Data-X Spring 2019: Homework 04

Name: Shun Lin

SID: 26636176

In this homework, you will do some exercises with plotting.

REMEMBER TO DISPLAY ALL OUTPUTS. If the question asks you to do something, make sure to print your results.

1.

Data:

Data Source: Data file is uploaded to bCourses and is named: **Energy.csv**

The dataset was created by Angeliki Xifara (Civil/Structural Engineer) and was processed by Athanasios Tsanas, Oxford Centre for Industrial and Applied Mathematics, University of Oxford, UK).

Data Description:

The dataset contains eight attributes of a building (or features, denoted by X1...X8) and response being the heating load on the building, y1.

- X1 Relative Compactness
- X2 Surface Area
- X3 Wall Area
- X4 Roof Area
- X5 Overall Height
- X6 Orientation
- X7 Glazing Area
- X8 Glazing Area Distribution
- y1 Heating Load

Q1.1

Read the data file in python. Check if there are any NaN values, and print the results.

In [1]:
```
# your code

# Load required modules
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

df=pd.read_csv('Energy.csv')

sum_of_nans = sum(len(df) - df.count())
print("There are " + str(sum_of_nans) + " Nan values in the dataframe")
df.head()
```

```
There are 0 Nan values in the dataframe
```

Out[1]:

	X1	X2	X3	X4	X5	X6	X7	X8	Y1
0	0.98	514.5	294.0	110.25	7.0	2	0.0	0	15.55
1	0.98	514.5	294.0	110.25	7.0	3	0.0	0	15.55
2	0.98	514.5	294.0	110.25	7.0	4	0.0	0	15.55
3	0.98	514.5	294.0	110.25	7.0	5	0.0	0	15.55
4	0.90	563.5	318.5	122.50	7.0	2	0.0	0	20.84

Q 1.2

Describe (using python function) data features in terms of type, distribution range (max and min), and mean values.

In [2]:
```
# your code
df.describe(include='all')
```

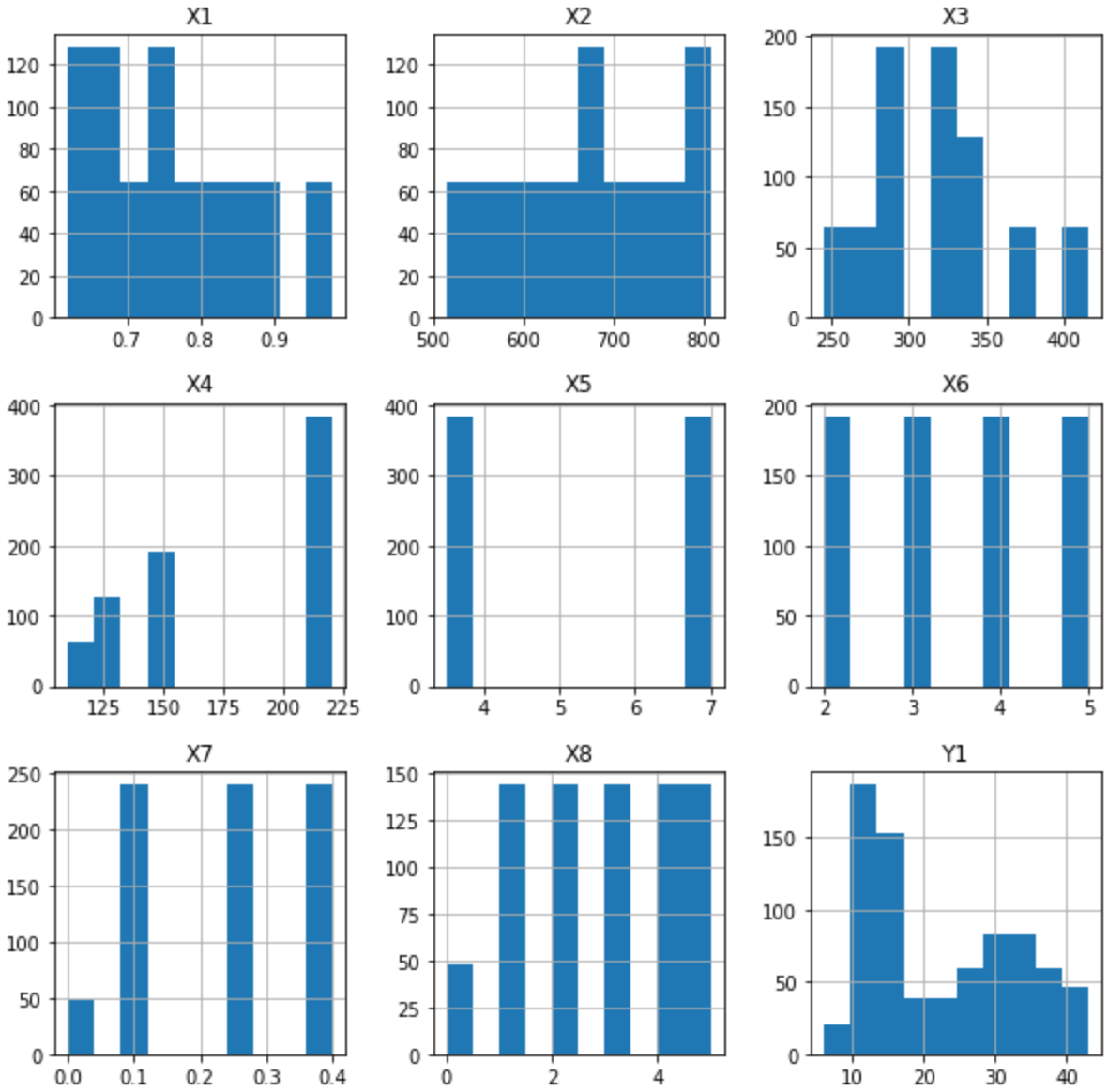
Out[2]:

	X1	X2	X3	X4	X5	X6	X7	
count	768.000000	768.000000	768.000000	768.000000	768.00000	768.000000	768.000000	768.0
mean	0.764167	671.708333	318.500000	176.604167	5.25000	3.500000	0.234375	2.8
std	0.105777	88.086116	43.626481	45.165950	1.75114	1.118763	0.133221	1.5
min	0.620000	514.500000	245.000000	110.250000	3.50000	2.000000	0.000000	0.0
25%	0.682500	606.375000	294.000000	140.875000	3.50000	2.750000	0.100000	1.7
50%	0.750000	673.750000	318.500000	183.750000	5.25000	3.500000	0.250000	3.0
75%	0.830000	741.125000	343.000000	220.500000	7.00000	4.250000	0.400000	4.0
max	0.980000	808.500000	416.500000	220.500000	7.00000	5.000000	0.400000	5.0

Q 1.3

Plot feature distributions for all the attributes in the dataset (Hint - Histograms are one way to plot data distributions). This step should give you clues about data sufficiency.

In [3]:
```
# your code
plot1_3 = df.hist(figsize=(10, 10))
```

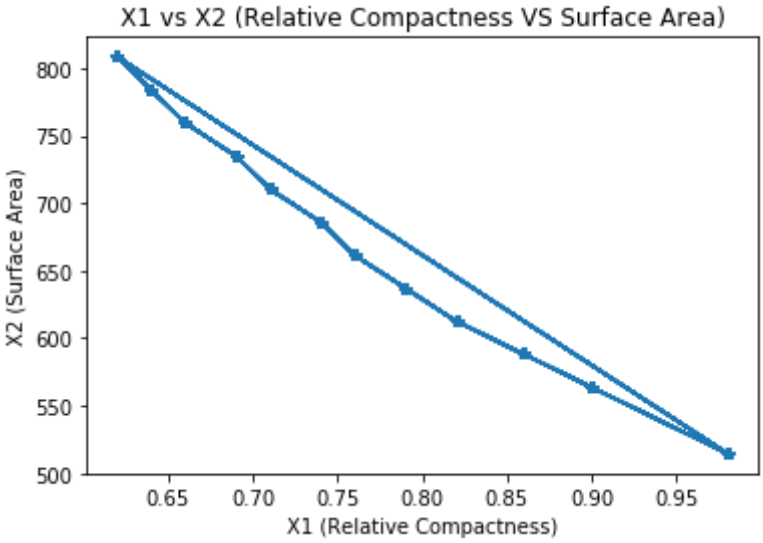


Q1.4

Create a combined line and scatter plot for attributes 'X1' and 'X2' with a marker (*). You can choose either of the attributes as x & y. Label your axes and give a title to your plot.

In [4]:
```
# your code
plot1_4 = df.plot(x='X1', y='X2', style='*-', title="X1 vs X2 (Relative
 Compactness VS Surface Area)", legend=False)
plot1_4.set_xlabel("X1 (Relative Compactness)")
plot1_4.set_ylabel("X2 (Surface Area)")
```

Out[4]:
```
Text(0, 0.5, 'X2 (Surface Area)')
```

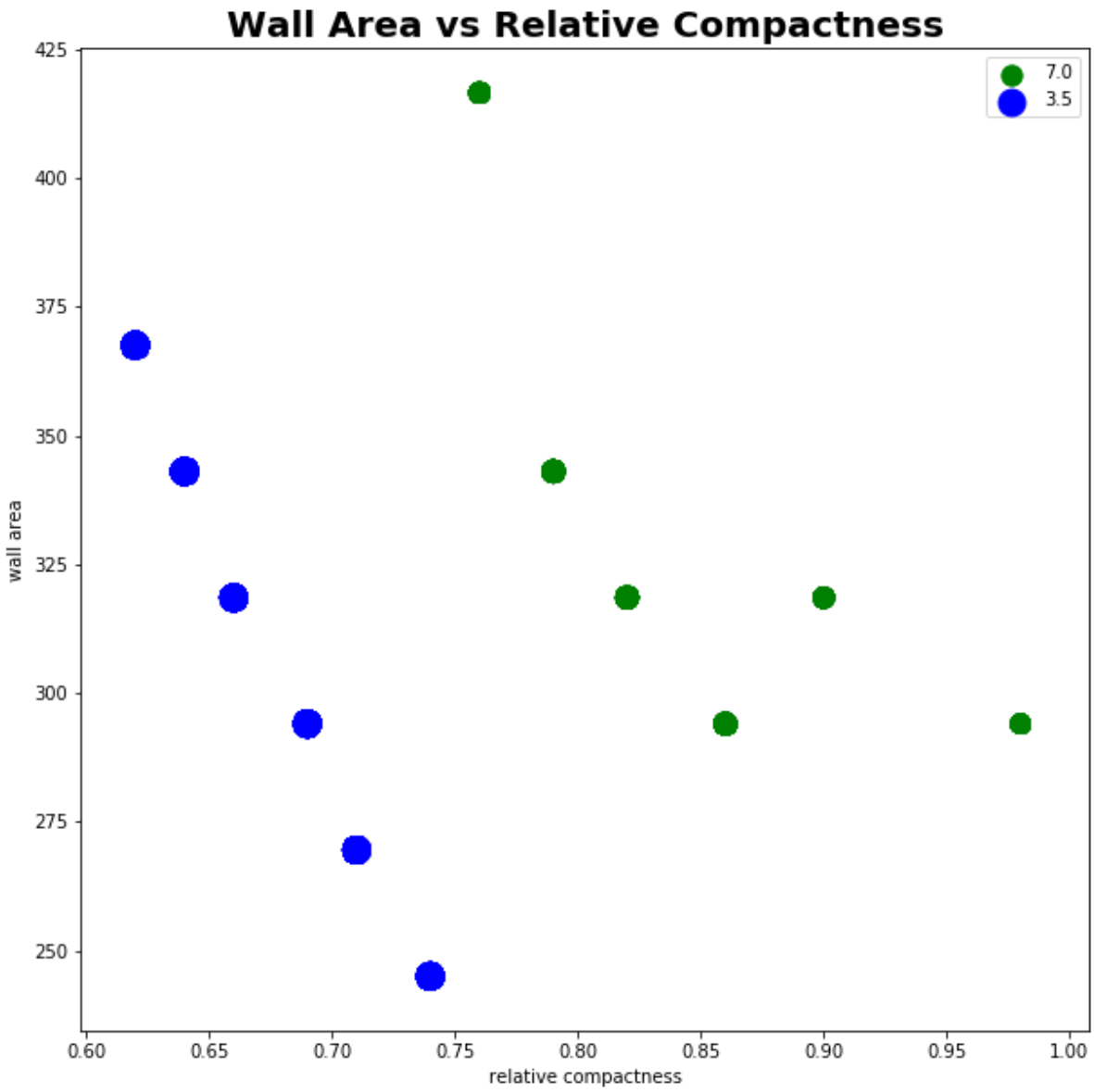


Q1.5

Create a scatter plot for how 'Wall Area' changes with 'Relative Compactness'. Give different colors for different 'Overall Height' and different bubble sizes by 'Roof Area'. Label the axes and give a title. Add a legend to your plot.

In [5]:
```
# your code
wall_area = df["X3"]
relative_compactness = df["X1"]
overall_height = df["X5"]
max_overall_height = max(overall_height)
roof_area = df["X4"]
# create plots
f, ax = plt.subplots(figsize=(10, 10))
colors_list = ["green", "blue"]
lines_map = ()
heights_map = ()
for i, height in enumerate(overall_height.unique()):
    color = colors_list[i]
    x = df[overall_height == height]["X1"]
    y = df[overall_height == height]["X3"]
    sizes = df[overall_height == height]["X4"]
    lines_map = lines_map + (ax.scatter(x, y, c=color, s=sizes),)
    heights_map = heights_map + (height,)
ax.set_ylabel("wall area")
ax.set_xlabel("relative compactness")
ax.set_title("Wall Area vs Relative Compactness", fontsize=20, fontweigh
t="bold")
ax.legend(lines_map, heights_map, loc="best")
```

Out[5]: <matplotlib.legend.Legend at 0x1a211f3ba8>



2.

Q 2.1a.

Create a dataframe called `icecream` that has column `Flavor` with entries `Strawberry`, `Vanilla`, and `Chocolate` and another column with `Price` with entries `3.50`, `3.00`, and `4.25`. Print the dataframe.

In [6]:
```
# your code
icecream = pd.DataFrame([["Strawberry", 3.5], ["Vanilla", 3.00],["Chocol
ate", 4.25]], columns=['Flavor', 'Price'])
print(icecream)
```

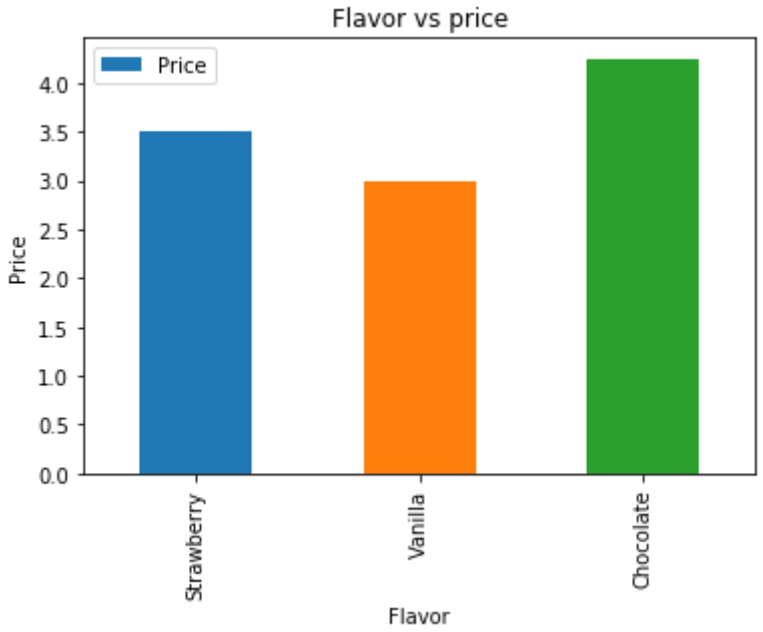
```
       Flavor  Price
0  Strawberry   3.50
1     Vanilla   3.00
2   Chocolate   4.25
```

Q 2.1b

Create a bar chart representing the three flavors and their associated prices. Label the axes and give a title.

In [7]:
```
# your code
plot2_1b = icecream.plot.bar(x="Flavor", y="Price", title="Flavor vs pri
ce")
plot2_1b.set_ylabel("Price")
```

Out[7]: Text(0, 0.5, 'Price')



Q 2.2

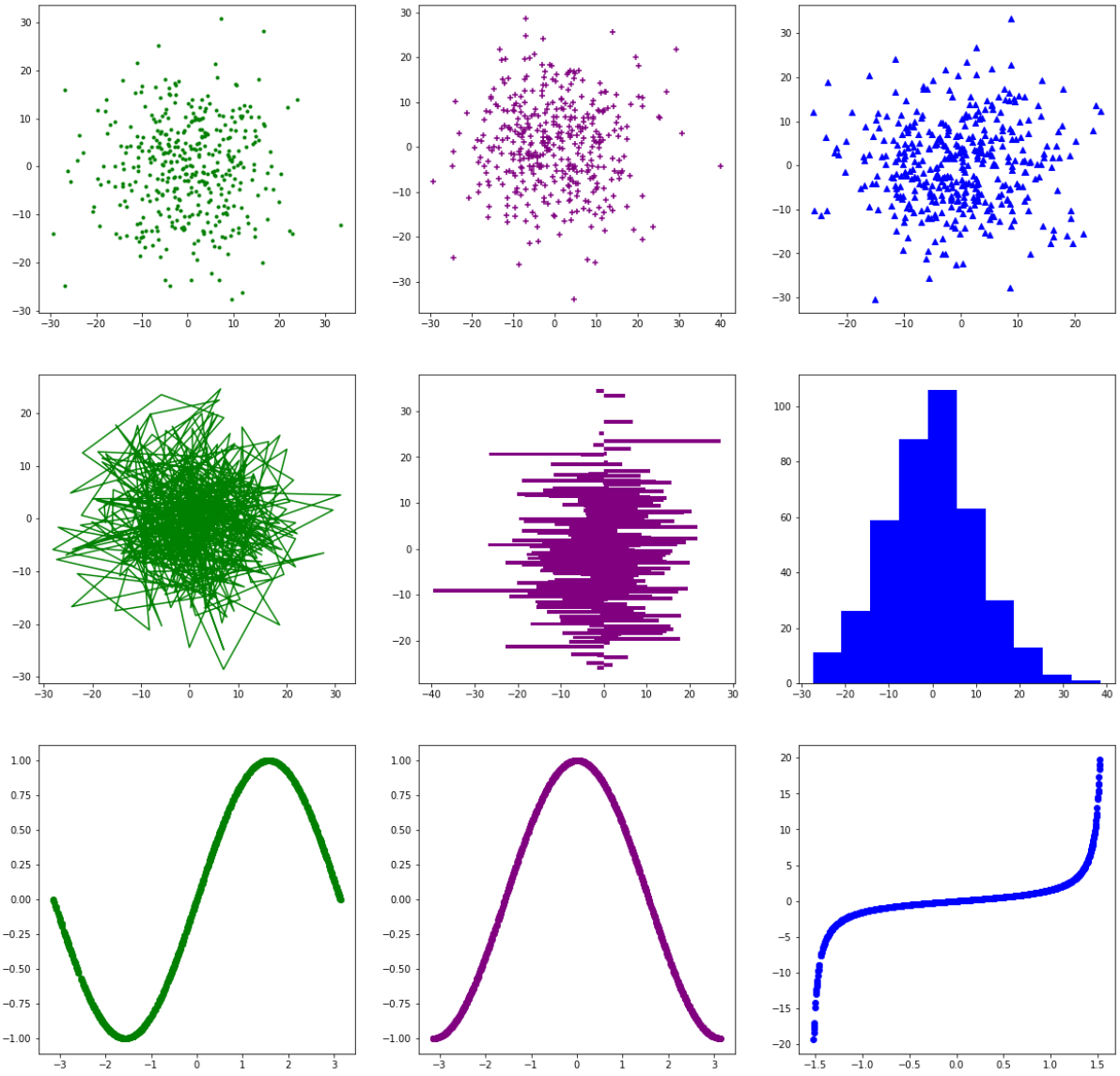
Create 9 random plots in a figure (Hint: There is a numpy function for generating random data).

The top three should be scatter plots (one with green dots, one with purple crosses, and one with blue triangles. The middle three graphs should be a line graph, a horizontal bar chart, and a histogram. The bottom three graphs should be trignometric functions (one sin, one cosine, one tangent). Keep in mind the range and conditions for the trignometric functions.

All these plots should be on the same figure and not 9 independent figures.

In [8]:
```
# your code

# generate different random data for each plot
random_data = {}
for i in range(1, 10):
    random_x = 10 * np.random.normal(size=400)
    if i >= 7:
        random_x = np.random.uniform(-np.pi, np.pi, 1000)
        if i == 7:
            random_y = np.sin(random_x)
        elif i == 8:
            random_y = np.cos(random_x)
        elif i == 9:
            eps = 0.05
            random_x = np.random.uniform(-np.pi/2+eps, np.pi/2 - eps, 10
00)
            random_y = np.tan(random_x)
    else:
        random_y = 10 * np.random.normal(size=400)
    random_data[i] = {}
    random_data[i]["x"] = random_x
    random_data[i]["y"] = random_y

# create plots
f, ax = plt.subplots(nrows=3,ncols=3, figsize=(20, 20))

# top three plots
ax[0,0].scatter(random_data[1]["x"], random_data[1]["y"], marker=".", c=
"green")
ax[0,1].scatter(random_data[2]["x"], random_data[2]["y"], marker="+", c=
"purple")
ax[0,2].scatter(random_data[3]["x"], random_data[3]["y"], marker="^", c=
"blue")

# middle three plots
ax[1,0].plot(random_data[4]["x"], random_data[4]["y"], color="green")
ax[1,1].barh(random_data[5]["x"], random_data[5]["y"], color="purple")
ax[1,2].hist(random_data[6]["x"], color="blue")

# bottom three plots
# sin plot
ax[2,0].scatter(random_data[7]["x"], random_data[7]["y"], c="green")
# cos plot
ax[2,1].scatter(random_data[8]["x"], random_data[8]["y"], c = "purple")
# tangent plot
ax[2,2].scatter(random_data[9]["x"], random_data[9]["y"], c="blue")
```

Out[8]: <matplotlib.collections.PathCollection at 0x1a21f769b0>



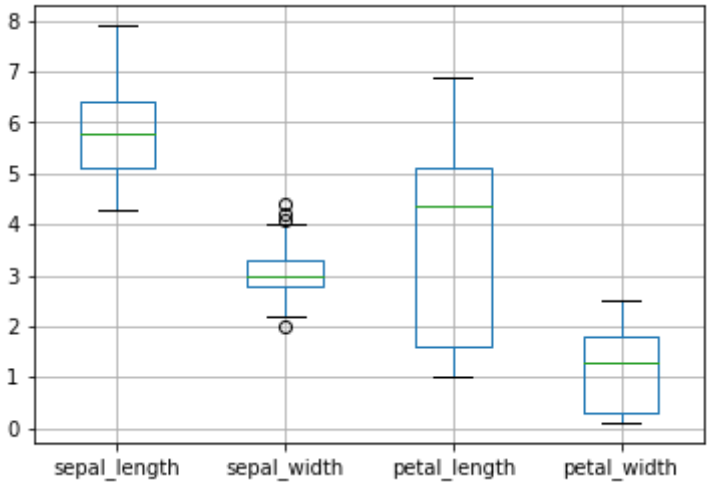
3.

Q 3.1

Load the 'Iris' dataset using seaborn. Create a box plot for the attributes 'sepal_length', sepal_width', 'petal_length' and 'petal_width' in the Iris dataset.

In [9]:
```
# your code
iris = sns.load_dataset("iris")
iris.boxplot(column=["sepal_length", "sepal_width", "petal_length", "petal_width"])
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a228641d0>
```



Q 3.2

In a few sentences explain what can you interpret from the above box plot.

In [10]:
```
solution_text = "The box plot above shows that there are some outliers from the sepal_width column, that the mean for sepal_length is the highest while the mean for petal_width is the smallest. We can also see that sepal_width has less spread (values are closer to one another), while petal_length has the most spread and values varies from one another. From those box plots we can also see that sepal_length has the highest value while petal_width has the lowest value."
print(solution_text)
```

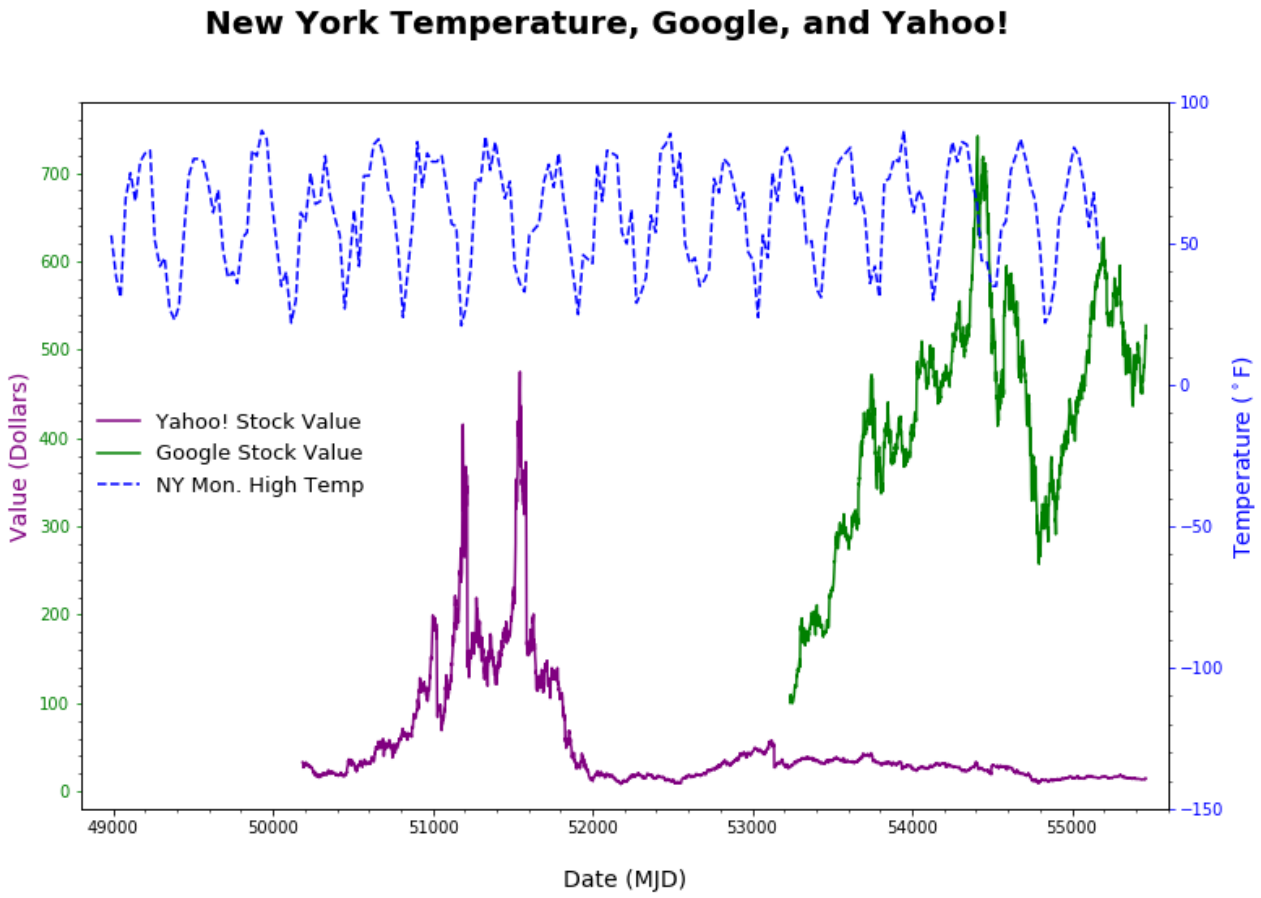
```
The box plot above shows that there are some outliers from the sepal_width column, that the mean for sepal_length is the highest while the mean for petal_width is the smallest. We can also see that sepal_width has less spread (values are closer to one another), while petal_length has the most spread and values varies from one another. From those box plots we can also see that sepal_length has the highest value while petal_width has the lowest value.
```

Q 4.

The data files needed:

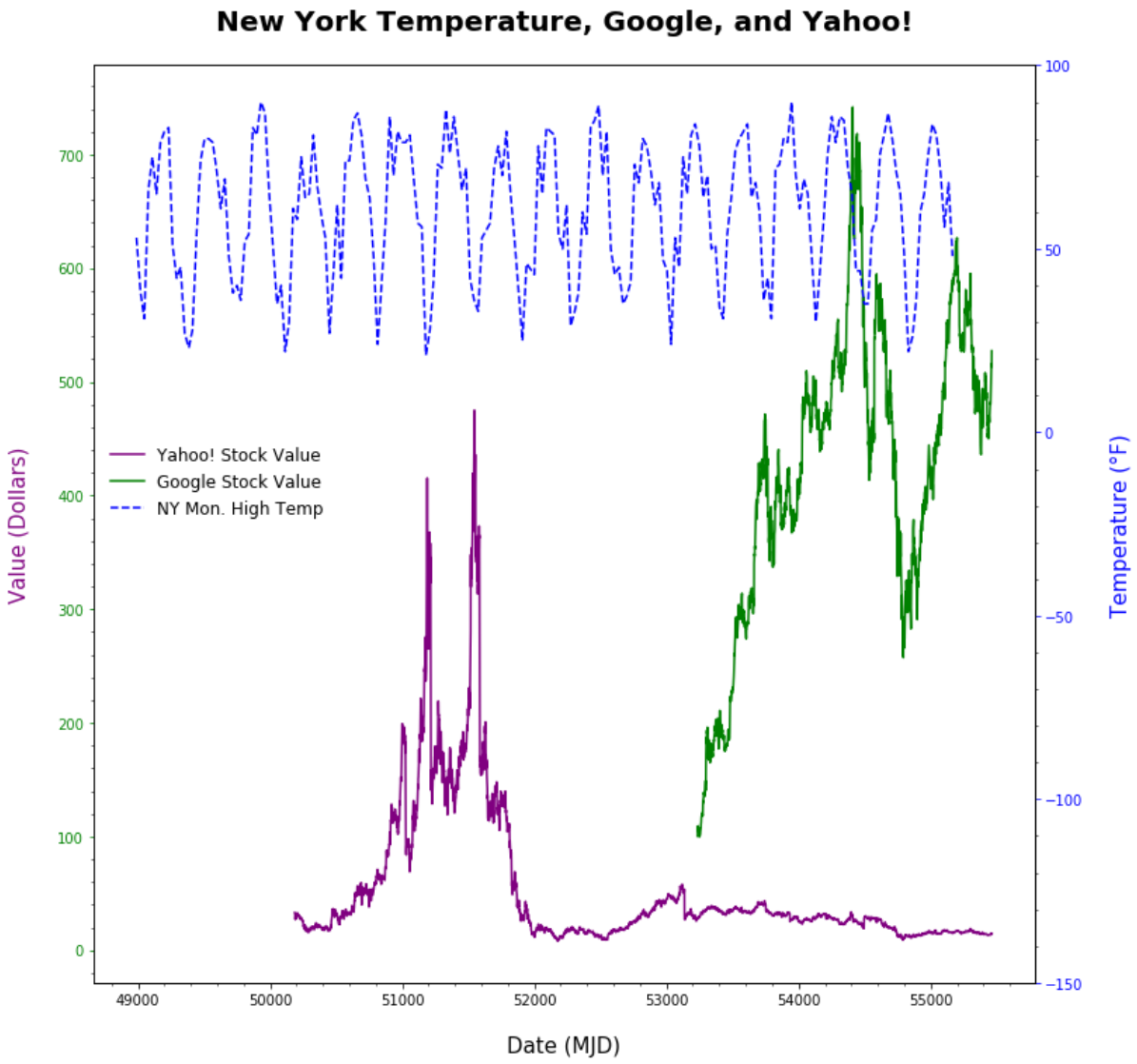
`google_data.txt, ny_temps.txt & yahoo_data.txt`

Use your knowledge with `Python, NumPy, pandas and matplotlib` to reproduce the plot below:



In [11]:
```
# your code

# import data
google_data = pd.read_csv('google_data.txt',sep="\t")
yahoo_data = pd.read_csv('yahoo_data.txt',sep="\t")
ny_temps = pd.read_csv('ny_temps.txt',sep='\t')
```

In [12]:
```
f3, ax3 = plt.subplots(figsize=(12, 12))

# plot yahoo and google
l1, = ax3.plot(yahoo_data["Modified Julian Date"], yahoo_data["Stock Val
ue"], color="purple", label='Yahoo! Stock Value')
l2, = ax3.plot(google_data["Modified Julian Date"], google_data["Stock V
alue"], color="green", label="Google Stock Value")

# set left y label
ax3.set_ylabel("Value (Dollars)",color="purple", labelpad=20, fontsize=1
5)

# set x label and size
ax3.set_xlabel("Date (MJD)", labelpad=20, fontsize=15)

# set title
ax3.set_title("New York Temperature, Google, and Yahoo!", pad=25, fontsi
ze=20, fontweight="bold")

# turn on left minor tick
plt.minorticks_on()

# create right axis for ny temperature
ax4 = ax3.twinx()

# plot ny_temperature
l3, = ax4.plot(ny_temps["Modified Julian Date"], ny_temps["Max Temperatu
re"], "--", color="blue", label="NY Mon. High Temp")

# set right y label
ax4.set_ylabel("Temperature (°F)",color="blue", labelpad=20, fontsize=15)

# set left and right ticks color
ax3.tick_params("y", colors="green")
ax4.tick_params("y", colors="blue")

# set lengends
plt.legend((l1,l2, l3), ('Yahoo! Stock Value', "Google Stock Value", "NY
 Mon. High Temp"), loc=[0.01,0.5], frameon=False, prop={'size': 12})

# set right y axis tick range
plt.yticks(np.arange(-150, 150, 50))

# turn on right minor ticks
plt.minorticks_on()
```


New York Temperature, Google, and Yahoo!
Yahoo! Stock Value
Google Stock Value
NY Mon. High Temp
Value (Dollars)
Temperature (°F)
Date (MJD)
0
100
200
300
400
500
600
700
−150
−100
−50
0
50
100
49000
50000
51000
52000
53000
54000
55000

In [ ]: