ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

BÁO CÁO LAB 03
TRỰC QUAN HÓA DỮ LIỆU VỚI NUMPY, PANDAS, MATPLOTLIB

Bộ môn: Trực quan hóa dữ liệu
Giảng viên hướng dẫn: Lê Ngọc Thành

2020 - 2021

MỤC LỤC

I. THÔNG TIN NHÓM VÀ MỨC ĐỘ HOÀN THÀNH ĐỒ ÁN

1. Thông tin nhóm

Họ và tên	MSSV	Công việc	Mức độ hoàn thành
Đinh Phan Kim Ngân	18120476	Lab 3 electricpower.py	100%
Lê Hoàng Phương Nhi	18120496	Lab 3 carmpg.py	100%
Nguyễn Thị Hồng Nhung	18120498	Lab 3 electricpower.py	100%
Nguyễn Thành Phát	18120501	Lab 3 carmpg.py	100%
Lê Thị Như Quỳnh	18120530	Lab 3 electricpower.py	100%

2. Mức độ hoàn thành đồ án

Các tiêu chí	Mức độ hoàn thành
Mô tả các thư viện liên quan đến chức năng trực quan hóa dữ liệu	100%
Thực hiện theo yêu cầu và giải thích chi tiết các kiến thức liên quan	100%
Minh chứng thực hiện (hình chụp màn hình)	100%

II. NỘI DUNG CHI TIẾT

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

Matplotlib:

Là một thư viện vẽ đồ thị rất mạnh mẽ hữu ích cho những người làm việc với python và Numpy.

- ✓ Khái niệm chung:
  - Figure: Như một cái cửa sổ chứa tất cả những gì bạn sẽ vẽ trên đó.
  - Axes: Thành phần chính của một figure là các axes (những khung nhỏ hơn để vẽ hình lên đó). Một figure có thể chứa một hoặc nhiều axes. Nói cách khác, figure chỉ là khung chứa, chính các axes mới thật sự là nơi các hình vẽ được vẽ lên.
  - Asix: chúng là dòng số giống như các đối tượng và đảm nhiệm việc tạo các giới hạn biểu đồ.
  - Artist: Mọi thứ mà bạn có thể nhìn thấy trên figure là một artist như Text objects, Line2D objects, collection objects, Hầu hết các Artists được gắn với Axes.

2. Các yêu cầu

2.1. Lab3_carmpg

2.1.1. Có bao nhiêu ô tô và bao nhiêu thuộc tính trong tập dữ liệu?

Sử dụng câu lệnh **data.car_name.duplicated().sum()** để đếm số tên xe trùng lắp trong data set.

```
data.car_name.duplicated().sum()
```

```
94
```

Sử dụng câu lệnh **data.duplicated().sum()** để đếm số dòng trùng lắp trong data set.

```
data.duplicated().sum()
```

```
0
```

⇨ Ta có thể thấy dù tên xe có bị trùng lắp nhưng lại không có dòng trùng nào trong tập dữ liệu tức là dù xe có trùng tên nhưng các thông số kỹ thuật lại khác nhau. Do đó số xe ô tô trong tập dữ liệu chính bằng số dòng của data set.

Sử dụng câu lệnh **data.shape** để trả về kích thước (số dòng, số cột) của data set. Khi đó số xe ô tô và số thuộc tính trong tập dữ liệu chính bằng số dòng và số cột của data set.

```
row, col = data.shape
print(row)
print(col)
```

```
406
9
```

⇨ Như vậy có 406 ô tô và 9 thuộc tính trong tập dữ liệu.

2.1.2. Có bao nhiêu công ty xe hơi khác nhau được đại diện trong tập dữ liệu? Tên xe có MPG tốt nhất? Hãng xe nào sản xuất nhiều xe 8 xi lanh nhất? Tên xe ô tô 3 xi lanh? Thực hiện một số tìm kiếm trên Internet và cho biết về lịch sử và sự phổ biến của những chiếc ô tô 3 xi lanh?

✓ Sau khi xem xét thuộc tính **car_name** thì nhận thấy tên công ty xe hơi là từ đầu tiên trong chuỗi tên xe. Vì vậy sử dụng câu lệnh **data.car_name.str.split(" ").str[0]** để trả về tên công ty xe hơi. Và tiến hành thêm cột **company** vào data.

```
#Thêm cột `company` vào data
data['company'] = data.car_name.str.split(" ").str[0]
data
```

✓ Sau khi xem xét các giá trị ở cột **company** thì thấy có một số tên hãng xe trùng lặp hoặc bị sai vì vậy tiến hành chỉnh sửa các giá trị đó

```
#Chỉnh các tên hãng chưa đúng hoặc trùng lắp
data['company'] = data['company'].str.replace('volkswagen','VW')
data['company'] = data['company'].str.replace('vokswagen','VW')
data['company'] = data['company'].str.replace('vw','VW')
data['company'] = data['company'].str.replace('maxda','mazda')
data['company'] = data['company'].str.replace('toyouta','toyota')
data['company'] = data['company'].str.replace('mercedes-benz','mercedes')
data['company'] = data['company'].str.replace('chevy','chevrolet')
data['company'] = data['company'].str.replace('chevroelt','chevrolet')
data['company'] = data['company'].str.replace('datsun','nissan')
data['company'] = data['company'].str.replace('capri','ford')
```

✓ Sau đó sử dụng lệnh **data.company.nunique()** để đếm số lượng công ty xe hơi khác nhau trong data set.

```
: data.company.nunique()
: 29
```

⇨ Vậy có 29 công ty xe hơi khác nhau được đại diện trong tập dữ liệu.

✓ Sử dụng câu lệnh **data.mpg.max()** để trả về MPG tốt nhất. Vì vậy sử dụng câu lệnh **data[data.mpg == data.mpg.max()].car_name** để trả về tên của xe ô tô có MPG tốt nhất.

```
data[data.mpg == data.mpg.max()].car_name
329    mazda glc
Name: car_name, dtype: object
```

⇨ Vậy xe ô tô có MPG tốt nhất là: mazda glc

✓ Sử dụng câu lệnh **data[data.cylinders == 8].company** để trả về các hãng xe có sản xuất ô tô 8 xi lanh. Sau đó dùng câu lệnh **value_counts()** để trả về số lượng của từng giá trị riêng biệt.

```
data[data.cylinders == 8].company.value_counts() #Số lượng
```

```
ford          22
chevrolet     21
dodge         12
plymouth      11
amc            9
pontiac        7
oldsmobile     7
buick          7
mercury        5
chrysler       4
cadillac       2
hi             1
Name: company, dtype: int64
```

⇨ Vậy hãng xe sản xuất nhiều xe 8 xi lanh nhất là: ford

✓ Sử dụng câu lệnh **data[data.cylinders == 3].car_name** để trả về tên các xe ô tô có 3 xi lanh.

```
data[data.cylinders == 3].car_name
```

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

✓ Vậy các xe có 3 xi lanh là: mazda rx2 coupe, maxda rx3, mazda rx-4, mazda rx-7 gs.

✓ Lịch sử và sự phổ biến của những chiếc ô tô 3 xi lanh:

- Trước đây, đã có rất nhiều những ý kiến và niềm tin sai, lạc hậu trong ngành công nghiệp ô tô. Một số nhà phê bình đã đưa ra các lập luận rằng: để có đủ sức mạnh cho động cơ 3 xi lanh phải đòi hỏi kỹ thuật quá mức, nhiều bộ phận chuyển động hơn sẽ có thể mang lại nhiều sự cố hơn,... Cụ thể, ô tô 3 xi lanh trong quá khứ có tiếng giật rung thật sự tệ bởi vì chúng không có nhiều năng lượng và chiếc đầu tiên được bán ở Hoa Kỳ là Geo Metro, nó khá là lớn và có động cơ 3 xi lanh 1 lít được sản xuất bởi Suzuki chỉ đưa ra 55 hp.

- Tuy nhiên, động cơ 3 xi lanh mới (hiện đại) đã khắc phục được những khiếm khuyết trong quá khứ của động cơ 3 xi lanh cũ nhờ những tiến bộ trong kĩ thuật. Kể từ khi những chiếc ô tô 3 xi lanh đầu tiên xuất hiện trên đường vào những năm 1970 thì kỹ thuật ô tô đã đi một con đường dài. Nhờ những tiến bộ bượt bậc trong kĩ thuật mà động cơ 3 xi lanh hiện đại như Ford Ecosport đã mượt mà hơn, việc rung giật không cân bằng không còn là mối đe dọa như trước đây nữa, những động cơ 3 xi lanh 1.0 hiện đại này cho ra 123 hp khác xa so với loại 55 hp lúc trước có trong Geo Metro. Ngoài ra, việc đưa bộ tăng áp vào động cơ giúp nó có nhiều công suất hơn trong khi sử dụng ít nhiên liệu hơn, có thể nói sự cải tiến trong ngành công nghệ tăng áp là sự bổ sung hoàn hảo cho sự phát triển tiếp tục của động cơ 3 xi lanh. Nói chung, động cơ 3 xi lanh hiện đại đã mạnh mẽ, hiệu quả và đáng tin cậy hơn rất nhiều.
- Ngày càng có nhiều loại xe 3 xi lanh trở nên phổ biến vì thật sự không thể phủ nhận được tính thực dụng của nó. Nó cần ít nhiên liệu nhưng năng lượng lại lớn, vì cần ít nhiên liệu nên chi phí nhiên liệu cũng thấp hơn. Hơn nữa, 3 xi lanh thì sẽ dễ dàng chăm sóc và bảo dưỡng hơn. Các nhà sản xuất ô tô cũng được hưởng lợi từ những động cơ nhỏ hơn này: một điều quan trọng là cắt giảm kích thước động cơ, trong lượng nhẹ và kích thước nhỏ của động cơ giúp dễ dàng thiết kế xung quanh, và một chiếc ô tô 3 xi lanh rõ ràng là rẻ hơn để sản xuất so với chiếc 4 xi lanh. Tại Ford, các chuyên gia rất tự tin về tiềm năng của động cơ 3 xi lanh 1.0 lít này của họ và họ đã ngừng bán động cơ 4 xi lanh 1.6 lít - một mặt hàng chủ lực của tất cả các nhà sản xuất thị trường đại chúng trong nhiều thập kỷ. Động cơ 3 xi lanh cũng được thừa nhận là động cơ tinh tế nhất hiện có trên Focus.

2.1.3. Phạm vi, giá trị trung bình và độ lệch chuẩn của từng thuộc tính? Chú ý đến tiềm năng giá trị bị thiếu.

Để tính các giá trị trên thì các thuộc tính phải có dạng số. Vì vậy sử dụng câu lệnh **num_cols = data.select_dtypes(include = 'number').columns** để chọn ra các cột có dạng số trong

data set. Sau đó sử dụng câu lệnh **data[num_cols].describe()** để đưa ra các thông tin về: **count, mean, std, min, 25%, 50%, 70%, max** của các thuộc tính.

```
num_cols = data.select_dtypes(include = 'number').columns
```

```
data[num_cols].describe()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model	origin
count	398.000000	406.000000	406.000000	400.000000	406.000000	406.000000	406.000000	406.000000
mean	23.514573	5.475369	194.779557	105.082500	2979.413793	15.519704	75.921182	1.568966
std	7.815984	1.712160	104.922458	38.768779	847.004328	2.803359	3.748737	0.797479
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.000000	1.000000
25%	17.500000	4.000000	105.000000	75.750000	2226.500000	13.700000	73.000000	1.000000
50%	23.000000	4.000000	151.000000	95.000000	2822.500000	15.500000	76.000000	1.000000
75%	29.000000	8.000000	302.000000	130.000000	3618.250000	17.175000	79.000000	2.000000
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.000000	3.000000

✓ Như vậy phạm vi của từng thuộc tính thì xem xét ở các dòng min, max; giá trị trung bình của từng thuộc tính thì xem xét ở dòng mean; độ lệch chuẩn của từng thuộc tính thì xem xét ở dòng std.

Xây dựng hàm **missing_ratio** để xem xét về tiềm năng các giá trị bị thiếu ở từng thuộc tính.

```
def missing_ratio(data):
    return (data.isna().mean() * 100).round(1)
data[num_cols].agg([missing_ratio])
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model	origin
missing_ratio	2.0	0.0	0.0	1.5	0.0	0.0	0.0	0.0

2.1.4. Vẽ biểu đồ cho từng thuộc tính. Chú ý đến việc lựa chọn số lượng bins phù hợp. Viết 2-3 câu tóm tắt về khía cạnh thú vị của dữ liệu bằng cách xem biểu đồ.

Đầu tiên, xây dựng hàm **def bin_s(x)** để tính toán số lượng bins cho từng thuộc tính. Sử dụng quy tắc Freedman – Diaconis để tính độ rộng từng bin từ đó suy ra được số lượng bins bằng cách lấy (max – min)/binwidth. Quy tắc Freedman – Diaconis như sau:

$$\text{Bin width} = 2\,\frac{\text{IQR}(x)}{\sqrt[3]{n}}$$

Trong đó: IQR(x) = Q3(x) – Q1(x)

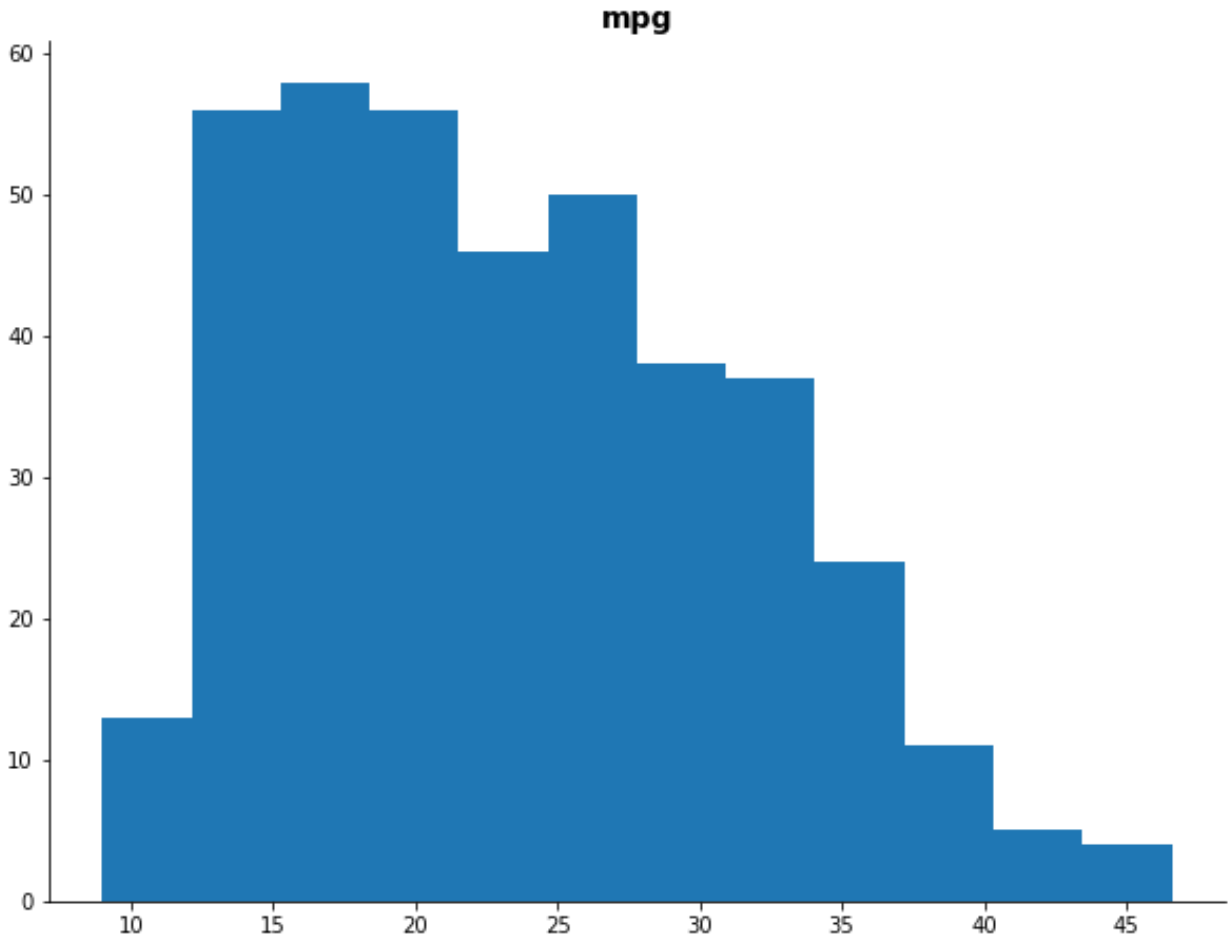
Từ đó, xây dựng được hàm tính bins như sau:

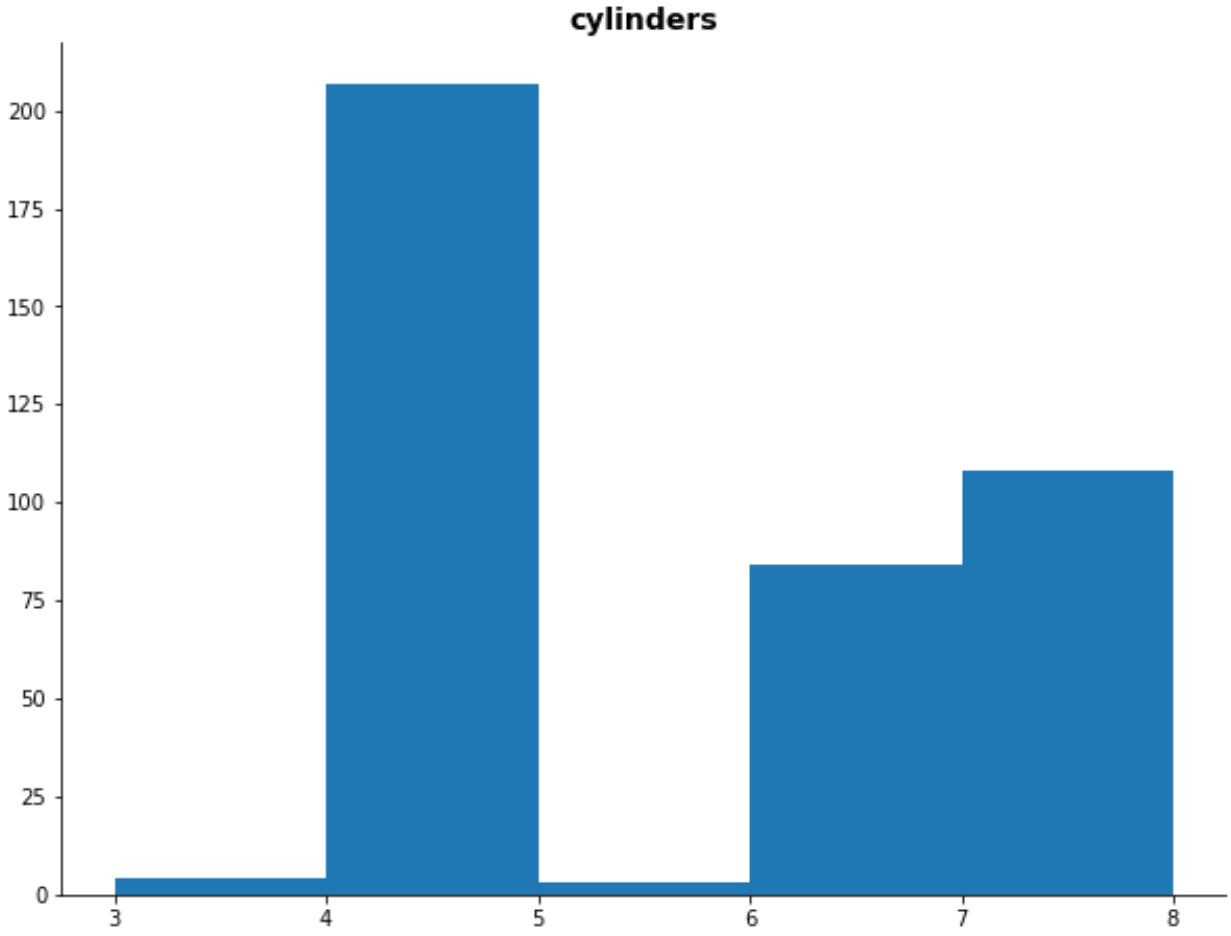
```
def bin_s(x):
    q25, q75 = data[x].quantile([0.25,0.75])
    bin_width = 2*(q75 - q25)*len(data[x])**(-1/3)
    bins = round((data[x].max() - data[x].min())/bin_width)
    return bins
```

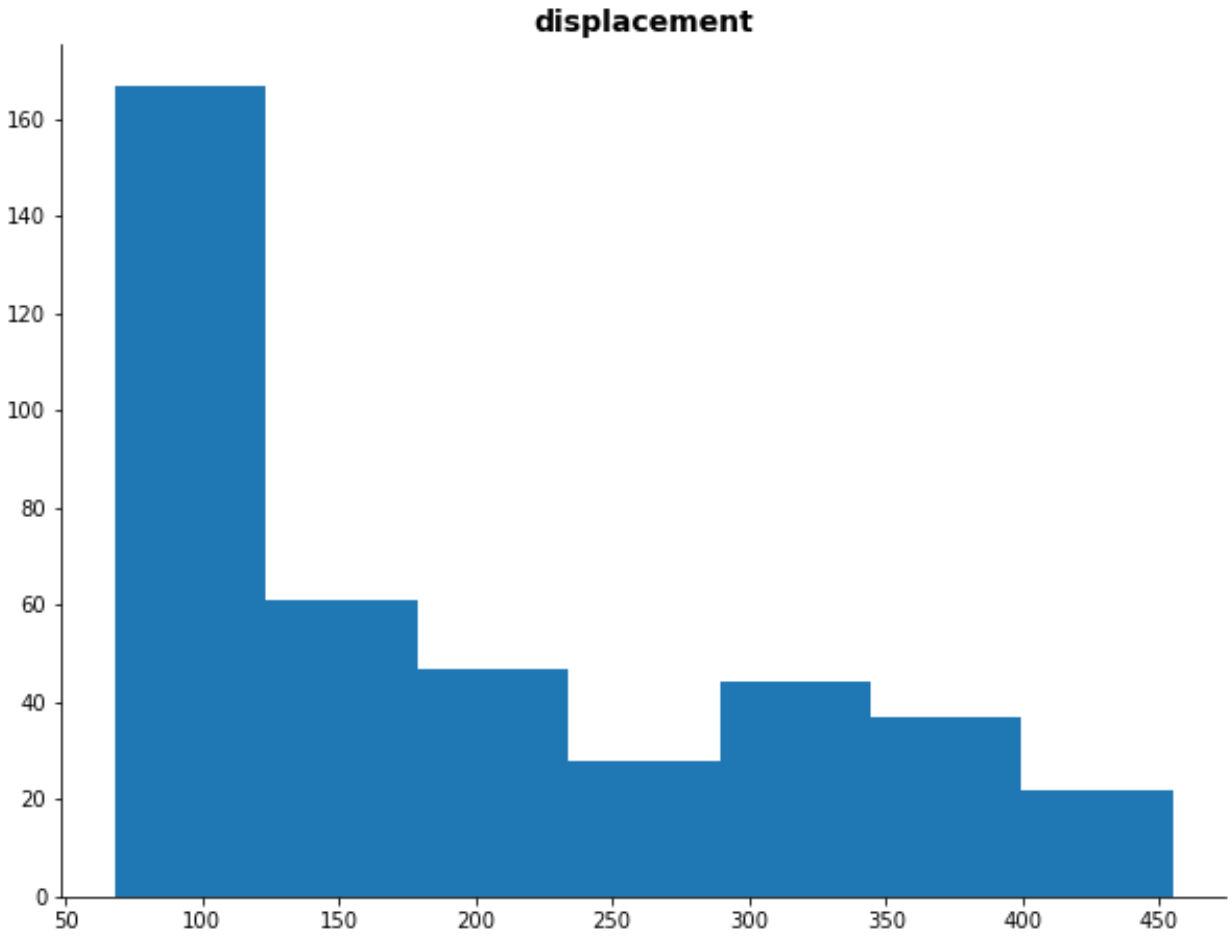
Sau đó, xây dựng hàm để vẽ biểu đồ histogram cho các thuộc tính. Sử dụng câu lệnh **plt.hist()** để vẽ.

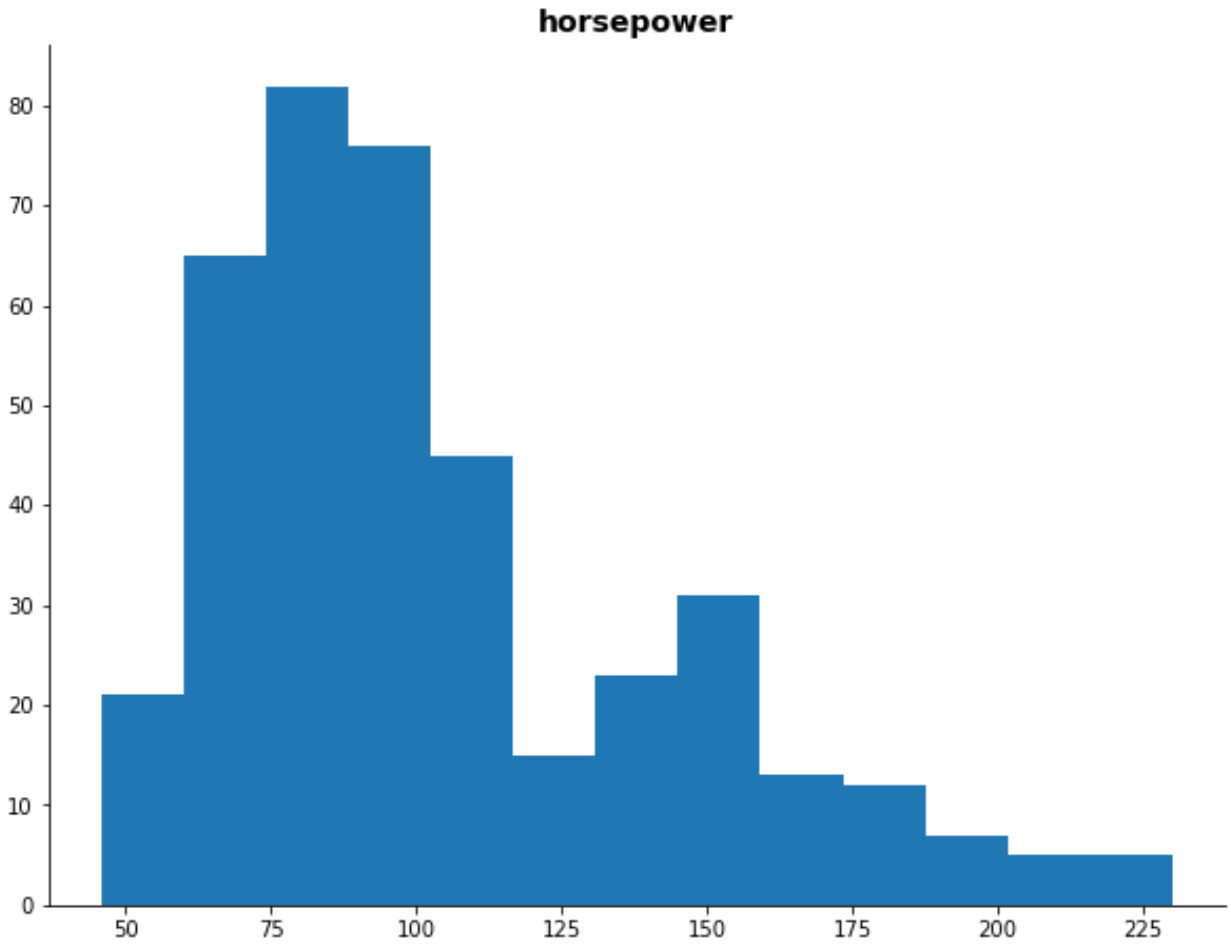
```
def hist_plot(x):
    fig_obj = plt.figure(figsize=(10, 7.5))
    ax = plt.subplot(111)

    ax.spines["bottom"].set_visible(True)  # Set the spines, or b
    ax.spines["left"].set_visible(True)
    ax.spines['right'].set_visible(False)
    ax.spines['top'].set_visible(False)

    ''' Plot the histogram of '''
    temp = data[x]
    bins = bin_s(x)
    p = plt.hist(temp, bins = bins)
    plt.title(x, fontsize=14, fontweight='bold')

    plt.show()
```

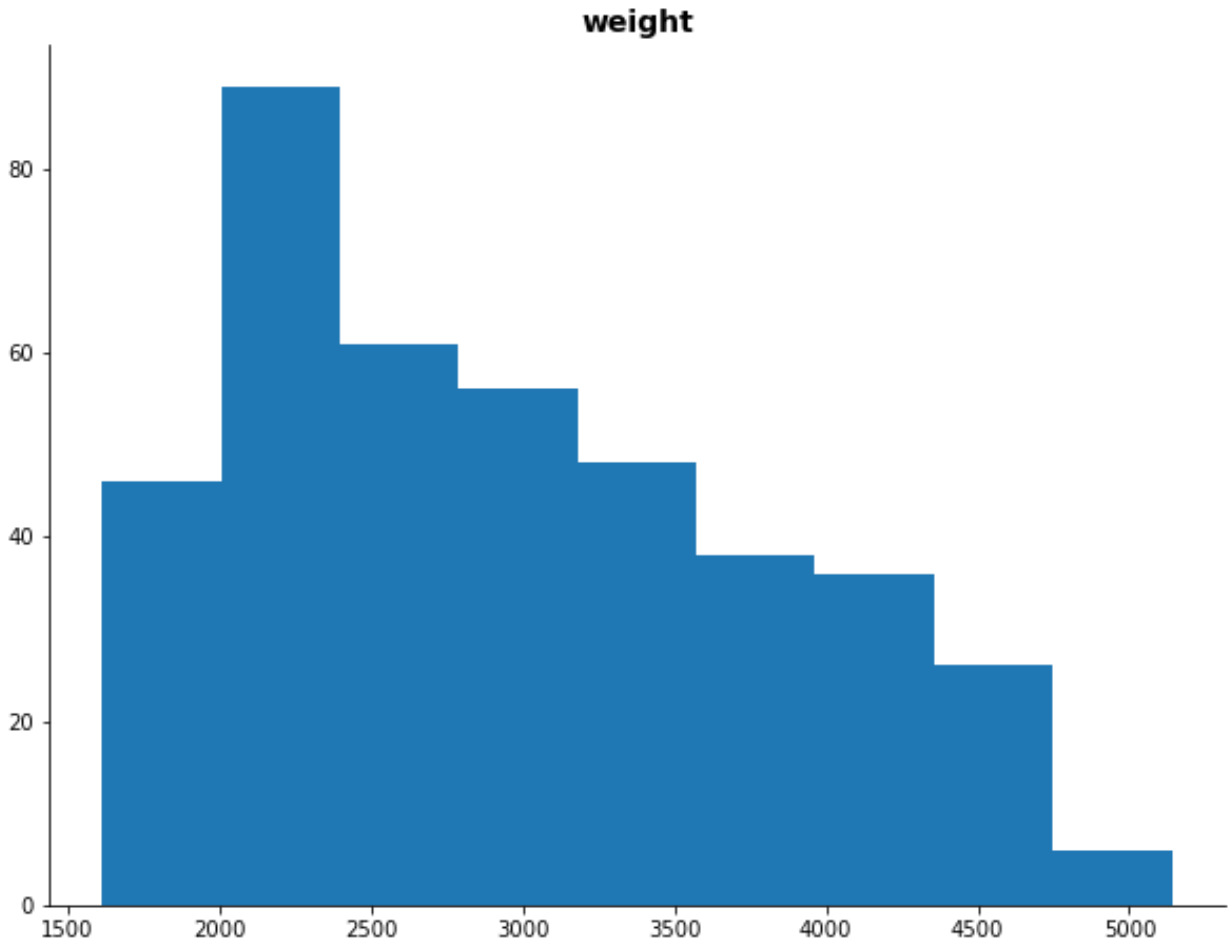
Cuối cùng, chạy vòng for để vẽ biểu đồ histogram cho từng thuộc tính số. Ta được 8 biểu đồ sau:

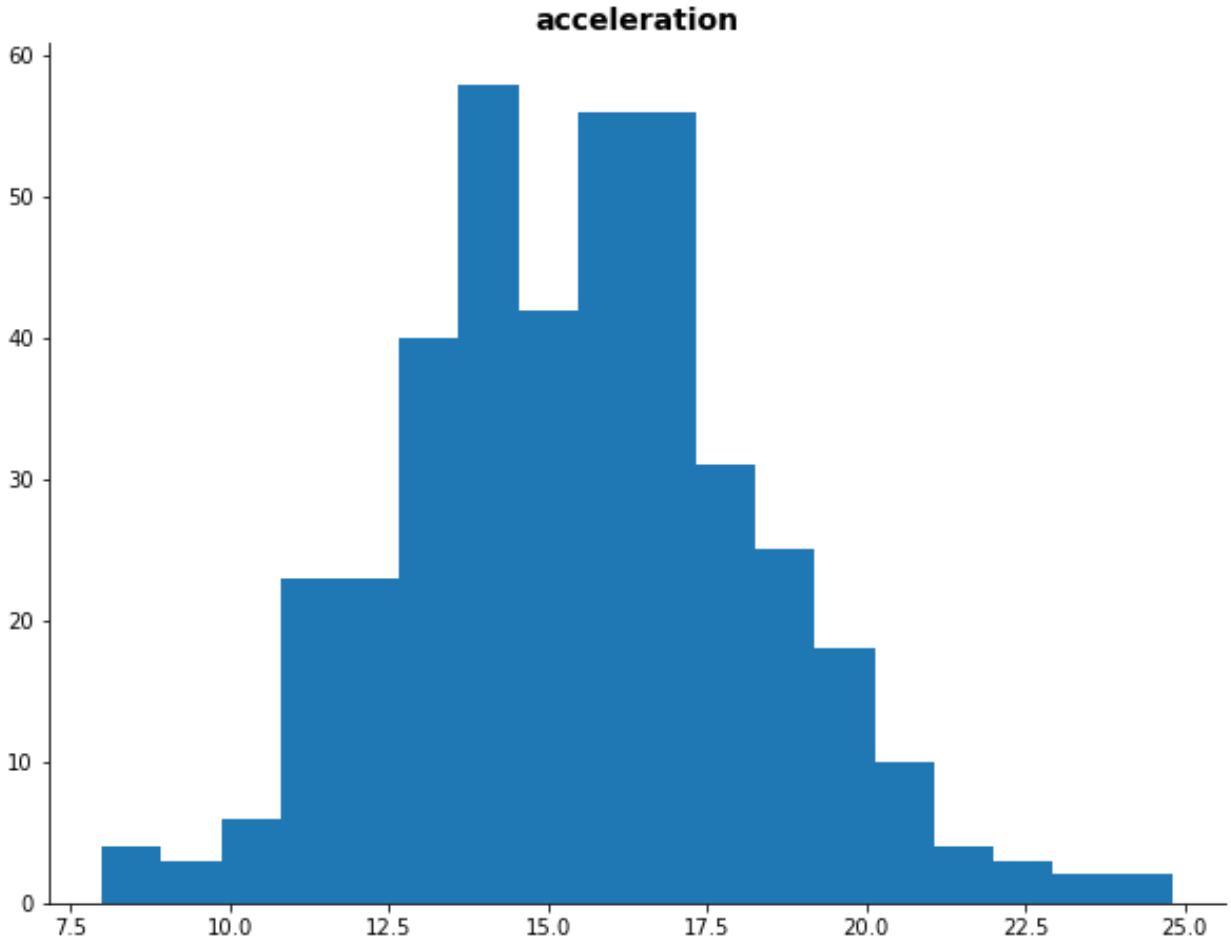

mpg
60
50
40
30
20
10
0
10
15
20
25
30
35
40
45

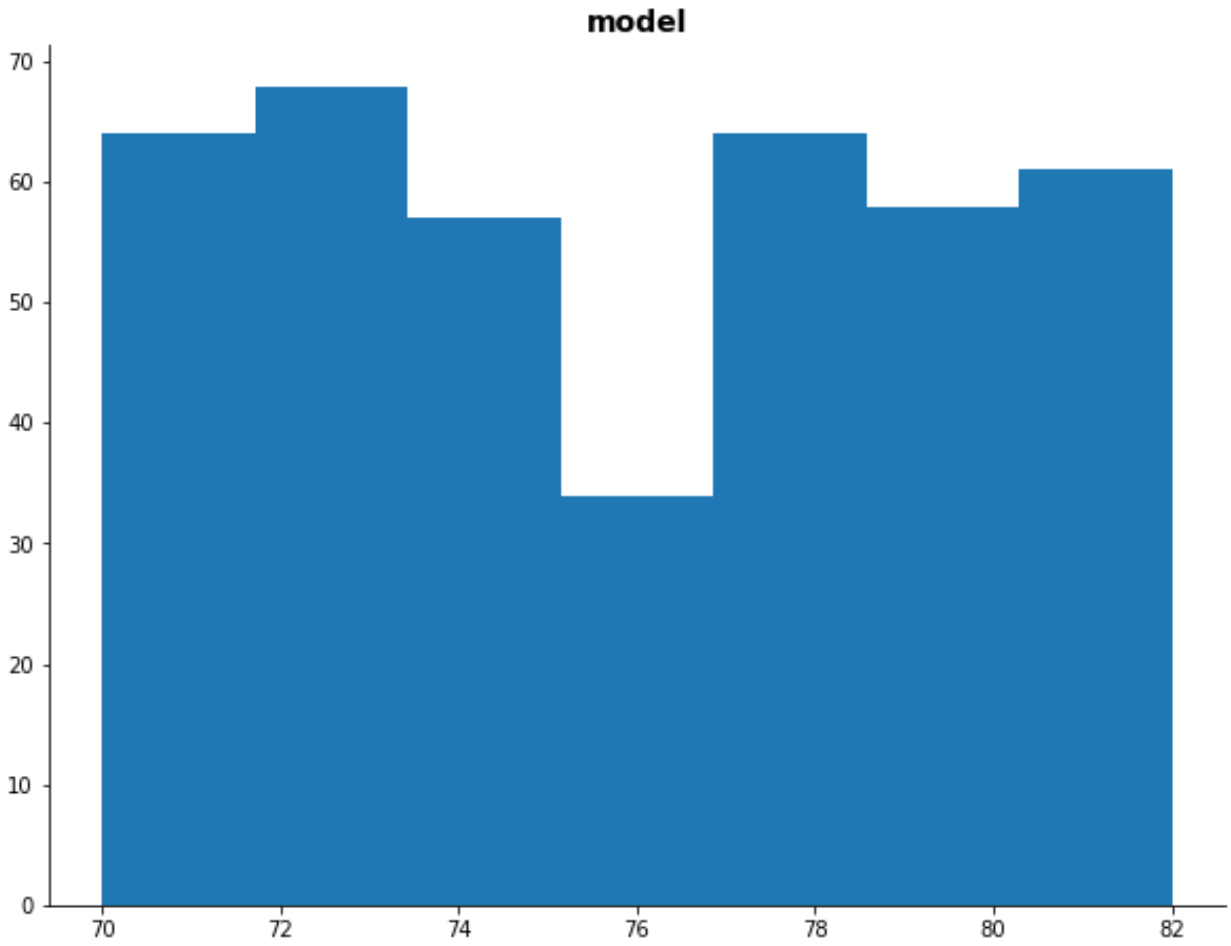

cylinders
200
175
150
125
100
75
50
25
0
3
4
5
6
7
8

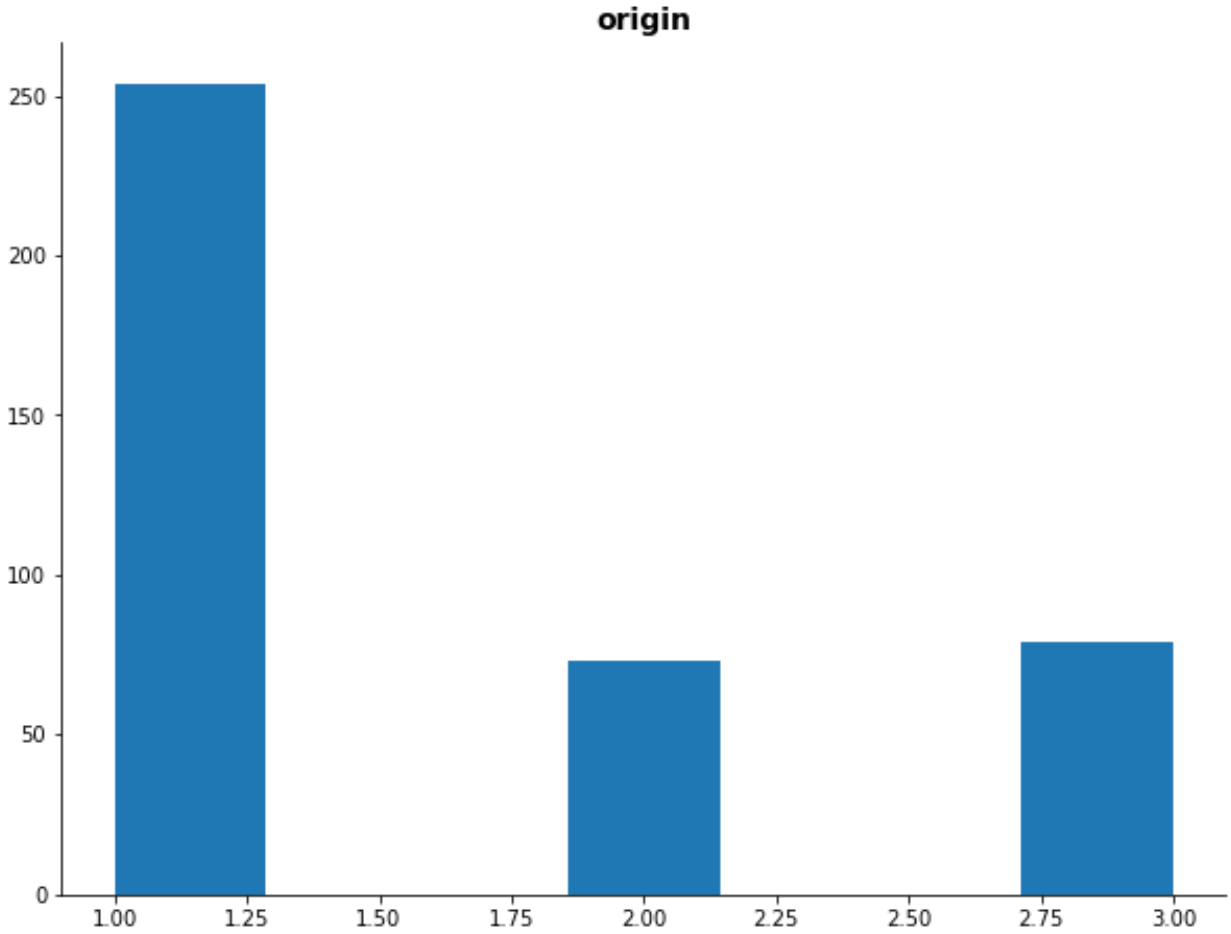

displacement
160
140
120
100
80
60
40
20
0
50
100
150
200
250
300
350
400
450


horsepower
80
70
60
50
40
30
20
10
0
50
75
100
125
150
175
200
225


weight
80
60
40
20
0
1500
2000
2500
3000
3500
4000
4500
5000


acceleration
60
50
40
30
20
10
0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0


model
70
60
50
40
30
20
10
0
70
72
74
76
78
80
82



Một số khía cạnh thú vị của dữ liệu:

- ✓ Từ biểu đồ của thuộc tính **origin** có thể thấy số lượng xe ô tô được sản xuất ở khu vực 1 chiếm đa số, nhiều vượt trội so với 2 khu vực còn lại (trên 250 so với tầm 70). Và theo tìm hiểu thì khu vực 1 là châu Mỹ, khu vực 2 là châu Âu, khu vực 3 là châu Á.
- ✓ Từ biểu đồ của thuộc tính **cylinders** có thể thấy các xe ô tô có số lượng xi lanh chủ yếu ở khoảng 4-5.
- ✓ Từ biểu đồ của thuộc tính **model** có thể thấy số lượng xe ô tô mới được sản xuất ra qua các năm trong giai đoạn từ năm 70-82 khá đồng đều, chỉ có vào khoảng những năm 76 số lượng này bị giảm đi tầm một nửa.

2.1.5. Vẽ biểu đồ phân tán của những thuộc tính weight vs. MPG. Có kết luận gì về mối quan hệ giữa các thuộc tính? Hệ số tương quan giữa 2 thuộc tính?

- ✓ Sử dụng câu lệnh **plt.scatter(data.weight,data.mpg,alpha=0.5)** để vẽ biểu đồ phân tán giữa weight vs. MPG. Để biểu đồ chi tiết và rõ ràng hơn thì sử dụng thêm các câu lệnh như:
    - plt.figure(figsize=(10, 7.5)): đặt kích cỡ cho biểu đồ.
    - plt.title('Scatterplot of weight vs. MPG', fontsize = 20): đặt tên biểu đồ và chỉnh cỡ chữ cho tên biểu đồ.

- plt.xlabel('weight', fontsize = 16): đặt tên cho trục x và chỉnh cỡ chữ cho tên trục x.
- plt.ylabel('mpg', fontsize = 16): đặt tên cho trục y và chỉnh cỡ chữ cho tên trục y.

```
plt.figure(figsize=(10, 7.5))
plt.scatter(data.weight,data.mpg,alpha=0.5)
plt.title('Scatterplot of weight vs. MPG', fontsize = 20)
plt.xlabel('weight', fontsize = 16)
plt.ylabel('mpg', fontsize = 16)
plt.show()
```

✓ Như vậy, ta sẽ được biểu đồ sau:

✓ Mối quan hệ giữa 2 thuộc tính: Từ biểu đồ phân tán trên ta thấy mpg lớn thì weight nhỏ và ngược lại mpg nhỏ thì weight lớn. Như vậy, 2 thuộc tính này tương quan nghịch.

✓ Sử dụng câu lệnh **data.mpg.corr(data.weight)** để tìm hệ số tương quan của 2 thuộc tính weight và mpg.

```
data.mpg.corr(data.weight)
```

```
-0.8317409332443354
```

⇨ Như vậy, hệ số tương quan giữa 2 thuộc tính là -0.8317409332443354

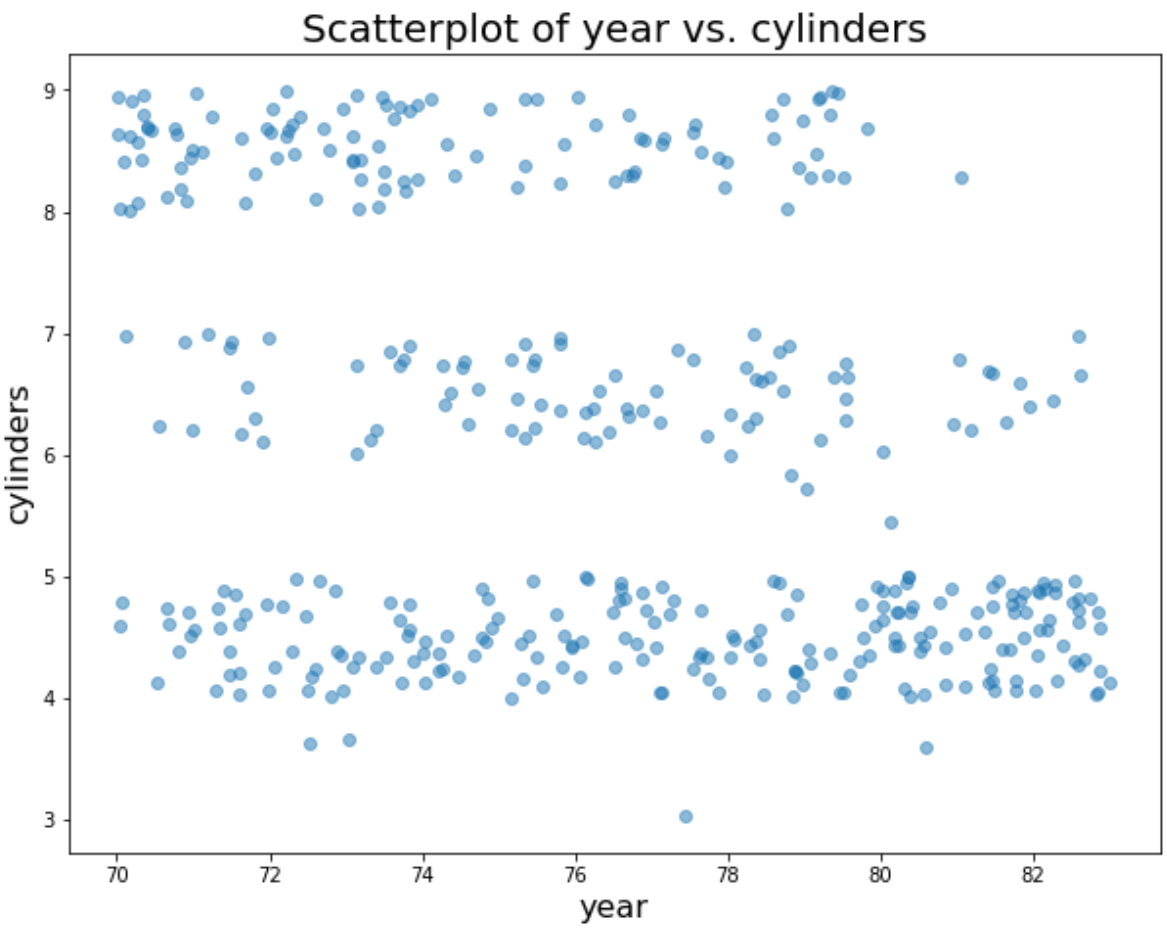
2.1.6. Vẽ biểu đồ phân tán của những thuộc tính year vs. cylinders. Thêm nhiễu ngẫu nhiên nhỏ vào các giá trị để scatterplot trông đẹp hơn. Bạn có thể kết luận điều gì? Thực hiện một số tìm kiếm trên Internet về lịch sử của ngành xe hơi trong suốt những năm 70 để có thể giải thích kết quả.

✓ Câu lệnh **np.random.random(len(data.model))** dùng để thêm các nhiễu ngẫu nhiên nhỏ cho thuộc tính model. Vì vậy, Sử dụng câu lệnh **plt.scatter(data.model + np.random.random(len(data.model)),data.cylinders+np.random.random(len(data.cylinders)), alpha=0.5)** để vẽ biểu đồ phân tán giữa year vs. cylinders. Để biểu đồ chi tiết và rõ ràng hơn thì sử dụng thêm các câu lệnh như:

- plt.figure(figsize=(10, 7.5)): đặt kích cỡ cho biểu đồ.
- plt.title('Scatterplot of year vs. cylinders' , fontsize = 20): đặt tên biểu đồ và chỉnh cỡ chữ cho tên biểu đồ.
- plt.xlabel('year', fontsize = 16): đặt tên cho trục x và chỉnh cỡ chữ cho tên trục x.
- plt.ylabel('cylinders', fontsize = 16): đặt tên cho trục y và chỉnh cỡ chữ cho tên trục y.

```
plt.figure(figsize=(10, 7.5))
plt.scatter(data.model + np.random.random(len(data.model)), data.cylinders + np.random.random(len(data.cylinders)), alpha=0.5)
plt.title('Scatterplot of year vs. cylinders', fontsize = 20)
plt.xlabel('year', fontsize = 16)
plt.ylabel('cylinders', fontsize = 16)
plt.show()
```

⇨ Như vậy, ta sẽ được biểu đồ sau:



⇨ Từ biểu đồ trên ta có thể thấy số lượng xi lanh có 3 mức là 4-5, 6-7, 8-9 và ở mức 4-5 là nhiều nhất. Nó phân bố khá đồng đều qua từng giai đoạn, tuy nhiên có vẻ như từ sau những năm 80 thì số ô tô có số xi lanh ở mức 8-9 là rất ít.

✓ Lịch sử ngành xe hơi trong suốt những năm 70:

- Mặc dù các loại ô tô chạy bằng hơi nước được sản xuất sớm hơn, nhưng nguồn gốc của ngành công nghiệp ô tô bắt nguồn từ sự phát triển của động cơ đốt trong trong những năm 1860 và 70, chủ yếu ở Pháp và Đức.
- Động cơ đốt trong là loại động cơ sử dụng nhiên liệu cháy nổ để đẩy piston trong xi-lanh, chuyển động tịnh tiến của piston làm quay trục cơ, sau đó làm bánh xe chuyển động nhờ xích tải hoặc trục chuyển động. Các loại nhiên liệu phổ biến nhất cho ô tô là xăng và dầu hỏa.
- Năm 1858, Jean Joseph Lenoir, một kỹ sư người Bỉ xin cấp bằng sáng chế động cơ đốt trongtác động kép, đánh lửa điện, sử dụng nhiên liệu khí than (1860).
- Vào năm 1863, Lenoir gắn động cơ này (đã được cải tiến, sử dụng nhiên liệu xăng và bộ chế hòa khí đơn giản) vào một chiếc xe coòng 3 bánh và thực hiện thành công chuyến đi mang tính lịch sử với quãng đường 50 dặm.

- 1862, kỹ sư người Pháp ông Alphanse Beau De Rochas đệ đơn cấp bằng sáng chế động cơ 4 kì số 52593 ngày 16/1/1862 (nhưng đã không sản xuất)
- 1864, Siegried Marus, một kỹ sư người Áo đã chế tạo một loại động cơ xi-lanhvowis bộ chế hòa khí rất thô sơ và sau đó gắn lên một chiếc xe ngựa và đã vận hành thành công trên quãng đường đá dài 500 foot (152,4 m). Vài năm sau đó, Marcus thiết kế một chiếc xe có thể vận hành với tốc độ 10 dặm/giờ và một số sử gia cho rằng đây mới chính là chiếc xe sử dụng động cơ xăng đầu tiên trên thế giới.
- 1866, 2 kỹ sư người Đức Eugen Langen và Nikolas August Otto cải tiến các thiết kế của Lenior và De Rochas và đã tạo ra được động cơ chạy gas có hiệu suất lớn hơn.
- 1873, kỹ sư người Mỹ George Brayton phát triển (nhưng không thành công) loại động cơ 2 kì chạy dầu hỏa, loại động cơ này dùng 2 xi-lanh bơm ngoài. Tuy vậy, loại động cơ này được coi như là loại động cơ dầu an toàn có giá trị ứng dụng đầu tiên.
- 1876, Nikolas August Otto phát minh thành công và được cấp bằng sáng chế động cơ 4 kì thì 2 loại động cơ này thường được gọi là "Chu kì Otto".
- 1876, Dougald Clerk chế tạo thành công động cơ 2 kì đầu tiên.

2.1.7. Vẽ thêm 2 biểu đồ phân tán mà bạn thấy thú vị. Thảo luận về những gì bạn thấy.

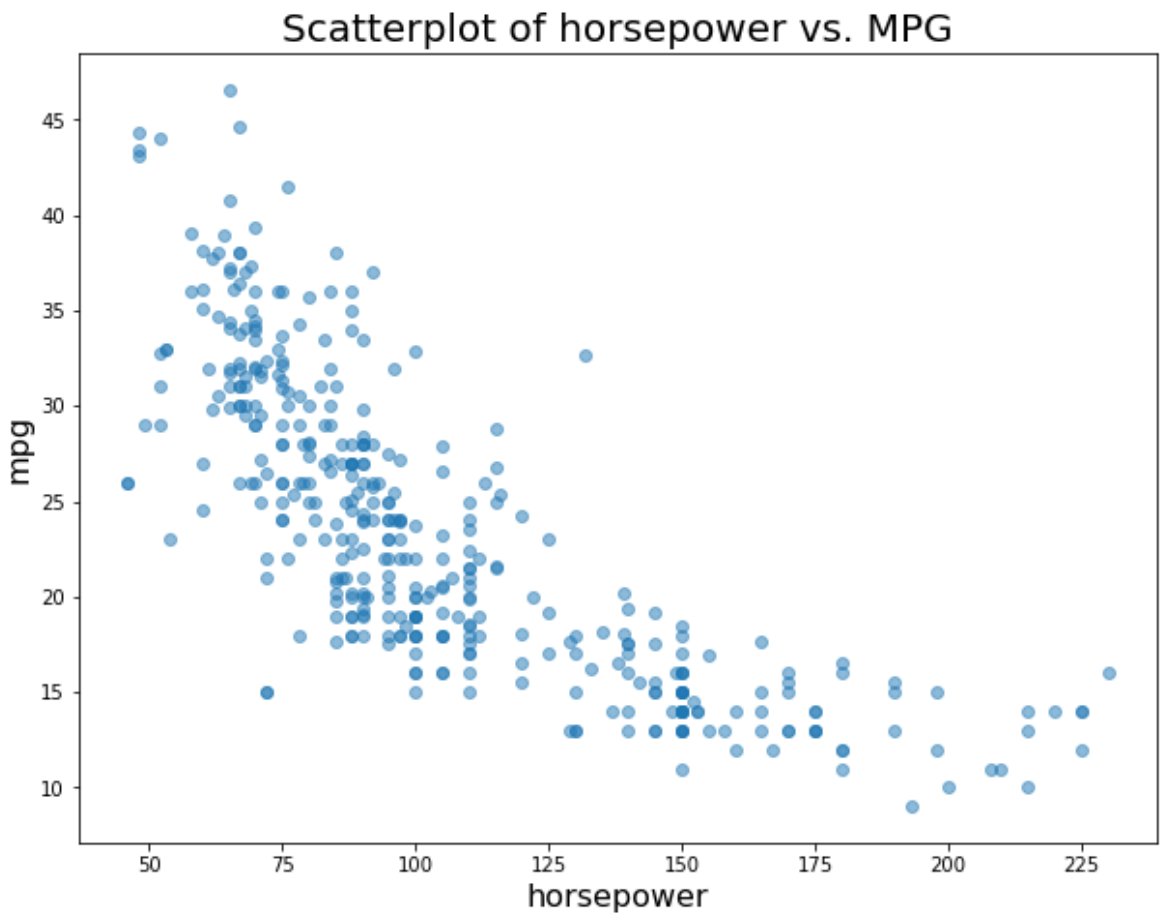
✓ Biểu đồ phân tán giữa **horsepower** và **MPG**:

Sử dụng câu lệnh **plt.scatter(data.horsepower,data.mpg,alpha=0.5)** để vẽ biểu đồ phân tán giữa horsepower vs. MPG. Để biểu đồ chi tiết và rõ ràng hơn thì sử dụng thêm các câu lệnh như:

- plt.figure(figsize=(10, 7.5)): đặt kích cỡ cho biểu đồ.
- plt.title('Scatterplot of horsepower vs. MPG', fontsize = 20): đặt tên biểu đồ và chỉnh cỡ chữ cho tên biểu đồ.
- plt.xlabel(' horsepower ', fontsize = 16): đặt tên cho trục x và chỉnh cỡ chữ cho tên trục x.
- plt.ylabel('mpg', fontsize = 16): đặt tên cho trục y và chỉnh cỡ chữ cho tên trục y.

```
plt.figure(figsize=(10, 7.5))
plt.scatter(data.horsepower,data.mpg,alpha=0.5)
plt.title('Scatterplot of horsepower vs. MPG', fontsize = 20)
plt.xlabel('horsepower', fontsize = 16)
plt.ylabel('mpg', fontsize = 16)
plt.show()
```

⇨ Như vậy, ta sẽ được biểu đồ sau:



⇨ Nhận xét về mối quan hệ giữa 2 thuộc tính: Từ biểu đồ phân tán trên ta thấy mpg lớn thì horsepower nhỏ và ngược lại mpg nhỏ thì horsepower lớn. Như vậy, 2 thuộc tính này tương quan nghịch.

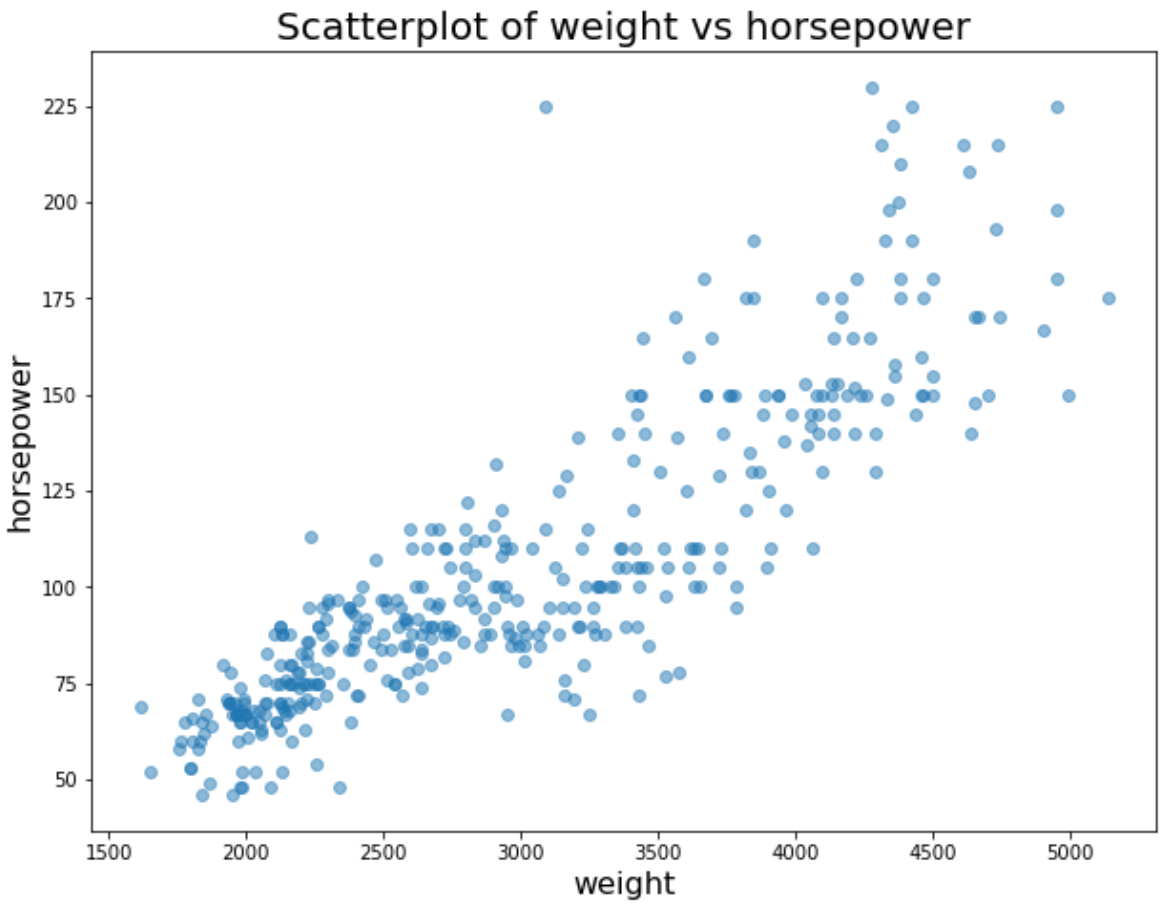
✓ Biểu đồ phân tán giữa **weight** và **horsepower**:

Sử dụng câu lệnh **plt.scatter(data.weight, data.horsepower, alpha=0.5)** để vẽ biểu đồ phân tán giữa weight và horsepower. Để biểu đồ chi tiết và rõ ràng hơn thì sử dụng thêm các câu lệnh như:

- plt.figure(figsize=(10, 7.5)): đặt kích cỡ cho biểu đồ.
- plt.title('Scatterplot of weight vs horsepower ', fontsize = 20): đặt tên biểu đồ và chỉnh cỡ chữ cho tên biểu đồ.
- plt.xlabel('weight', fontsize = 16): đặt tên cho trục x và chỉnh cỡ chữ cho tên trục x.
- plt.ylabel('horsepower', fontsize = 16): đặt tên cho trục y và chỉnh cỡ chữ cho tên trục y.

```
plt.figure(figsize=(10, 7.5))
plt.scatter(data.weight,data.horsepower,alpha=0.5)
plt.title('Scatterplot of weight vs horsepower', fontsize = 20)
plt.xlabel('weight', fontsize = 16)
plt.ylabel('horsepower', fontsize = 16)
plt.show()
```

⇨ Như vậy, ta sẽ được biểu đồ sau:



⇨ Nhận xét về mối quan hệ giữa 2 thuộc tính: Từ biểu đồ phân tán trên ta thấy weight lớn thì horsepower lớn, weight nhỏ thì horsepower nhỏ. Như vậy, 2 thuộc tính này tương quan thuận.

2.1.8. Vẽ một chuỗi thời gian cho tất cả các công ty hiển thị số lượng ô tô mới mà họ giới thiệu mỗi năm. Bạn có thấy một số xu hướng thú vị?

✓ Tạo hàm **line_plot** để vẽ biểu đồ biểu diễn số lượng xe của từng hãng qua mỗi năm

```
def line_plot(x,y,title):
    fig_obj = plt.figure(figsize=(15,7))
    ax = plt.subplot(111)

    ax.spines["bottom"].set_visible(True)  # Set the spines, or box bo
    ax.spines["left"].set_visible(True)
    ax.spines['right'].set_visible(False)
    ax.spines['top'].set_visible(False)

    ''' Plot the line of '''
    p = plt.plot(x,y,alpha = 0.5)
    plt.yticks(ticks = [0,1,2,3,4,5,6])
    plt.title(title, fontsize=14, fontweight='bold')
    plt.xlabel('Year', fontsize = 14)
    plt.ylabel('Number of cars', fontsize = 14)
    plt.show()
```

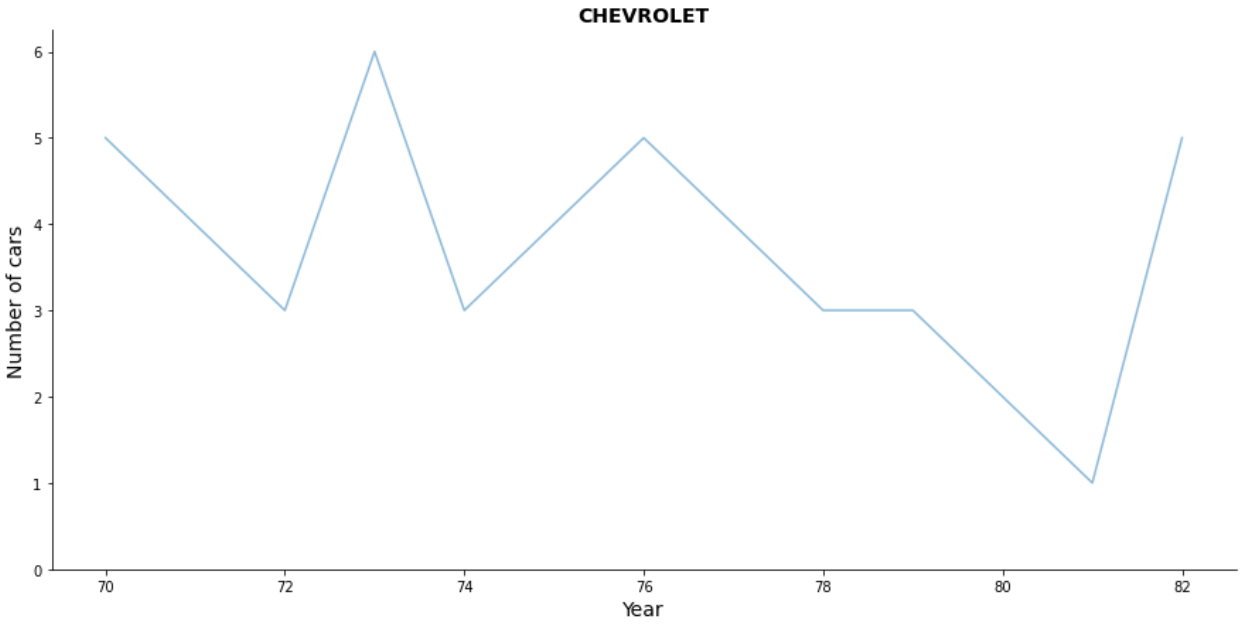
✓ Duyệt vòng for qua tất cả các hãng xe để gọi hàm vẽ:
Vì dữ liệu không liên tục, nghĩa là có những năm mà các hãng xe có thể không có xe nào. Vì vậy, ở những năm đó ta sẽ cập nhật là 0. Để xử lý việc này, làm các bước sau:

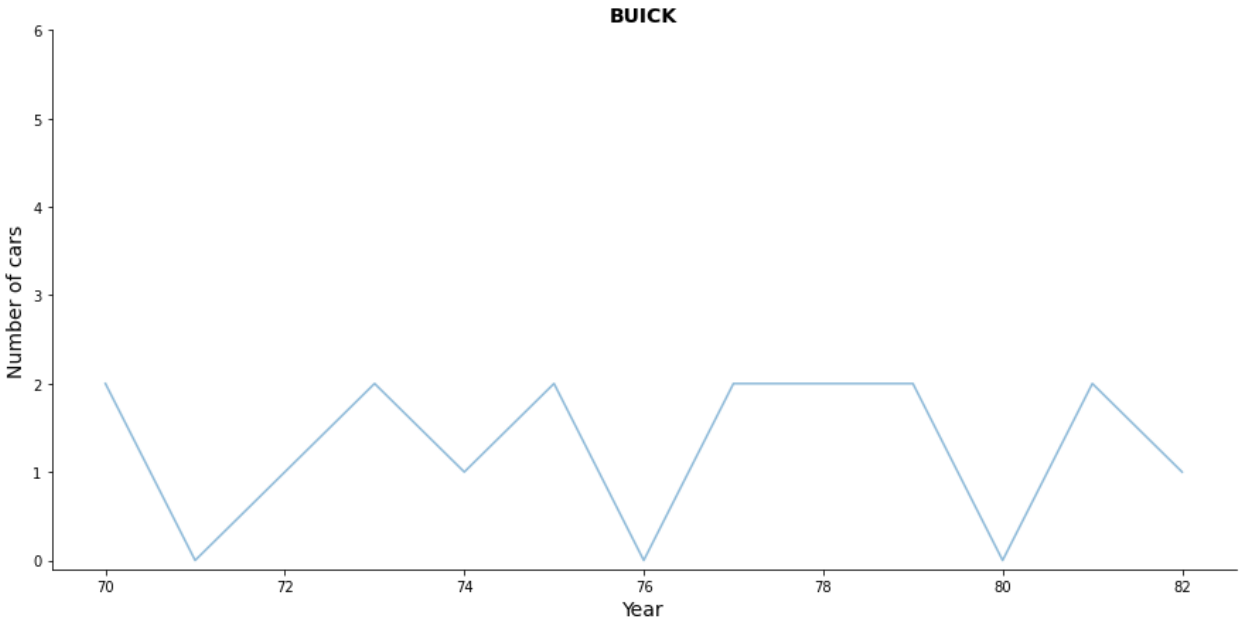
- Tạo 1 dataframe **temp** với 2 cột **model** và **count** biểu diễn cho năm và số lượng xe từng năm. Và ở đây model sẽ là các model có trong data, count sẽ là 0.
- Tạo 1 dataframe **sen** gồm các thuộc tính **company** (công ty), **model** (năm), **count** (số lượng xe của hãng qua mỗi năm) bằng cách groupby theo company và model đối với mỗi company sau đó đếm số lượng car_name.
- Sau đó, duyệt qua tất cả các năm có trong dataframe **sen** và tiến hành cập nhật cột count của dataframe **temp** theo cột **count** của dataframe **sen**.

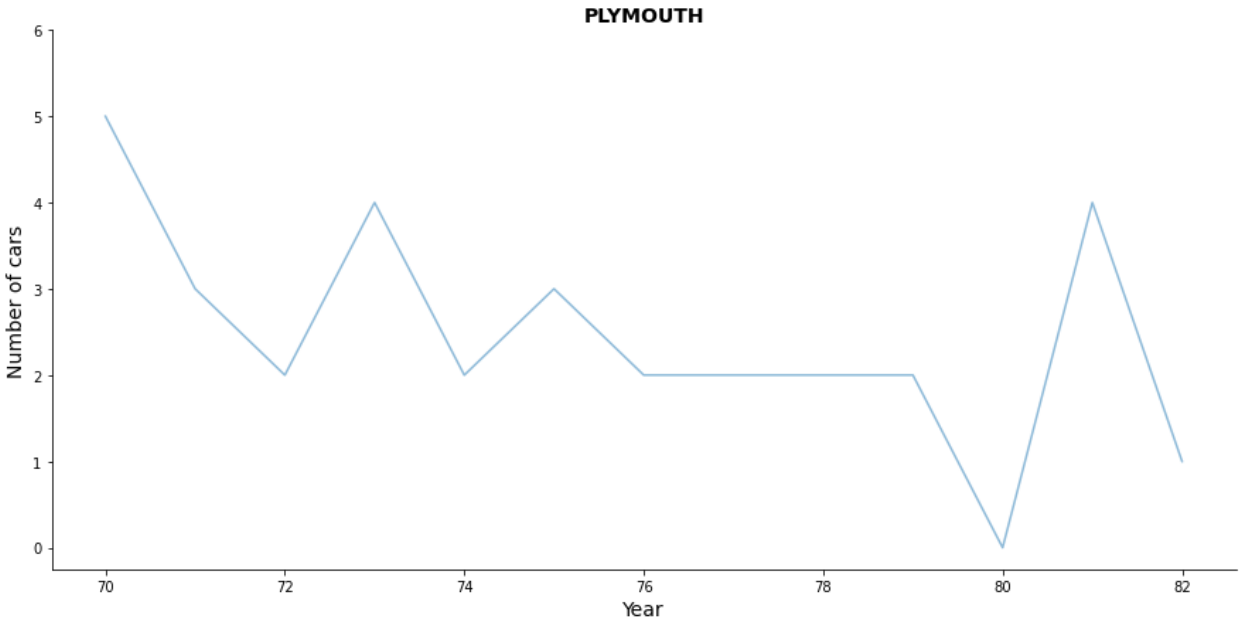
Gọi hàm **line_plot** để tiến hành vẽ biểu đồ.

```
#Vẽ biểu đồ cho từng hãng xe
for company in data.company.unique():
    #Tạo 1 dataframe với 2 cột model và count
    temp = pd.DataFrame()
    temp['model'] = data.model.unique()
    temp['count'] = 0

    sen = data[data['company'] == company].groupby(['company', 'model'])['car_name'].count().to_frame('count').reset_index()
    #Với từng hãng xe, năm nào có xe mới thì sẽ cập nhật lại count
    for j in sen.model.unique():
        temp.iloc[temp[temp.model == j].index[0],1]= sen.iloc[sen[sen.model == j].index[0],2]
    #Gọi hàm vẽ
    line_plot(temp['model'], temp['count'], company.upper())
```

⇨ Được 29 biểu đồ tương ứng với 29 hãng xe (trong báo cáo em chỉ đưa ra vài hình minh họa vì hơi nhiều ạ):


CHEVROLET
Number of cars
Year
0
1
2
3
4
5
6
70
72
74
76
78
80
82


BUICK
Number of cars
Year
0
1
2
3
4
5
6
70
72
74
76
78
80
82



⇨ Xu hướng thú vị: Có thể thấy hầu hết các hãng xe xuất hiện sớm (tầm khoảng năm 1970 – 1972) thì số lượng xe trong những năm 70s có dao động nhẹ nhưng đến khoảng năm 1980 – 1982 thì giảm mạnh (có trường hợp = 0). Có thể lí giải điều này vì có lẽ những hãng xuất hiện sớm đã hết ý tưởng hoặc kinh doanh lỗ vốn vì vậy mà không sản xuất xe mới nữa. Có những trường hợp ngoại lệ là những hãng xe nổi tiếng, đến tận bây giờ vẫn ở đỉnh cao như: BMW, Toyota, Ford, Honda, Chevrolet, Audi,...

2.1.9. Tính toán mối tương quan theo từng cặp thuộc tính và vẽ heatmap bằng Matplotlib. Bạn có thấy một số tương quan thú vị?

✓ Để tính mối tương quan giữa các thuộc tính thì các thuộc tính đó phải có dạng số. Vì vậy sử dụng câu lệnh **data.iloc[:,0:8].corr()** để chọn ra các thuộc tính có dạng số và tính toán mối tương quan giữa chúng. Độ tương quan của từng cặp thuộc tính được thể hiện ở hình dưới đây:

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

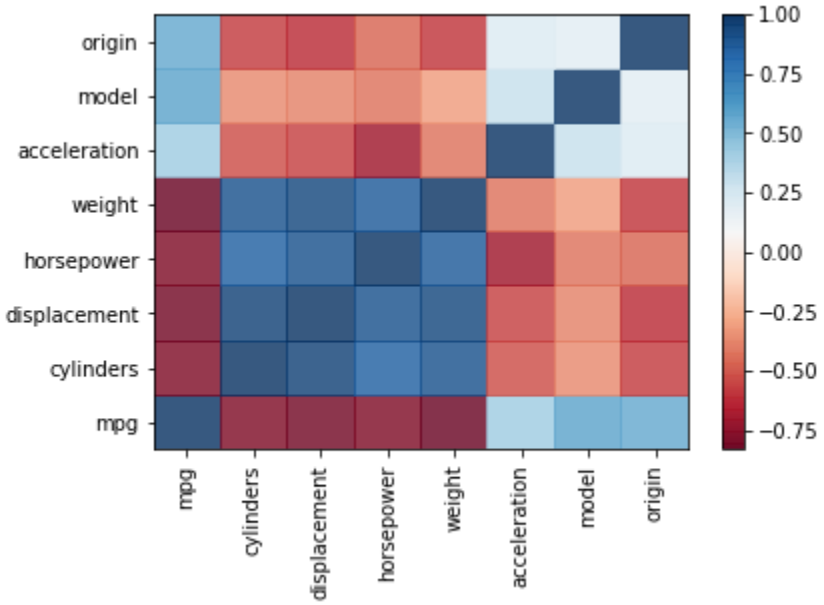
	mpg	cylinders	displacement	horsepower	weight	acceleration	model	origin
mpg	1.000000	-0.775396	-0.804203	-0.778427	-0.831741	0.420289	0.579267	0.563450
cylinders	-0.775396	1.000000	0.951787	0.844158	0.895220	-0.522452	-0.360762	-0.567478
displacement	-0.804203	0.951787	1.000000	0.898326	0.932475	-0.557984	-0.381714	-0.613056
horsepower	-0.778427	0.844158	0.898326	1.000000	0.866586	-0.697124	-0.424419	-0.460033
weight	-0.831741	0.895220	0.932475	0.866586	1.000000	-0.430086	-0.315389	-0.584109
acceleration	0.420289	-0.522452	-0.557984	-0.697124	-0.430086	1.000000	0.301992	0.218845
model	0.579267	-0.360762	-0.381714	-0.424419	-0.315389	0.301992	1.000000	0.187656
origin	0.563450	-0.567478	-0.613056	-0.460033	-0.584109	0.218845	0.187656	1.000000

✓ Sử dụng câu lệnh **heatmap = ax.pcolor(data.iloc[:,0:8].corr(), cmap='RdBu', alpha=0.8)** để vẽ heatmap bằng Matplotlib. Để biểu đồ chi tiết và rõ ràng hơn thì sử dụng thêm các câu lệnh như:

- ax.set_xticks(np.arange(8) + 0.5, minor=False): để chỉnh các lables ở trục x về giữa các ô thay vì ở bên mép trái.
- ax.set_yticks(np.arange(8) + 0.5, minor=False): để chỉnh các lables ở trục y về giữa các ô thay vì ở mép dưới.
- lables = data.columns[0:8] và ax.set_xticklabels(lables, minor=False, rotation = 90): thêm các lables cũng chính là tên từng thuộc tính vào trục x, sau đó xoay 90 độ để dễ nhìn hơn.
- ax.set_yticklabels(lables, minor=False): thêm các lables cũng chính là tên từng thuộc tính vào trục y.
- plt.colorbar(heatmap): thêm thanh hiển thị độ tương quan ứng với từng mức màu của heatmap.

```
fig, ax = plt.subplots()
heatmap = ax.pcolor(data.iloc[:,0:8].corr(), cmap='RdBu', alpha=0.8)
lables = data.columns[0:8]
ax.set_xticks(np.arange(8) + 0.5, minor=False)
ax.set_yticks(np.arange(8) + 0.5, minor=False)
ax.set_xticklabels(lables, minor=False, rotation = 90)
ax.set_yticklabels(lables, minor=False)
plt.colorbar(heatmap)
plt.show()
```

⇨ Như vậy, ta được heatmap như hình dưới đây:

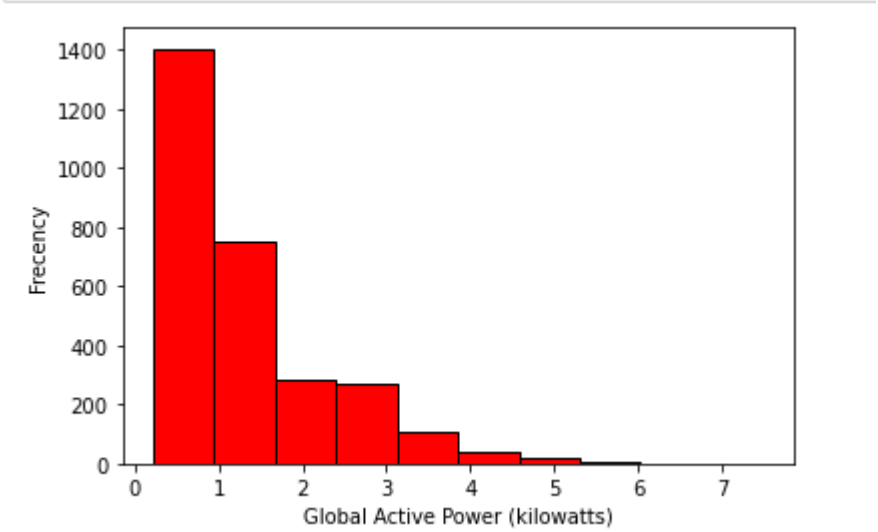


- ✓ Một số tương quan thú vị:
  - Nhìn chung thì các thuộc tính đều có mối tương quan khá mạnh với nhau (dù nghịch hay thuận). Chỉ có mối tương quan theo từng cặp giữa 3 thuộc tính **acceleraion, model, origin** là khá yếu.
  - Ta thấy **cylinders** và **displacement** có mối tương quan mạnh nhất và là tương quan thuận (0.951787). Điều này cũng dễ hiểu, vì thực tế displacement chính là tổng dung tích của tất cả xi lanh, vì vậy nhiều xi lanh thì tổng dung tích cũng nhiều hơn.

2.2. Lab3_electricpower

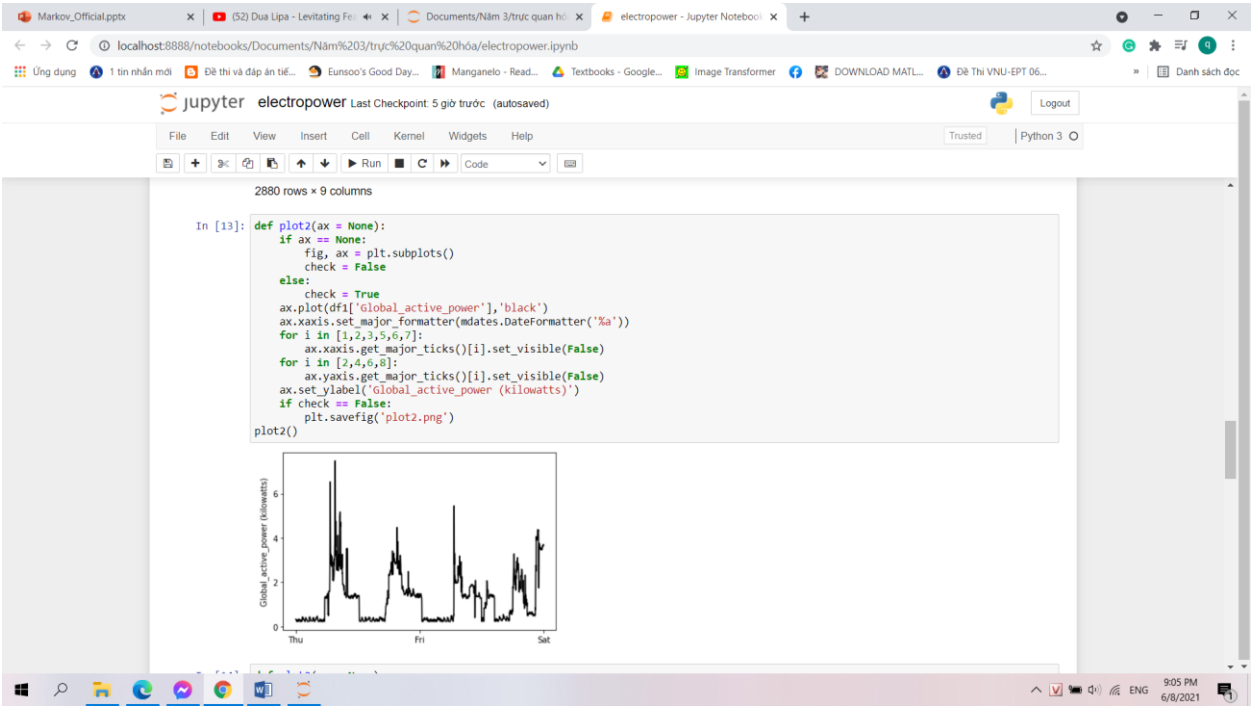
a. Xây dựng plot và lưu nó thành tập PNG

- ✓ Plot1:



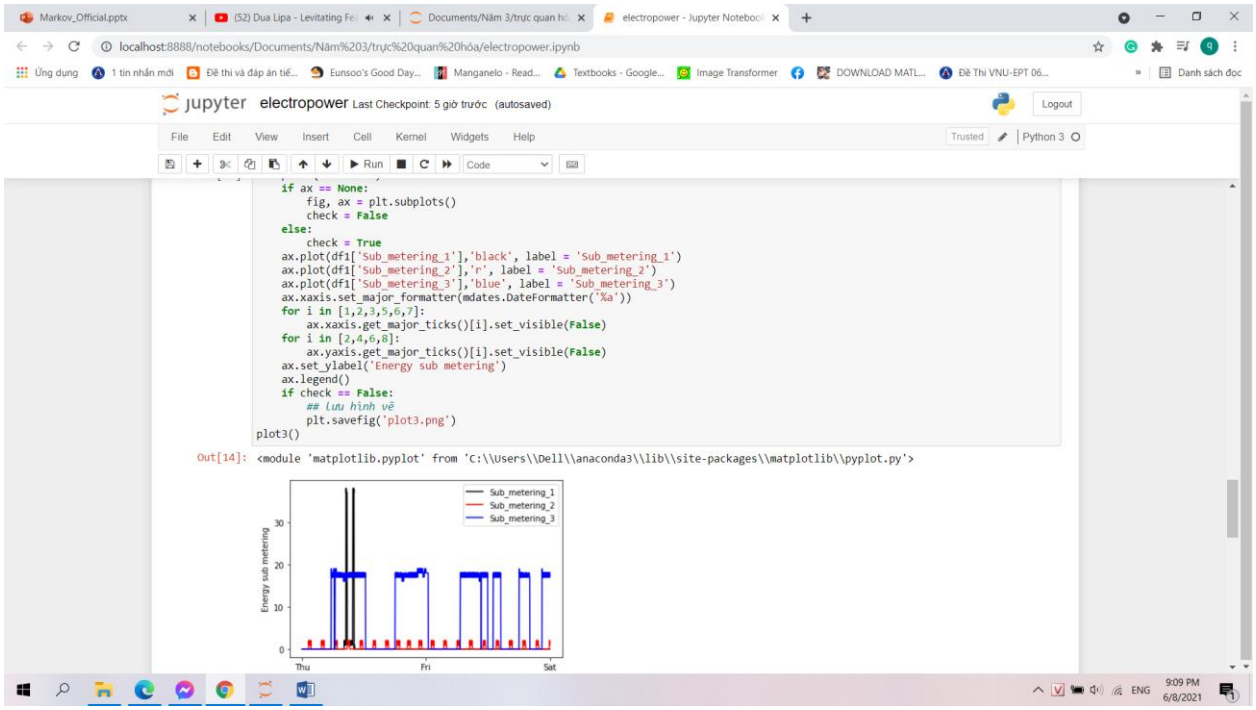
- ax.hist(df['Global_active_power'],color = 'red',edgecolor='black') : vẽ biểu đồ có trục x thể hiện global active power , trục y là tần số tương ứng với mỗi giá trị y.
- ax.set_xlabel("Global Active Power (kilowatts)") : gán nhãn trục x - biểu diễn Global Active Power (kilowatts).
- ax.set_ylabel("Frecency") : gán nhãn trục y - biểu diễn tần số .
- plt.savefig("plot1.png") : lưu biểu đồ.

✓ Plot 2:



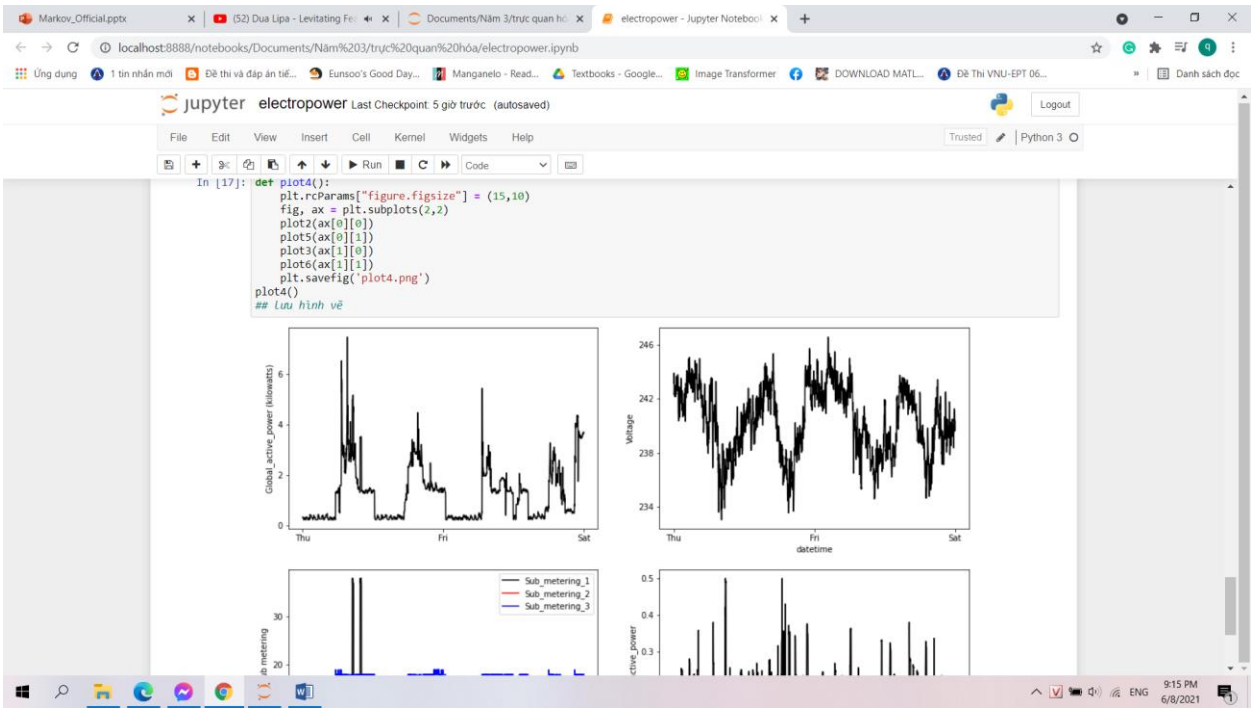
- ax.plot(df1['Global_active_power'],'black') : vẽ biểu đồ đường có trục y biểu diễn global active power, trục x là giờ trong hai ngày (2-1-2007 và 2-2-207).
- x.xaxis.set_major_formatter(mdates.DateFormatter('%a')): thay đổi các giá trị trục y, từ ngày và giờ sang dạng ngày trong tuần.
- ax.xaxis.get_major_ticks()[i].set_visible(False) : ẩn giá trị không mong muốn trên trục y
- ax.yaxis.get_major_ticks()[i].set_visible(False) : ẩn giá trị không mong muốn trên trục x..
- ax.set_ylabel('Global_active_power (kilowatts)') : hiển thị trục y đang đại diện cho đại lượng nào.
- plt.savefig('plot2.png') : lưu hình dưới dạng .png.

✓ Plot 3:



- ax.plot(df1['Sub_metering_1'],'black', label = 'Sub_metering_1'): vẽ biểu đồ đường có trục y biểu diễn 'Sub_metering_1', trục x là giờ trong hai ngày (2-1-2007 và 2-2-207)
- ax.plot(df1['Sub_metering_2'],'r', label = 'Sub_metering_2'): vẽ biểu đồ đường có trục y biểu diễn 'Sub_metering_1', trục x là giờ trong hai ngày (2-1-2007 và 2-2-207)
- ax.plot(df1['Sub_metering_3'],'blue', label = 'Sub_metering_3'): vẽ biểu đồ đường có trục y biểu diễn 'Sub_metering_1', trục x là giờ trong hai ngày (2-1-2007 và 2-2-207)
- ax.xaxis.set_major_formatter(mdates.DateFormatter('%a')): thay đổi các giá trị trục y, từ ngày và giờ sang dạng ngày trong tuần
- ax.xaxis.get_major_ticks()[i].set_visible(False) : ẩn giá trị không mong muốn trên trục y
- ax.yaxis.get_major_ticks()[i].set_visible(False) : ẩn giá trị không mong muốn trên trục x
- ax.set_ylabel('Energy sub metering') : hiển thị trục y đang đại diện cho đại lượng nào”
- ax.legend() : vẽ chú thích
- plt.savefig('plot3.png'): lưu hình vẽ dưới dạng .png

✓ Plot4 :



- plt.rcParams["figure.figsize"] = (15,10) : thay đổi kích thước của khung chứa các hình.
- fig, ax = plt.subplots(2,2) : tạo ra nhiều subplot.
- plot2(ax[0][0]), plot5(ax[0][1]), plot3(ax[1][0]), plot6(ax[1][1]): Hàm vẽ các biểu đồ .
- plt.savefig('plot4.png') : hàm lưu hình gồm 4 biểu đồ trên.

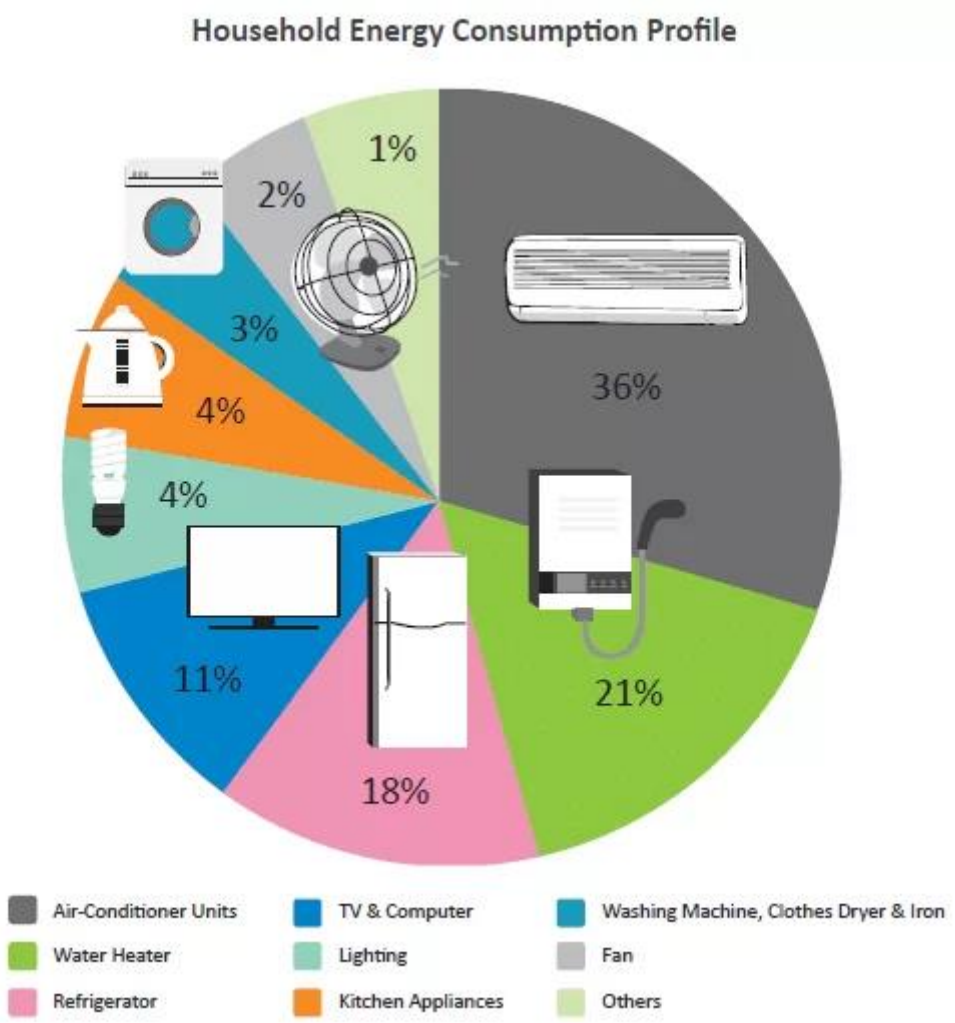
b. Story

Với sự phát triển của công nghệ đặc biệt là sự gia tăng của đồng hồ đo điện thông minh và việc áp dụng rộng rãi công nghệ phát điện như tấm pin mặt trời, có rất nhiều dữ liệu về việc sử dụng điện được thu thập và lưu trữ có sẵn. Các dữ liệu này đại diện cho một chuỗi thời gian đa biến của các biến liên quan đến điện năng, do đó có thể được sử dụng để lập mô hình và thậm chí dự báo mức tiêu thụ điện trong tương lai.

Trong lab này, chúng ta sẽ khám phá tập dữ liệu tiêu thụ điện năng trong gia đình, hiểu rõ hơn về dữ liệu thô bằng cách sử dụng phân tích và trực quan. Vì chỉ sử dụng dữ liệu trong 2 ngày (1/2/2007 và 2/2/2007) nên chúng ta không thể đưa ra các dự đoán về xu hướng sử dụng điện năng của hộ gia đình theo tuần, tháng, năm hay các mùa trong năm. Tuy nhiên sự tập trung nhận xét trực quan dữ liệu của 2 ngày ngẫu nhiên bất kỳ này cho ta cái nhìn cụ thể, rõ nét về mức độ sử dụng điện của hộ gia đình trong ngày và cụ thể hơn là trong các mốc thời gian trong ngày.

Ở *Plot 1* trực quan dữ liệu công suất tiêu thụ trung bình (Global active power) theo mức độ thường xuyên sử dụng. Nhận xét thấy mối quan hệ giữa công suất tiêu thụ và mức độ sử dụng là tương quan nghịch. Cụ thể, công suất tiêu thụ thấp (0 - 2 kilowatts) được sử dụng thường xuyên nhất, kế đến là công suất tiêu thụ trung bình (2 - 4 kilowatts) và công suất tiêu thụ cao

(4 - 6 kilowatts). Điều này đúng với thực tế các thiết bị được sử dụng thường xuyên như thiết bị chiếu sáng, thiết bị sạc, máy giặt, quạt điện,...thường có công suất tiêu thụ tương đối thấp. Tương tự với các thiết bị có công suất tiêu thụ trung bình như tivi, tủ lạnh,... và các thiết bị có công suất tiêu thụ cao như điều hòa, hệ thống sưởi, máy nước nóng,...



Nếu có thể duy trì tương quan thuận này thì hóa đơn tiền điện mỗi tháng của gia đình sẽ không có sự thay đổi đáng kể. Cũng chính vì thế, sẽ dễ dàng truy vết được nguyên nhân đột nhiên hóa đơn điện cuối tháng tăng bất thường, chú ý mức độ thường xuyên sử dụng các thiết bị có công suất cao và trung bình trong nhà, đặc biệt trong thời kỳ thực hiện giãn cách xã hội, thời gian ở nhà ắt hẳn sẽ tăng và nhu cầu đối với các thiết bị có công suất cao là điều rất dễ phát hiện.

Ở *Plot 2* trực quan dữ liệu công suất tiêu thụ trung bình (Global active power) theo thời gian. Nhìn chung, trực quan của công suất tiêu thụ chia một ngày thành các mốc thời gian tương đối cố định và có sự tương đồng nhất định ở 2 ngày. Công suất tiêu thụ của các bắt đầu từ sáng sớm (khoảng 6-7 giờ sáng), mức tiêu thụ giảm vào giữa ngày, điều này có thể có ý nghĩa nếu hầu hết các thành viên trong gia đình đều ra khỏi nhà. Điều này cũng giải thích cho việc công suất tiêu thụ tăng vào cuối ngày. Còn một vài nguyên nhân có thể tác động đến sự tăng

vào thời điểm cuối ngày ví dụ như các quốc gia có nhiệt độ thấp về đêm dẫn đến việc sử dụng hệ thống sưởi làm tăng mức tiêu thụ qua đêm mạnh mẽ hay nhu cầu sử dụng các thiết bị nhà bếp có công suất cao vào các ngày lễ,... Đồng thời có thể xem xét đến sự khác nhau giữa hai ngày, nếu giải thích theo hướng thời gian có người ở nhà thì sự khác nhau giữa hai ngày là dễ hiểu đối với một ngày trong tuần và cuối tuần.

Ở *Plot 3* trực quan dữ liệu đo sáng phụ năng lượng số 1, 2, 3 (Sub metering 1, Sub metering 2, Sub metering 3), trực quan này phân tích chi tiết và tách biệt công suất tiêu thụ của các thiết bị cụ thể, từ đó giúp dễ dàng nhận thấy nguyên nhân làm gia tăng đột ngột tổng công suất tiêu thụ trong ngày. Cụ thể, ở ngày 1/2/2007 tổng công suất tiêu thụ cao vượt bậc là do tác động trực tiếp dữ liệu Sub metering 1, có thể giải thích buổi sáng này các thiết bị như máy rửa bát, lò nướng và lò vi sóng có thể đã được sử dụng. Điều này hợp lý với dữ liệu Sub metering 2, Sub metering 3 tương ứng với các thiết bị như máy giặt, máy sấy quần áo, tủ lạnh, đèn chiếu sáng, máy nước nóng, máy lạnh,... được sử dụng thường xuyên và đều đặn.

Plot 4 tổng kết từ các plot trước đó và cụ thể hóa các mốc thời gian cụ thể (datetime) cho thấy rõ sự tương đồng của 4 plot về đỉnh của các đồ thị tương ứng với các mốc thời gian có nhu cầu sử dụng các thiết bị có công suất cao trong ngày. Kịch bản chính dễ thấy nhất theo mốc thời gian tuyến tính này đó là dựa theo các giả thuyết về thời gian trong ngày, để tính khả năng mọi người có ở nhà hay không, xem xét là ngày trong tuần hay cuối tuần, có phải là ngày lễ ở địa điểm lấy dữ liệu hay không,... Những yếu tố này có thể ít ảnh hưởng đáng kể đối với dự báo dữ liệu hàng tháng và có lẽ ở một mức độ nào đó đối với dữ liệu hàng tuần, tuy nhiên xem xét các yếu tố con người đơn giản trên có thể hữu ích trong các tính năng kỹ thuật từ dữ liệu, do đó có thể giúp dự báo chu kỳ hàng ngày cụ thể dễ dàng hơn.

III. TÀI LIỆU THAM KHẢO

Documents của các thư viện ‘matplotlib’, ‘pandas’.
https://www.motorbiscuit.com/three-cylinder-engine-back-and-here-to-stay/
https://www.youtube.com/watch?v=nSy_N7JEp7o&t=166s
https://torontopubliclibrary.typepad.com/business_personal_finance/2015/03/automobile-industry.html
https://bachkhoahanoi.edu.vn/tin-tuc/su-ra-doi-va-phat-trien-cua-nganh-cong-nghe-o-to-tren-the-gioi/
https://sites.google.com/site/carssworld914/qua-trinh-phat-trien
https://chandat.net/kien-thuc/meo-vat/cong-suat-cua-cac-thiet-bi-dien-gia-dung-trong-nha/
https://chandat.net/testx/wp-content/uploads/2018/11/Dien-nang-tieu-thu-cua-cac-thiet-bi-dien-gia-dung-0.png