TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN TRỰC QUAN HÓA DỮ LIỆU

-----📖-----

BÁO CÁO LAB 3

Sinh viên: 18120505 Đào Quốc Phong

18120528 Nguyễn Như Quang

18120525 Đoàn Thanh Quang

MỤC LỤC

1. Thông tin nhóm.

MSSV	Họ và tên	Email
18120505	Đào Quốc Phong	18120505@student.hcmus.edu.vn
18120528	Nguyễn Như Quang	18120528@student.hcmus.edu.vn
18120525	Đoàn Thanh Quang	18120525@student.hcmus.edu.vn

2. Mức độ hoàn thành.

2.1. Công việc và mức độ hoàn thành

STT	Chi tiết yêu cầu	Mức độ hoàn thành
1	Thu thập và tiền xử lý dữ liệu.	100%
2	Chọn lựa, giải thích, trực quan các trường và các mối quan hệ giữa chúng.	100%
3	Rút ra ý nghĩa hợp lý sau mỗi dữ liệu được trực quan.	100%
4	Xem xét trên nhiều quan hệ, nhiều góc nhìn khác nhau.	100%
5	Mô tả thư viện liên quan đến chức năng trực quan hóa dữ liệu	100%
6	Báo cáo	100%

2.2. Phân chia công việc

MSSV	Họ và tên	Công việc
18120505	Đào Quốc Phong	Thu thập và tiền xử lý dữ liệu. Làm Electric power consumption data. Rút ra ý nghĩa hợp lý sau mỗi dữ liệu được trực quan. Báo cáo.
18120528	Nguyễn Như Quang	Làm Exploratory analysis of Car MPG data. Mô tả thư viện liên quan đến chức năng trực quan hóa dữ liệu. Rút ra ý nghĩa hợp lý sau mỗi dữ liệu được trực quan. Báo cáo.
18120525	Đoàn Thanh Quang	Làm Exploratory analysis of Car MPG data. Mô tả thư viện liên quan đến chức năng trực quan hóa dữ liệu. Rút ra ý nghĩa hợp lý sau mỗi dữ liệu được trực quan. Báo cáo.

3. Báo cáo

3.1. Tìm hiểu về Numpy

Khi xử lý dữ liệu, chúng ta thường cần một cách làm việc với mảng nhiều chiều và phải áp dụng một số phép toán thống kê và toán học cơ bản trên dữ liệu đó.Numpy hỗ trợ cho các mảng lớn, nhiều chiều và tích hợp cho nhiều phép toán thống kê và toán học cấp cao. Nổi bật nhất của Numpy là Ndarrays, nó nhanh hơn list của Python, dữ liệu đồng nhất giúp giảm lãng phí bộ nhớ và thời gian truy cập tốt hơn.

3.2. Tìm hiểu về Pandas

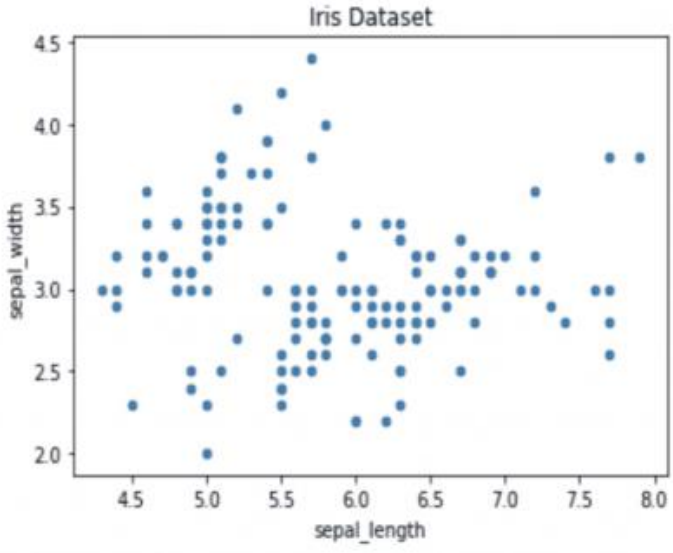
Pandas là một thư viện hiệu suất cao, dễ sử dụng, cung cấp các cấu trúc dữ liệu. Pandas Visualization dễ dàng tạo ra các plot từ một khung dữ liệu và chuỗi pandas. Nó cũng có API cấp cao hơn Matplotlib và do đó chúng ta cần ít code hơn cho cùng kết quả.

3.2.1. Scatter Plot

Để tạo một scatter plot trong Pandas, chúng ta gọi <dataset>.plot.scatter() và truyền cho nó hai đối số, tên của cột x cũng như tên của cột y.

```
1  iris.plot.scatter(x='sepal_length', y='sepal_width', title='Iris Dataset')
```

pandas_simple_scatterplot.py hosted with ❤ by GitHub view raw



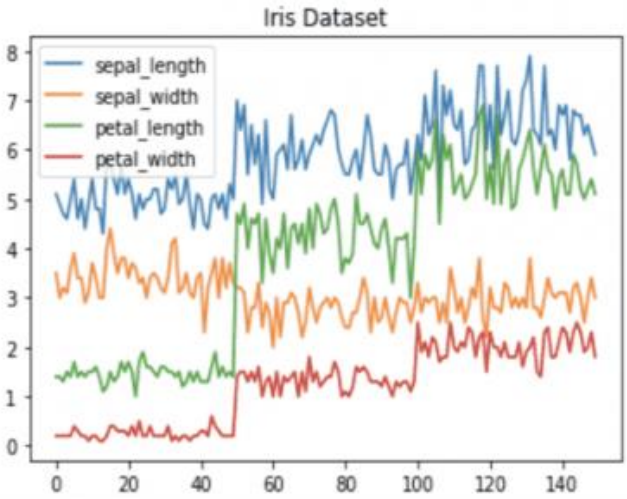
3.2.2. Line Chart

Để tạo một line chart trong Pandas, chúng ta gọi <dataframe>.plot.line(). Trong khi ở Matplotlib, chúng ta cần lặp lại từng cột mà chúng ta muốn vẽ, trong Pandas

chúng ta không cần phải làm điều này vì nó tự động vẽ tất cả các cột số có sẵn (ít nhất là nếu chúng tôi không chỉ định một cột cụ thể).

```
1  iris.drop(['class'], axis=1).plot.line(title='Iris Dataset')
```

pandas_simple_linechart.py hosted with ❤ by GitHub view raw

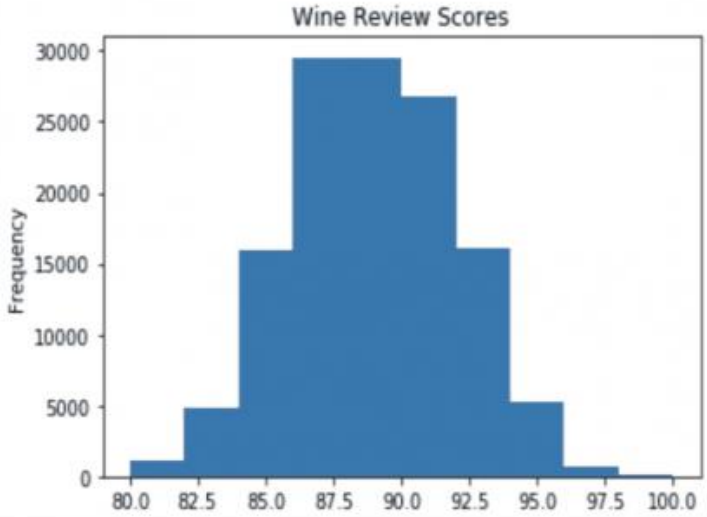


3.2.3. Histogram

Chúng ta tạo một Histogram với phương thức plot.hist. Không có bất kỳ đối số bắt buộc nào nhưng chúng ta có thể tùy ý chuyển một số như kích thước.

```
1  wine_reviews['points'].plot.hist()
```

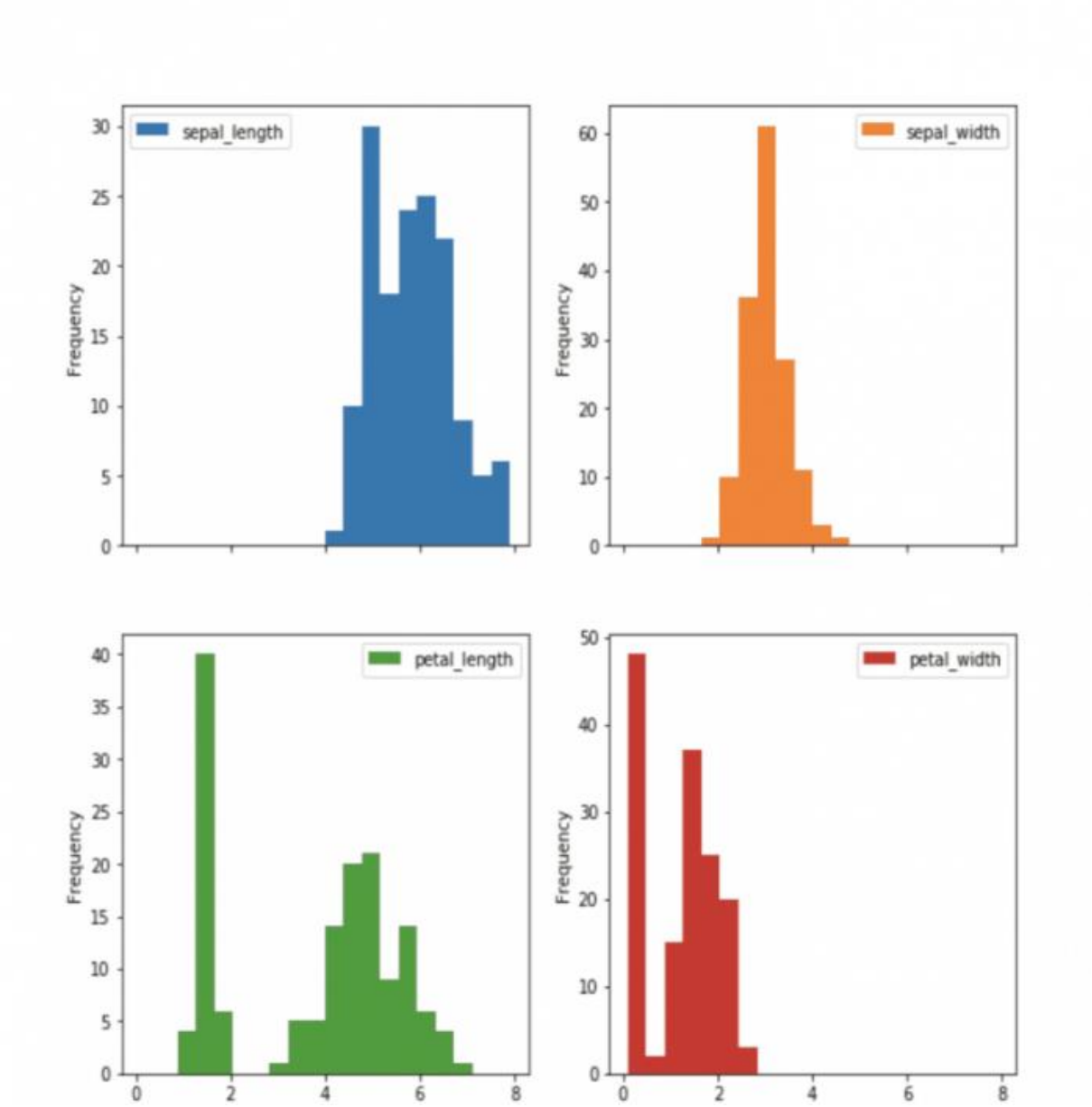
pandas_simple_histogram.py hosted with ❤ by GitHub view raw



Tạo nhiều histogram cũng rất dễ dàng

```
1  iris.plot.hist(subplots=True, layout=(2,2), figsize=(10, 10), bins=20)
```

pandas_multiple_histograms.py hosted with ❤ by GitHub view raw



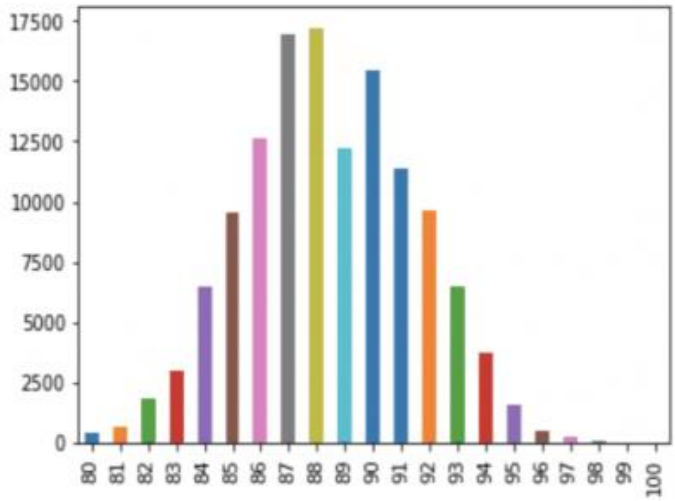
Đối số subplots chỉ định rằng chúng ta muốn có một plot riêng cho từng tính năng và layout chỉ định số lượng plot trên mỗi hàng và cột.

3.2.4. Bar chart

Để vẽ bar chart, chúng ta sử dụng phương thức plot.bar(). Trước khi chúng ta gọi nó, chúng ta cần lấy dữ liệu của mình. Trước tiên chúng ta sẽ đếm các lần xuất hiện bằng phương thức value_count() và sau đó sắp xếp các lần xuất hiện từ nhỏ nhất đến lớn nhất bằng phương thức sort_index().

```
1  wine_reviews['points'].value_counts().sort_index().plot.bar()
```

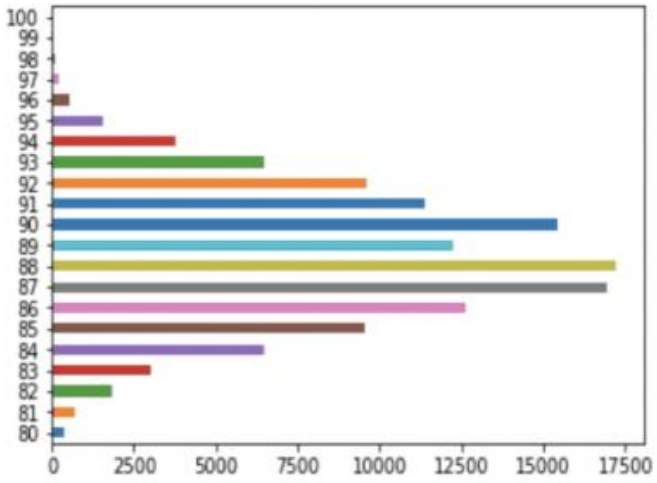
pandas_simple_barchart_vertical.py hosted with ❤ by GitHub view raw



Có thể tạo một biểu đồ thanh ngang rất đơn giản.

```
1  wine_reviews['points'].value_counts().sort_index().plot.barh()
```

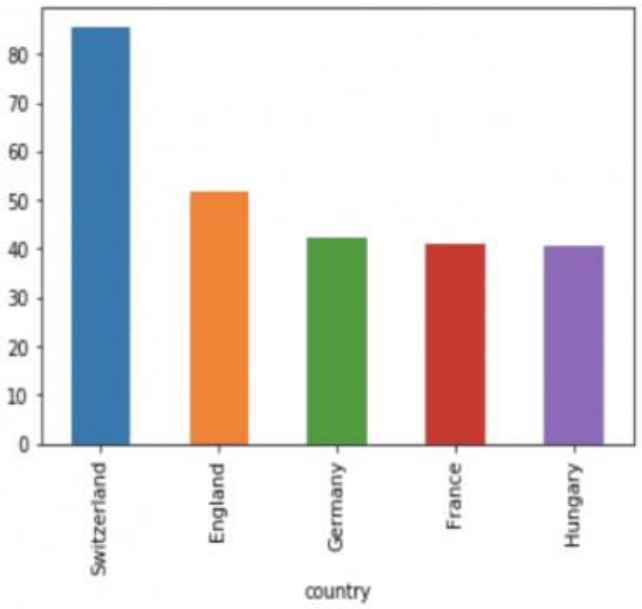
pandas_simple_barchart_horizontal.py hosted with ❤ by GitHub view raw



Ngoài ra còn có thể vẽ thêm nhiều dữ liệu khác vào.

```
1  wine_reviews.groupby("country").price.mean().sort_values(ascending=False)[:5].plot.ba
```

pandas_simple_barchart_vertical_2.py hosted with ❤ by GitHub view raw



3.3. Tìm hiểu về Matplotlib

Matplotlib là thư viện python phổ biến nhất.

Matplotlib đặc biệt tốt để tạo các biểu đồ cơ bản như line charts, bar charts, histograms và nhiều hơn nữa. Nó có thể được nhập bằng cách gõ:

import matplotlib.pyplot as plt

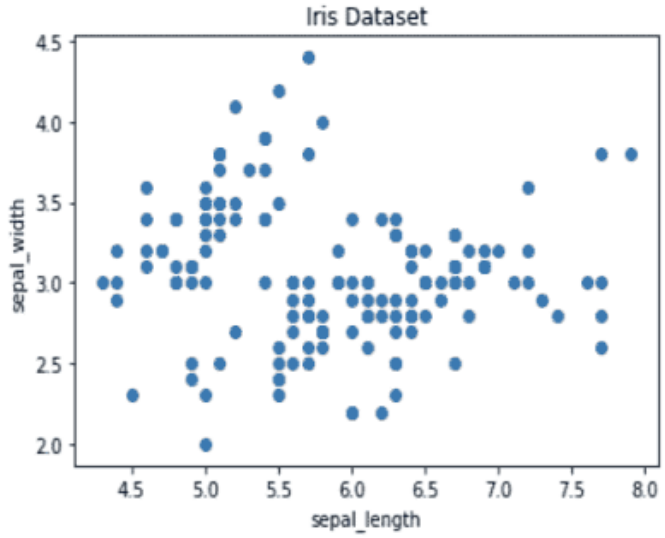
3.3.1. Scatter Plot

Chúng ta sử dụng phương thức scatter để tạo một scatter plot trong Matplotlib. Chúng ta cũng sẽ tạo một hình và một trục bằng cách sử dụng plt.subplots để cung cấp cho biểu đồtiêu đề và nhãn.

```python
# create a figure and axis
fig, ax = plt.subplots()

# scatter the sepal_length against the sepal_width
ax.scatter(iris['sepal_length'], iris['sepal_width'])
# set a title and labels
ax.set_title('Iris Dataset')
ax.set_xlabel('sepal_length')
ax.set_ylabel('sepal_width')
```

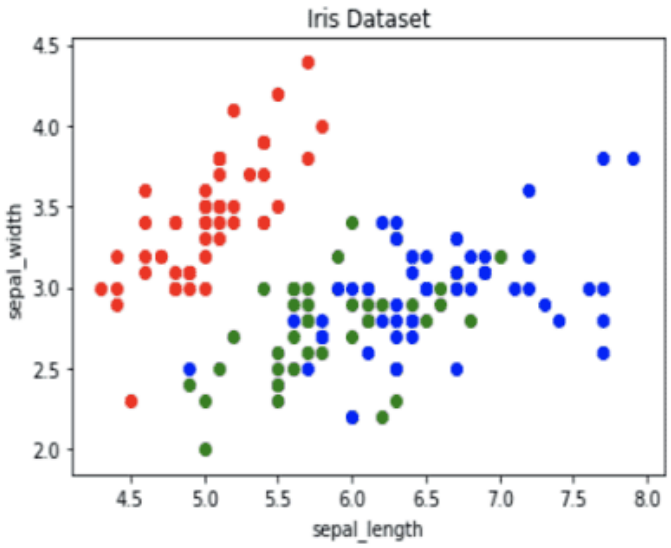
matplotlib_simple_scatterplot.py hosted with ❤ by GitHub view raw



Để biểu đồ nhiều ý nghĩa hơn chúng ta có thể tô màu trong mỗi điểm dữ liệu theo lớp của nó. Điều này có thể được thực hiện bằng cách tạo một từ điển ánh xạ từ lớp sang màu và sau đó tự phân tán từng điểm bằng cách sử dụng vòng lặp for và chuyển màu tương ứng.

```
# create color dictionary
colors = {'Iris-setosa':'r', 'Iris-versicolor':'g', 'Iris-virginica':'b'}
# create a figure and axis
fig, ax = plt.subplots()
# plot each data-point
for i in range(len(iris['sepal_length'])):
    ax.scatter(iris['sepal_length'][i], iris['sepal_width'][i],color=colors[iris['class'
# set a title and labels
ax.set_title('Iris Dataset')
ax.set_xlabel('sepal_length')
ax.set_ylabel('sepal_width')
```

matplotlib_simple_scatterplot_with_colors.py hosted with ❤ by GitHub view raw

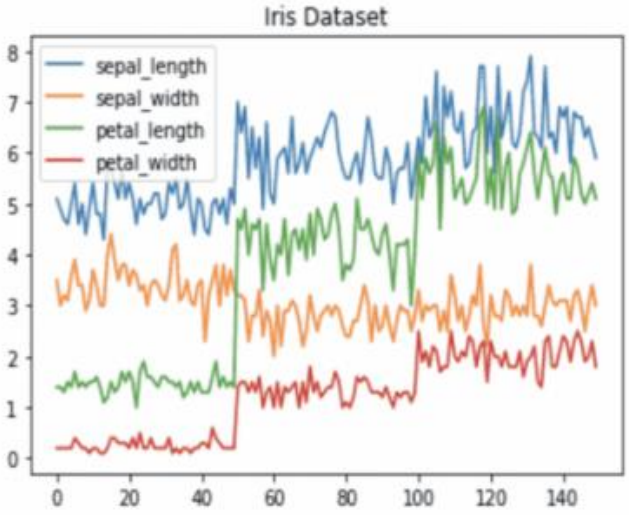


3.3.2. Line Chart

Chúng ta tạo một line chart bằng phương thức plot. Chúng ta cũng có thể vẽ nhiều cột trong một biểu đồ, bằng cách lặp qua các cột mà chúng ta muốn và vẽ từng cột trên cùng một trục.

```
# get columns to plot
columns = iris.columns.drop(['class'])
# create x data
x_data = range(0, iris.shape[0])
# create figure and axis
fig, ax = plt.subplots()
# plot each column
for column in columns:
    ax.plot(x_data, iris[column], label=column)
# set title and legend
ax.set_title('Iris Dataset')
ax.legend()
```

matplotlib_simple_linechart.py hosted with ❤ by GitHub view raw

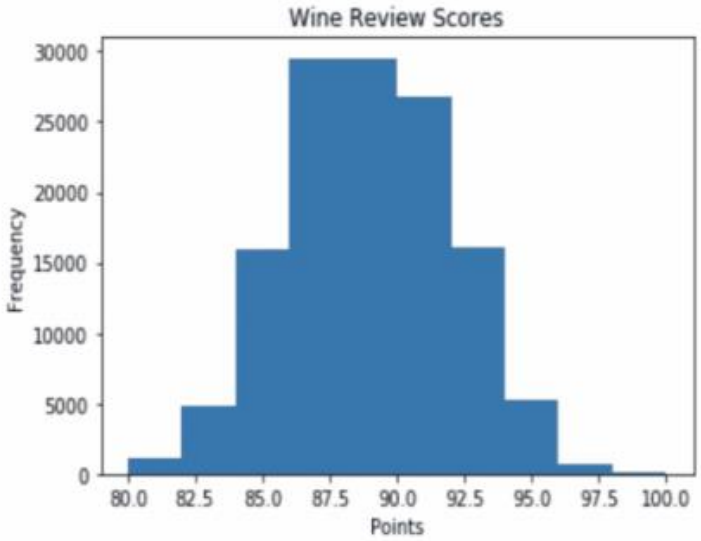


3.3.3. Histogram

Chúng ta tạo Histogram bằng phương thức hist. Nếu chúng ta chuyển dữ liệu phân loại như cột điểm từ bộ dữ liệu đánh giá, nó sẽ tự động tính toán tần suất mỗi lớp xảy ra.

```
# create figure and axis
fig, ax = plt.subplots()
# plot histogram
ax.hist(wine_reviews['points'])
# set title and labels
ax.set_title('Wine Review Scores')
ax.set_xlabel('Points')
ax.set_ylabel('Frequency')
```

matplotlib_simple_histogram.py hosted with ❤ by GitHub view raw

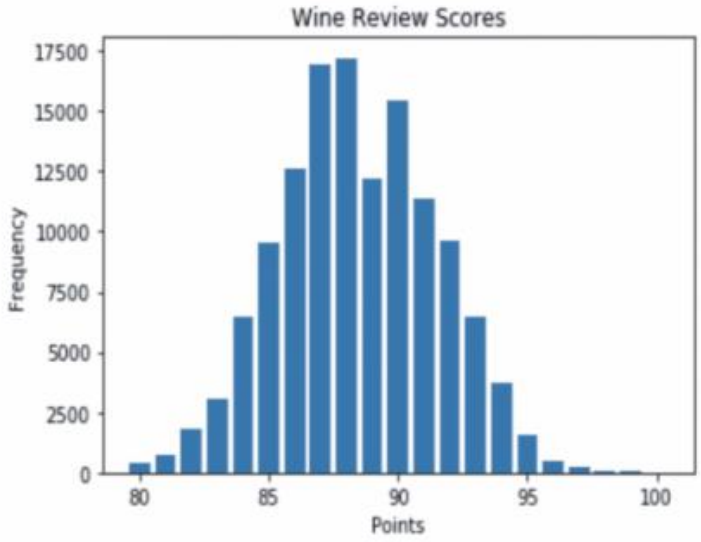


3.3.4. Bar chart

Một bar chart có thể được tạo bằng phương thức bar. Bar chart không tự động tính toán tần suất của danh mục nên chúng ta sẽ sử dụng chức năng value_counts pandas để thực hiện việc này.

```python
# create a figure and axis
fig, ax = plt.subplots()
# count the occurrence of each class
data = wine_reviews['points'].value_counts()
# get x and y data
points = data.index
frequency = data.values
# create bar chart
ax.bar(points, frequency)
# set title and labels
ax.set_title('Wine Review Scores')
ax.set_xlabel('Points')
ax.set_ylabel('Frequency')
```

matplotlib_simple_barchart.py hosted with ❤ by GitHub view raw



3.4. Trực quan hóa và nhận xét dữ liệu (Phân tích dữ liệu về Car MPG)

3.4.1. Số lượng xe hơi và thuộc tính

```
In [45]: data.shape
Out[45]: (406, 9)
```

3.4.2. Số công ty xe hơi đại diện cho tập thể dữ liệu

```
In [46]: data['brand'] = data.car_name.str.split().str[0]
         len(data['brand'].unique())
```

Tên xe có chỉ số MPG tốt nhất

```
In [47]: data.iloc[data.mpg.argmax()].car_name
Out[47]: 'mazda glc'
```

Hãng xe sản xuất nhiều xe 8 xi lanh nhất:

Out[48]:

	brand	cylinders
0	amc	9
1	buick	7
2	cadillac	2
3	chevrolet	19
4	chevy	2
5	chrysler	4
6	dodge	12
7	ford	22
8	hi	1
9	mercury	5
10	oldsmobile	7
11	plymouth	11
12	pontiac	7

```
In [49]: cyl8_brand.loc[cyl8_brand.cylinders.argmax()].brand
Out[49]: 'ford'
```

Tên của các xe loại 3 xi lanh:

```
In [50]: cyl3 = data[data.cylinders==3]
         cyl3.car_name
Out[50]: 78      mazda rx2 coupe
         118           maxda rx3
         250         mazda rx-4
         341       mazda rx-7 gs
         Name: car_name, dtype: object
```

Lịch sử và sự phổ biến của xe 3 xi lanh:

Nhiều người đã từng có những định kiến không tốt về động cơ 3 xi lanh cho xe hơi, nhưng với công nghệ hiện nay, và nhìn nhận lại, thì động cơ 3 xi lanh có ưu điểm riêng của nó, và hiện nay động cơ 3 xi lanh rất phổ biến, được lắp đặt trên rất nhiều xe hơi.

3.4.3. Phạm vi, giá trị trung bình và độ lệch chuẩn của từng thuộc tính

In [51]:
```
data.describe()
```

Out[51]:

	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

Các giá trị bị thiếu:

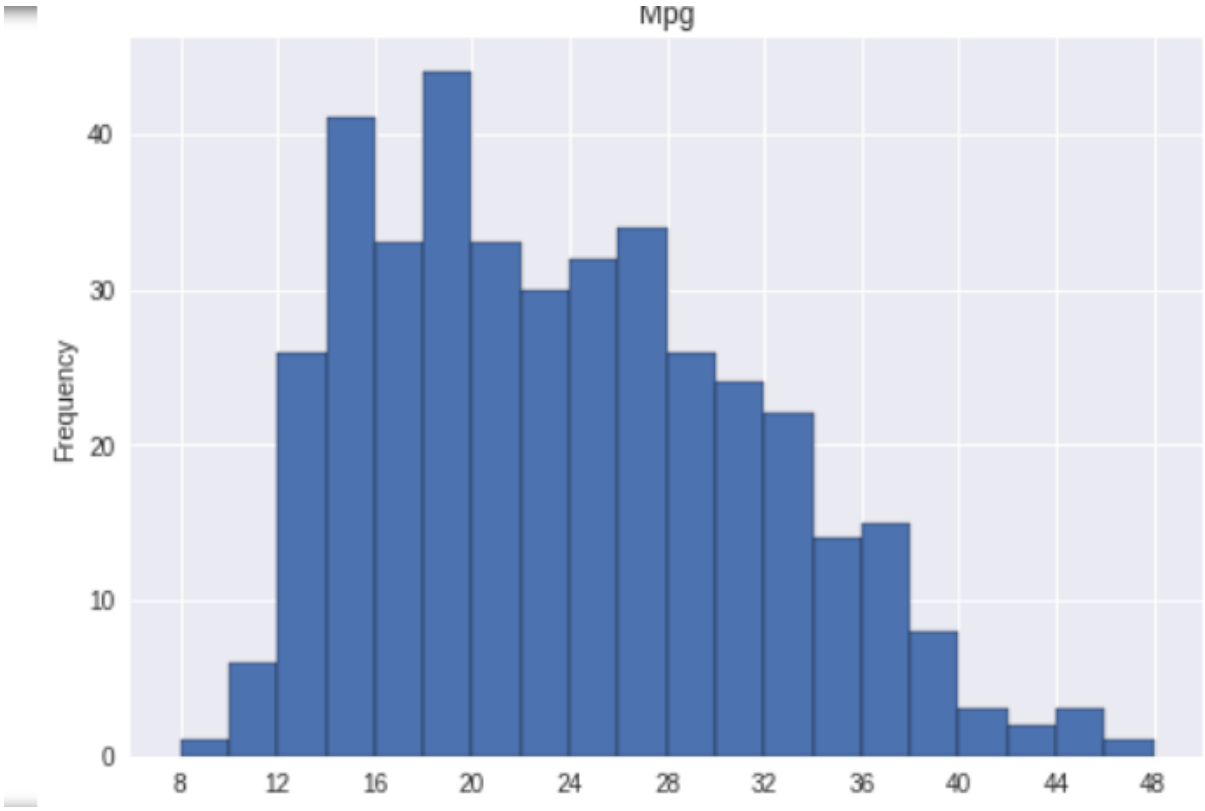
In [52]:
```
data.isna()
```

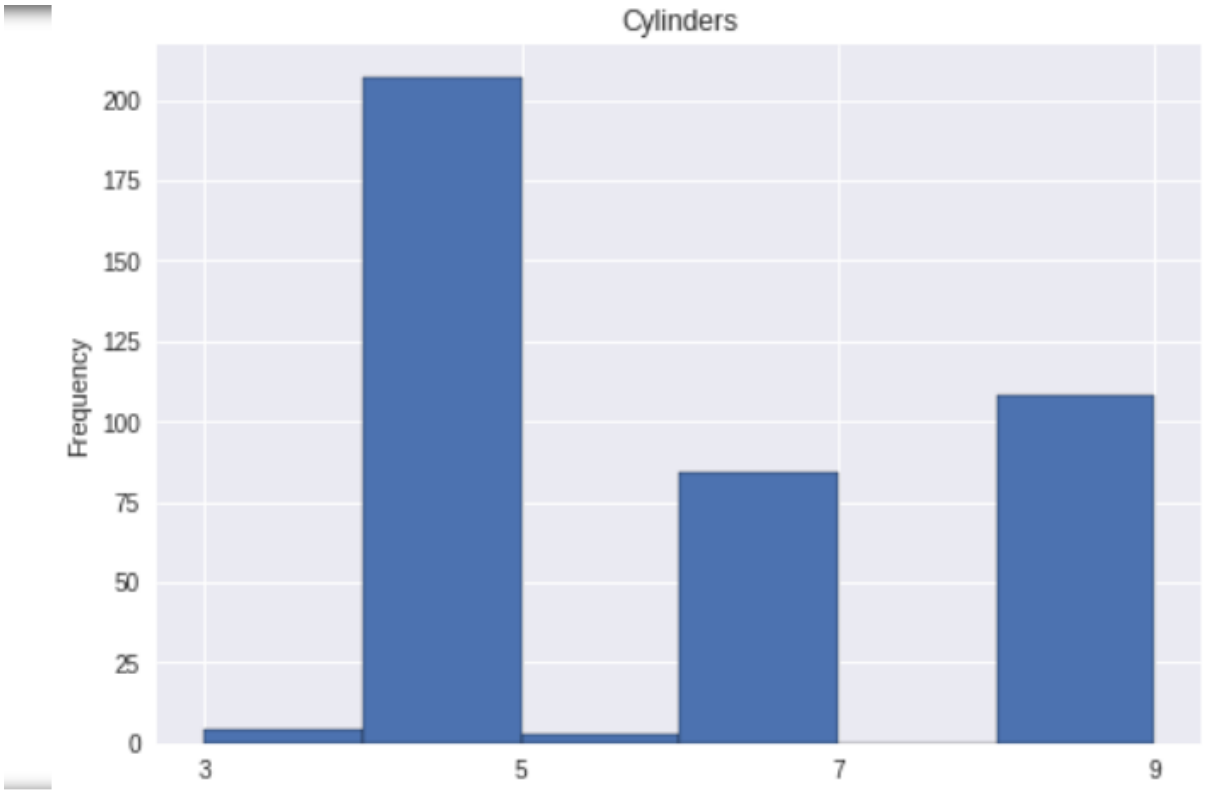
Out[52]:

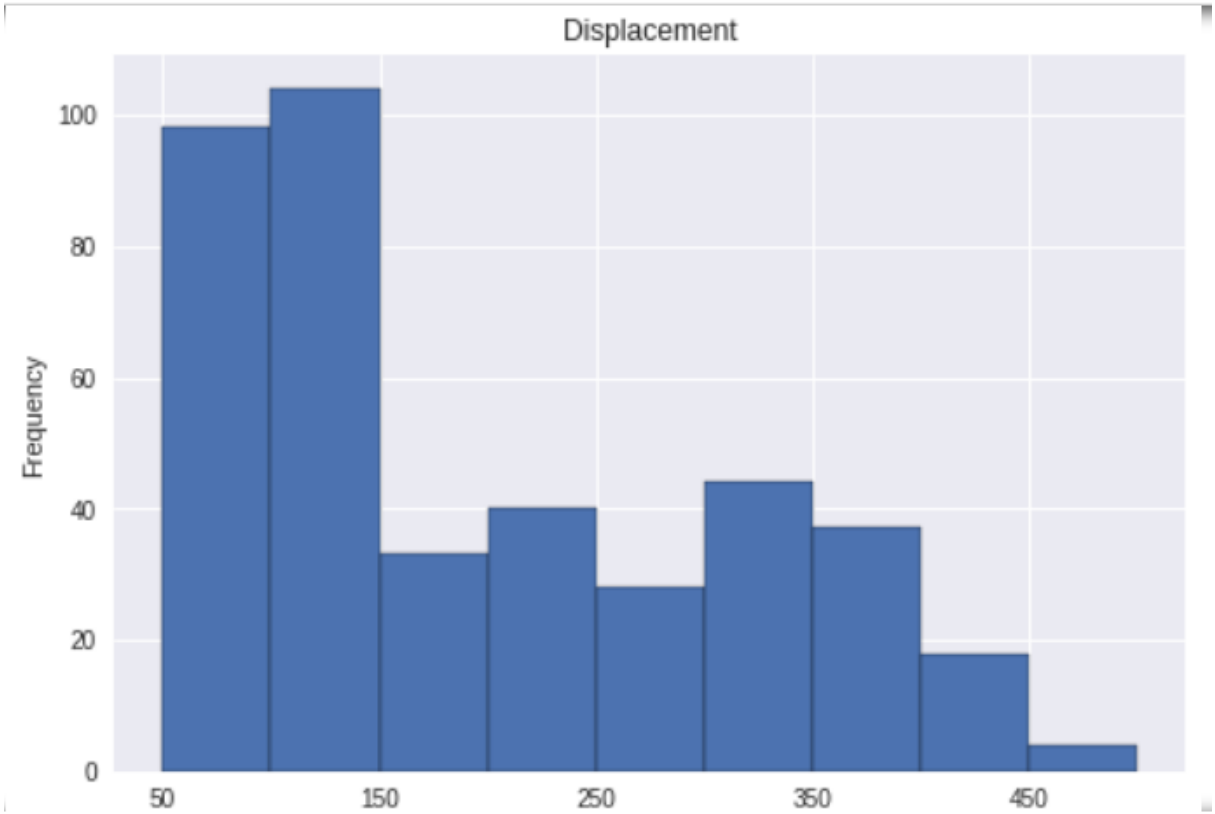
	mpg	cylinders	displacement	horsepower	weight	acceleration	model	origin	car_name	brand
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...
401	False	False	False	False	False	False	False	False	False	False
402	False	False	False	False	False	False	False	False	False	False
403	False	False	False	False	False	False	False	False	False	False
404	False	False	False	False	False	False	False	False	False	False
405	False	False	False	False	False	False	False	False	False	False

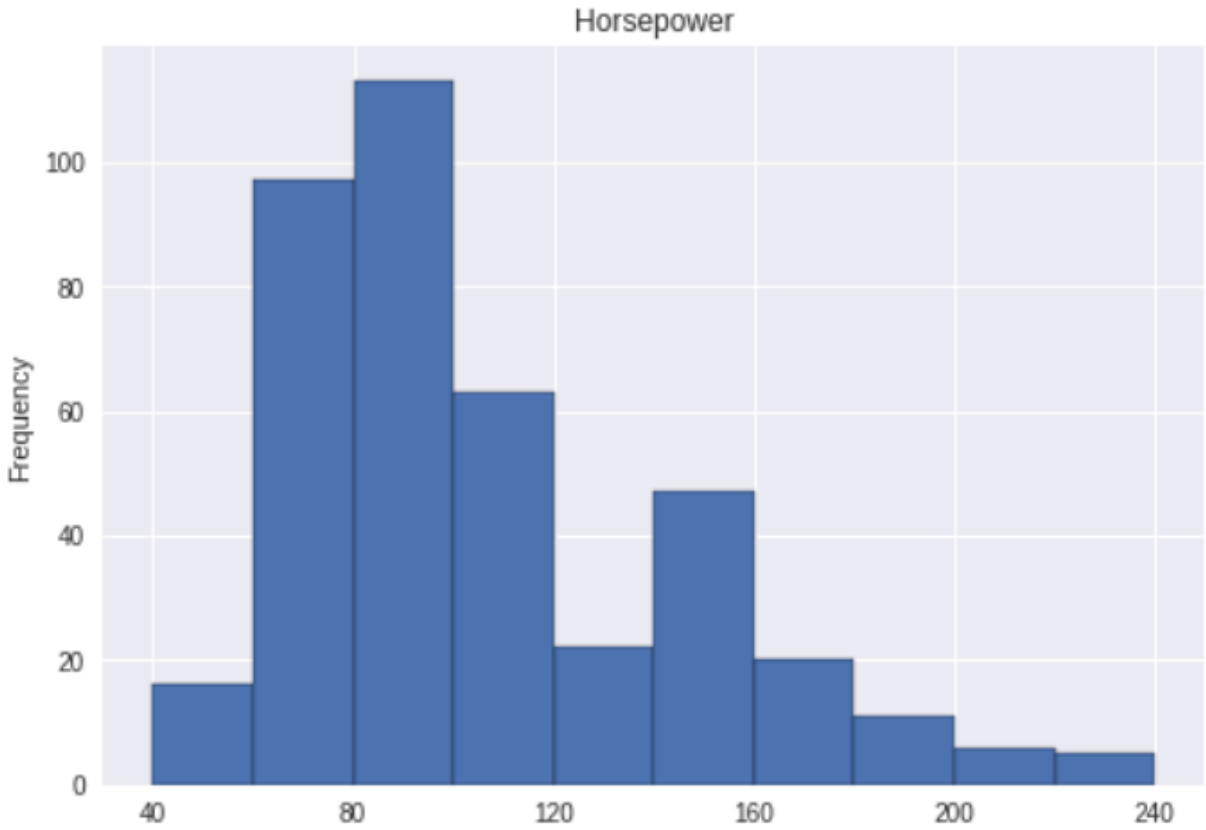
406 rows × 10 columns

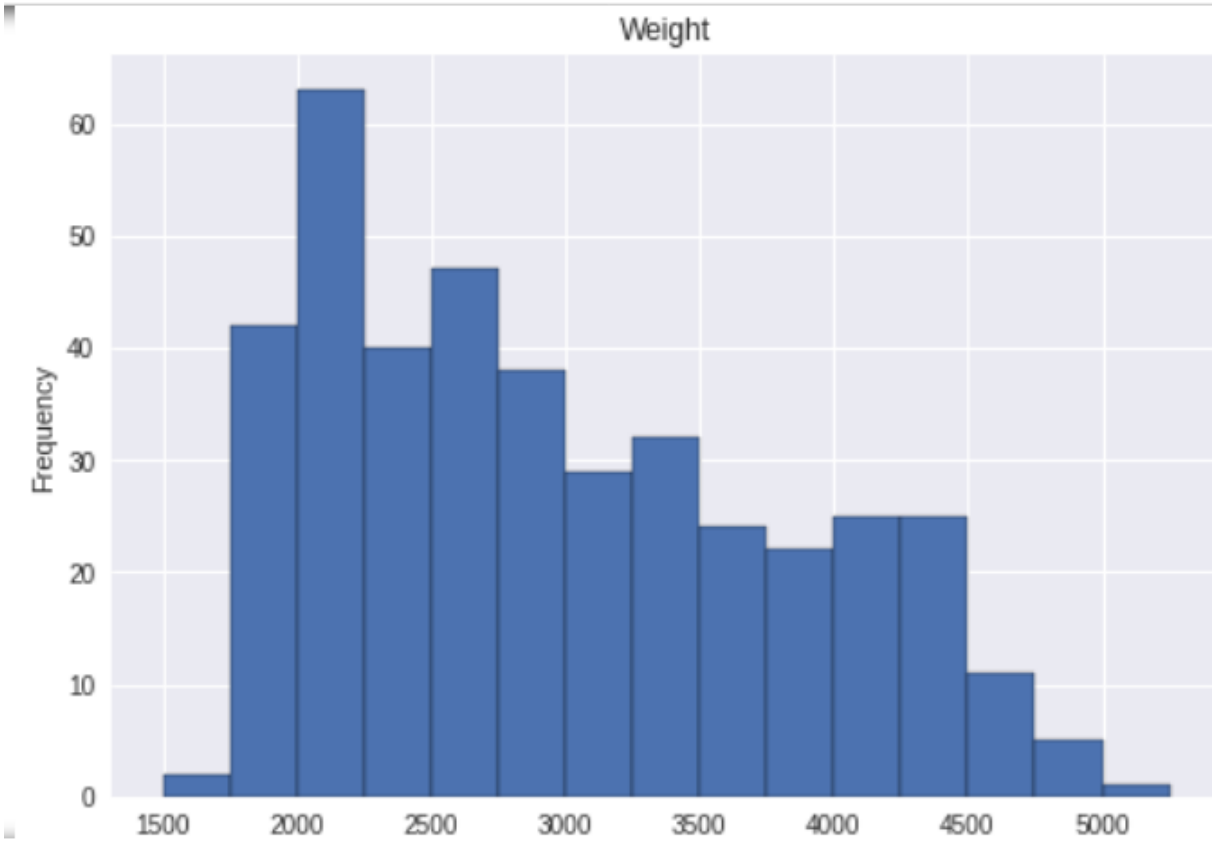
3.4.4. Biểu đồ cho các thuộc tính

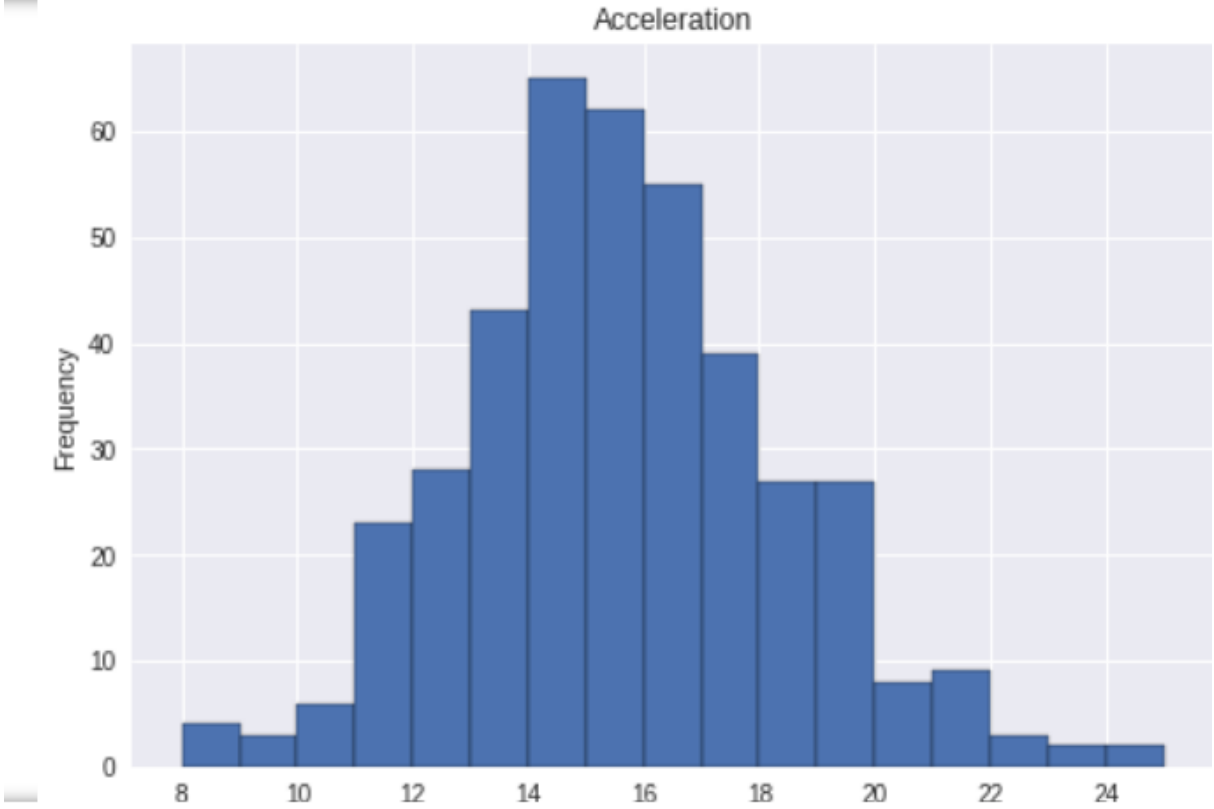


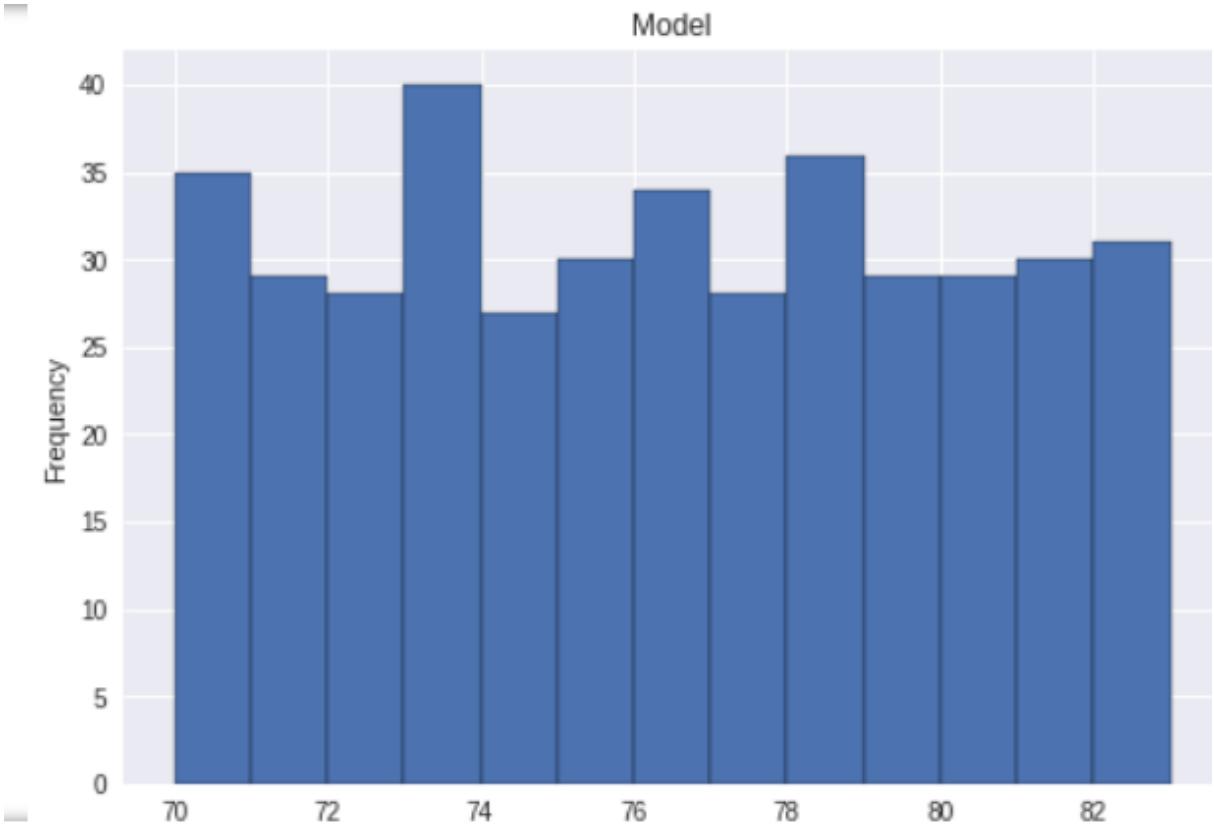


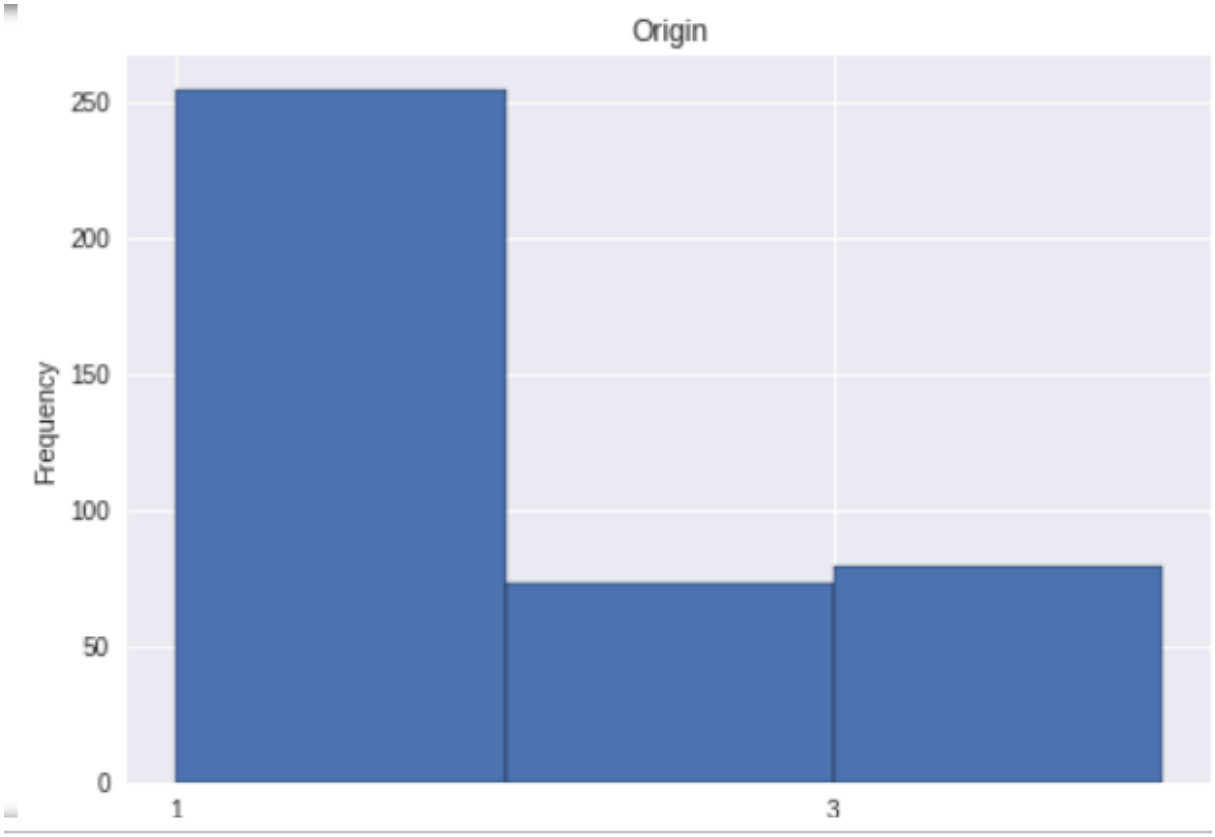

Displacement
Frequency
0
20
40
60
80
100
50
150
250
350
450


Horsepower
Frequency
0
20
40
60
80
100
40
80
120
160
200
240


Weight
Frequency
60
50
40
30
20
10
0
1500
2000
2500
3000
3500
4000
4500
5000

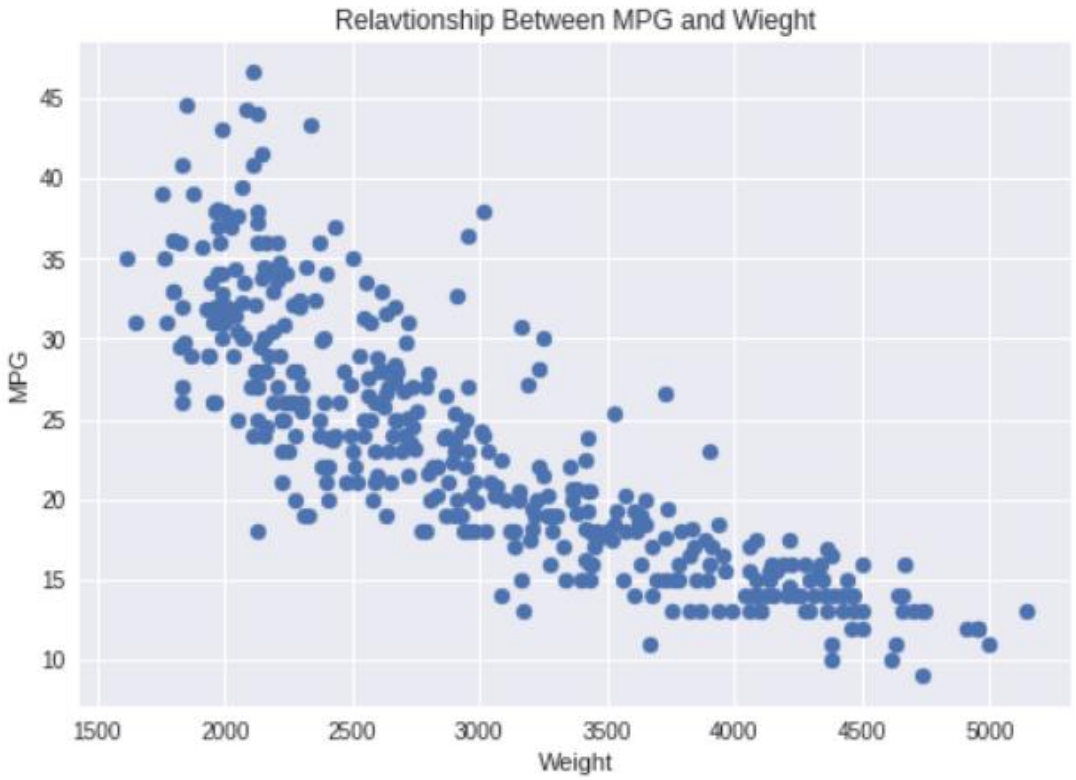

Acceleration
Frequency
60
50
40
30
20
10
0
8
10
12
14
16
18
20
22
24


Model
Frequency
0
5
10
15
20
25
30
35
40
70
72
74
76
78
80
82


Origin
Frequency
0
50
100
150
200
250
1
3

Nhận xét: Đa phần các xe hơi vẫn khá tốn nhiên liệu, phần lớn các xe có mpg dưới 30, rất ít xe tiết kiệm nhiên liệu (trên 40 mpg). Phân nữa các loại xe là động cơ 4 xi lanh, nữa còn lại là 6 và 8 xi lanh, xe 3 và 5 xi lanh có rất ít trong dữ liệu, 7 xi lanh gần như không có. Hầu hết các xe khá nhẹ, có mã lực thấp, và có khả năng thay thế.

3.4.5. Biểu đồ phân tán trọng lượng so với MPG



Cân nặng và độ hao xăng tỉ lệ nghịch với nhau. Xe càng nặng thì càng tốn nhiên liệu, xe nhẹ thì tiết kiệm nhiên liệu hơn.

Hệ số tương quan:

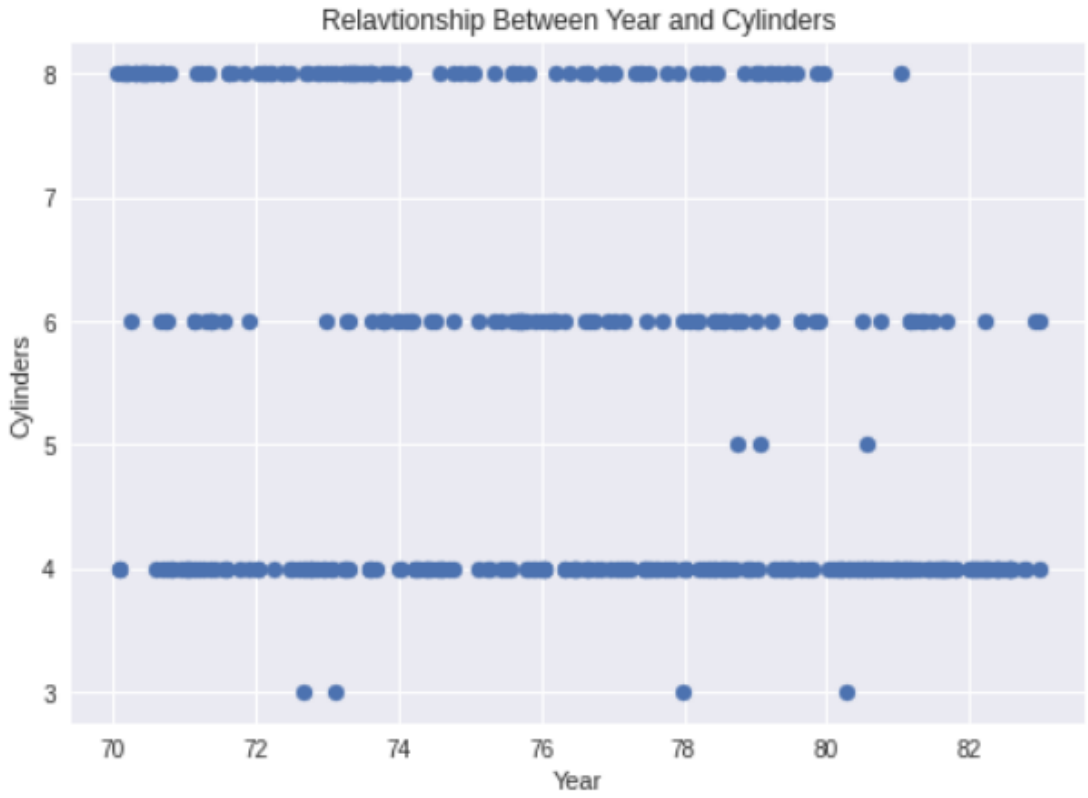
```
In [56]: weight_mpg_df = data[['weight', 'mpg']]
         weight_mpg_df.corr()
```

Out[56]:

	weight	mpg
weight	1.000000	-0.831741
mpg	-0.831741	1.000000

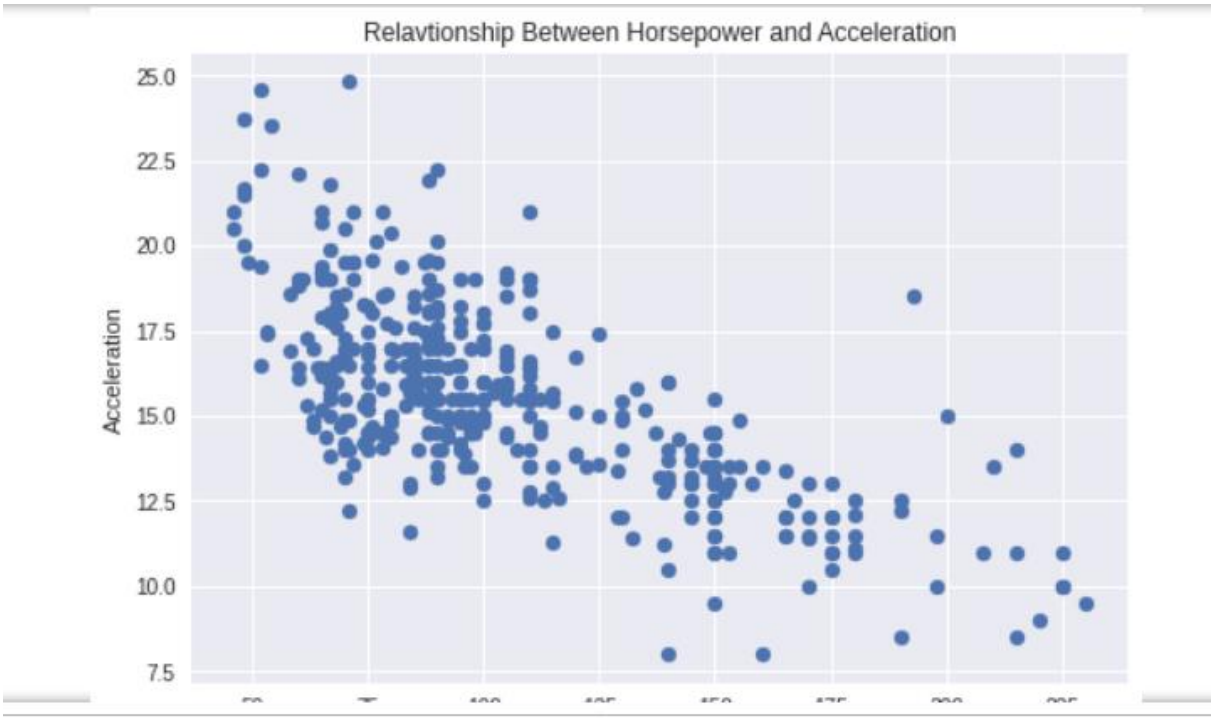
Correlation coefficient betwene Weight and MPG is: -0.831741

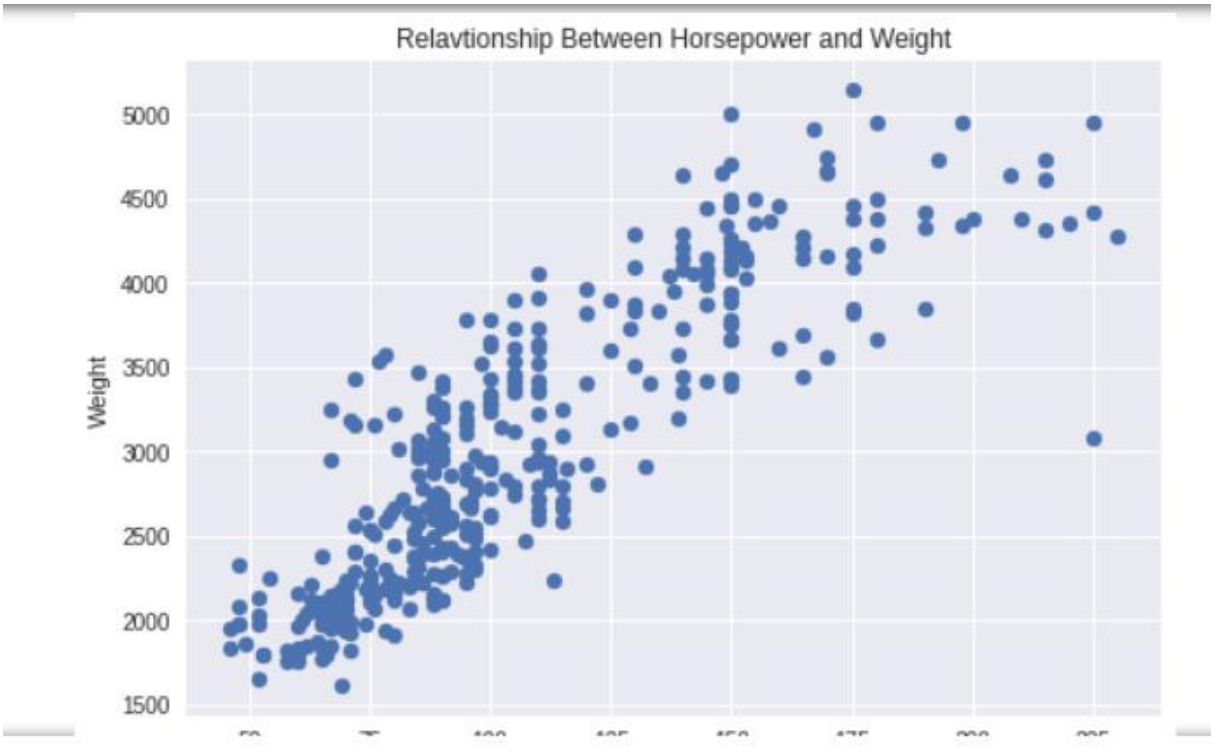
3.4.6. Biểu đồ phân tán các năm so với xi lanh



Hầu hết các xe hơi được sản xuất theo công nghệ động cơ 4 xi lanh, động cơ 6 và 8 xi lanh cũng khá phổ biến trong thời gian này. Bên cạnh đó động cơ 3 và 5 xi lanh không được ưa chuộng và ít được sản xuất.

3.4.7. Hai biểu đồ phân tán

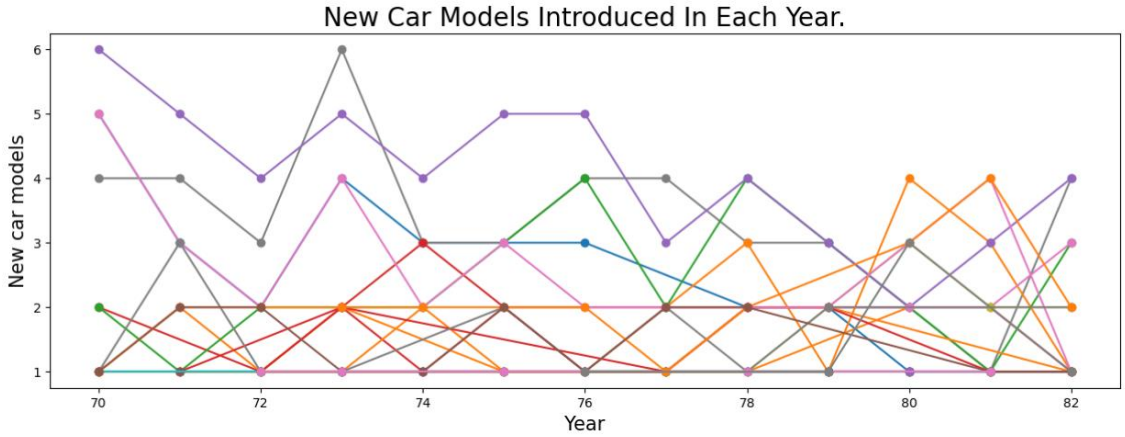




Nhận xét: Dựa vào hình một ta có thể thấy một cách rõ ràng là mã lực tỉ lệ nghịch với khả năng tăng tốc của xe. Đây là một điều khá lạ theo như thường lệ thì hai thông số này phải tỉ lệ thuận với nhau, động cơ mạnh thì xe chạy nhanh hơn. Để tìm hiểu thêm về điều này ta vẽ biểu đồ thứ hai về sự tương quan của mã lực và cân nặng của xe. Ta thấy mã lực tỉ lệ thuận với cân nặng của xe, và cân nặng có tốc độ tăng trưởng cao hơn

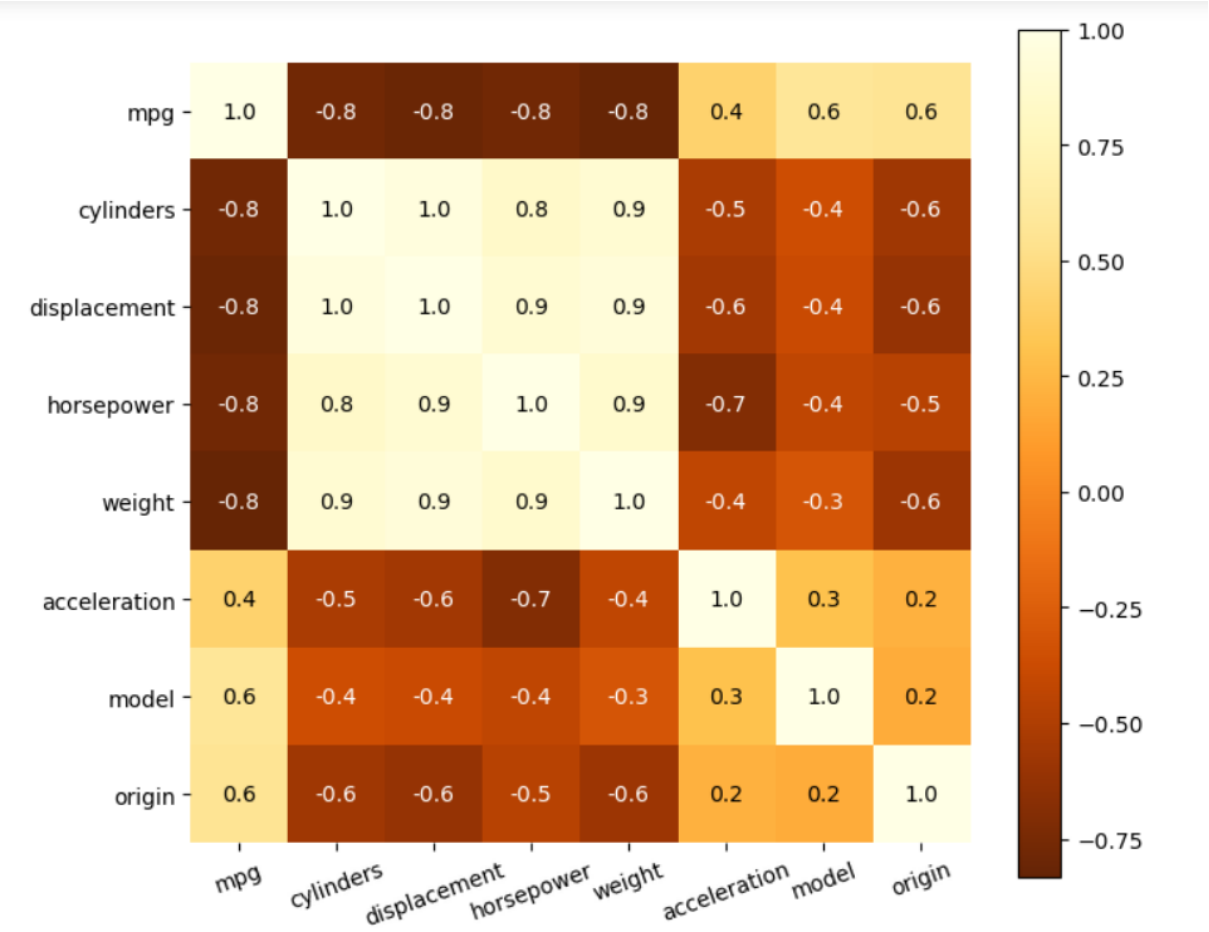
một xíu. Điều này lý giải vấn đề ở hình thứ nhất. Tuy mã lực cao nhưng vì xe quá nặng nên độ gia tốc của xe không hề tăng lên mà lại giảm xuống.

3.4.8. Biểu đồ số xe mới mỗi năm



Nhận xét: Hầu hết các hãng xe ít ra mắt xe mới hơn theo thời gian. Trung bình số xe mới mà các hãng ra mắt hằng nằm là 2 cho đến 3 mẫu xe.

3.4.9. Biểu đồ nhiệt



Nhận xét: Heatmap cho thấy mã lực và cân nặng tỉ lệ nghịch với mpg. Điều này cho thấy chỉ xe nhẹ cộng cơ vừa phải sẽ giúp xe tiết kiệm được nhiên liệu hơn. Đáng

ngạc nhiên khi model theo các năng lại tỉ lệ thuận với mpg (chỉ số tiêu thụ nhiên liệu), qua các năm, các xe không hề giảm tiêu thụ nhiên liệu mà còn tăng lên. Nhìn vào model và cân nặng , mã lực của xe thì ta thấy chúng tỉ lệ nghịch với nhau. Điều này lý giải tại sai sau các năm các xe lại tốn nhiên liệu hơn. Vậy ta có thể thấy vào thời đó các hãng xe không để ý tới việc tạo ra các xe ít tốn nhiên liệu hơn mà tập trung vào sản xuất xe to hơn, mạnh hơn để thu hút khách hàng.

3.5. Electric power consumption data

3.5.1. Load dữ liệu:

Vì dữ liệu thiếu ở dạng dấu "?" nên khi load dữ liệu cần phải chỉ ra kí tự "?" được xem như là dữ liệu thiếu.

```
import pandas as pd
data1 = pd.read_csv('household_power_consumption.txt', sep = ';', na_values = '?')
data1
```

	Date	Time	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3
0	16/12/2006	17:24:00	4.216	0.418	234.84	18.4	0.0	1.0	17.0
1	16/12/2006	17:25:00	5.360	0.436	233.63	23.0	0.0	1.0	16.0
2	16/12/2006	17:26:00	5.374	0.498	233.29	23.0	0.0	2.0	17.0
3	16/12/2006	17:27:00	5.388	0.502	233.74	23.0	0.0	1.0	17.0
4	16/12/2006	17:28:00	3.666	0.528	235.68	15.8	0.0	1.0	17.0
...	...	...	...	...	...	...	...	...	...
2075254	26/11/2010	20:58:00	0.946	0.000	240.43	4.0	0.0	0.0	0.0
2075255	26/11/2010	20:59:00	0.944	0.000	240.00	4.0	0.0	0.0	0.0
2075256	26/11/2010	21:00:00	0.938	0.000	239.82	3.8	0.0	0.0	0.0
2075257	26/11/2010	21:01:00	0.934	0.000	239.70	3.8	0.0	0.0	0.0
2075258	26/11/2010	21:02:00	0.932	0.000	239.55	3.8	0.0	0.0	0.0

2075259 rows × 9 columns

3.5.2. Ước lượng bộ nhớ cần thiết:

Đối với kiểu dữ liệu object, cần 50 MB bộ nhớ cho mỗi biến để lưu trữ các phương thức như length(), split(),..., đối với character đầu tiên, cần 8 bytes, các character còn lại mỗi character 1 bytes.

Đối với kiểu dữ liệu float64 8 bytes cho mỗi biến.

- Đối với cột Time, mỗi biến có 10 kí tự:
  (8+9+50)*2075259 = 134891835 bytes
- Đối với cột Date, trung bình mỗi biến có 12 kí tự:
  (8+11+50)*2075259 = 143192871 bytes
- 7 cột numeric, kiểu dữ liệu là float64:
  8*2075259 = 16602072 bytes
- Index: 128 bytes
- Tổng:
  (134891835 + 143192871 + 16602072 * 7 + 128) / 2^{20} = 376.03 MB

Dung lượng bộ nhớ thật tế:

```
data1.info(memory_usage='deep')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2075259 entries, 0 to 2075258
Data columns (total 9 columns):
 #   Column                 Dtype
---  ------                 -----
 0   Date                   object
 1   Time                   object
 2   Global_active_power    float64
 3   Global_reactive_power  float64
 4   Voltage                float64
 5   Global_intensity       float64
 6   Sub_metering_1         float64
 7   Sub_metering_2         float64
 8   Sub_metering_3         float64
dtypes: float64(7), object(2)
memory usage: 370.0 MB
```

```
data1.memory_usage(deep = True)
```

```
Index                       128
Date                  136859313
Time                  134891835
Global_active_power    16602072
Global_reactive_power  16602072
Voltage                16602072
Global_intensity       16602072
Sub_metering_1         16602072
Sub_metering_2         16602072
Sub_metering_3         16602072
dtype: int64
```

3.5.3. Tiền xử lý dữ liệu:

Định dạng dữ liệu trước khi tiền xử lý:

```
data1.dtypes
```

```
Date                   object
Time                   object
Global_active_power   float64
Global_reactive_power float64
Voltage               float64
Global_intensity      float64
Sub_metering_1        float64
Sub_metering_2        float64
Sub_metering_3        float64
dtype: object
```

Cần tạo ra 1 cột DateTime để lưu trữ dữ liệu kiểu DateTime được tạo từ cột Date và cột Time.

Lọc lấy dữ liệu của 2 ngày 01/02/2007 và 02/02/2007. Xoá các dòng có dữ liệu thiếu trong 2 ngày này.

Định dạng dữ liệu sau khi tiền xử lý:

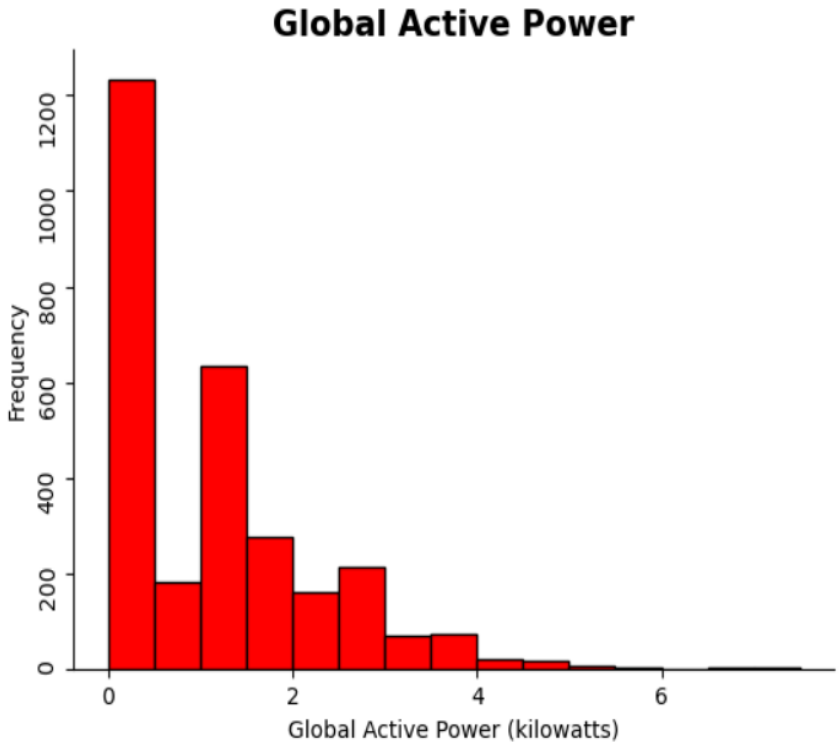
```
Date                            object
Time                            object
Global_active_power            float64
Global_reactive_power          float64
Voltage                        float64
Global_intensity               float64
Sub_metering_1                 float64
Sub_metering_2                 float64
Sub_metering_3                 float64
DateTime                datetime64[ns]
dtype: object
```

Dữ liệu sau khi tiền xử lý:

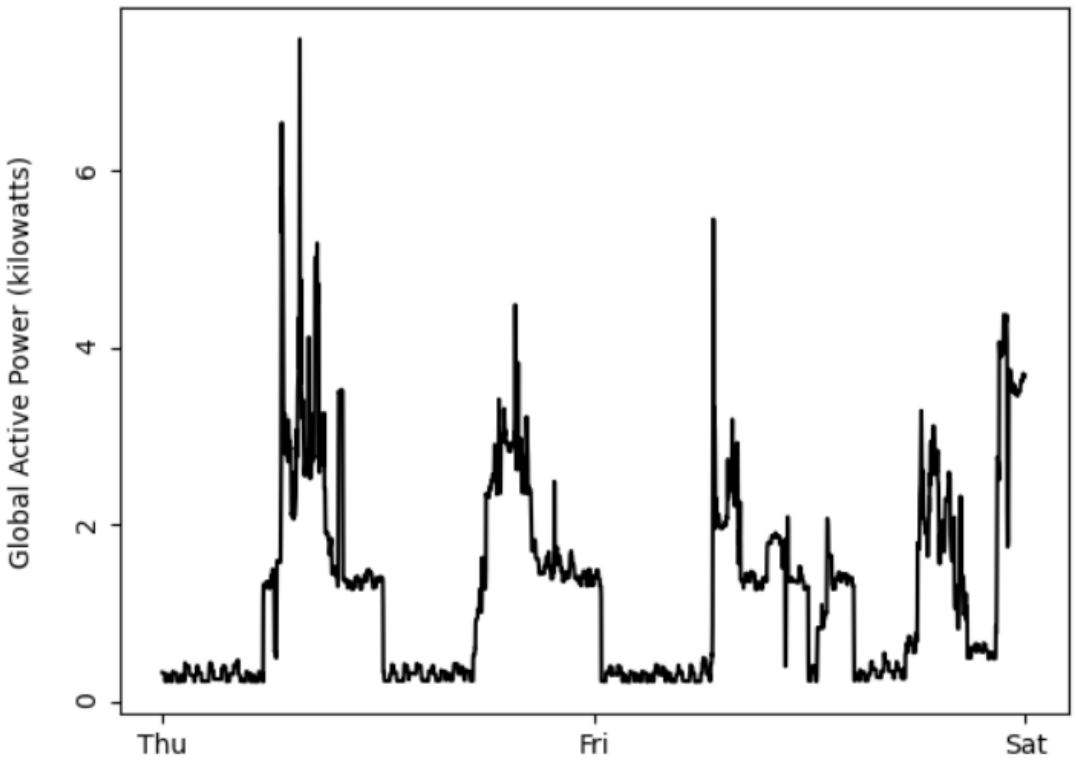
	Date	Time	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3	DateTime
66636	1/2/2007	00:00:00	0.326	0.128	243.15	1.4	0.0	0.0	0.0	2007-02-01 00:00:00
66637	1/2/2007	00:01:00	0.326	0.130	243.32	1.4	0.0	0.0	0.0	2007-02-01 00:01:00
66638	1/2/2007	00:02:00	0.324	0.132	243.51	1.4	0.0	0.0	0.0	2007-02-01 00:02:00
66639	1/2/2007	00:03:00	0.324	0.134	243.90	1.4	0.0	0.0	0.0	2007-02-01 00:03:00
66640	1/2/2007	00:04:00	0.322	0.130	243.16	1.4	0.0	0.0	0.0	2007-02-01 00:04:00
...	...	...	...	...	...	...	...	...	...	...
69511	2/2/2007	23:55:00	3.696	0.226	240.90	15.2	0.0	1.0	18.0	2007-02-02 23:55:00
69512	2/2/2007	23:56:00	3.698	0.226	241.02	15.2	0.0	2.0	18.0	2007-02-02 23:56:00
69513	2/2/2007	23:57:00	3.684	0.224	240.48	15.2	0.0	1.0	18.0	2007-02-02 23:57:00
69514	2/2/2007	23:58:00	3.658	0.220	239.61	15.2	0.0	1.0	17.0	2007-02-02 23:58:00
69515	2/2/2007	23:59:00	3.680	0.224	240.37	15.2	0.0	2.0	18.0	2007-02-02 23:59:00

2880 rows × 10 columns

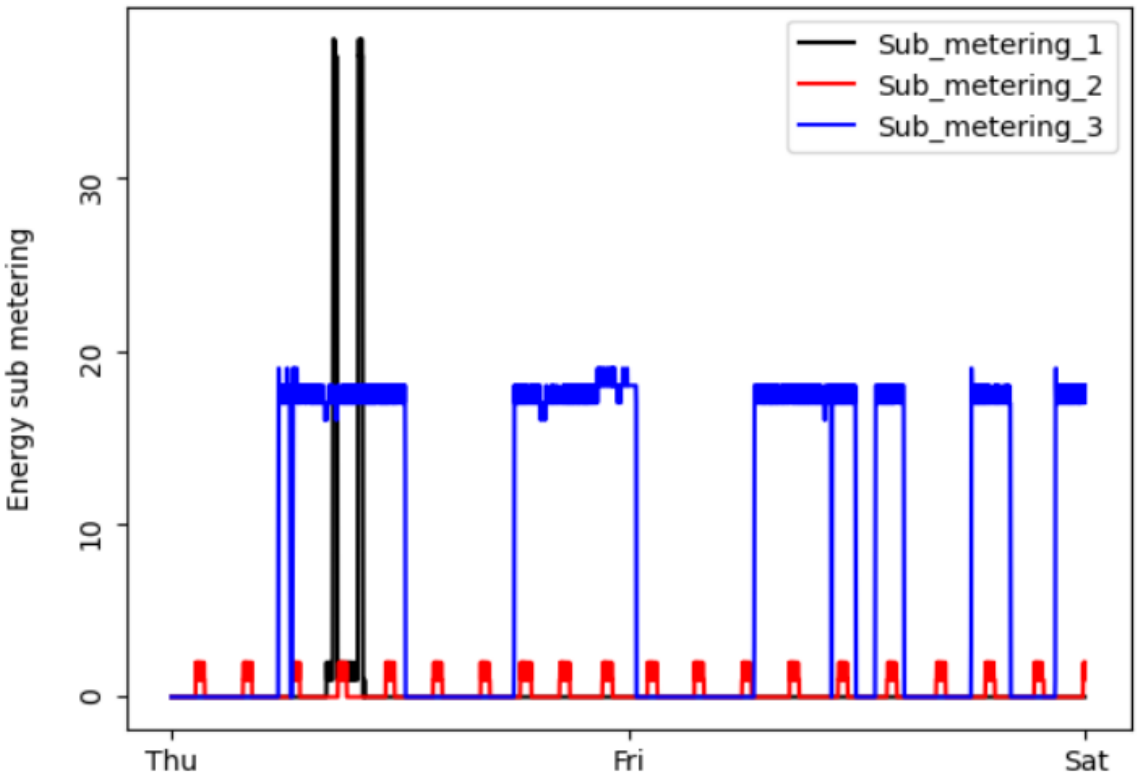
3.5.4. Trực quan theo yêu cầu đề bài:



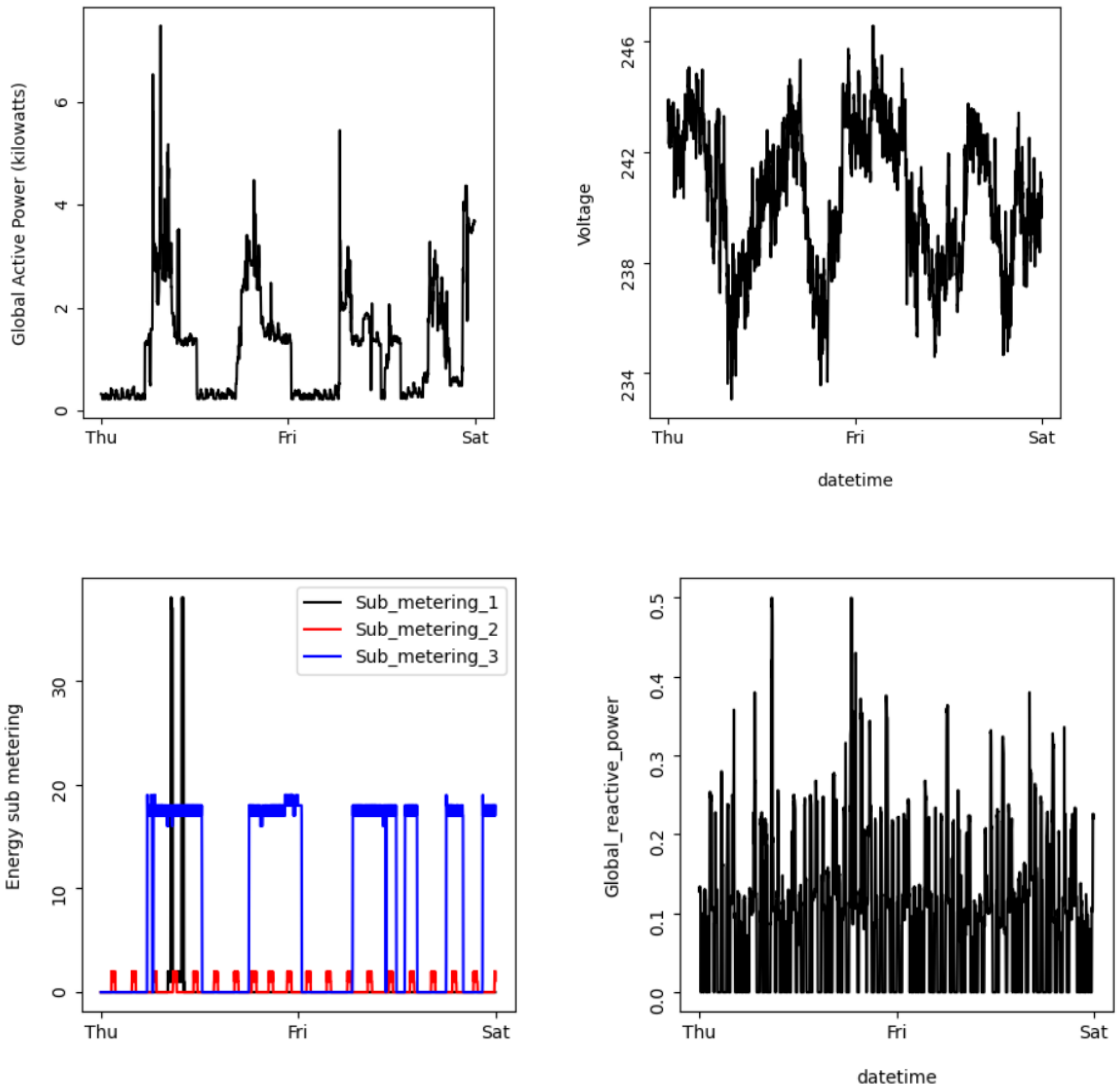
Plot 1


Global Active Power (kilowatts)
0
2
4
6
Thu
Fri
Sat

Plot 2


Sub_metering_1
Sub_metering_2
Sub_metering_3
Energy sub metering
0
10
20
30
Thu
Fri
Sat

Plot 3



Plot 4

3.5.5. Rút Kết:

Câu chuyện về việc sử dụng năng lượng của hộ gia đình trong 2 ngày

Trong 2 ngày ta thấy được hộ gia đình có tần suất sử dụng điện ở mức công suất từ trung bình tới thấp(Plot 1), trong đó điện được sử dụng nhiều nhất là vào khoảng buổi trưa và buổi tối(Plot 2). Dựa vào Plot 3 ta thấy được vào buổi trưa lượng điện được sử dụng nhiều vào các thiết bị bếp như máy rửa chén, lò vi sóng, còn buổi trưa, tối sẽ là máy sưởi, máy nước nóng.