FACULTY OF INFORMATION TECHNOLOGY
UNIVERSITY OF SCIENCE, HO CHI MINH NATIONAL UNIVERSITY

DATA VISUALIZATION BASICS

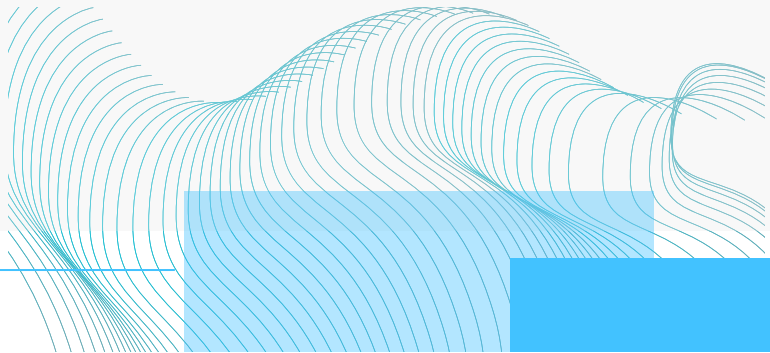
LAB 03
SUBJECT: DATAVISUALIZATION

2022 - 2023

SUBMITTED TO:
LECTURER. LE NGOC THANH

Due date: 09/05/2023

TABLE OF CONTENTS



ABOUT US

Võ Văn Hoàng
20127028

Nguyễn Đức Minh
20127049

Ngô Văn Trung Nguyên
20127054

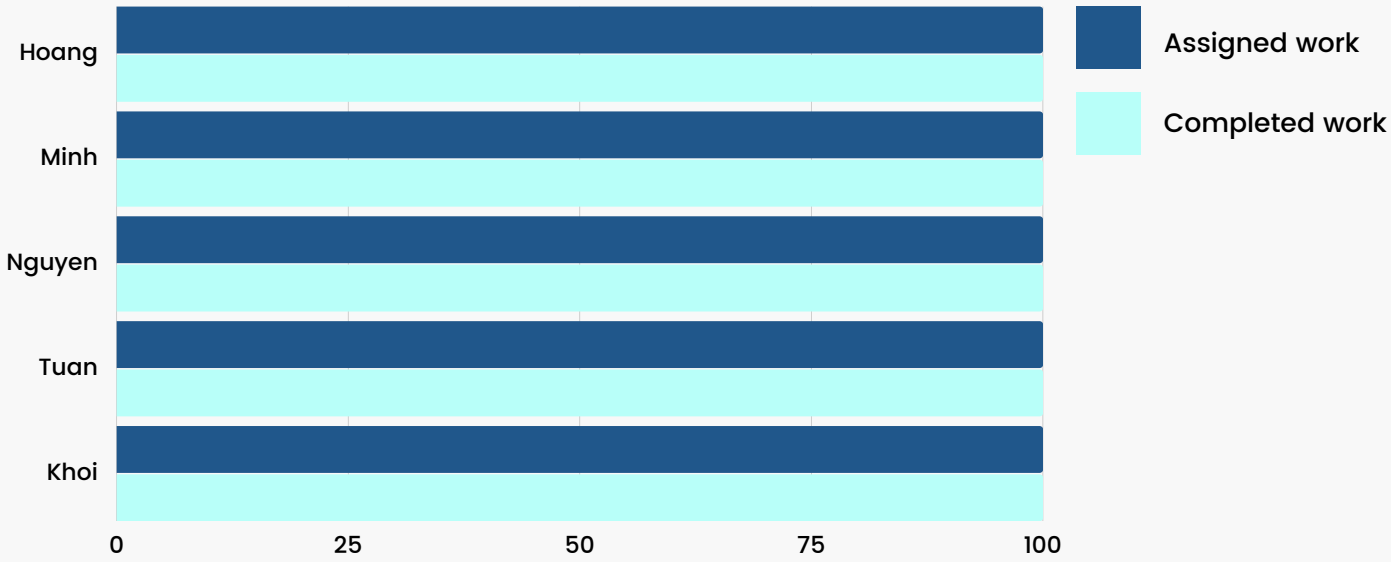
Nguyễn Minh Tuấn
20127092

Nguyễn Trương Minh Khôi
20127214

Some information about our group

We are K20 students from Faculty of Information Technology, all of us have the same field: Data Science, Data Visualization is a subject that we are all interested in.

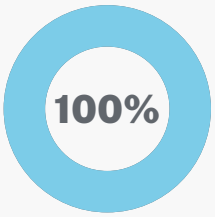
TASK ALLOCATION



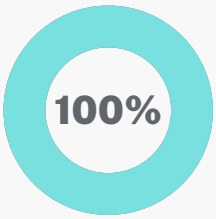
IN DETAIL

Vo Van Hoang - 100% 20127028	Writing report Do Electric power consumption data
Nguyen Duc Minh - 100% 20127049	Pandas Researching Do Car MPG data
Ngo Van Trung Nguyen - 100% 20127054	Numpy Researching Do Electric power consumption data
Nguyen Minh Tuan - 100% 20127092	Writing report Do Car MPG data
Nguyen Truong Minh Khoi - 100% 20127214	Matplotlib Researching Do Car MPG data

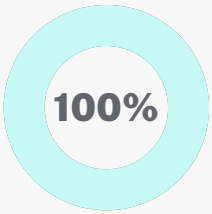
COMPLETED RATE



REPORT COMPLETED RATE

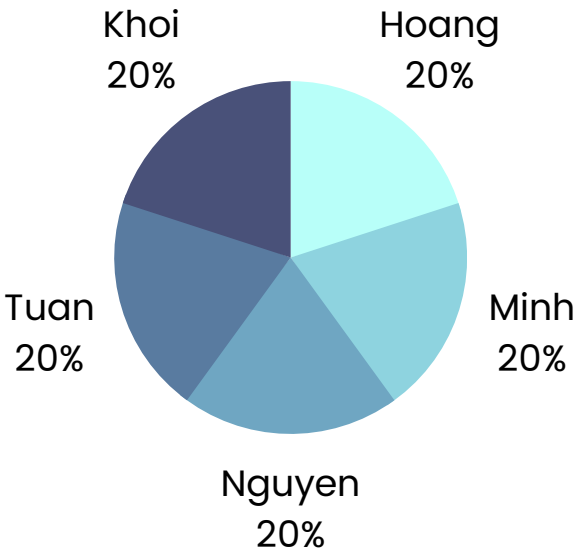


REQUIREMENT 1 COMPLETED RATE



REQUIREMENT 2 COMPLETED RATE

IN DETAIL



Week 1

Research and Planning

We shared work equally and team member had started researching their responsibility

Week 2

Speed up

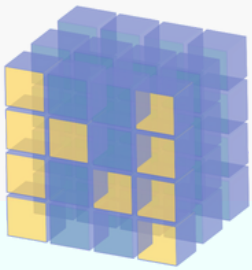
Because we only have 3 week to finish this lab so that we try our best completing all the requirements in the second week.

Week 3

Review and submit

All of us check the whole work of our team, give evaluation to improve and finish this lab.

LIBRARY DESCRIPTION



NumPy

pandas

matplotlib

1. NumPy

What is Numpy?

- Numpy is a very popular and powerful math library of Python, it allows us to work efficiently with matrics and arrays.
- Especially, matrix data and large arrays with much faster processing speed compared to Python without Numpy.

Main Functions

- **Statistics:** NumPy provides functions for statistical analysis.
- **Indexing and slicing:** NumPy includes sophisticated indexing and slicing capabilities for accessing and modifying specific components or parts of an array.
- **Array reshaping and resizing:** NumPy has routines for reshaping and resizing arrays.
- **Linear algebra:** NumPy includes a sophisticated linear algebra package contains functions for matrix operations, linear equation solving, computing eigenvalues and eigenvectors.
- **Mathematical operations:** NumPy has a large number of mathematical functions.
- **Array creation:** NumPy has a number of routines for array creation.
- **Random number generation:** NumPy routines for generating random numbers are available.

2. pandas

What is Pandas ?

- Pandas is a Python package that provides high-performance, user-friendly data structures and data analysis capabilities.
- It is based on NumPy and provides efficient methods for working with and manipulating data in tabular formats
- Pandas is frequently used in data science, finance, economics, and other professions that need data analysis.

Main Functions

- Dealing with missing values, removing duplicates, and replacing values.
- Data filtering, sorting, grouping, and aggregation.
- Time series analysis, including resampling, shifting, and rolling window calculations.
- Reading and writing data in a variety of formats, including CSV, Excel, SQL databases,...

3. matplotlib

What is Matplotlib ? 🔎

- Matplotlib is a popular Python data visualization toolkit that includes a variety of tools for building static, animated, and interactive displays.
- It is commonly used in data science, engineering, and other professions that require data visualization.

Main Functions 🛠️

- Plotting functions: Matplotlib has a number of plotting functions for constructing various types of plots, including line plots, scatter plots, bar plots, histograms, heatmaps,...
- Subplots: Matplotlib lets you to create numerous subplots within a single figure, allowing you to compare multiple plots at the same time.
- Animation: Matplotlib supports the creation of animated graphs, which allows for the visualization of dynamic data and processes.
- Saving plots: Matplotlib supports saving plots in a variety of formats, including PNG, PDF,...

REQUIREMENT 1
EXPLORATORY ANALYSIS OF CAR MPG DATA

CONTENT OF DATA

It is the analysis of Car MPG data from the UC Irvine Machine Learning Repository. The "Auto MPG Data Set" contains information about various cars such as the miles per gallon (mpg) that they can achieve, the number of cylinders in the engine, the displacement of the engine, the horsepower of the engine, the weight of the car, the acceleration time from 0 to 60 mph, the model year of the car, and the origin of the car. The dataset consists of 398 instances, and the goal is to predict the mpg of a car based on its other features.

MAY / 2023

Car MPG data

Information of column

Column	Description
mpg	Miles per gallon- the number of miles a car can travel on a gallon
cylinders	The number of cylinders of the vehicle
displacement	The volume of all cylinders of the vehicle
weight	Actual weight of the vehicle
acceleration	Vehicle acceleration
model	Generation of vehicle
origin	Version of vehicle
car_name	Name of car

1. How many cars and how many attributes are in the data set?

```
#1. How many cars and how many attributes are in the data set.
num_cars = len(data)
num_attributes = len(data.columns)
print("Number of cars: ", num_cars)
print("Number of attributes: ", num_attributes)
```

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

Car MPG data

2. How many distinct car companies are represented in the data set? What is the name of the car with the best MPG? What car company produced the most 8-cylinder cars? What are the names of 3-cylinder cars? Do some internet search that can tell you about the history and popularity of those 3-cylinder cars.

```
num_companies = len(data['car_name'].str.split().str[0].unique())
print("Number of distinct car companies: ", num_companies)
best_mpg_car = data.loc[data['mpg'].idxmax()]['car_name']
print("Car with the best MPG: ", best_mpg_car)
most_8cyl_company = data.loc[data['cylinders'] == 8]['car_name'].str.split().str[0].mode()[0]
print("Car company that produced the most 8-cylinder cars: ", most_8cyl_company)
cyl3_cars = data.loc[data['cylinders'] == 3]['car_name'].unique()
print("Names of 3-cylinder cars: ", cyl3_cars)
```

```
Number of distinct car companies:  38
Car with the best MPG:  mazda glc
Car company that produced the most 8-cylinder cars:  ford
Names of 3-cylinder cars:  ['mazda rx2 coupe' 'maxda rx3' 'mazda rx-4' 'mazda rx-7 gs']
```

- Three-cylinder engines have been around for decades, but only recently have they gained popularity for mainstream vehicles. They are known for delivering good fuel economy while still providing adequate power. Many manufacturers have introduced small, fuel-efficient cars with three-cylinder engines, and some have even introduced larger vehicles with these engines. However, concerns about their reliability and durability still exist among some consumers.

3. What is the range, mean, and standard deviation of each attribute? Pay attention to potential missing values.

```
# 3. What is the range, mean, and standard deviation of each attribute? Pay attention to potential missing values
desc_stats = data.describe()

# Print the range, mean, and standard deviation of each attribute
for col in desc_stats.columns:
    if col != 'count':
        range = desc_stats[col]['max'] - desc_stats[col]['min']
        mean = desc_stats[col]['mean']
        standard_deviation = desc_stats[col]['std']

        print(f"{col}:")
        print(f"    Range: {range}")
        print(f"    Mean: {mean}")
        print(f"    Standard deviation: {standard_deviation}\n")
```

Car MPG data

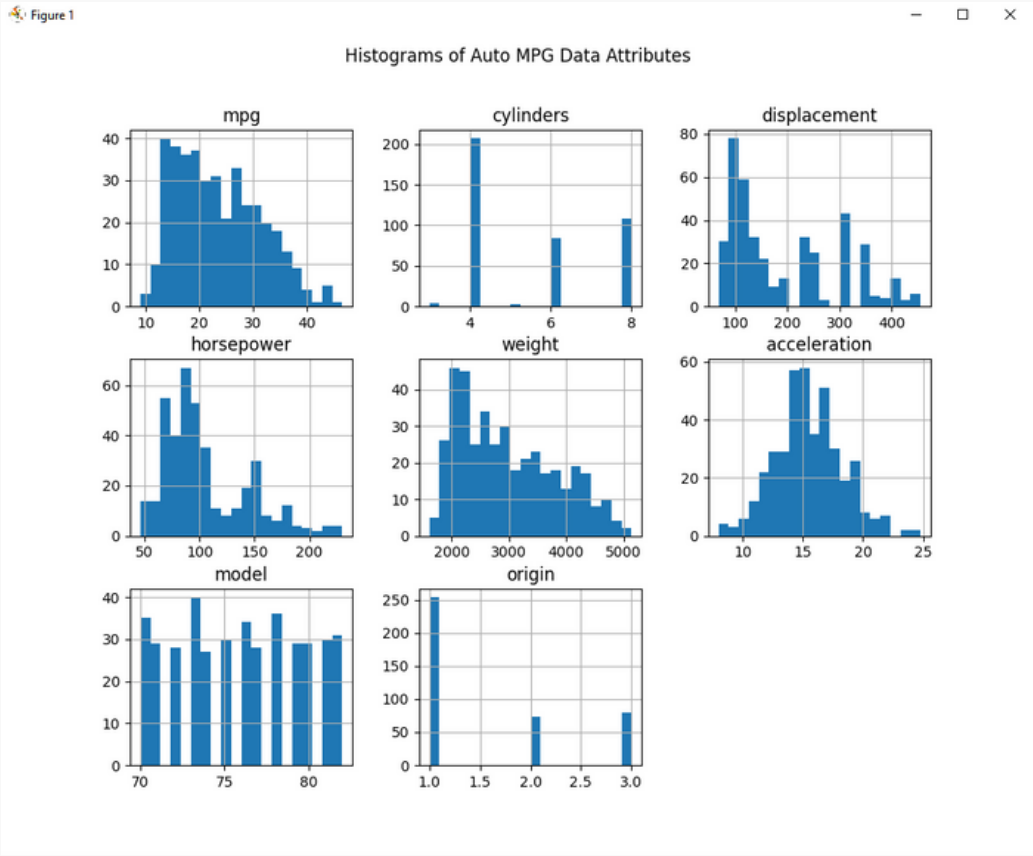
```
mpg:
    Range: 37.6
    Mean: 23.514572864321607
    Standard deviation: 7.8159843125657815

cylinders:
    Range: 5.0
    Mean: 5.475369458128079
    Standard deviation: 1.7121596315485297

displacement:
    Range: 387.0
    Mean: 194.7795566502463
    Standard deviation: 104.92245837948875

horsepower:
    Range: 184.0
    Mean: 105.0825
    Standard deviation: 38.7687791831052

weight:
    Range: 3527.0
    Mean: 2979.4137931034484
    Standard deviation: 847.0043282393509

acceleration:
    Range: 16.8
    Mean: 15.519704433497537
    Standard deviation: 2.803358816342546

model:
    Range: 12.0
    Mean: 75.92118226600985
    Standard deviation: 3.74873734545588

origin:
    Range: 2.0
    Mean: 1.5689655172413792
    Standard deviation: 0.7974789993244706
```

Above is the range, mean, and standard deviation of each attribute

Car MPG data

4. Plot histograms for each attribute. Pay attention to the appropriate choice of number of bins. Write 2-3 sentences summarizing some interesting aspects of the data by looking at the histograms.

```
#4. Plot histograms for each attribute. Pay attention to the appropriate choice of number of bins.
#Write 2-3 sentences summarizing some interesting aspects of the data by looking at the histograms.
data.hist(bins=20, figsize=(10, 8))
plt.suptitle('Histograms of Auto MPG Data Attributes')
plt.show()
```

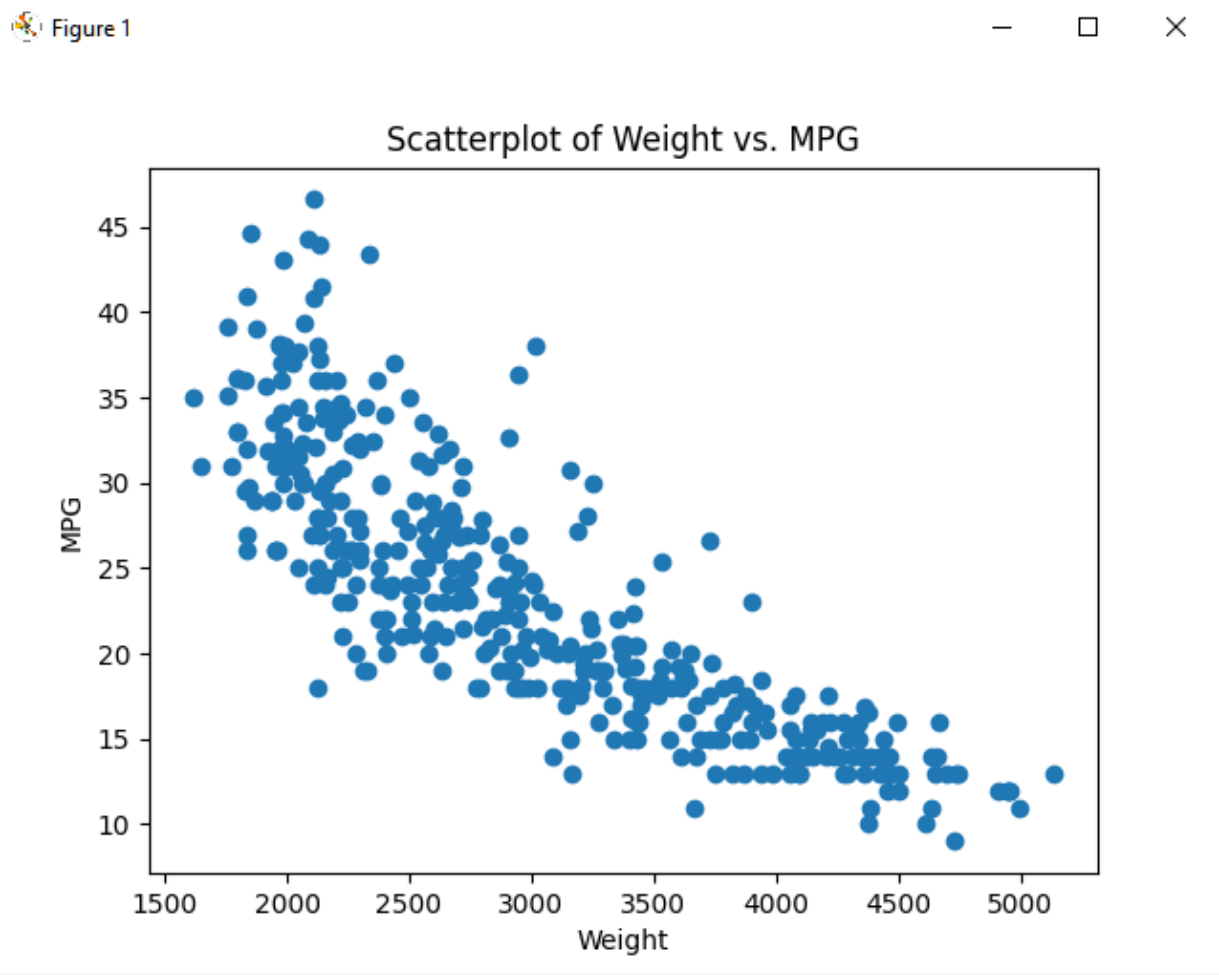


- The cylinders attribute is heavily skewed towards 4 and 8 cylinder cars, with relatively few 3 and 5 cylinder cars.
- The horsepower attribute is also skewed, with a large number of cars having low horsepower and a long tail to the right.

5. Plot a scatterplot of weight vs. MPG attributes. What do you conclude about the relationship between the attributes? What is the correlation coefficient between the 2 attributes?

```
plt.scatter(data['weight'], data['mpg'])
plt.xlabel('Weight')
plt.ylabel('MPG')
plt.title('Scatterplot of Weight vs. MPG')
plt.show()
```

Car MPG data




```
print(data[['weight','mpg']].corr())
```

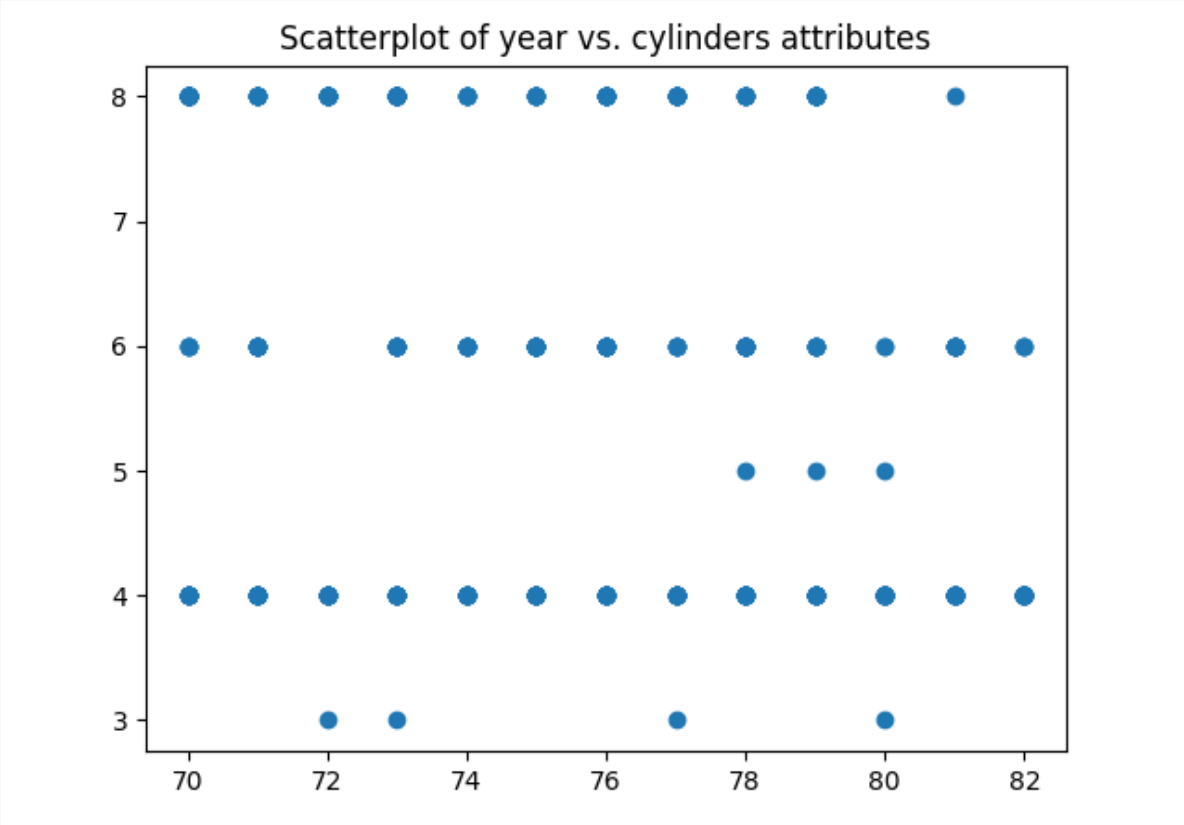
```
            weight       mpg
weight    1.000000 -0.831741
mpg      -0.831741  1.000000
```

- As for the relationship between the attributes and the correlation coefficient, we can see from the scatterplot that there seems to be a negative correlation between weight and MPG, meaning that as weight increases, MPG tends to decrease.

6. Plot a scatterplot of year vs. cylinders attributes. Add a small random noise to the values to make the scatterplot look nicer. What can you conclude? Do some internet search about the history of car industry during 70's that might explain the results.

```
plt.scatter(data=data, x="model", y="cylinders");
plt.title('Scatterplot of year vs. cylinders attributes')
plt.show()
```

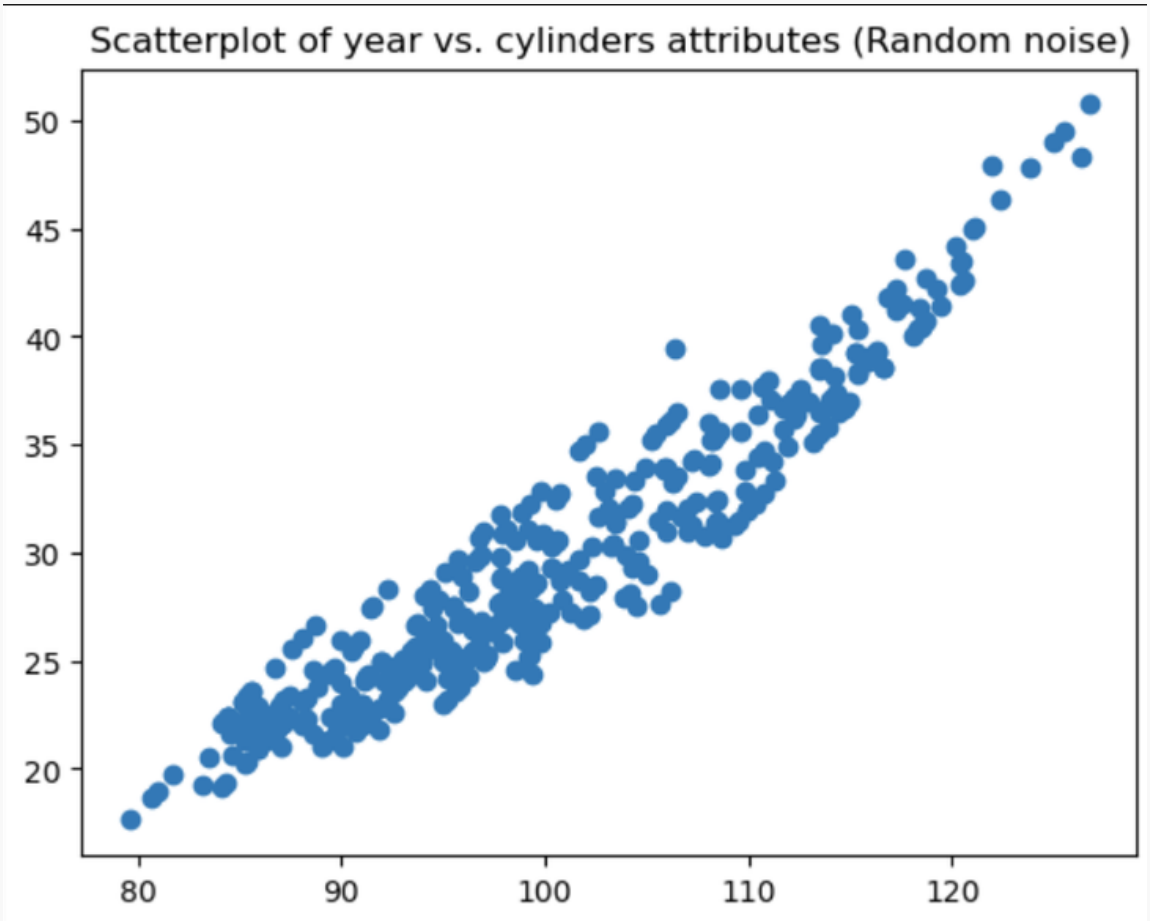
Car MPG data



- From the scatterplot, we can see that the number of cylinders in cars decreased over time from the 1970s to the early 1980s
- According to what i search on the internet, in 1970 the American automobile industry was under threat from several angles. Falling sales, a 57-day strike at General Motors idling around 347,000 workers, and higher quality foreign cars were the primary culprits,
- oil crisis, which led to increased demand for more fuel-efficient cars. As a result, many car manufacturers began to reduce the number of cylinders in their cars to improve fuel efficiency.

```
noise = data.mpg + np.random.random(len(data.mpg))
data1 = data.copy()
data1['model'] += noise
data1['cylinders'] += noise
plt.scatter(data=data1, x="model", y="cylinders")
plt.title('Scatterplot of year vs. cylinders attributes (Random noise)')
plt.show()
```

Car MPG data



- Using the original data, we can see that when the number of cylinders is high, the fuel consumption is correspondingly high (seen with an even number of cylinders).
- However, after introducing a random quantity of noise, the aforementioned statement becomes clearer than ever when the data points form a straight line..

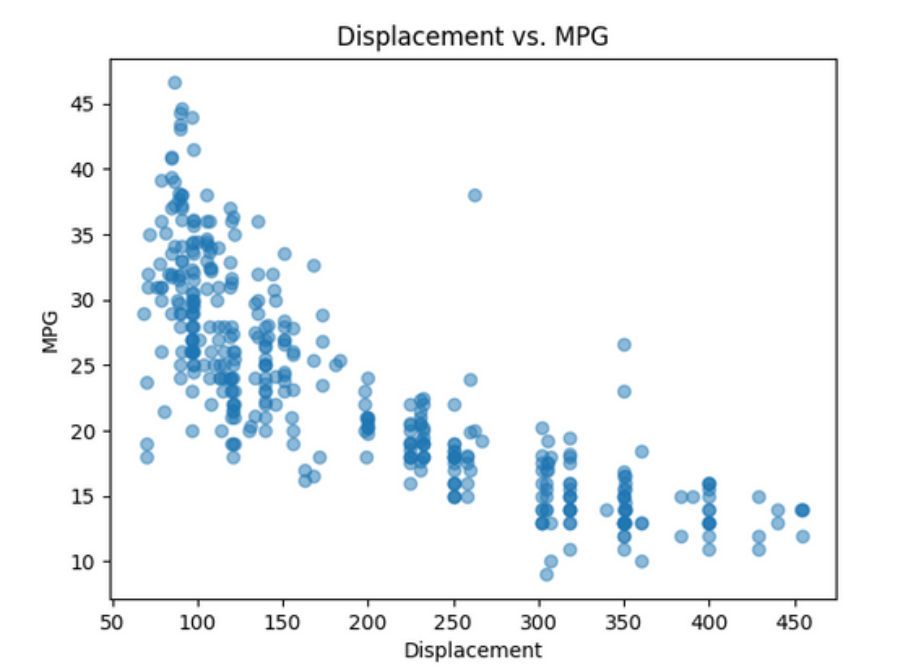
7. Show 2 more scatterplots that are interesting do you. Discuss what you see

Displacement vs MPG

```
#Displacement vs MPG
plt.scatter(data['displacement'], data['mpg'], alpha=0.5)
plt.xlabel('Displacement')
plt.ylabel('MPG')
plt.title('Displacement vs. MPG')
plt.show()
```

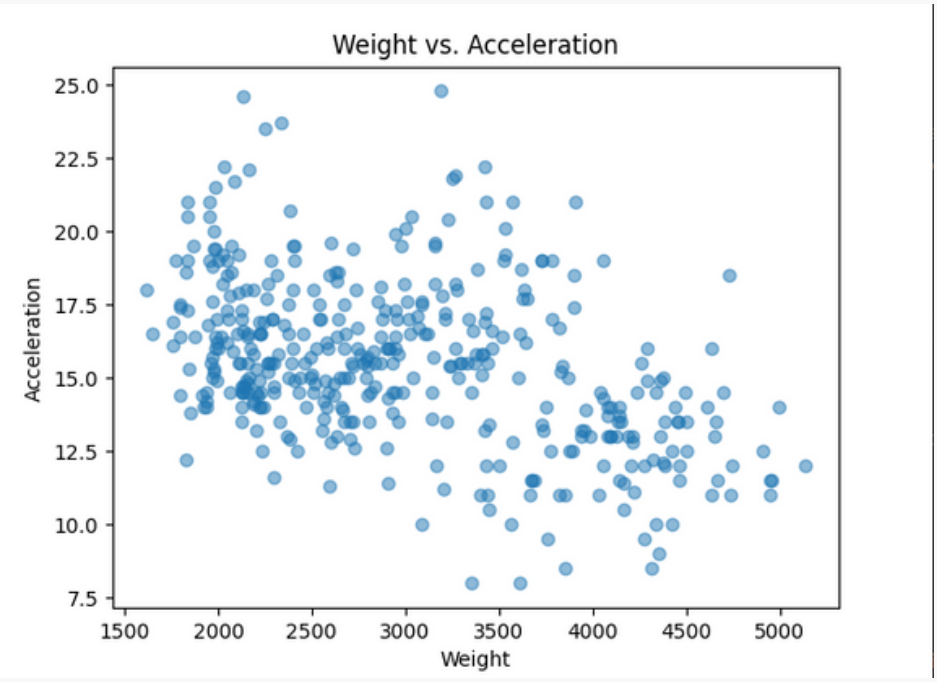
- Displacement vs. MPG: The scatterplot bellow can show the relationship between the size of the engine and the fuel efficiency of the car. It can be expected that larger engines consume more fuel and therefore have lower MPG.

Car MPG data



Weight vs. Acceleration

```
plt.scatter(data['weight'], data['acceleration'], alpha=0.5)
plt.xlabel('Weight')
plt.ylabel('Acceleration')
plt.title('Weight vs. Acceleration')
plt.show()
```

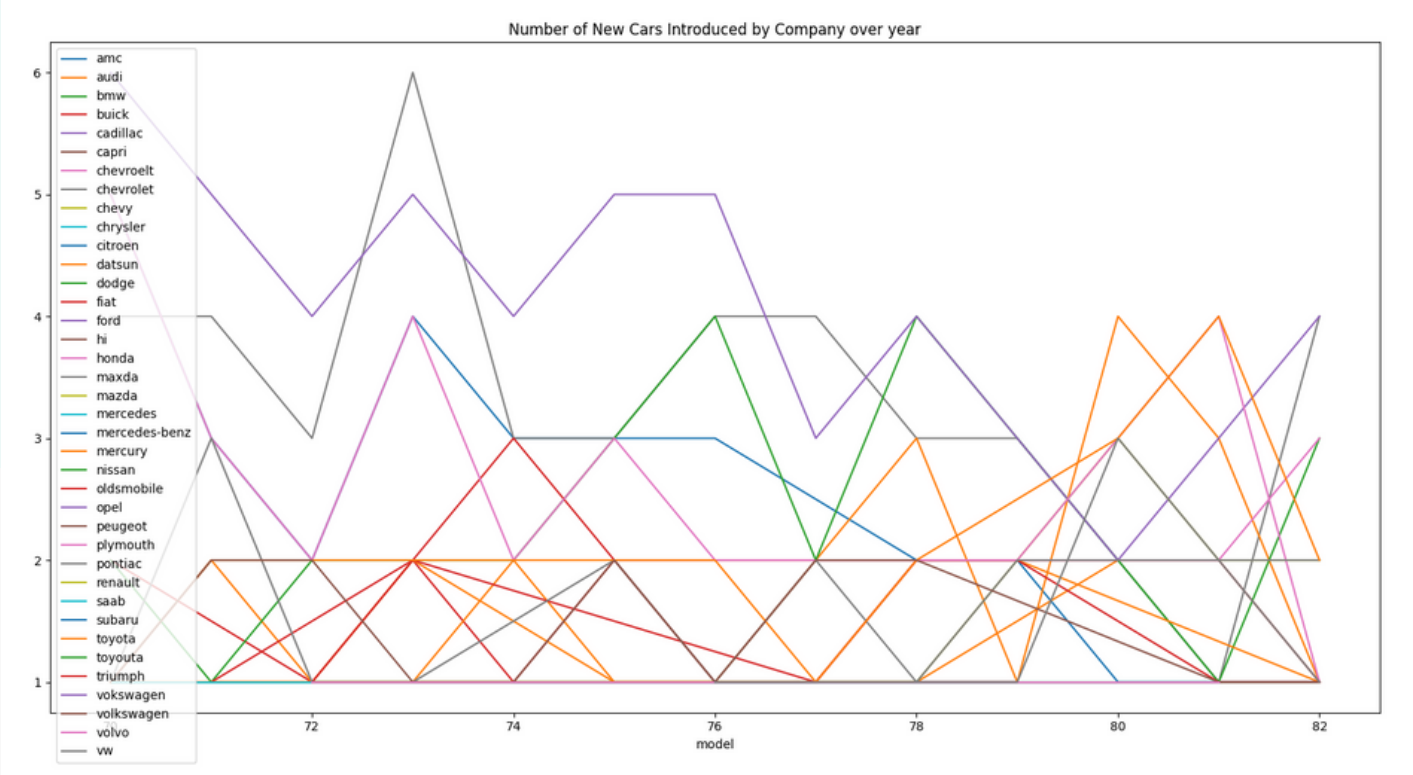


- Weight vs. Acceleration: This scatterplot can show the relationship between the weight of the car and how quickly it can accelerate. It can be expected that heavier cars have slower acceleration times.

Car MPG data

8. Plot a time series for all the companies that show how many new cars they introduces during each year. Do you see some interesting trends?

```
data['company'] = data['car_name'].str.split().str[0]
new_cars_by_company = data.groupby(['company', 'model'])['car_name'].count()
fig, ax = plt.subplots(figsize=(15, 7))
for company in new_cars_by_company.index.get_level_values('company').unique():
    new_cars_by_company.loc[company].plot(ax=ax, label=company)
ax.set_title('Number of New Cars Introduced by Company over year')
ax.legend()
plt.show()
```



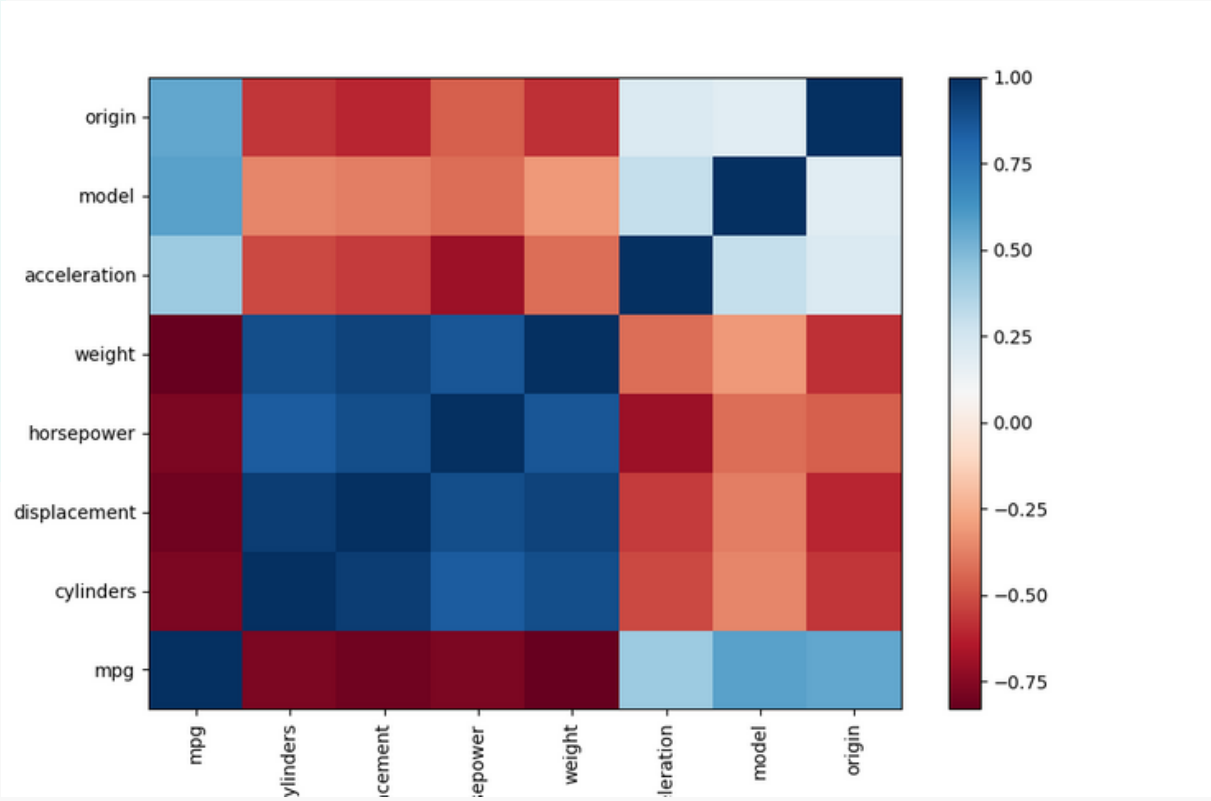
Some interesting trends:

- Firstly, there is a clear decline in the number of new cars introduced by American companies from the 1970s to the 1980s.
- Secondly, Japanese car companies such as Toyota, Nissan and Honda showed a steady increase in the number of new cars introduced during this time period
- Thirdly, European car companies such as Volkswagen, Volvo and BMW showed fluctuations in the number of new cars introduced

Car MPG data

9. Calculate the pairwise correlation, and draw the heatmap with Matplotlib. Do you see some interesting correlation?

```
corr_matrix = data.iloc[:,0:8].corr()
plt.pcolor(corr_matrix, cmap='RdBu')
plt.colorbar()
plt.xticks(np.arange(0.5, len(corr_matrix.columns), 1), corr_matrix.columns, rotation=90)
plt.yticks(np.arange(0.5, len(corr_matrix.index), 1), corr_matrix.index)
plt.show()
```



There are some interesting correlations

- Strong positive correlation between horsepower and weight of the car.
- Strong negative correlation between miles per gallon (mpg) and both horsepower and weight.
- Positive correlation between displacement and horsepower, as well as between displacement and weight.
- Negative correlation between mpg and acceleration.

REQUIREMENT 2
ELECTRIC POWER CONSUMPTION

CONTENT OF DATA

It is the measurements of electric power consumption in one household with a one-minute sampling rate over a period of almost 4 years. Different electrical quantities and some sub-metering values are available.

REQUIRED TASKS

- Examine how household energy usage varies over a 2-day period (2007-02-01 and 2007-02-02) in February, 2007 and reconstruct the following plots below.
- Construct the plot and save it to a PNG file and name each of the plot files as Plot1.png, Plot2.png,...
- Complete four functions in the skeleton Python script.
- Write a story about the energy use of the household during the two days. Base the story on the plots you have made.

ELECTRIC POWER CONSUMPTION

1. PREPROCESSING

01 Create a datetime column which consists of both date and time

```
# Preprocessing by making datetime has the value
df['datetime'] = df['Date'] + ' ' + df['Time']
df.datetime = pd.to_datetime(df.datetime, infer_datetime_format=True)
```

02 Replace missing value (?) by 0

```
# Preprocessing by replacing missing value by 0 index
df = df.replace({'?': 0})
```

03 Transform related columns from string to numeric

```
# Transform Global_active_power from string type to numeric type
df['Global_active_power'] = df['Global_active_power'].apply(lambda x: pd.to_numeric(x, errors='coerce'))
# Transform Global_reactive_power from string type to numeric type
df['Global_reactive_power'] = df['Global_reactive_power'].apply(lambda x: pd.to_numeric(x, errors='coerce'))
# Transform Voltage from string type to numeric type
df['Voltage'] = df['Voltage'].apply(lambda x: pd.to_numeric(x, errors='coerce'))
# Transform Sub_metering_1 from string type to numeric type
df['Sub_metering_1'] = df['Sub_metering_1'].apply(lambda x: pd.to_numeric(x, errors='coerce'))
# Transform Sub_metering_2 from string type to numeric type
df['Sub_metering_2'] = df['Sub_metering_2'].apply(lambda x: pd.to_numeric(x, errors='coerce'))
```

04 Create data of 2007-02-01 and 2007-02-02 to a dataframe and call it datetime

```
# Create a dataframe including the 2 researching date
data = df[df.Date.isin(['1/2/2007','2/2/2007'])]
data = data.set_index('datetime')
```

05 Checking result on Notebook

datetime	Date	Time	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3
2007-02-01 00:00:00	1/2/2007	00:00:00	0.326	0.128	243.15	1.400	0.0	0.0	0.0
2007-02-01 00:01:00	1/2/2007	00:01:00	0.326	0.130	243.32	1.400	0.0	0.0	0.0
2007-02-01 00:02:00	1/2/2007	00:02:00	0.324	0.132	243.51	1.400	0.0	0.0	0.0
2007-02-01 00:03:00	1/2/2007	00:03:00	0.324	0.134	243.90	1.400	0.0	0.0	0.0
2007-02-01 00:04:00	1/2/2007	00:04:00	0.322	0.130	243.16	1.400	0.0	0.0	0.0
2007-02-01 00:05:00	1/2/2007	00:05:00	0.320	0.126	242.29	1.400	0.0	0.0	0.0
2007-02-01 00:06:00	1/2/2007	00:06:00	0.320	0.126	242.46	1.400	0.0	0.0	0.0
2007-02-01 00:07:00	1/2/2007	00:07:00	0.320	0.126	242.63	1.400	0.0	0.0	0.0
2007-02-01 00:08:00	1/2/2007	00:08:00	0.320	0.128	242.70	1.400	0.0	0.0	0.0
2007-02-01 00:09:00	1/2/2007	00:09:00	0.236	0.000	242.89	1.000	0.0	0.0	0.0

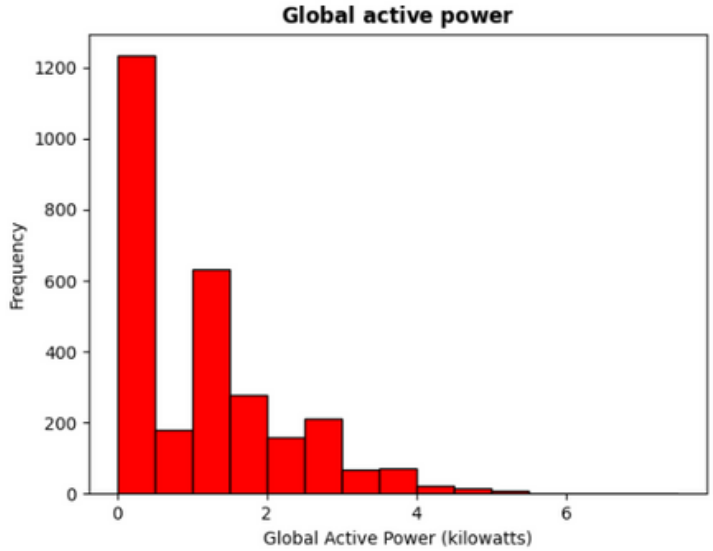
ELECTRIC POWER CONSUMPTION

2. Plot 1

01 Analysis and definition for the requirement

- The problem data involves estimating electricity usage in a household at a sampling rate of one minute over a period of 4 years.
- The visual problem we use only uses 2 days from February 1 to February 2, 2007 so that we need to filter data only takes 2 dates 1/2/2007 and 2/2/2007.
- We will have to re-visualize the existing image using the numpy, pandas and matplotlib libraries .
- We need to plot the column 'Global Active Power (kilowatt)' with the X-axis being 'Global Active Power' and the Y-axis being 'Frequency'.
- Draw histogram

02 Display Plot1.png



03 Code demo and explanation

```
def plot1():
    plt.hist(data['Global_active_power'], edgecolor='black', color='red', bins=np.arange(0, 8, 0.5));
    plt.ylabel("Frequency")
    plt.xlabel("Global Active Power (kilowatts)")
    plt.title(r"$\bf{Global\ active\ power}$")
    plt.xticks([0, 2, 4, 6]);
    plt.savefig('Plot1.png')
    # pass
```

- **edgecolor='black':** make the color of edges in this plot in black
- **color='red':** make the color of bars in this histogram in red
- **bins=np.arange(0, 8, 0.5):** create a histogram that bins are between 0 and 8 with size of 0.5.
- **plt.ylabel("Frequency"):** identify label for the y-axis is "Frequency".
- **plt.xlabel("Global Active Power (kilowatts)"):** identify the label for the x-axis is "Global Active Power (kilowatts)".
- **plt.title(r"$\bf{Global\ active\ power}$"):** set the title "Global active power" in bold font.
- **plt.xticks([0, 2, 4, 6]):** set the positions of the tick note on the x-axis.
- **plt.savefig('Plot1.png')** save the current plot to 'Plot1.png'

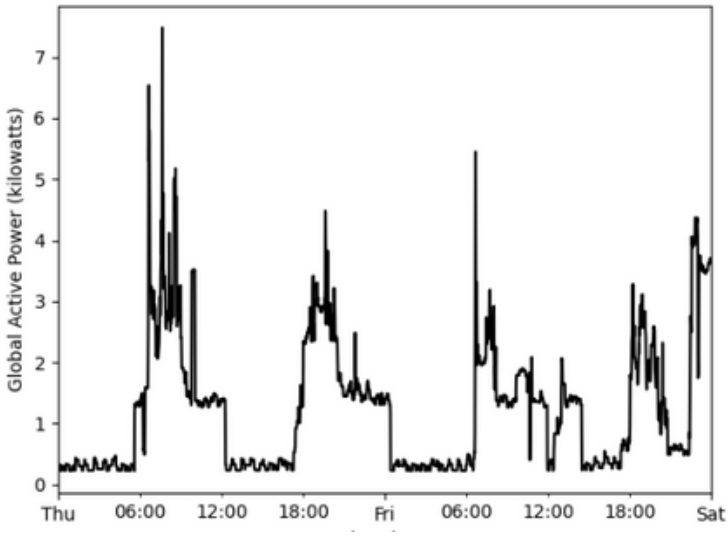
ELECTRIC POWER CONSUMPTION

3. Plot 2

01 Analysis and definition for the requirement

- The chart is still using the 'Global Active Power (kilowatt)' column chart with the x-axis being the hours of the day starting from Thursday (February 1, 2007) to the end of Friday (February 2, 2007) and the y-axis is 'Global Active Power' (actual values for each hour period).
- The visual problem we use only uses 2 days from February 1 to February 2, 2007 so that we need to filter data only takes 2 dates 1/2/2007 and 2/2/2007.
- We will have to re-visualize the existing image using the Numpy, Pandas and Matplotlib libraries .

02 Display Plot2.png



03 Code demo and explanation

```
def plot2():
    fig,ax1 = plt.subplots()
    data['Global_active_power'].plot(ax=ax1, color='black')
    ax1.set_xticklabels(['Thu','Fri','Sat'])
    ax1.set_ylabel('Global Active Power (kilowatts)')
    plt.savefig('Plot2.png')
    # pass
```

- **fig,ax1 = plt.subplots():** generate a new plottable figure and axes object and subplots() function returns fig and ax1.
- **data['Global_active_power'].plot(ax=ax1, color='black'):** plot the variable 'Global_active_power' on the ax1 axes with the black line.
- **ax1.set_xticklabels(['Thu','Fri','Sat']):** use the set_xticklabels() method to set the labels for the x-axis tick markers.
- **ax1.set_ylabel('Global Active Power (kilowatts)'):** set the label for the y-axis of the plot to 'Global Active Power (kilowatts)'.
- **plt.savefig('Plot2.png'):** save the current plot to 'Plot2.png'.

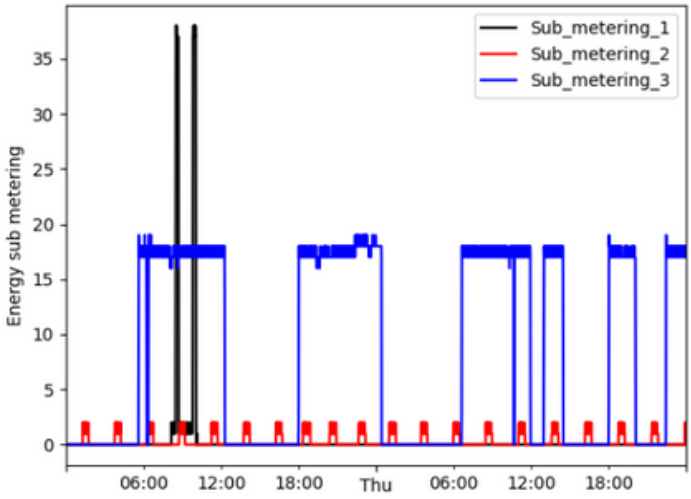
ELECTRIC POWER CONSUMPTION

4. Plot 3

01 Analysis and definition for the requirement

- The figure shows the comparison of the level as well as the time of using electrical appliances in the house at different times.
- **Sub_metering_1** refers to kitchen electricity (dishwasher, microwave, etc.).
- **Sub_metering_2** is the electricity in the laundry room (washing machine, dryer, refrigerator, or lamp).
- **Sub_metering_3** is for water heaters and air conditioners.

02 Display Plot3.png



03 Code demo and explanation

```
def plot3():
    fig,ax = plt.subplots()
    data['Sub_metering_1'].plot(color='black',ax=ax)
    data['Sub_metering_2'].plot(color='red',ax=ax)
    data['Sub_metering_3'].plot(color='blue',ax=ax)
    ax.set_xticklabels(['','Thu','','Fri','','Sat',''])
    plt.legend()
    plt.ylabel('Energy sub metering')
    plt.savefig('Plot3.png')
    # pass
```

- **fig,ax = plt.subplots():** Using the matplotlib library's subplots() method, this produces a new figure and axis object.
- **data['Sub_metering_1'].plot(color='black',ax=ax):** make line graph, utilize the plot() method of Pandas, Sub_metering_1 is shown in black.
- **data['Sub_metering_2'].plot(color='red',ax=ax):** make line graph, utilize the plot() method of Pandas, Sub_metering_2 is shown in red.
- **data['Sub_metering_3'].plot(color='blue',ax=ax):** make line graph, utilize the plot() method of Pandas, Sub_metering_3 is shown in blue.
- **ax.set_xticklabels(['','Thu','','Fri','','Sat','']):** change the x-axis tick labels in the plot to a list of strings.
- **plt.legend():** make a legend to the chart.
- **plt.ylabel('Energy sub metering'):** identify label for the y-axis is 'Energy sub metering'.
- **plt.savefig('Plot3.png'):** save the current plot to 'Plot3.png'.

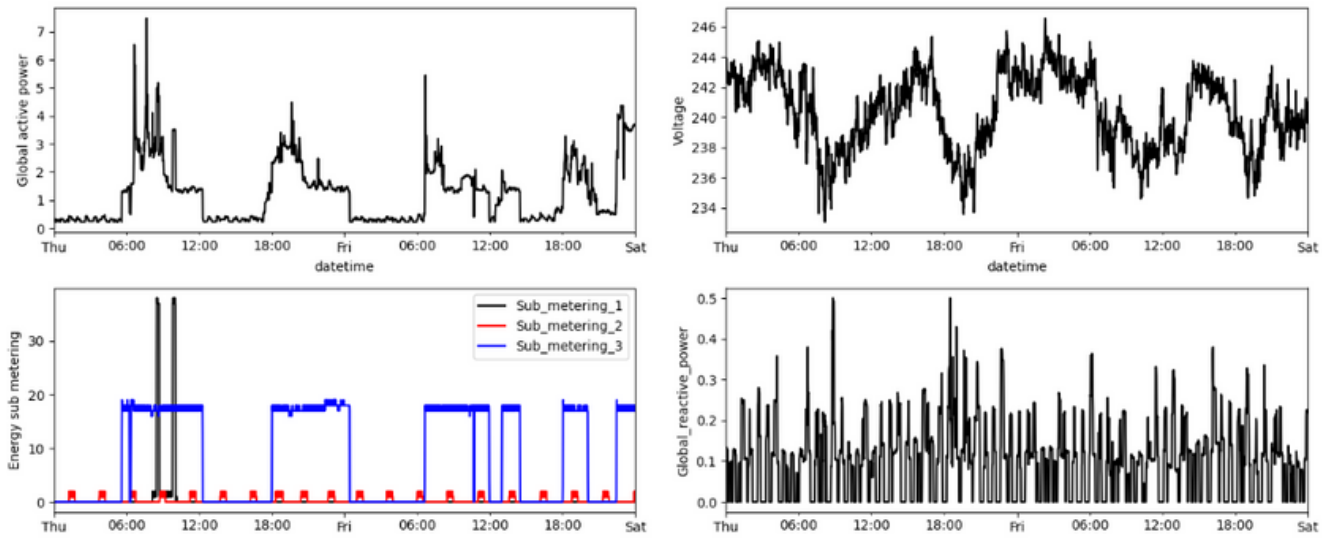
ELECTRIC POWER CONSUMPTION

5. Plot 4

01 Analysis and definition for the requirement

- **Global active power** is the power used in the whole household (kilowatt unit)
- **Global reactive power** is also the electrical power used. However, if active power is the actual power consumed in the circuit, then **reactive power** is useless power that only runs between the source and the load (doesn't do any work).
- **Energy sub metering** is electrical energy (in watt/h)
- **Voltage** is the average voltage per minute

02 Display Plot4.png



03 Code demo and explanation

```
def plot4():
    fig,((ax1,ax2),(ax3,ax4)) = plt.subplots(2,2,figsize=(14,6))

    # Plot a
    data['Global_active_power'].plot(ax=ax1, color='black')
    ax1.set_xticklabels(['Thu','Fri','Sat'])
    ax1.set_ylabel('Global active power')

    # Plot b
    data['Voltage'].plot(ax=ax2, color='black')
    ax2.set_xticklabels(['Thu','Fri','Sat'])
    ax2.set_ylabel('Voltage')

    # Plot c
    data['Sub_metering_1'].plot(color='black',ax=ax3)
    data['Sub_metering_2'].plot(color='red',ax=ax3)
    data['Sub_metering_3'].plot(color='blue',ax=ax3)
    ax3.set_xticklabels(['Thu','Fri','Sat'])
    ax3.set_ylabel('Energy sub metering')
    ax3.legend()

    # Plot d
    data['Global_reactive_power'].plot(ax=ax4, color='black')
    ax4.set_xticklabels(['Thu','Fri','Sat'])
    ax4.set_ylabel('Global_reactive_power')
    plt.tight_layout()
    plt.savefig('Plot4.png')
    # pass
```

- **plt.tight_layout():** adjust the positions of all the chart automatically so that the charts do not overlap.
- The rest lines of code have the same meaning with all the definition I had explained above.

6. A story about the energy use of the household during the two days

To begin with **Plot 1, 2, 3**, we can find out that

Amount of electricity in the kitchen(Sub_metering_1)

The amount of electricity in the kitchen increased throughout two days but only peaked from 9 AM to 11 AM on Thursday, implying that each cooking period in this residence will cook enough food for two days.

Electricity in the laundry room(Sub_metering_2)

Over the course of both days, the electricity in the laundry room seemed to fluctuate. That's because the laundry facilities and refrigerator appear to be in use at all times (even late at night and early in the morning when everyone is sleeping).

Amount of electricity for water heaters,...(Sub_metering_3)

The amount of electricity used by water heaters and air conditioners changes, which may appear weird at first, but there are certain standards in place. The two gadgets will be used the most between 6 AM and 12 PM. The amount of electricity utilized varies very little, often between 15-20 sub-meters.

Now, we will take a deep look in **Plot 4** to have some comments

- Electrical energy is separated into three sections, and we can see that sub_metering_3 energy fluctuates similarly to the power capacity consumed in one day.
- Power consumption often peaks in the morning and evening, and wasted electricity appears to be spent at all hours of the day, although in little amounts.
- Last but not least, the average voltage per minute changes in the opposite direction, decreasing during the busiest times of the day and increasing during less busy times, such as the afternoon or late at night.

REFERENCES

Lecture slide by Dr. Bui Tien Len

Theory lesson every Friday of Dr. Bui Tien Len

Visualization Analysis & Design - Tamara Munzner

The Big Book of DashBoards

Pandas theory: http://pandas.pydata.org/pandas-docs/stable/

Matplotlib theory: https://matplotlib.org/2.0.2/gallery.html

Thank You Teacher!!!

Because of all the knowledge you delivered to us