Data-X Spring 2019: Homework 04

Name: Jack Xie

SID: 3032163590

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]:
```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib as mpl
import matplotlib.pyplot as plt  # always import pyplot module as plt (s
tandard)
import seaborn as sns
%matplotlib inline
```

In [2]:
```
df = pd.read_csv('Energy.csv')
df
```

Out[2]:

	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
5	0.90	563.5	318.5	122.50	7.0	3	0.0	0	21.46
6	0.90	563.5	318.5	122.50	7.0	4	0.0	0	20.71
7	0.90	563.5	318.5	122.50	7.0	5	0.0	0	19.68
8	0.86	588.0	294.0	147.00	7.0	2	0.0	0	19.50
9	0.86	588.0	294.0	147.00	7.0	3	0.0	0	19.95
10	0.86	588.0	294.0	147.00	7.0	4	0.0	0	19.34
11	0.86	588.0	294.0	147.00	7.0	5	0.0	0	18.31
12	0.82	612.5	318.5	147.00	7.0	2	0.0	0	17.05
13	0.82	612.5	318.5	147.00	7.0	3	0.0	0	17.41
14	0.82	612.5	318.5	147.00	7.0	4	0.0	0	16.95
15	0.82	612.5	318.5	147.00	7.0	5	0.0	0	15.98
16	0.79	637.0	343.0	147.00	7.0	2	0.0	0	28.52
17	0.79	637.0	343.0	147.00	7.0	3	0.0	0	29.90
18	0.79	637.0	343.0	147.00	7.0	4	0.0	0	29.63
19	0.79	637.0	343.0	147.00	7.0	5	0.0	0	28.75
20	0.76	661.5	416.5	122.50	7.0	2	0.0	0	24.77
21	0.76	661.5	416.5	122.50	7.0	3	0.0	0	23.93
22	0.76	661.5	416.5	122.50	7.0	4	0.0	0	24.77
23	0.76	661.5	416.5	122.50	7.0	5	0.0	0	23.93
24	0.74	686.0	245.0	220.50	3.5	2	0.0	0	6.07
25	0.74	686.0	245.0	220.50	3.5	3	0.0	0	6.05
26	0.74	686.0	245.0	220.50	3.5	4	0.0	0	6.01
27	0.74	686.0	245.0	220.50	3.5	5	0.0	0	6.04
28	0.71	710.5	269.5	220.50	3.5	2	0.0	0	6.37
29	0.71	710.5	269.5	220.50	3.5	3	0.0	0	6.40
...	...	...	...	...	...	...	...	...	...
738	0.79	637.0	343.0	147.00	7.0	4	0.4	5	41.09
739	0.79	637.0	343.0	147.00	7.0	5	0.4	5	40.79
740	0.76	661.5	416.5	122.50	7.0	2	0.4	5	38.82

	X1	X2	X3	X4	X5	X6	X7	X8	Y1
741	0.76	661.5	416.5	122.50	7.0	3	0.4	5	39.72
742	0.76	661.5	416.5	122.50	7.0	4	0.4	5	39.31
743	0.76	661.5	416.5	122.50	7.0	5	0.4	5	39.86
744	0.74	686.0	245.0	220.50	3.5	2	0.4	5	14.41
745	0.74	686.0	245.0	220.50	3.5	3	0.4	5	14.19
746	0.74	686.0	245.0	220.50	3.5	4	0.4	5	14.17
747	0.74	686.0	245.0	220.50	3.5	5	0.4	5	14.39
748	0.71	710.5	269.5	220.50	3.5	2	0.4	5	12.43
749	0.71	710.5	269.5	220.50	3.5	3	0.4	5	12.63
750	0.71	710.5	269.5	220.50	3.5	4	0.4	5	12.76
751	0.71	710.5	269.5	220.50	3.5	5	0.4	5	12.42
752	0.69	735.0	294.0	220.50	3.5	2	0.4	5	14.12
753	0.69	735.0	294.0	220.50	3.5	3	0.4	5	14.28
754	0.69	735.0	294.0	220.50	3.5	4	0.4	5	14.37
755	0.69	735.0	294.0	220.50	3.5	5	0.4	5	14.21
756	0.66	759.5	318.5	220.50	3.5	2	0.4	5	14.96
757	0.66	759.5	318.5	220.50	3.5	3	0.4	5	14.92
758	0.66	759.5	318.5	220.50	3.5	4	0.4	5	14.92
759	0.66	759.5	318.5	220.50	3.5	5	0.4	5	15.16
760	0.64	784.0	343.0	220.50	3.5	2	0.4	5	17.69
761	0.64	784.0	343.0	220.50	3.5	3	0.4	5	18.19
762	0.64	784.0	343.0	220.50	3.5	4	0.4	5	18.16
763	0.64	784.0	343.0	220.50	3.5	5	0.4	5	17.88
764	0.62	808.5	367.5	220.50	3.5	2	0.4	5	16.54
765	0.62	808.5	367.5	220.50	3.5	3	0.4	5	16.44
766	0.62	808.5	367.5	220.50	3.5	4	0.4	5	16.48
767	0.62	808.5	367.5	220.50	3.5	5	0.4	5	16.64

768 rows × 9 columns

Q 1.2

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

In [3]:
```
df.describe()
```

Out[3]:

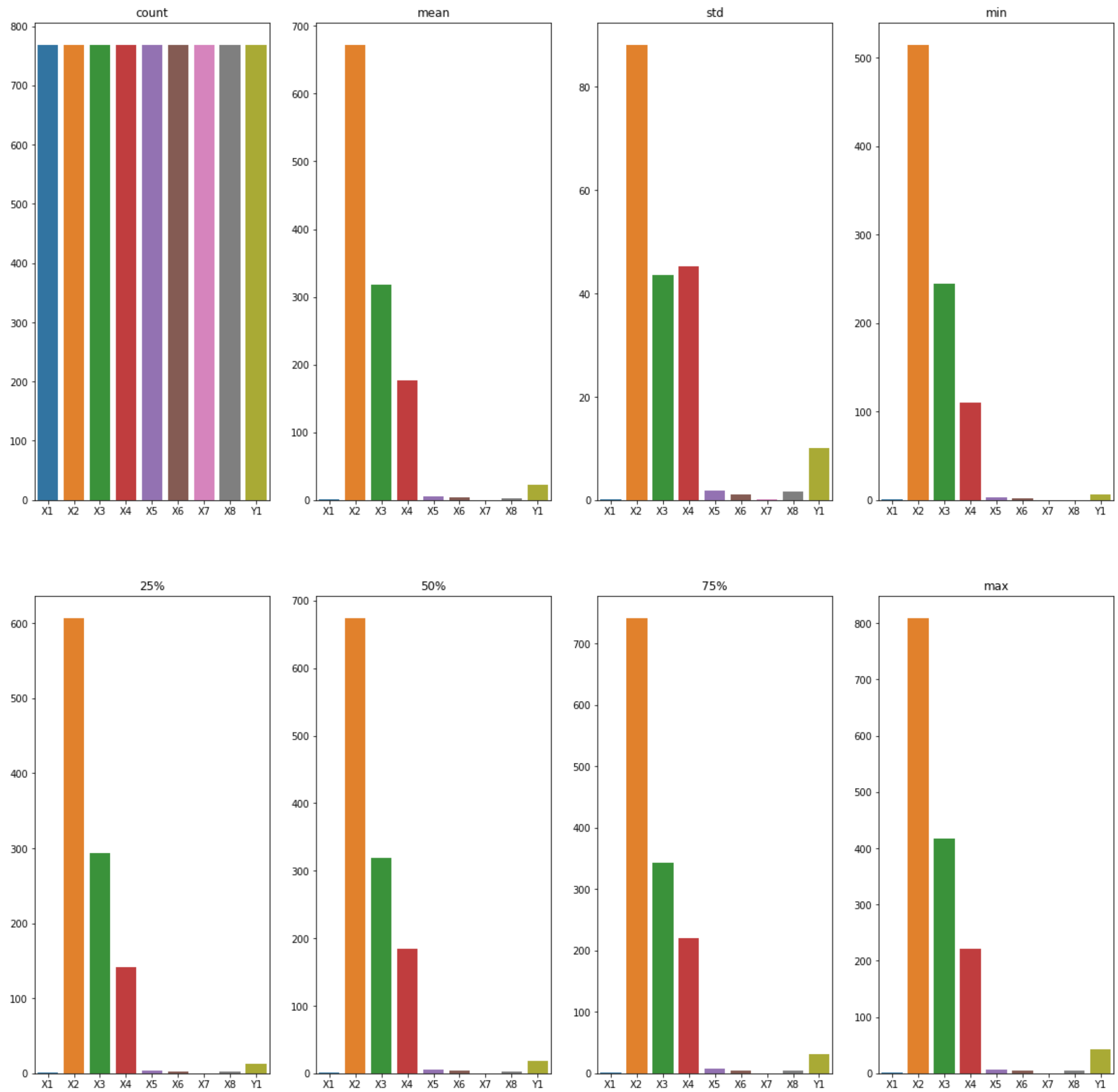
	X1	X2	X3	X4	X5	X6	X7	
count	768.000000	768.000000	768.000000	768.000000	768.00000	768.000000	768.000000	768.0
mean	0.764167	671.708333	318.500000	176.604167	5.25000	3.500000	0.234375	2.8
std	0.105777	88.086116	43.626481	45.165950	1.75114	1.118763	0.133221	1.5
min	0.620000	514.500000	245.000000	110.250000	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 [4]:
```
IPython_default = plt.rcParams.copy() # save default styling
```

In [5]:
```python
a, b, c = 0, 0, 0
fig, ax = plt.subplots(2, 4, figsize=(20, 20))
for i in [df.describe().loc[i].tolist() for i in df.describe().index]:
    sns.barplot(y=i, x=df.columns, ax=ax[a, b]).set_title(df.describe().
index[c])
    c+=1
    if b > 2:
        a, b = 1, 0
    else:
        b+=1
```

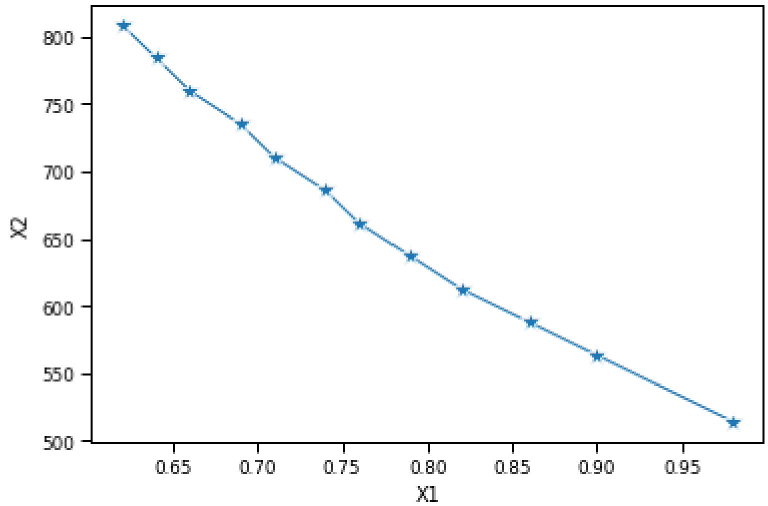


Q1.4

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 [7]:
```
x1, x2 = df['X1'], df['X2']
sns.lineplot(x=x1, y=x2, marker='*')
paper_rc = {'lines.linewidth': 1, 'lines.markersize': 10}

sns.set_context("paper", rc = paper_rc)
plt.show()
```



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.

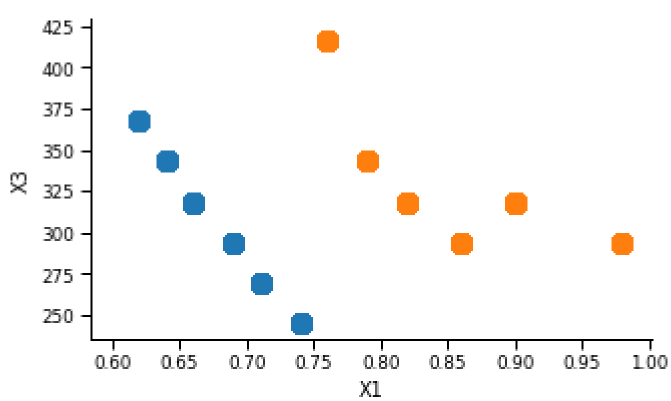
In [72]:
```
a=df['X5'].astype('int64')
```

In [34]:
```
df.groupby(df['X5']).count().index
```

Out[34]:
```
Float64Index([3.5, 7.0], dtype='float64', name='X5')
```

In [74]:
```
fg = sns.FacetGrid(data=df, hue='X5',aspect=1.61)
fg.map(plt.scatter, 'X1', 'X3')
```

Out[74]:
```
<seaborn.axisgrid.FacetGrid at 0x1296a08d0>
```



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 [9]:
```
icecream = pd.DataFrame({'Flavor':['Strawberry', 'Vanilla', 'Chocolate'
], 'Price':[3.50, 3.00, 4.25]})
icecream
```

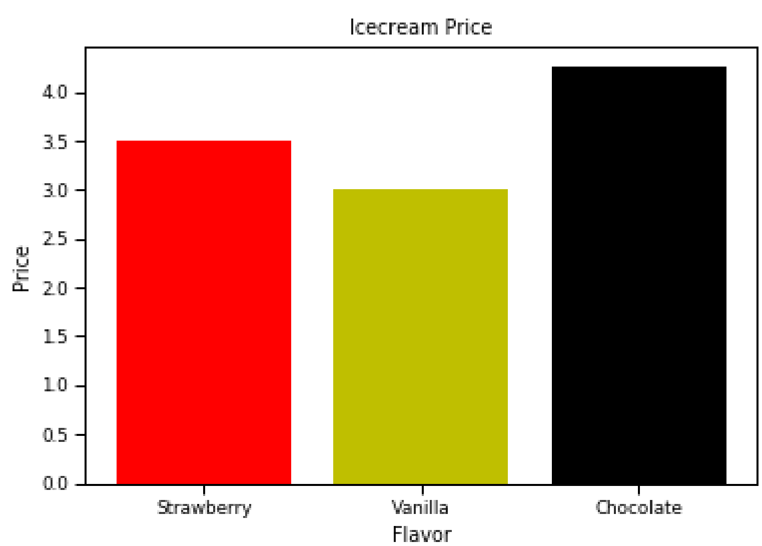
Out[9]:

	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 [10]:
```
ax  = plt.gca()
ax.bar(data=icecream, x='Flavor', height='Price', color='ryk')
ax.set_title('Icecream Price')
ax.set_xlabel('Flavor')
ax.set_ylabel('Price')
plt.show()
```



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 [11]:
```
IPython_default = plt.rcParams.copy() # save default styling
```

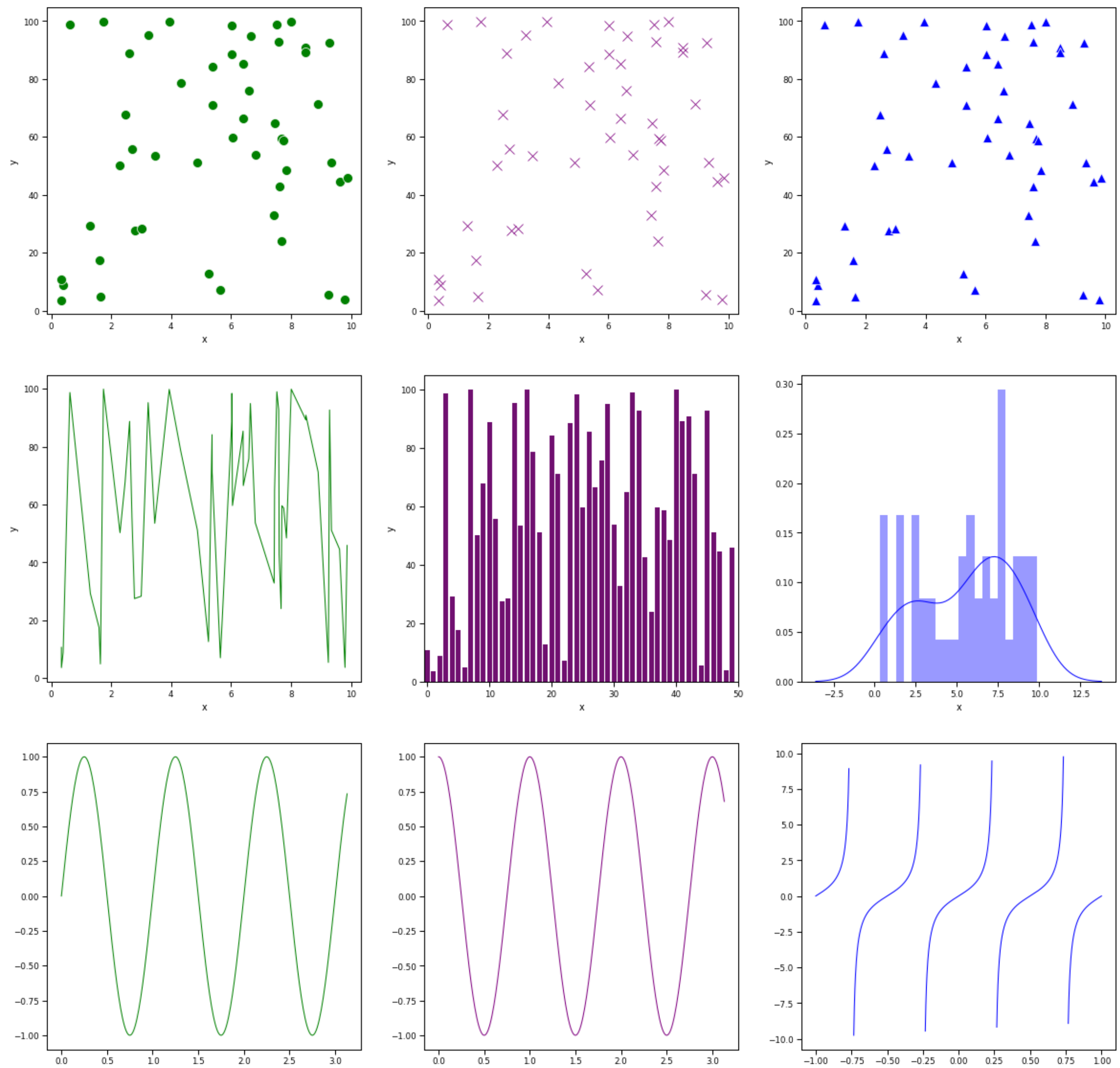
In [12]:
```
data = pd.DataFrame({'x' : np.random.random(50)*10, 'y' :
np.random.random(50)*100})
```

In [13]:
```
a1, b1, c1 = 0, 0, 0
sine = np.sin(data['x'])
cos = np.cos(data['x'])
tan = np.tan(data['x'])
fig, ax = plt.subplots(3, 3, figsize=(20, 20))
sns.scatterplot(data=data, x='x', y='y', ax=ax[0, 0], color='g')
sns.scatterplot(data=data, x='x', y='y', ax=ax[0, 1], color='purple', ma
rker='x')
sns.scatterplot(data=data, x='x', y='y', ax=ax[0, 2], color='b', marker=
'^')
sns.lineplot(data=data, x='x', y='y', ax=ax[1, 0], color='g')
sns.barplot(data=data, x='x', y='y', ax=ax[1, 1], color='purple')
plt.sca(ax[1, 1])
plt.xticks(np.arange(0, 55, 10), labels=np.arange(0, 55, 10))
sns.distplot(data['x'], ax=ax[1, 2], color='b', bins=20)
plt.sca(ax[2, 0])
s = np.arange(0.0, np.pi, np.pi/300)
plt.plot(s, np.sin(np.pi *2*s),'g')
plt.sca(ax[2, 1])
plt.plot(s, np.cos(np.pi *2*s),'purple')
plt.sca(ax[2, 2])
a = np.linspace(-1.0, 1.0, 1000)
t = (np.sin(2 * np.pi * a)) / (np.cos(2 * np.pi * a))
t[t > 10] = np.nan
t[t < -10] = np.nan
plt.plot(a, t, 'b-', lw=1)
```

```
/Users/jackxie/anaconda3/lib/python3.6/site-packages/ipykernel_launche
r.py:23: RuntimeWarning: invalid value encountered in less
```

Out[13]:

```
[<matplotlib.lines.Line2D at 0x125dd6f98>]
```



3.

Q 3.1

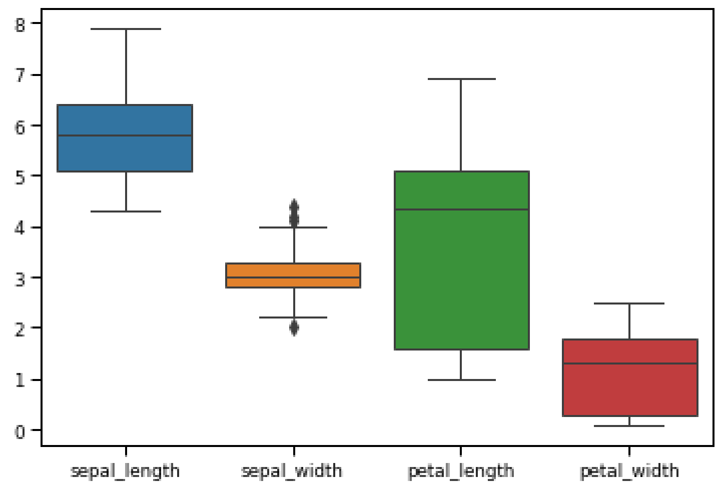
Load the 'Iris' dataset using seaborn. Create a box plot for the attributes 'sepal_length', sepal_width', 'petal_length' and 'petal_width' in the Iris dataset.

In [14]:
```
iris = sns.load_dataset('iris')
iris.head()
```

Out[14]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [15]:
```
ax = sns.boxplot(data=iris)
```



Q 3.2

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

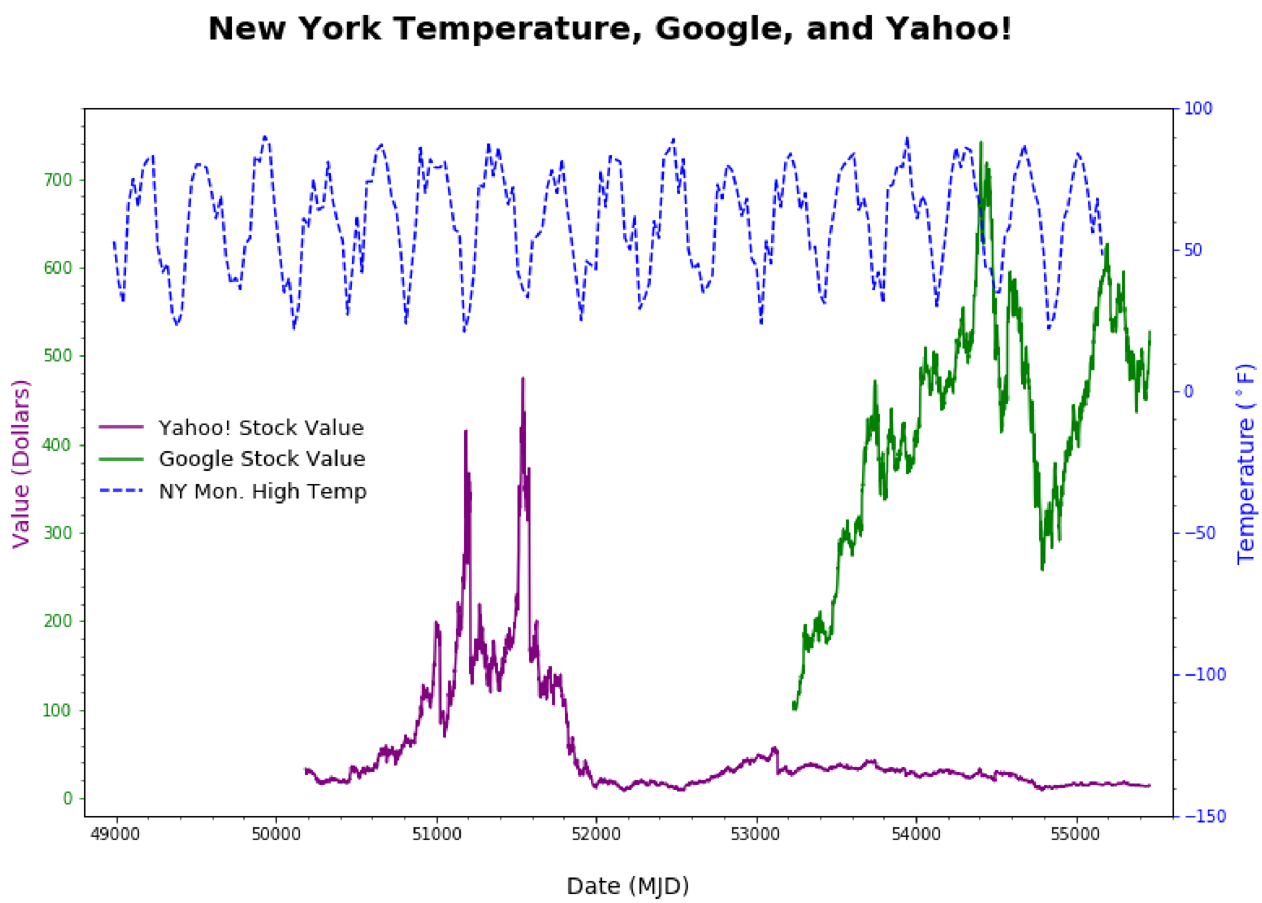
In gernal, sepal is larger in length and width than petal. But petal has more diversed range of the length and width than sepal.

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 [16]:
```
gg = pd.read_csv('google_data.txt')
yh = pd.read_csv('yahoo_data.txt')
ny = pd.read_csv('ny_temps.txt')
```

In [17]:
```
gg['Date (MJD)'] = gg['Modified Julian Date\tStock Value'].str.slice(sto
p=5).astype('float64')
gg['Value (Dollars)'] = gg['Modified Julian Date\tStock Value'].str.slic
e(6).astype('float64')
```

In [36]:
```
yh['Date (MJD)'] = yh['Modified Julian Date\tStock Value'].str.slice(stop=5).astype('float64')
yh['Value (Dollars)'] = yh['Modified Julian Date\tStock Value'].str.slice(6).astype('float64')
```

In [19]:
```
ny['Date (MJD)'] = ny['Modified Julian Date\tMax Temperature'].str.slice(stop=5).astype('float64')
ny['Temperature ($^\circ$F)'] = ny['Modified Julian Date\tMax Temperature'].str.slice(6).astype('float64')
```

In [20]:
```
fig, ax2 = plt.subplots(figsize=(15, 8))
ax2.title.set_position([.5, 1.08])
ax2.set_title('New York Temperature, Google, and Yahoo!',
              color='k',fontweight='bold',fontsize=20)
ax2.set_xlabel('Date (MJD)',color='k', fontsize=15)
ax2.set_ylabel('Value (Dollars)', color='purple', fontsize=15)
ax2.tick_params('y', colors='g')
ax2.set_yticks(np.arange(0, 800, 100), minor=True)
ax2.set_ylim(-50, 780)
ax2.set_xticks(np.arange(49000, 55400, 1000), minor=True)
ax2.set_xlim(48800, 55600)
sns.lineplot(x=yh['Date (MJD)'], y=yh['Value (Dollars)'], color='purple'
, linewidth=1.5, label='Yahoo! Stock Value')
sns.lineplot(x=gg['Date (MJD)'], y=gg['Value (Dollars)'], color='g', lin
ewidth=1.5, label='Google Stock Value')


ax3 = ax2.twinx()
sns.lineplot(x=ny['Date (MJD)'], y=ny['Temperature ($^\circ$F)'], color=
'b', linewidth=1.5, label='NY Mon. High Temp', legend=False)
ax3.lines[0].set_linestyle("--")
ax3.set_ylabel('Temperature ($^\circ$F)', fontsize=15, color='b')
ax3.tick_params('y', colors='b')
ax3.set_yticks(np.arange(-150, 140, 50), minor=True)

lines, labels = ax2.get_legend_handles_labels()
lines2, labels2 = ax3.get_legend_handles_labels()
ax2.legend(lines + lines2, labels + labels2, loc=6, prop={'size': 15}, f
rameon=False)
```

Out[20]: <matplotlib.legend.Legend at 0x125651908>