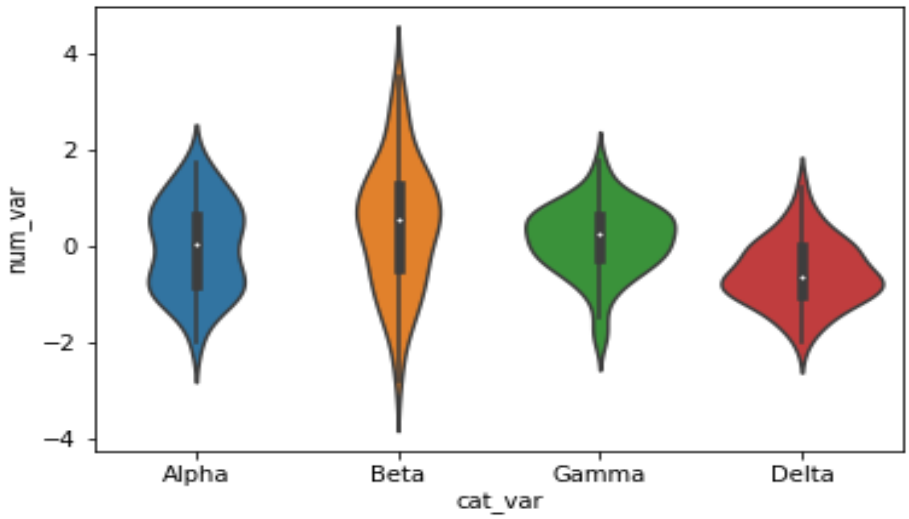
3.4.1 Boxplot

Boxplot đưa ra một bản tóm tắt dữ liệu tốt đẹp. Nó giúp hiểu rõ hơn về phân phối của các biến. Nó nên được sử dụng khi chúng ta yêu cầu sử dụng thông tin thống kê tổng thể về việc phân phối dữ liệu. Nó có thể được sử dụng để phát hiện các ngoại lệ trong dữ liệu. ví dụ: Điểm tín dụng của Khách hàng. Chúng tôi có thể nhận được giá trị lớn nhất, nhot nhất và nhiều thông tin khác về số điểm.



3.4.2 Violin plot

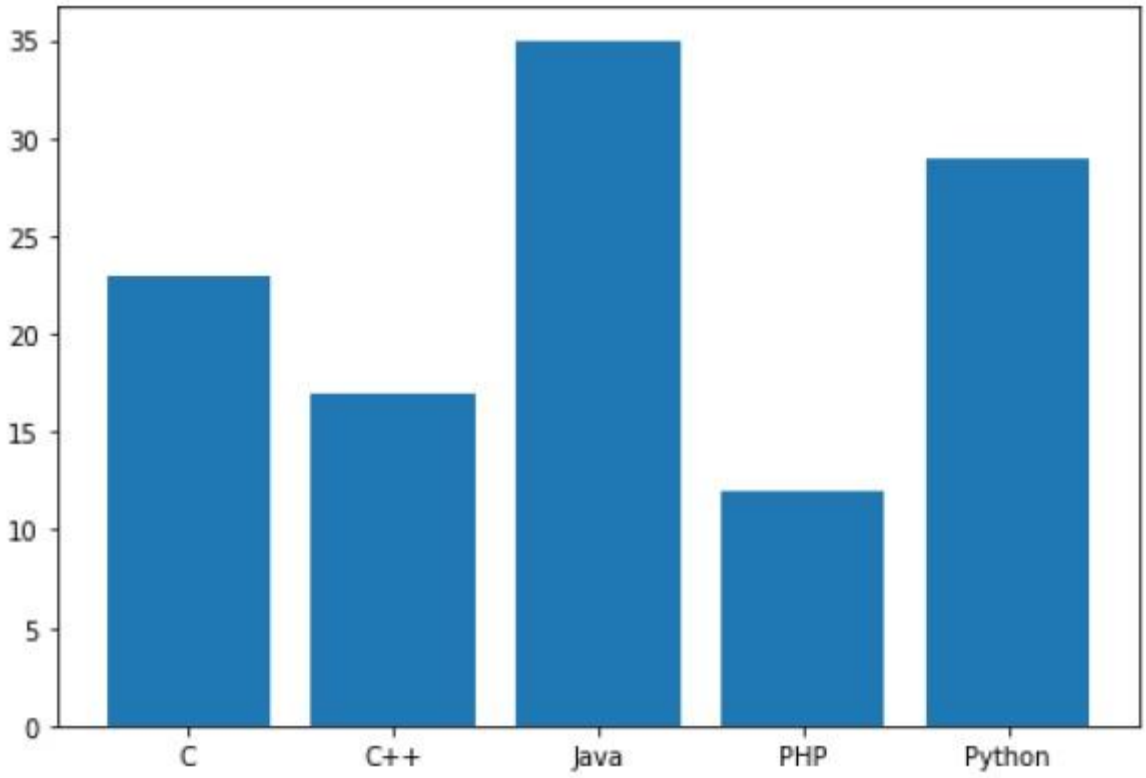
Violin là một biểu đồ tốt hơn so với boxplot vì nó mang lại hiểu biết rộng hơn nhiều về sự phân bố dữ liệu. Nó giống như một cây đàn vĩ cầm và các khu vực phình ra chỉ ra sự phân phối dữ liệu



3.5 Bar Plot

Biểu đồ cột hiển thị sự phân bố dữ liệu qua một số nhóm. Nó thường bị nhầm lẫn với biểu đồ histogram, trong khi histogram chỉ lấy dữ liệu số để vẽ biểu đồ. Nó được sử dụng khi cần so sánh giữa một số nhóm.

Hàm sử dụng: plt.bar()

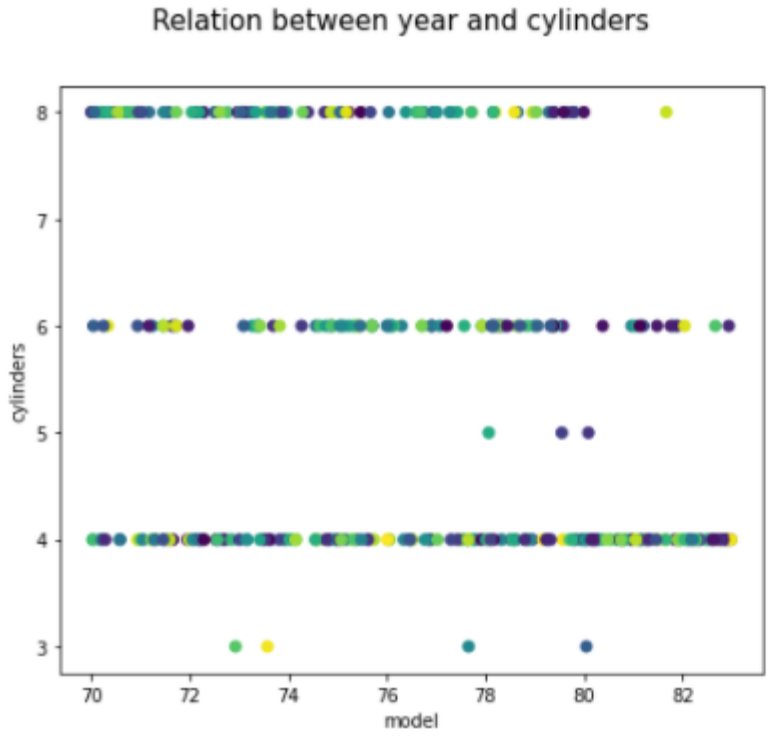


3.6 Scatter Plot

Biểu đồ phân tán giúp ta hình dung 2 biến. Nó giúp xác định mối quan hệ của dữ liệu với mỗi biến, tức là các mẫu tương quan hoặc xu hướng. Nó cũng giúp phát hiện những giá trị ngoại lai. Scatter Plot được sử dụng trong các khái niệm

Máy học như hồi quy, trong đó x và y là các biến liên tục. Nó cũng được sử dụng để phát hiện các ngoại lệ.

```
fig, ax = plt.subplots(figsize=(7,6))
fig.suptitle("Relation between year and cylinders", size=15)
colors = np.random.random(len(data.cylinders))
plt.scatter(data.model + np.random.random(len(data.model)), data.cylinders,c=colors)
ax.set_xlabel("model")
ax.set_ylabel("cylinders")
plt.show()
```



plt.scatter () nhận 2 đối số số để phân tán các điểm dữ liệu trong biểu đồ. Nó tương tự như biểu đồ đường ngoại trừ không có các đường thẳng nối các điểm.

II. Phân tích khám phá dữ liệu Car MPG

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

Trong data set có 406 ô tô và có 9 thuộc tính

2. **Có bao nhiêu công ty xe hơi khác nhau được đại diện trong tập dữ liệu? Tên của chiếc xe có MPG tốt nhất là gì? Hãng xe nào sản xuất nhiều xe 8 xi lanh nhất? Những xe có 3 xi-lanh có tên là gì? Thực hiện một số tìm kiếm trên internet có thể cho bạn biết về lịch sử và sự phổ biến của những chiếc xe 3 xi-lanh.**

Theo quan sát ta thấy rằng tên của các xe đều bắt đầu từ tên của công ty đã sản xuất ra xe hơi đó. Do đó đầu tiên ta sẽ tiến hành lọc ra từ đầu tiên trong chuỗi car_name

```
list_car_names = data.car_name.str.split()
companies = [comp[0] for comp in list_car_names]
temp_df = data.copy()
print(len(companies))
```

```
406
```

Ta chuyển car_name sang dạng chuỗi sau đó dùng hàm split() để tách từng từ trong chuỗi tên. Sau đó ta sử dụng vòng lặp for để lấy ra từ đầu tiên trong mỗi chuỗi. Khi hoàn tất ta được một danh sách tên các công ty ứng với các xe.

```
len(set(companies))
```

```
38
```

Một công ty có thể sản xuất ra nhiều loại xe nên khi loại bỏ các tên trùng, ta thu được số công ty còn lại là 38. Tuy nhiên khi quan sát các giá trị trong set companies này, ta sẽ thấy có một số tên công ty bị viết sai chính tả, hoặc một số viết dưới dạng viết tắt nên bị hiểu thành hai công ty riêng biệt, một số khác là các nhãn hiệu khác nhau nhưng đều có chung một công ty. Sau khi tìm hiểu thêm, nhóm tiến hành gom nhóm các tên sai chính tả thành một công ty.

```
temp_df = temp_df.assign(company=companies)
temp_df.company.unique().shape
```

```
(38,)
```

```
temp_df.company.replace(['chevroelt','chevy','vokswagen','vw','mercedes-benz','toyouta','maxda'],\
                        ['chevrolet', 'chevrolet','volkswagen', 'volkswagen', 'mercedes','toyota', 'mazda'],inplace=True)
```

```
list_comp = list(temp_df.company.unique())
temp_df.company.nunique()
```

```
31
```

Tạo một temp_df từ data bằng lệnh copy(), sau đó thêm cột company vào temp_df, sau đó ở cột company sử dụng hàm replace để thay thế các giá trị sai chính tả hoặc viết tắt, sau đó ta sử dụng hàm unique() để tạo ra mảng ra các giá trị phân biệt và shape cho biết kích thước của mảng. Vậy tổng cộng có 31 công ty trong tập dữ liệu.

Chiếc xe có MPG lớn nhất là: mazda glc

```
#name of the car with the best MPG
mpg_max = data.mpg.max()
print(data.car_name[data.mpg == mpg_max])
```

```
329    mazda glc
Name: car_name, dtype: object
```

- Lấy giá trị lớn nhất trong cột mpg
- Lấy giá trị tại dòng có điều kiện data.mpg = mpg_max là True.

3. Công ty sản xuất nhiều xe ô tô có 8 xi lanh nhất:

```
cylin_8 = temp_df.loc[temp_df.cylinders == 8].company.value_counts()
cylin_8
```

```
ford          22
chevrolet     19
dodge         12
plymouth      11
amc            9
oldsmobile     7
buick          7
pontiac        7
mercury        5
chrysler       4
chevy          2
cadillac       2
hi             1
Name: company, dtype: int64
```

Pandas.DataFrame.loc cho phép ta lọc ra các dòng có điều kiện thuộc tính cylinders == 8 là True. Kết quả là một DataFrame có 108 dòng thỏa điều kiện. Tiếp theo ta chọn ra thuộc tính company và sử dụng chức năng value_counts () để liệt kê ra các giá trị phân biệt tương ứng với số lần xuất hiện của chúng trong cột. Công ty sản xuất nhiều xe ô tô có 8 xi lanh nhất là ford với 22 xe.

4. Chiếc xe có 3 xilanh là: mazda rx2 coupe, maxda rx3, mazda rx-4, mazda rx-7 gs

```
#names of 3-cylinder cars
print(data.car_name[data.cylinders == 3])
```

```
78     mazda rx2 coupe
118          maxda rx3
250         mazda rx-4
341      mazda rx-7 gs
Name: car_name, dtype: object
```

Lịch sử và độ phổ biến của các loại xe 3-cylinders

+ Mazda Rx2 được sản xuất từ năm 1970 đến năm 1976 được sản xuất tại Hiroshima, Nhật Bản và được bán rộng rãi tại nhiều quốc gia trên thế giới.

+ Mazda Savanna đã được sử dụng trên mẫu xe động cơ quay bán tại Nhật Bản, như một chiếc coupe, sedan và wagon. Trên thị trường quốc tế, nó được gọi là Mazda RX-3. Nó nhỏ hơn và thể thao hơn so với người anh em của nó, Capella Rotary / RX-2, và phần lớn giống với mẫu xe bốn bánh thông thường của nó, Mazda Grand Familia. Nó có sẵn từ tháng 9 năm 1971 đến năm 1978 dưới dạng Super Deluxe coupé, Deluxe sedan và station wagon. Chiếc coupe Super Deluxe nặng hơn (884 kg so với 864 kg) và có sọc thân xe tùy chọn, đồng hồ, tấm chắn sáng phía sau và bảng điều khiển trung tâm / tay vịn cao và cột lái có thể đóng mở. Được bán từ năm 1972 đến năm 1978 tại Hoa Kỳ, phiên bản RX-3 cực kỳ thành công.

+ Mazda RX-4 à một chiếc xe lớn hơn những chiếc xe cùng thời với động cơ quay, RX-2 dựa trên Capella và RX-3 dựa trên Grand Familia. Đối với thị trường Hoa Kỳ, RX-4 được bán từ năm 1974 đến năm 1978, khi RX-7 ra mắt.

+ Mazda RX-7 là một chiếc xe thể thao sử dụng động cơ quay trước/ giữa, dẫn động cầu sau, được sản xuất và bán ra thị trường bởi Mazda từ năm 1978 đến năm 2002 qua ba thế hệ, tất cả đều sử dụng động cơ nhỏ gọn, nhẹ Động cơ quay Wankel.

5. Phạm vi, giá trị trung bình và độ lệch chuẩn của từng thuộc tính là gì? Chú ý đến các giá trị tiềm ẩn còn thiếu.

```
data.mpg.unique()
```

```
array([18. , 15. , 16. , 17. , 14. ,  nan, 24. , 22. , 21. , 27. , 26. ,
       25. , 10. , 11. ,  9. , 28. , 19. , 12. , 13. , 23. , 30. , 31. ,
       35. , 20. , 29. , 32. , 33. , 17.5, 15.5, 14.5, 22.5, 24.5, 18.5,
       29.5, 26.5, 16.5, 31.5, 36. , 25.5, 33.5, 20.5, 30.5, 21.5, 43.1,
       36.1, 32.8, 39.4, 19.9, 19.4, 20.2, 19.2, 25.1, 20.6, 20.8, 18.6,
       18.1, 17.7, 27.5, 27.2, 30.9, 21.1, 23.2, 23.8, 23.9, 20.3, 21.6,
       16.2, 19.8, 22.3, 17.6, 18.2, 16.9, 31.9, 34.1, 35.7, 27.4, 25.4,
       34.2, 34.5, 31.8, 37.3, 28.4, 28.8, 26.8, 41.5, 38.1, 32.1, 37.2,
       26.4, 24.3, 19.1, 34.3, 29.8, 31.3, 37. , 32.2, 46.6, 27.9, 40.8,
       44.3, 43.4, 36.4, 44.6, 40.9, 33.8, 32.7, 23.7, 23.6, 32.4, 26.6,
       25.8, 23.5, 39.1, 39. , 35.1, 32.3, 37.7, 34.7, 34.4, 29.9, 33.7,
       32.9, 31.6, 28.1, 30.7, 24.2, 22.4, 34. , 38. , 44. ])
```

```
data.horsepower.unique()
```

```
array([130., 165., 150., 140., 198., 220., 215., 225., 190., 115., 153.,
       175., 170., 160.,  95.,  97.,  85.,  88.,  46.,  87.,  90., 113.,
       200., 210., 193.,  nan,  48., 100., 105., 180., 110.,  72.,  86.,
        70.,  76.,  65.,  69.,  60.,  80.,  54., 208., 155., 112.,  92.,
       145., 137., 158., 167.,  94., 107., 230.,  49.,  75.,  91., 122.,
        67.,  83.,  78.,  52.,  61.,  93., 148., 129.,  96.,  71.,  98.,
        53.,  81.,  79., 120., 152., 102., 108.,  68.,  58., 149.,  89.,
        63.,  66., 139., 103., 125., 133., 138., 135., 142.,  77.,  62.,
       132.,  84.,  64.,  74., 116.,  82.])
```

Dùng hàm unique() của pandas, ta nhận thấy có hai cột có giá trị thiếu là mpg và horsepower. Ta sẽ tiến hành điền giá trị thiếu bằng pandas median() method.

```
data['mpg'].fillna(data['mpg'].median(), inplace=True)
data['horsepower'].fillna(data['horsepower'].median(), inplace=True)
```

Sau đó ta sẽ tính phạm vi, giá trị trung bình và độ lệch chuẩn của từng thuộc tính trong data

sử dụng hàm min(), max(), mean() và std(). Hàm mix() và max() chỉ sử dụng cho từng thuộc tính trong data, hàm mean() và std() áp dụng trên toàn bộ data và cho kết quả trên từng thuộc tính.

```
#range of each attribute
print("mpg          ",data.mpg.min(), "    to ", data.mpg.max())
print("cylinder     ",data.cylinders.min(), "    to ", data.cylinders.max())
print("displacement ",data.displacement.min(), "   to ", data.displacement.max())
print("horsepower   ",data.horsepower.min(), "   to ", data.horsepower.max())
print("weight       ",data.weight.min(), " to ", data.weight.max())
print("acceleration ",data.acceleration.min(), "    to ", data.acceleration.max())
print("model        ",data.model.min(), "   to ", data.model.max())
print("origin       ",data.origin.min(), "    to ", data.origin.max())
```

```
mpg           9.0     to  46.6
cylinder      3.0     to  8.0
displacement  68.0    to  455.0
horsepower    46.0    to  230.0
weight        1613.0  to  5140.0
acceleration  8.0     to  24.8
model         70.0    to  82.0
origin        1.0     to  3.0
```

```
#mean of each attribute
data.mean()
```

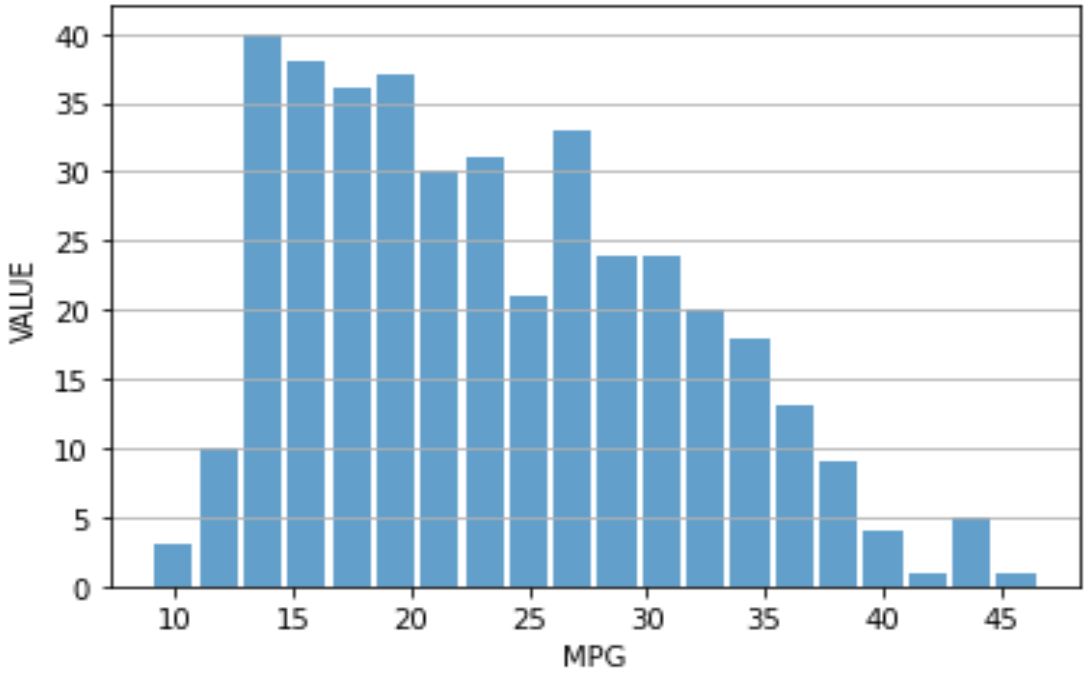
```
mpg              23.514573
cylinders         5.475369
displacement    194.779557
horsepower      105.082500
weight         2979.413793
acceleration     15.519704
model            75.921182
origin            1.568966
dtype: float64
```

```
# standard deviation of each attribute
data.std()
```

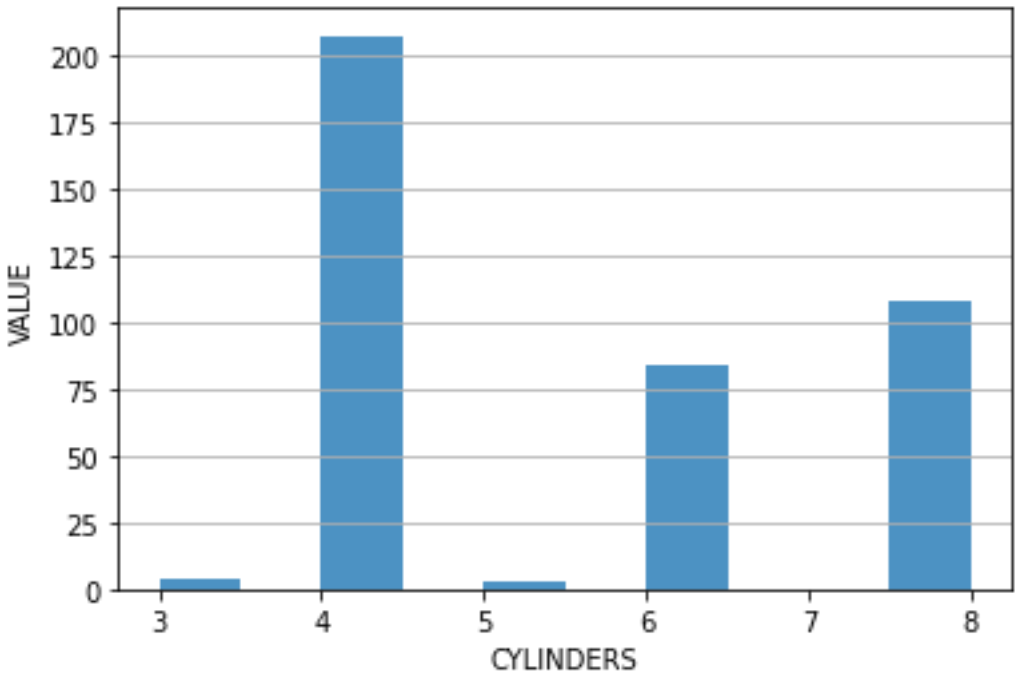
```
mpg              7.738404
cylinders        1.712160
displacement   104.922458
horsepower      38.480531
weight         847.004328
acceleration     2.803359
model            3.748737
origin           0.797479
dtype: float64
```

6. Vẽ biểu đồ cho từng thuộc tính. 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 xem biểu đồ.

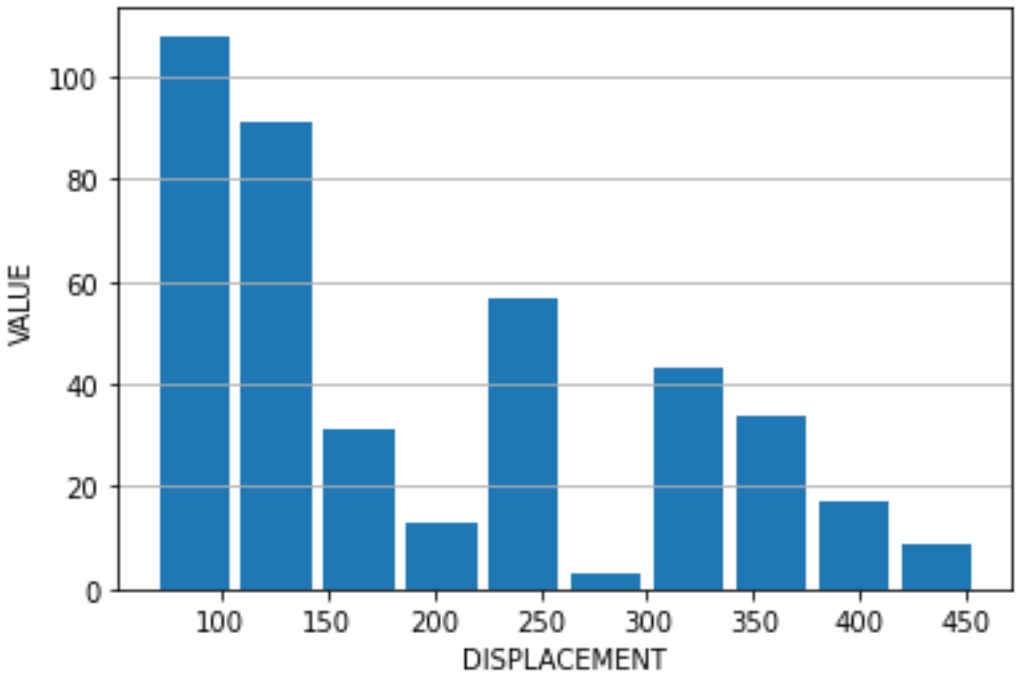
- MPG: ta thấy MPG được phân bố khá rộng, tập trung nhiều về 2 khoảng (13.0, 20.0) và (26.0, 27.0), trong đó số lượng giá trị nằm trong khoảng (14, 16) là nhiều nhất.



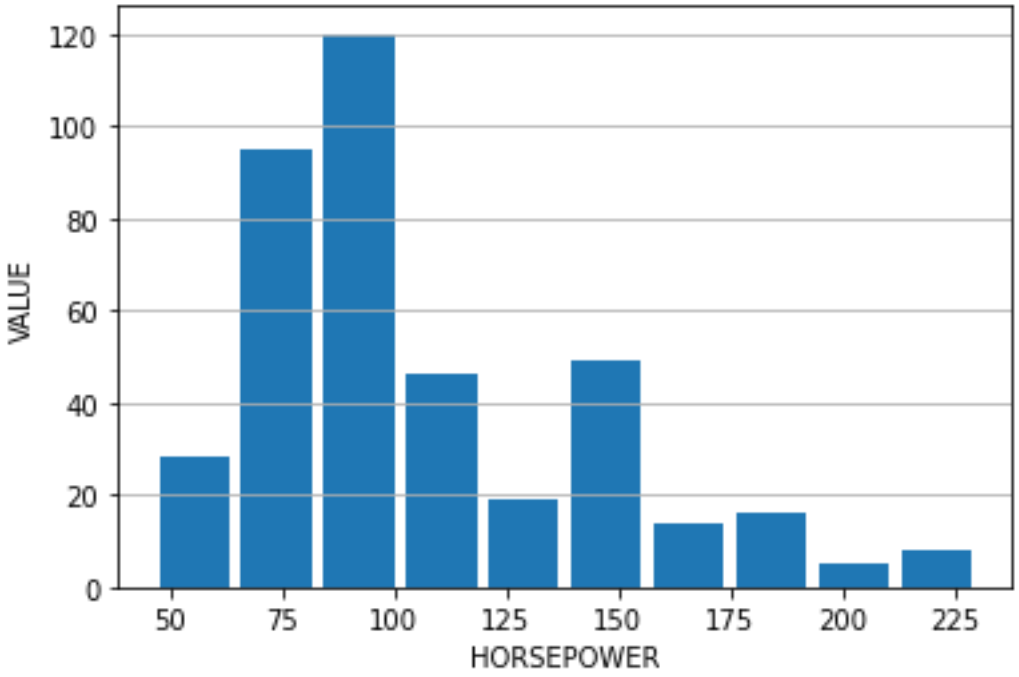
- Cylinders: gồm 5 loại (3, 4, 5, 6, 8). Đa số các xe có 4, 6 hoặc 8 cylinders. Hơn một nửa số xe trong tập dữ liệu có 4 cylinders. Số xe có 3 hoặc 5 cylinders chiếm tỷ lệ rất ít.



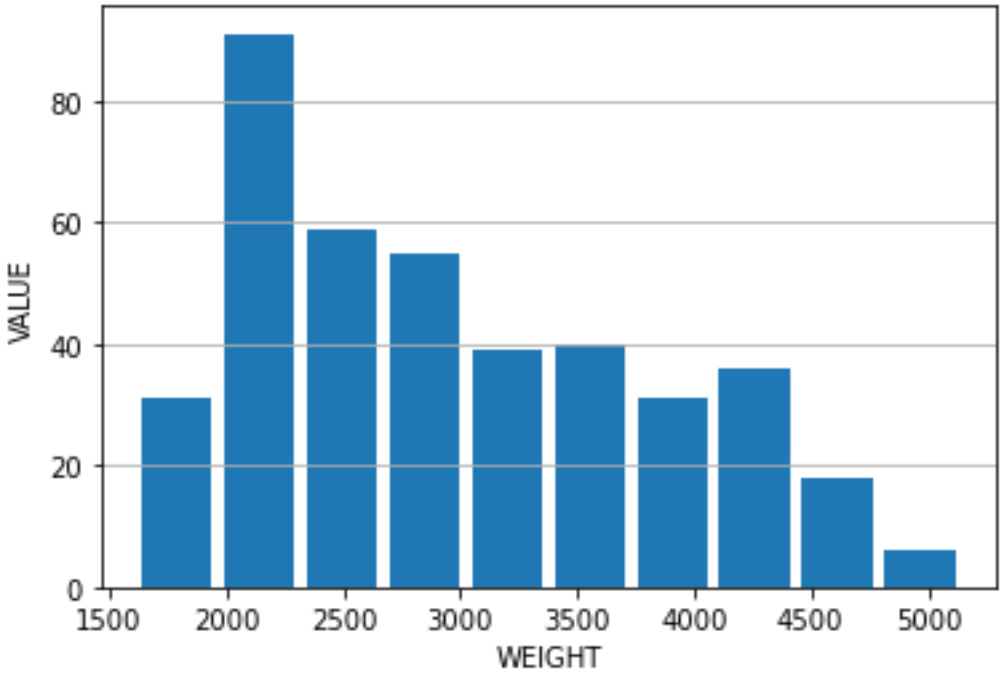
- Displacement: có độ lệch chuẩn rất lớn (104.793164), cho thấy displacement phân tán không đồng đều, xa giá trị trrung binh, lệch nhiều về phía bên trái. Hầu hết ô tô có displacement thấp.



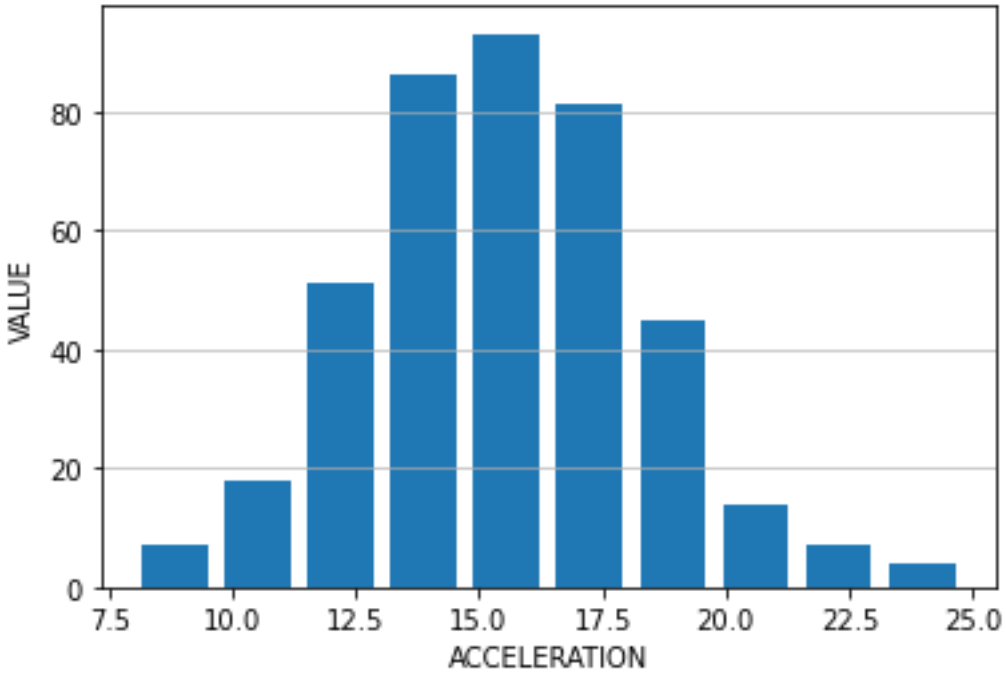
- Horsepower: giá trị phân bố không đồng đều và tập trung về phía bên trái, mức độ phân bổ bên phải rất ít chứng tỏ có ít loại xe có mã lực cao.



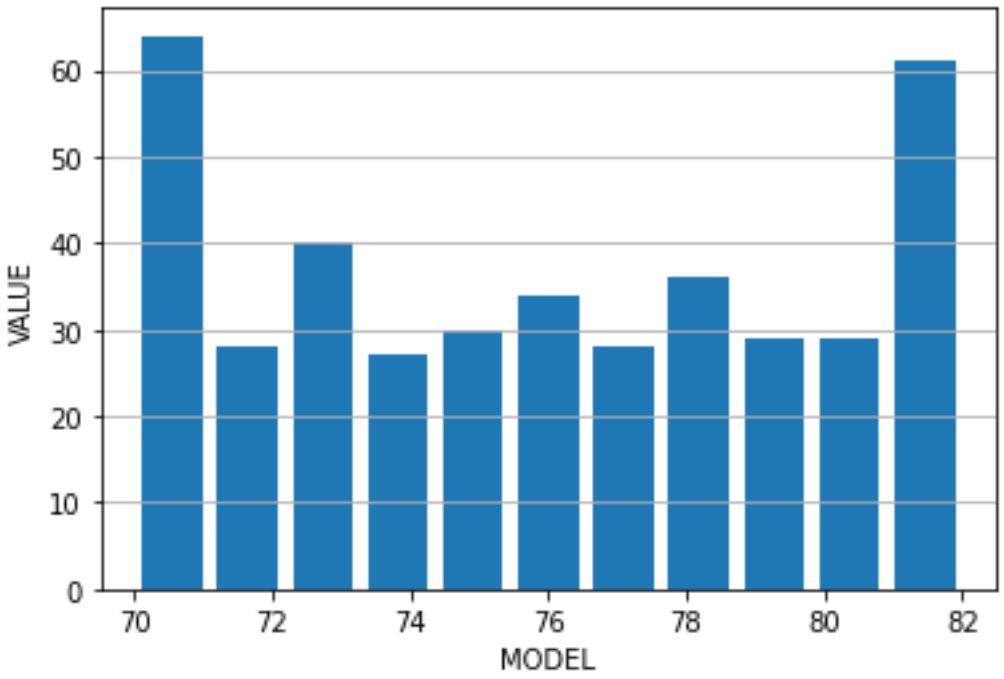
- Weight: ta thấy dữ liệu được phân bố rộng, giá trị tập trung nhiều nhất là khoảng (2000-2300), trải rộng từ (2400 - 4300)



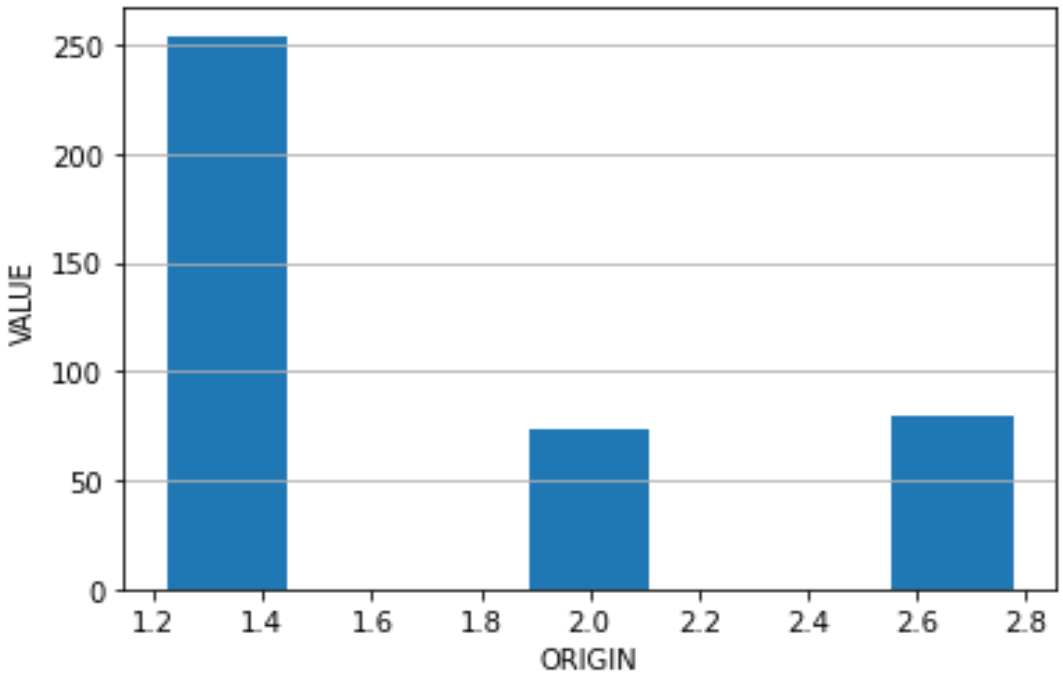
- Acceleration: dữ liệu có phân phối chuẩn, giá trị trung bình rơi vào khoảng 16-17



- Model: Khoảng giá trị là (70 - 82), ta thấy trải dài và đều khắp khoảng giá trị. Model 70 và 82 có số lượng giá trị cao nhất, các giá trị còn lại có phân bố gần tương đương nhau.



Origin: có 3 loại (1, 2, 3), dữ liệu tập trung nhiều nhất ở loại 1, chiếm gần 2/3 tập dữ liệu. Giá trị loại 2 và 3 có phân bố gần tương đương nhau.



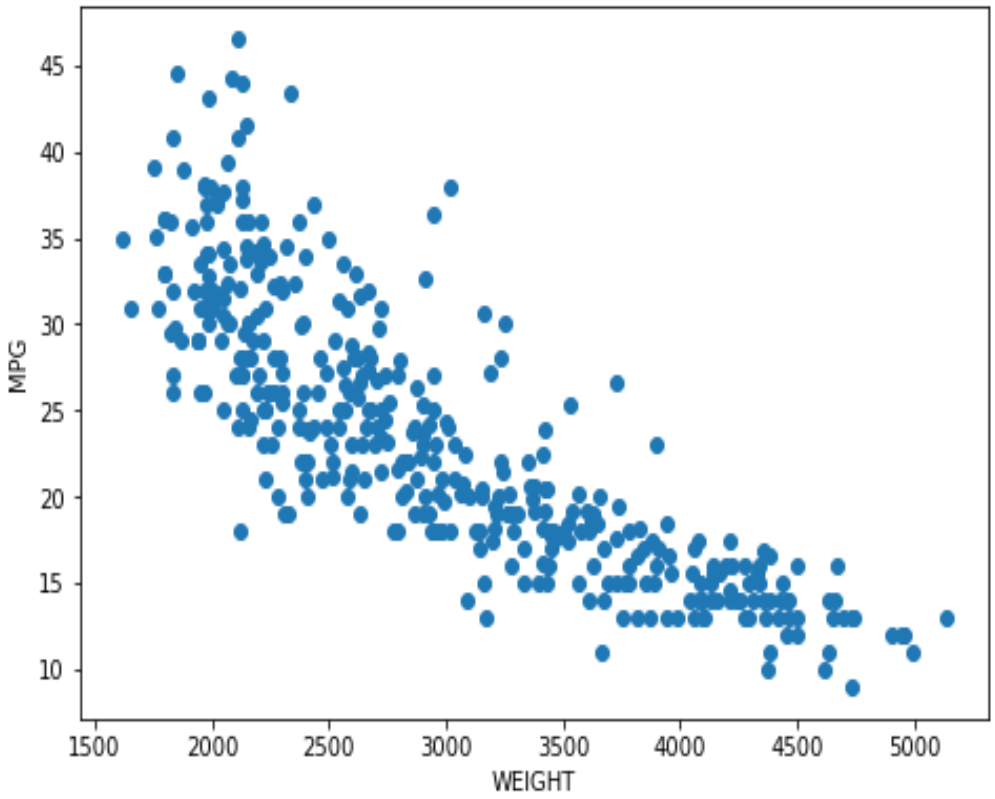
7. **Vẽ biểu đồ phân tán của trọng lượng so với thuộc tính MPG. Bạn kết luận gì về mối quan hệ giữa các thuộc tính? Hệ số tương quan giữa 2 thuộc tính là gì?**

Scatterplot: ta thấy giữa 2 thuộc tính có quan hệ nagative correlation - tức là khi Weight càng tăng thì MPG càng giảm, các điểm có dạng tuyến tính và tương quan mạnh với nhau.

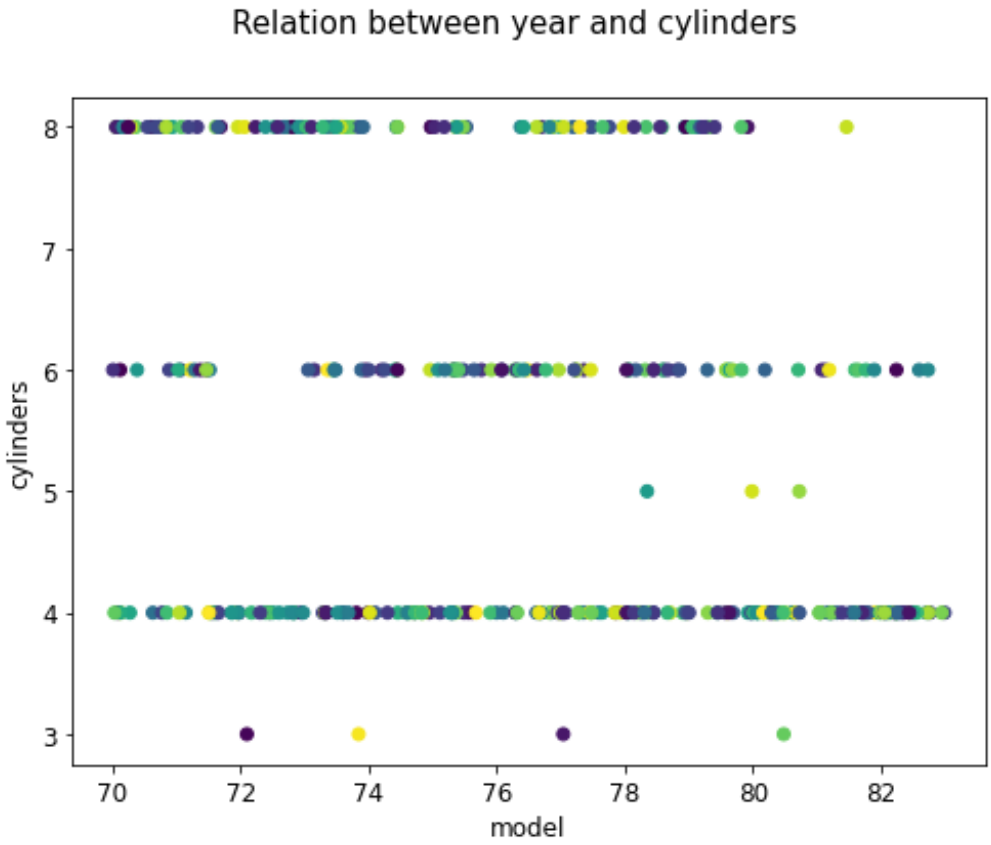
Hệ số tương quan giữa 2 thuộc tính là: -0.823985

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

	mpg	weight
mpg	1.000000	-0.823985
weight	-0.823985	1.000000



8. Vẽ biểu đồ phân tán giữa hai thuộc tính năm và số xi lanh. Thêm một số nhiễu nhỏ vào các giá trị để làm cho biểu đồ trông đẹp hơn. Bạn có thể đưa ra kết luận gì? Tìm kiếm trên mạng về lịch sử của ngành công nghiệp ô tô trong những năm 70s có thể giải thích kết quả.



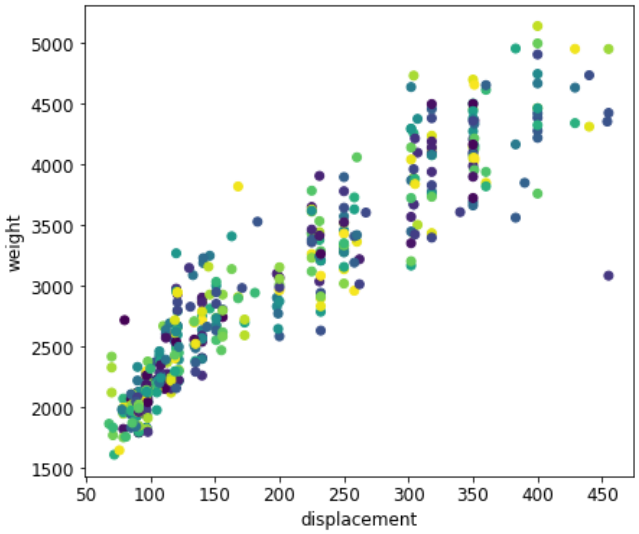
- Kết luận: Từ năm 70 đến trước năm 80, hầu hết các hãng xe đều sản xuất xe với 4, 6 hoặc 8 cylinders, tuy nhiên từ sau năm 80 trở đi, xe với 8 cylinders không còn được sản xuất nữa.

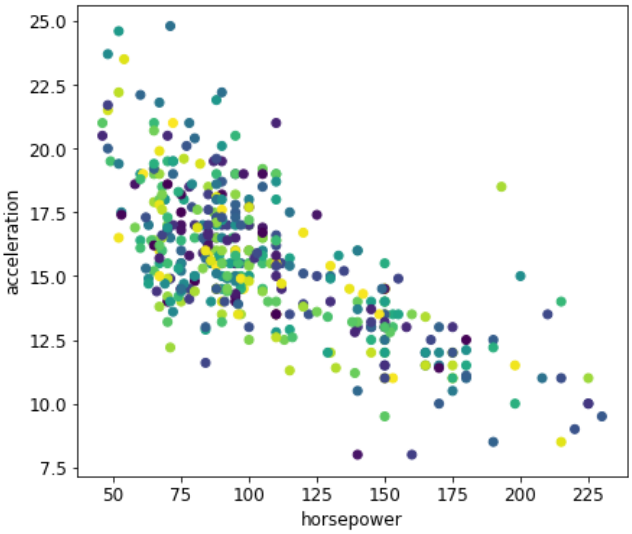
- Giải thích kết quả:

- Số lượng cylinders càng nhiều thì càng tiêu tốn nhiều nhiên liệu để động cơ hoạt động và như vậy thì không sử dụng năng lượng hiệu quả

- Vào những năm 70s đã xảy ra cuộc khủng hoảng năng lượng, các nhà chức trách của Mỹ áp đặt các quy định về thiết kế xe ô tô và buộc các nhà sản xuất ô tô phải lưu tâm vấn đề môi trường trong các thiết kế của mình. Các nhà sản xuất xe phải cân nhắc các biện pháp và phương án giảm số cylinders từ 8 còn 6 hoặc 4 cylinders được lựa chọn.

9. Hiển thị thêm 2 biểu đồ phân tán nữa mà bạn thấy thú vị. Thảo luận về những gì bạn thấy.





- Displacement và weight có hệ số tương quan tích cực (giá trị displacement tăng thì giá trị của weight cũng sẽ tăng và ngược lại), ta có thể vẽ một đường thẳng tuyến tính hướng lên đi qua các điểm và việc này cho thấy hai thuộc tính có tương quan mạnh.
- Horsepower và acceleration có hệ số tương quan tiêu cực (giá trị horsepower tăng thì giá trị acceleration sẽ giảm và ngược lại), ta có thể vẽ một đường tuyến tính hướng xuống dưới đi qua tập hợp các điểm và việc này cho thấy hai thuộc tính có tương quan mạnh. Có nhiều giá trị ngoại lai trong tập hợp các điểm.

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

```
comp_prod_count = temp_df.groupby(['company'])['model'].value_counts()
comp_prod_count
```

```
company  model
amc      70.0     5
         73.0     4
         71.0     3
         74.0     3
         75.0     3
                 ..
volvo    73.0     1
         75.0     1
         76.0     1
         78.0     1
         81.0     1
Name: model, Length: 211, dtype: int64
```

Trước khi vẽ time series, ta cần sử dụng pandas lọc ra tập dữ liệu cần thiết. Đầu tiên ta sẽ gom nhóm tập dữ liệu dựa trên thuộc tính company, sau đó đếm số giá trị của thuộc tính model, kết quả trả về series thể hiện số xe được sản xuất theo từng năm của company đó.

```
cols = list(list_comp)
new_data = pd.DataFrame(columns=cols, index = temp_df.model.unique())
for comp in list_comp:
    new_data[comp] = comp_prod_count[comp]
new_data.dropna(axis=1,thresh = 2,inplace=True)
```

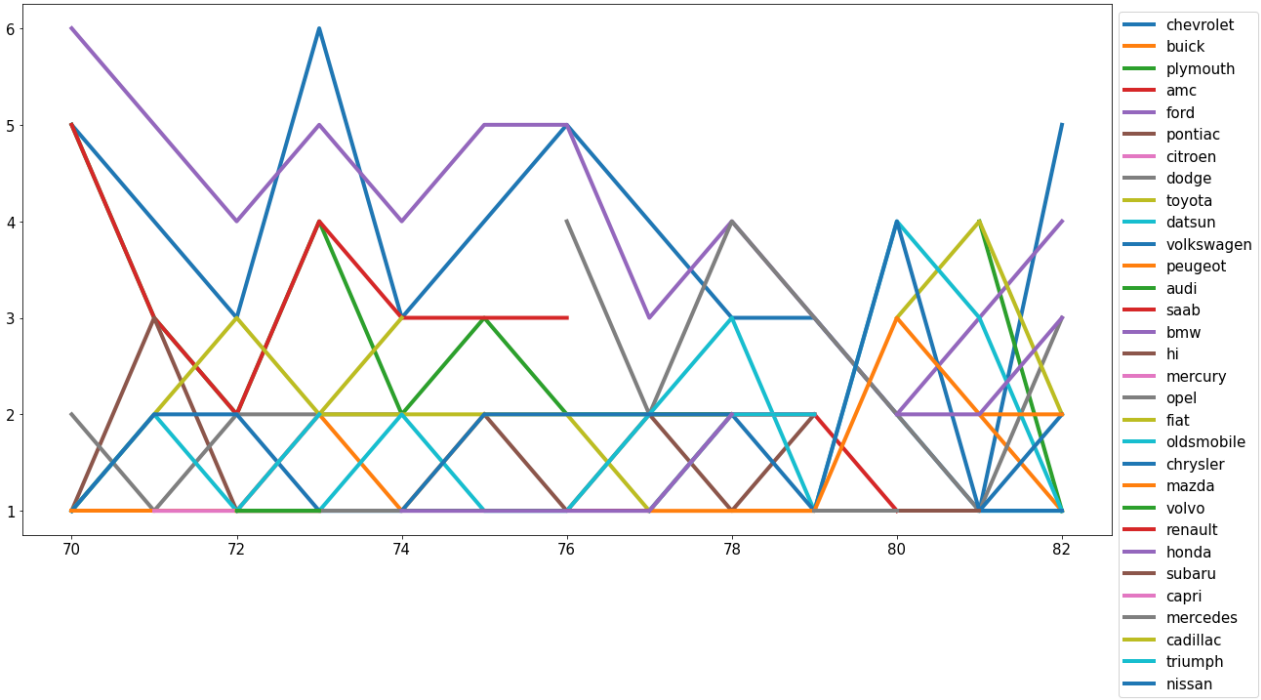
```
new_data
```

	chevrolet	buick	plymouth	amc	ford	pontiac	dodge	toyota	datsun	volkswagen	...	fiat	oldsmobile	chrysler	mazda	volvo	renault	honda	subi
70.0	5	2.0	5.0	5.0	6	1.0	2.0	1.0	1	1	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
71.0	4	NaN	3.0	3.0	5	3.0	1.0	2.0	2	2	...	1.0	NaN	NaN	NaN	NaN	NaN	NaN	N
72.0	3	1.0	2.0	2.0	4	1.0	2.0	3.0	1	2	...	NaN	1.0	1.0	1.0	1.0	1.0	NaN	N
73.0	6	2.0	4.0	4.0	5	1.0	2.0	2.0	1	1	...	2.0	2.0	1.0	1.0	1.0	NaN	NaN	N
74.0	3	1.0	2.0	3.0	4	NaN	2.0	2.0	2	1	...	3.0	NaN	NaN	NaN	NaN	NaN	1.0	
75.0	4	2.0	3.0	3.0	5	2.0	NaN	2.0	1	2	...	NaN	NaN	NaN	NaN	1.0	NaN	1.0	N
76.0	5	NaN	2.0	3.0	5	1.0	4.0	2.0	1	2	...	1.0	NaN	NaN	NaN	1.0	1.0	1.0	N
77.0	4	2.0	2.0	NaN	3	2.0	2.0	1.0	2	2	...	NaN	1.0	1.0	1.0	NaN	1.0	1.0	
78.0	3	2.0	2.0	2.0	4	1.0	4.0	2.0	3	2	...	NaN	2.0	NaN	1.0	1.0	NaN	2.0	N
79.0	3	2.0	2.0	2.0	3	2.0	3.0	NaN	1	1	...	1.0	2.0	1.0	1.0	NaN	NaN	NaN	N
80.0	2	NaN	NaN	1.0	2	NaN	2.0	3.0	4	4	...	NaN	NaN	NaN	3.0	NaN	1.0	2.0	
81.0	1	2.0	4.0	NaN	3	NaN	1.0	4.0	3	1	...	NaN	1.0	1.0	2.0	1.0	1.0	2.0	

Tạo DataFrame new_data bằng cách tạo các cột bằng danh sách công ty, đặt index là thuộc tính model, sau đó gán giá trị của từng cột bằng series tìm được

ở trên. Sau đó ta sẽ loại bỏ các cột có số giá trị khác NaN nhỏ hơn 2. Ta được kết quả cuối cùng của new_data như trên.

Có được tập dữ liệu cần thiết, ta tiến hành trực quan theo yêu cầu



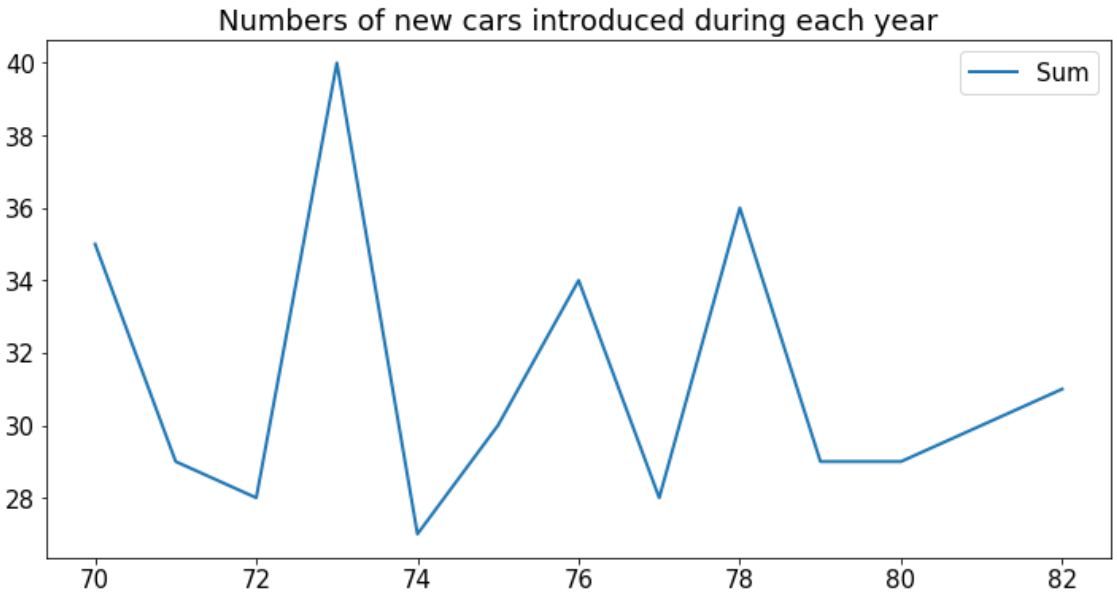
➢ **Phân tích:**

- Các công ty nhiều nhất cho ra mắt 6 mẫu xe mới trong một năm và ít nhất là 0. Đa số các công ty cho ra mắt 1 hoặc 2 mẫu xe mới trong năm. Chevrolet, Ford, Datsun và Volkswagen là 4 công ty duy nhất luôn cho ra mắt sản phẩm mới mỗi năm trong khoảng từ năm 1970 đến năm 1982. Cũng trong khoảng thời gian này, chỉ có Ford và Chevrolet đạt được mốc giới thiệu 6 mẫu xe mới trong vòng một năm.

- Trong năm 1981, hầu hết các công ty đều chỉ ra mắt thêm 1 mẫu xe mới và một năm sau đó mới cho thấy sự khởi sắc rõ rệt.

Do có nhiều cách hiểu đề nên nhóm sẽ trực quan thêm cách hiểu thứ hai. Ta sẽ tiến hành tính tổng tất cả các xe được giới thiệu của từng năm, bỏ qua giá trị NaN

```
new_data['Sum'] = new_data.dropna(how='all').sum(1)
new_data
```



➢ **Xu hướng:**

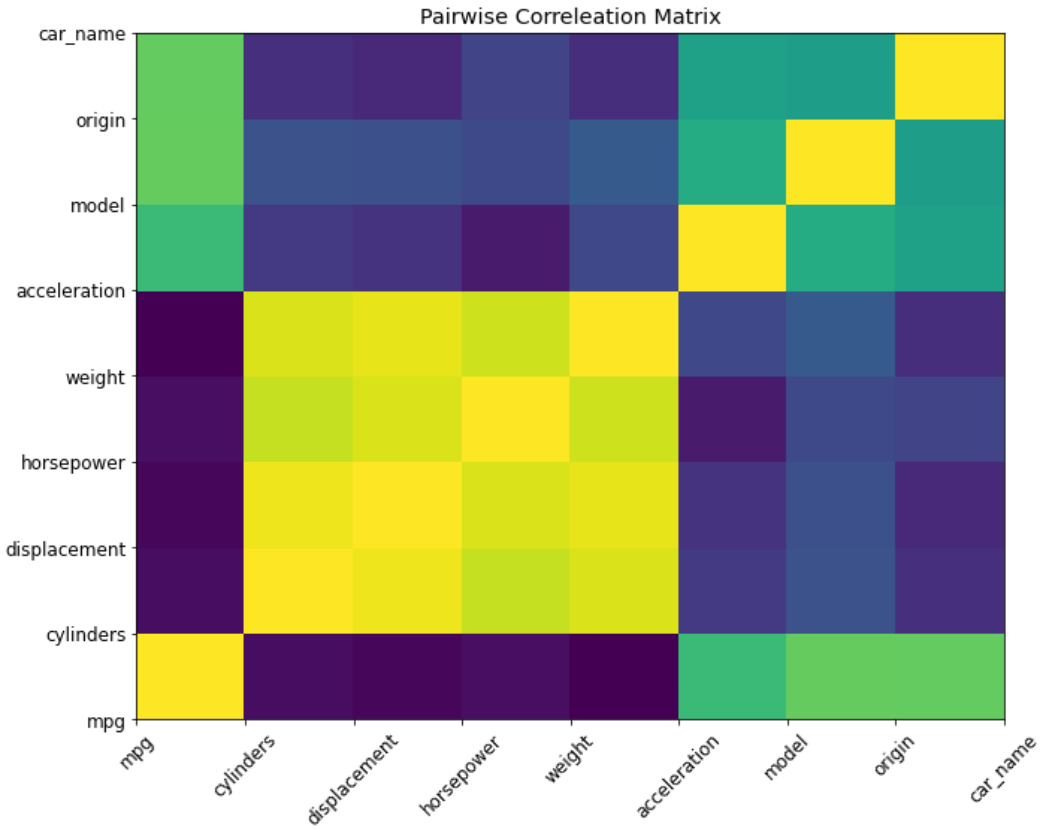
- Nhìn chung số mẫu xe mới được giới thiệu qua từng năm có tính chu kỳ (các điểm cực tiểu cách nhau từ 2 đến 3 năm), tức là nếu một năm nào đó có rất nhiều mẫu xe mới được giới thiệu thì sẽ có con số này xu hướng giảm mạnh vào năm sau. Những nếu một năm cụ thể có số lượng xe mới ít hơn đáng kể so với năm trước đó thì qua năm sau lượng xe có thể tiếp tục tụt giảm, hoặc có thể tăng đáng kể, hoặc có thể giữ nguyên. Năm 73 là năm chứng kiến sự ra đời của nhiều mẫu xe nhất với tổng cộng 40 xe được cho ra mắt thị trường. Ngay sau năm này, năm 74 lại cho thấy sự sụt giảm mạnh

mẽ, chỉ có 27 mẫu xe mới trong năm và là con số thấp nhất trong khoảng thời gian trên.

11. Tính toán mối tương quan theo từng cặp và vẽ bản đồ nhiệt bằng Matplotlib. Bạn có thấy mối tương quan thú vị nào đó không?

```
df_corr = data.iloc[:,0:8].corr()
df_corr
```

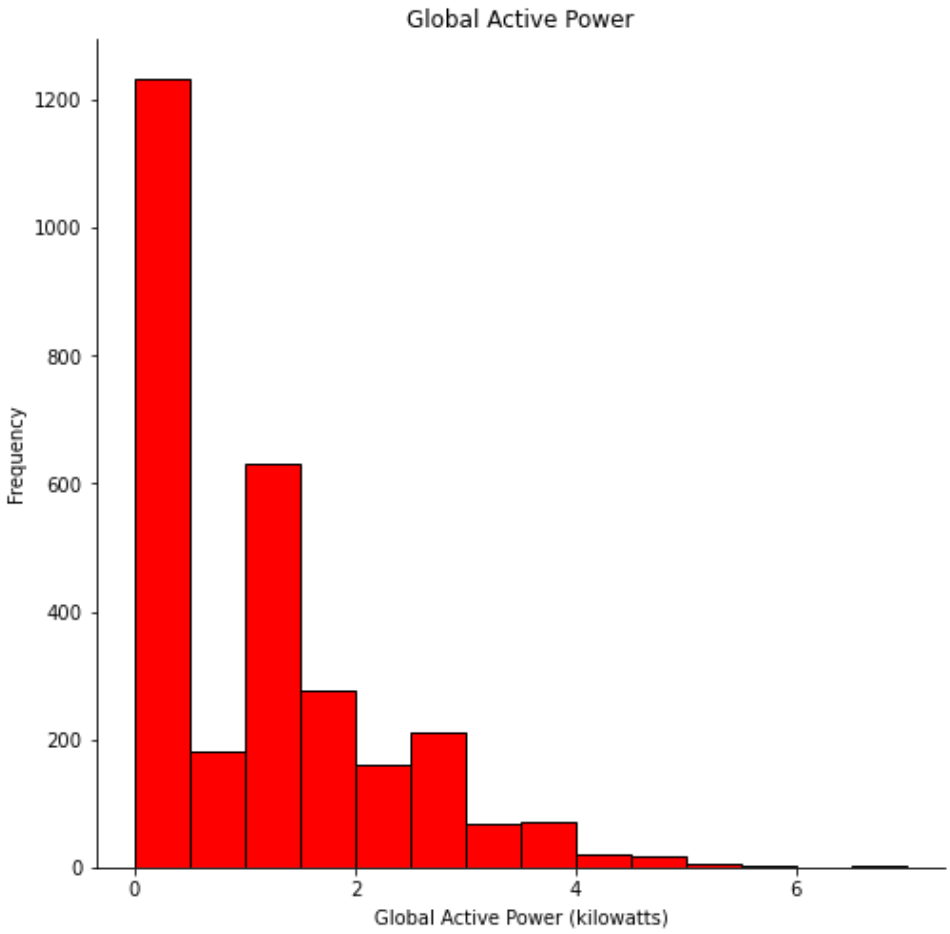
	mpg	cylinders	displacement	horsepower	weight	acceleration	model	origin
mpg	1.000000	-0.763451	-0.792077	-0.760926	-0.823985	0.410449	0.567223	0.561354
cylinders	-0.763451	1.000000	0.951787	0.842307	0.895220	-0.522452	-0.360762	-0.567478
displacement	-0.792077	0.951787	1.000000	0.896703	0.932475	-0.557984	-0.381714	-0.613056
horsepower	-0.760926	0.842307	0.896703	1.000000	0.864369	-0.694415	-0.421699	-0.457162
weight	-0.823985	0.895220	0.932475	0.864369	1.000000	-0.430086	-0.315389	-0.584109
acceleration	0.410449	-0.522452	-0.557984	-0.694415	-0.430086	1.000000	0.301992	0.218845
model	0.567223	-0.360762	-0.381714	-0.421699	-0.315389	0.301992	1.000000	0.187656
origin	0.561354	-0.567478	-0.613056	-0.457162	-0.584109	0.218845	0.187656	1.000000



- Bốn thuộc tính cylinders, displacement, horsepower và weight cho thấy những mối tương quan tích cực mạnh mẽ với nhau và mối tương quan tiêu cực với biến mpg
- Một số cặp thuộc tính khác có tương quan tiêu cực: (acceleration, house), (weight, origin)

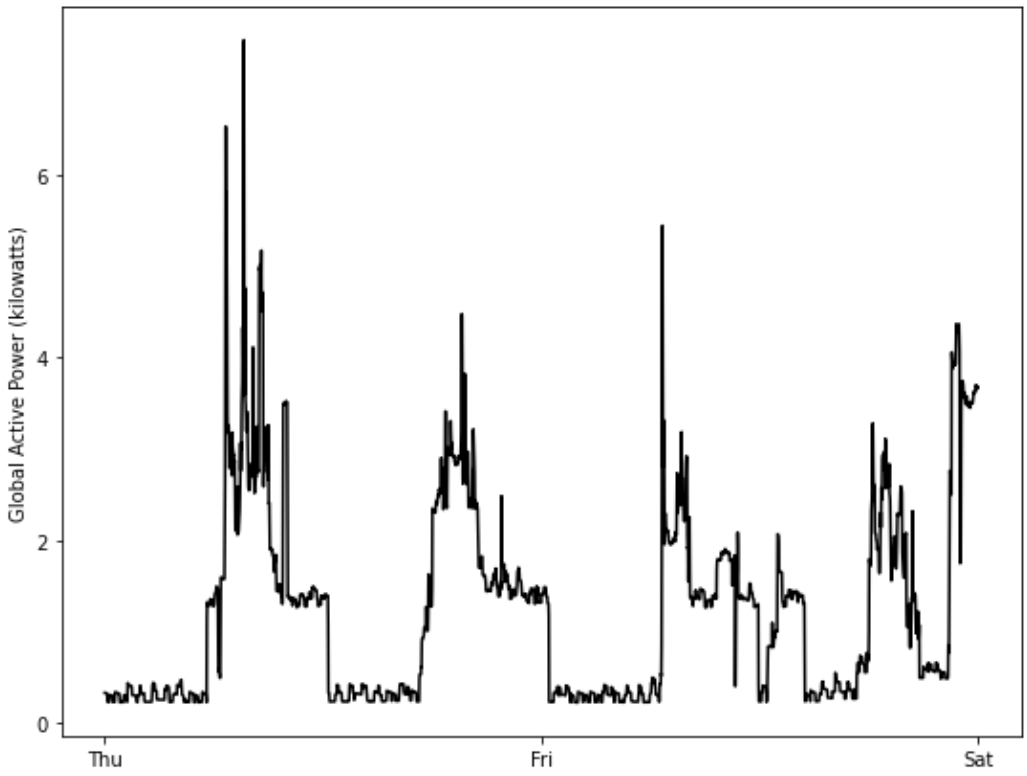
III. Electric power consumption data

```
def plot1():
    fig, ax= plt.subplots(figsize=(8, 8))
    ax.spines["right"].set_visible(False)
    ax.spines["top"].set_visible(False)
    bins =np.arange(0, max(data["Global_active_power"]), 0.5)
    ax.hist(data["Global_active_power"], color='red', edgecolor='black', bins=bins)
    plt.xticks(list(range(0, int(max(data["Global_active_power"])), 2)))
    plt.ylabel("Frequency")
    plt.xlabel("Global Active Power (kilowatts)")
    plt.title("Global Active Power")
    fig.savefig('./images/plot1.png', bbox_inches='tight')
    plt.show()

plot1();
```



Dựa vào biểu đồ phân phối, ta thấy được lượng điện được tiêu thụ ở mức từ 0 đến 2 kilowatt trong phần lớn thời gian. Lượng điện tiêu thụ trong một khoảng thời gian càng lớn thì tần suất xuất hiện càng nhỏ.

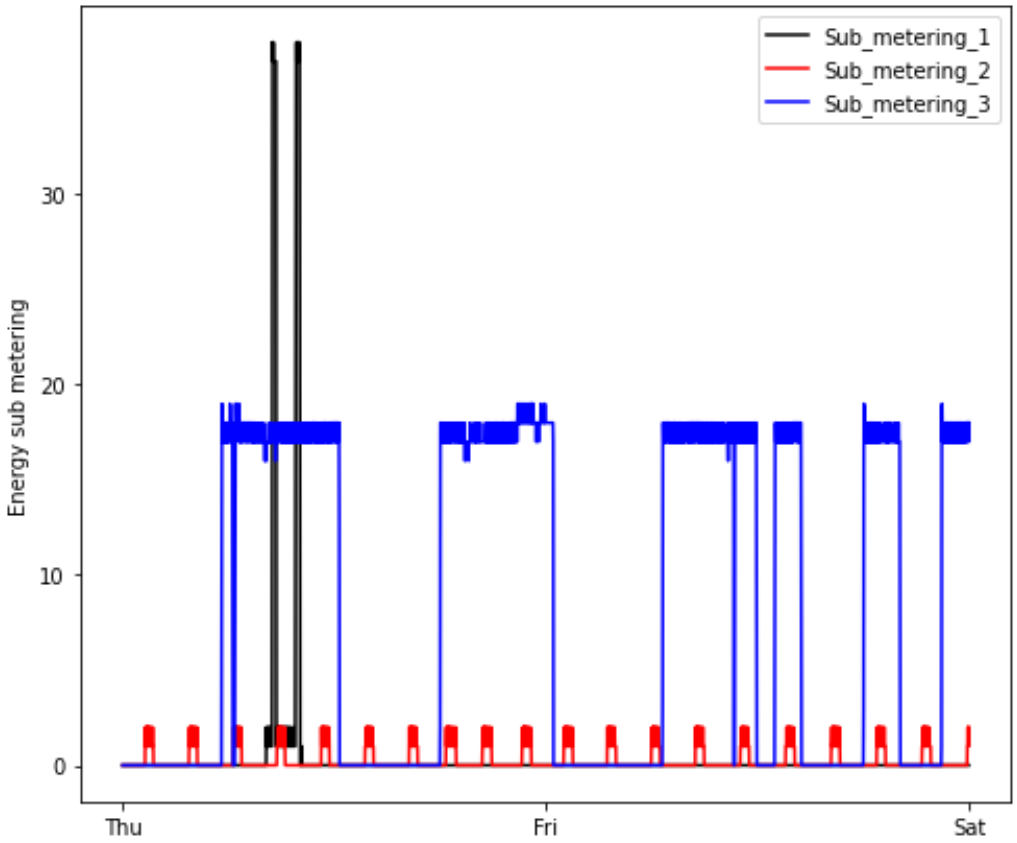
```
def plot2():
    fig, ax= plt.subplots(figsize=(9, 7))
    ax.plot(data["Global_active_power"], color='black')
    ticks = ["Thu", "Fri", "Sat"]
    yticks = list(range(0, int(max(data["Global_active_power"])), 2))
    plt.xticks([0, data.shape[0]/2, data.shape[0]], ticks)
    plt.yticks(yticks)
    plt.ylabel("Global Active Power (kilowatts)")
    fig.savefig('./images/plot2.png', bbox_inches='tight')
    plt.show()

plot2();
```



Dựa vào biểu đồ 2, ta thấy trong một ngày, mức tiêu thụ điện sẽ phân hóa thành 4 khoảng thời gian khác nhau. Khoảng thời gian đầu tiên là từ 0 giờ

- khoảng 6 giờ. Khoảng thời gian này phần lớn dân số dành để ngủ nên lượng điện tiêu thụ chỉ đến từ việc sử dụng các thiết bị gia dụng hoạt động trong thời gian dài và ít tiêu tốn điện năng. Khoảng thời gian thứ 2 kéo dài từ khoảng 6 giờ đến 12 giờ. Mốc thời gian này chứng kiến lượng điện tiêu thụ tăng cao đáng kể, một vài thời điểm tiêu thụ đến hơn 5 kilowatts. Lượng điện tiêu thụ tăng cao này có thể được giải thích bởi mọi người đã thức dậy và thực hiện các sinh hoạt cần phải tiêu thụ điện. Các hoạt động này yêu cầu sử dụng các thiết bị có lượng điện tiêu thụ lớn như các thiết bị tạo nhiệt (bếp, máy sấy tóc,…). Việc tồn tại một vài điểm lượng diện tiêu thụ tăng cao đột biến là bởi vì sự phân hóa của thời điểm bắt đầu làm việc của mọi người. Các công việc thời nay thường có sự liên kết chặt chẽ với nhau, dẫn đến việc sẽ có một nhóm các công việc có thời điểm làm việc giống nhau, dẫn đến việc mọi người sẽ được chia thành các nhóm có thời gian biểu tươngtwj nhau và lượng điện tiêu thụ vì thế sẽ có những thời điểm đột biến khi nhiều người cần sử dụng. Nhóm thứ 3 kéo dài từ khoảng 12 giờ đến 17 giờ, giai đoạn này phần lớn mọi người đang làm việc hoặc học tập nên sẽ không sử dụng nhiều đồ gia dụng, lượng điện tiêu thụ vì thế khá thấp. Khoảng thời gian còn lại là từ 17 giờ đến 24 giờ, khoảng thời gian này mọi người thường dành thời gian ở nhà nên lượng điện tiêu thụ sẽ khá cao. Nhưng do buổi tối chúng ta ít chịu ràng buộc từ các hoạt động xã hội hơn nên lượng điện tiêu thụ sẽ mang tính ngẫu nhiên hơn.

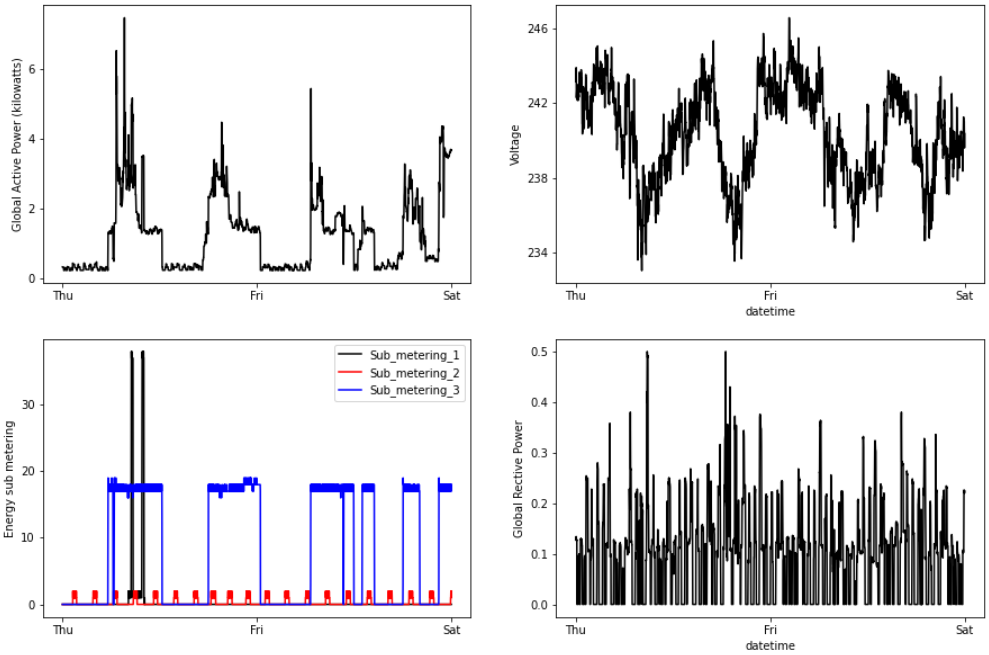
```
def plot3():
    fig, ax =plt.subplots(figsize=(8,7))
    ax.plot(data["Sub_metering_1"], color='black')
    ax.plot(data["Sub_metering_2"], color='red')
    ax.plot(data["Sub_metering_3"], color='blue')
    ax.legend(["Sub_metering_1", "Sub_metering_2", "Sub_metering_3"])
    ticks = ["Thu", "Fri", "Sat"]
    yticks = list(range(0, int((max(data["Sub_metering_1"]) )), 10))
    plt.yticks(yticks)
    plt.xticks([0, data.shape[0]/2, data.shape[0]], ticks)
    plt.ylabel("Energy sub metering")
    fig.savefig('./images/plot3.png', bbox_inches='tight')
    plt.show()

plot3()
```



Ở biểu đồ 3, ta thấy được lượng điện tiêu thụ cho từng nhóm đồ gia dụng. Nhóm thứ nhất là nhóm đồ dùng nhà bếp (được biểu diễn bằng đường màu đen). Ta có thể thấy lượng điện tiêu thụ của nhóm sản phẩm này tập

trung vào một thời điêm rất ngắn nhưng lượng điện tiêu thụ rất cao. Điều này có thể được giải thích bởi nhóm đồ gia dụng này tiêu tốn rất nhiều điện năng nhưng không thường xuyên được sử dụng trong một ngày. Nhóm sản phẩm thứ 2 liên quan đến các hoạt động thiết yếu trong cuộc sống. Lượng diện tiêu thụ ở nhóm sản phẩm này duy trì ở mức ổn định trong thời gian dài. Bù lại, ở từng thời điểm thì lượng điện tiêu thụ không quá lớn. Nhóm sản phẩm cuối cùng là các sản phẩm không quá thiết yếu (máy nước nóng và máy lạnh). Ta thấy lượng diện tiêu thụ ở nhóm sản phẩm này khá cao và duy trì trong thời gian khá dài. Đối với nhóm sản phẩm này, những người nào có trang bị thì sẽ sử dụng khi họ ở nhà và tắt đi khi ra ngoài. Vì thế, mỗi lần sử dụng thường kéo dài lâu và mỗi lần dừng sử dụng cũng khá lâu.

```
def plot4():
    fig = plt.figure(figsize=(15,10))

    ax1 = fig.add_subplot(221)
    ax1.plot(data["Global_active_power"], color='black')
    ticks = ["Thu", "Fri", "Sat"]
    plt.xticks([0, data.shape[0]/2, data.shape[0]], ticks)
    plt.yticks(list(range(0, int(max(data["Global_active_power"])), 2)))
    plt.ylabel("Global Active Power")

    ax2=fig.add_subplot(222)
    ax2.plot(data["Voltage"], color='black')
    yticks=list(range(int(data["Voltage"].min()+1), int(data["Voltage"].max()+1), 4))
    ticks = ["Thu", "Fri", "Sat"]
    plt.xticks([0, data.shape[0]/2, data.shape[0]], ticks)
    plt.yticks(yticks)
    plt.xlabel("datetime")
    plt.ylabel("Voltage")

    ax3=fig.add_subplot(223)
    ax3.plot(data["Sub_metering_1"], color='black')
    ax3.plot(data["Sub_metering_2"], color='red')
    ax3.plot(data["Sub_metering_3"], color='blue')
    ax3.legend(["Sub_metering_1", "Sub_metering_2", "Sub_metering_3"])
    yticks = list(range(0, int((max(data["Sub_metering_1"]) )), 10))
    ticks = ["Thu", "Fri", "Sat"]
    plt.yticks(yticks)
    plt.xticks([0, data.shape[0]/2, data.shape[0]], ticks)
    plt.ylabel("Energy sub metering")

    ax4=fig.add_subplot(224)
    ax4.plot(data["Global_reactive_power"], color='black')
    ticks = ["Thu", "Fri", "Sat"]
    plt.xticks([0, data.shape[0]/2, data.shape[0]], ticks)
    plt.ylabel("Global Rective Power")
    plt.xlabel("datetime")
    fig.savefig('./images/plot4.png', bbox_inches='tight')
    fig.show();

plot4()
```



Khi quan sát cả 4 biểu đồ ở biểu đồ 4, ta nhận thấy các biểu đồ này có điểm chung về mức độ tiêu thụ điện trong mỗi khoảng thời gian. Biểu đồ thể hiện Votage cao ở thời điểm ít tiêu thụ điện năng và thấp ở thời điểm điện năng có lượng tiêu thụ lớn. Điều khiến lượng điện tiêu thụ cao điểm của một ngày trùng với thời điểm mà lượng điện tiêu thụ cho các sản phẩm nâng cao chất lượng cuộc sống (nhóm 3) và các hoạt động nấu nướng lớn. Và ở mỗi thời điểm, công suất phản khảng kháng đều khá cao và ổn định bởi công suất phản kháng ít chịu ảnh hưởng từ vấn đề tiêu thụ điện của người sử dụng mà nó luôn xảy ra ở mọi hoàn cảnh. Tuy nhiên, khi lượng điện tiêu thụ tăng cao lên, dẫn đến việc diện phải được truyền tải nhiều hơn thì công suất phản kháng cũng sẽ tăng.

Tài liệu tham khảo

https://viblo.asia/p/gioi-thieu-ve-numpy-mot-thu-vien-chu-yeu-phuc-vu-cho-khoa-hoc-may-tinh-cua-python-maGK7kz9Kj2

https://pandas.pydata.org/

https://towardsdatascience.com/data-visualization-using-matplotlib-16f1aae5ce70