Data-X Spring 2019: Homework 04

Name: McClain Thiel

SID: 3034003600

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
import numpy as np
import pandas as pd
#import matplotlib.pyplot as plt

tb = pd.read_csv('Energy.csv')
print(tb.isnull().values.any())
tb.head()
```

```
False
```

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]:
```
tb.info()
tb.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
X1    768 non-null float64
X2    768 non-null float64
X3    768 non-null float64
X4    768 non-null float64
X5    768 non-null float64
X6    768 non-null int64
X7    768 non-null float64
X8    768 non-null int64
Y1    768 non-null float64
dtypes: float64(7), int64(2)
memory usage: 54.1 KB
```

Out[2]:

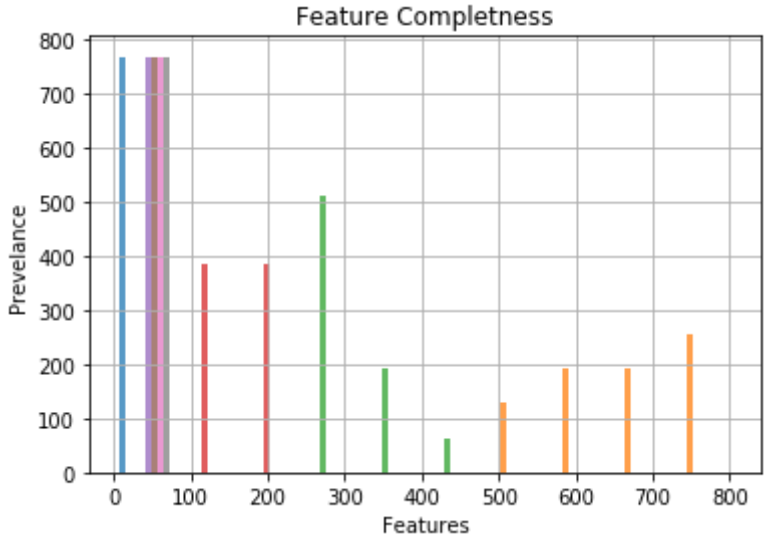
	X1	X2	X3	X4	X5	X6	X7	
count	768.000000	768.000000	768.000000	768.000000	768.00000	768.000000	768.000000	768.0
mean	0.764167	671.708333	318.500000	176.604167	5.25000	3.500000	0.234375	2.8
std	0.105777	88.086116	43.626481	45.165950	1.75114	1.118763	0.133221	1.5
min	0.620000	514.500000	245.000000	110.250000	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

In [3]:
```
# your code
```

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.

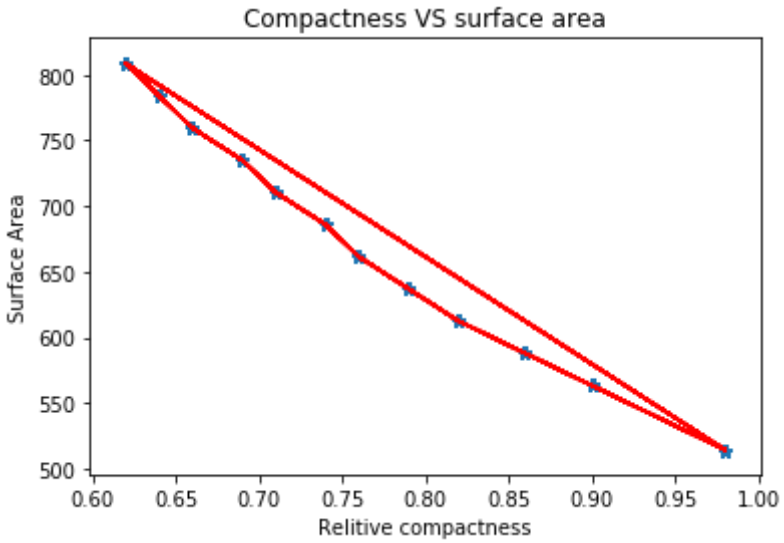
In [10]:
```
# your code
import matplotlib.pyplot as plt
plt.hist([tb['X1'],tb['X2'], tb['X3'], tb['X4'],tb['X5'],tb['X6'], tb['X
7'],tb['X8']],  alpha=0.75)
plt.xlabel('Features')
plt.ylabel('Prevelance')
plt.title('Feature Completness')
plt.grid(True)
plt.show()
```



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.

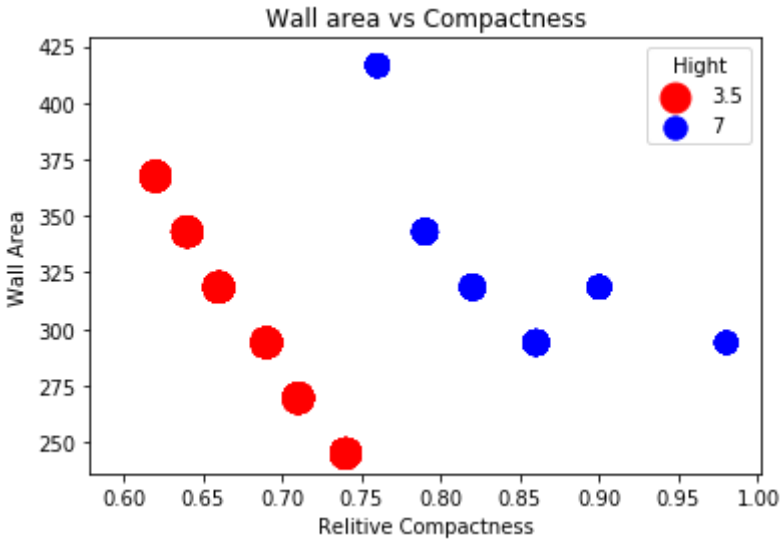
In [21]:
```
# your code
plt.close()
plt.plot(tb['X1'], tb['X2'], c='R')
plt.scatter(tb['X1'], tb['X2'],marker='*')
plt.xlabel('Relitive compactness')
plt.ylabel('Surface Area')
plt.title('Compactness VS surface area')
plt.show()
```



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 [56]:
```
# your code
plt.close()
#v = plt.scatter(tb['X1'], tb['X3'], s= tb['X4'], c=tb['X5'])
rc1 = tb['X1'].where(tb['X5'] == 3.5)
rc2 = tb['X1'].where(tb['X5'] == 7)
x = plt.scatter(rc1, tb['X3'], s = tb['X4'], c='R')
y = plt.scatter(rc2, tb['X3'], s = tb['X4'], c='B')
plt.xlabel('Relitive Compactness')
plt.ylabel('Wall Area')
plt.title('Wall area vs Compactness')
plt.legend((x,y),("3.5",'7'),title="Hight")
plt.show()
```



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 [70]:
```
# your code
temp = {"Flavor": ['Strawberry', 'Vanilla', 'Chocolate'],
        'Price': [3.50,3.00,4.25]}
icecream = pd.DataFrame(data=temp)
icecream
```

Out[70]:

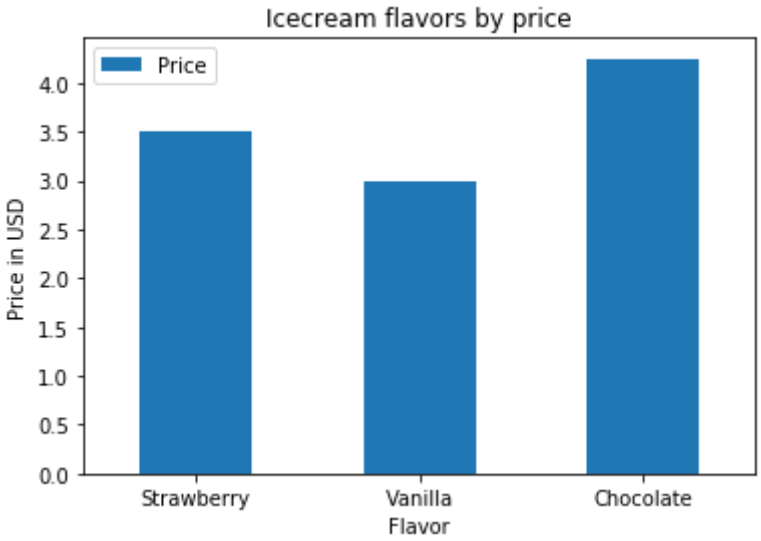
	Flavor	Price
0	Strawberry	3.50
1	Vanilla	3.00
2	Chocolate	4.25

Q 2.1b

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

In [77]:
```
# your code
icecream.plot.bar(x = 'Flavor', y = "Price", rot=0)
plt.title('Icecream flavors by price')
plt.ylabel('Price in USD')
```

Out[77]: Text(0, 0.5, 'Price in USD')



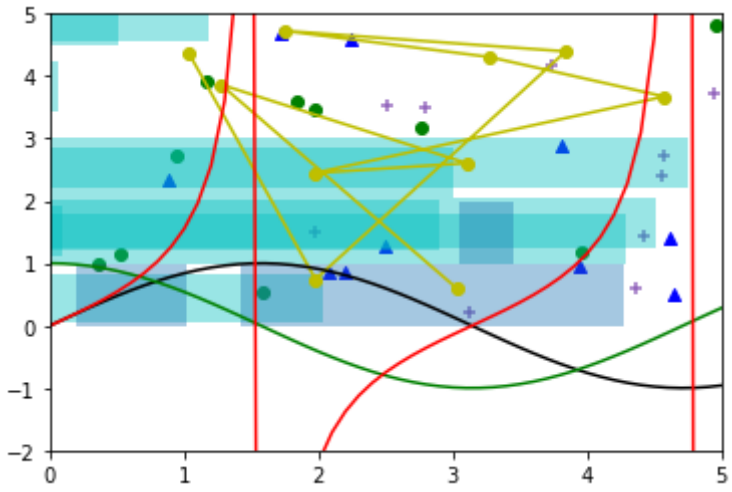
Q 2.2

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 [131]:
```
# your code
from numpy.random import random

#presets for trig functions
x1 =x2= np.arange(0,2*np.pi,0.1)
x3 = np.arange(0,2*np.pi,0.1)

#random scatter plots
a = plt.scatter(5*random(10), 5*random(10), c="G")
b = plt.scatter(5*random(10), 5*random(10), marker='+', c='tab:purple')
c  = plt.scatter(5*random(10), 5*random(10), marker='^', c='B')

#line, horoizontal bar, histogram
d  = plt.plot(5*random(10), 5*random(10), marker='o', c='y')
e  = plt.barh(5*random(10), 5*random(10), color='c', alpha = .4)
f = plt.hist(5*random(10),alpha = .4)

#sin, cos, tan
g = plt.plot(x1, np.sin(x),  c='k')
h = plt.plot(x2, np.cos(x),  c='g')
i = plt.plot(x3, np.tan(x),  color=colors[4])

#formatting
plt.axis([0,5,-2,5])

#plt.legend((a, b, c, d, e, f, g, h, i),
#           ('Scatter1', 'Scatter2', 'Scatter3', 'Line Plot', 'H-Bar',
 'Hist', 'Sin', 'Cos' ,'Tan'))

plt.show()
```

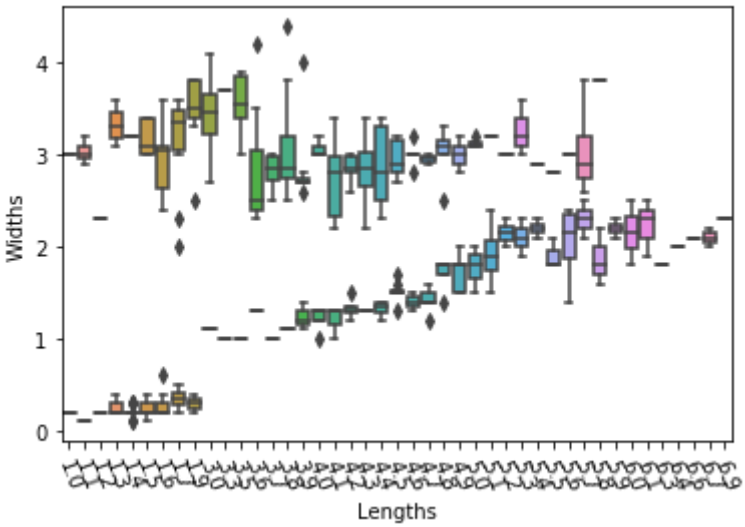


3.

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 [172]:
```
# your code
plt.close()
import seaborn as sns
iris = sns.load_dataset('iris')
x = sns.boxplot(x="sepal_length", y="sepal_width", data=iris)
y = sns.boxplot(x="petal_length", y="petal_width", data=iris)
plt.ylabel('Widths')
plt.xlabel('Lengths')
plt.xticks(rotation=-70)
# Show plot
plt.show()
```



Q 3.2

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

The relation between sepal length and width is positive, but the opposite is true for petal length as it relates to petal width. This might sugest that sepals benifit from the added support provided by the width of the sepal but the petals often are one or the other. However, the relation between petal length and width is pretty weak. It's a slight inverse but not very steep.

Q 4.

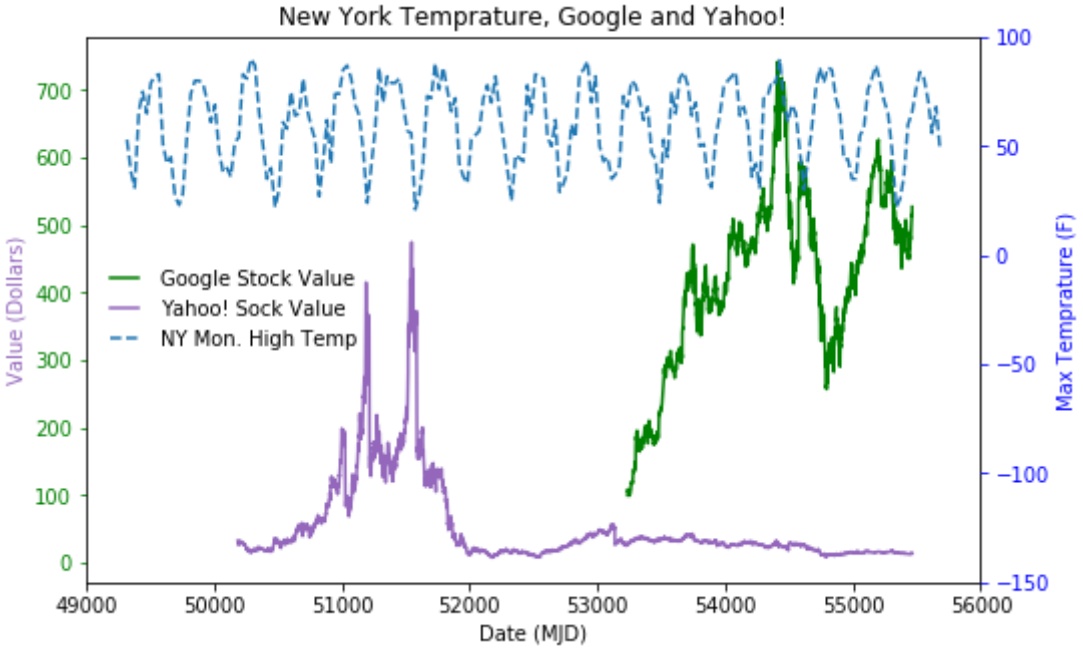
The data files needed:

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

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

In [268]:
```
google = pd.read_csv('google_data.txt' , sep= ' ')
nyt = pd.read_csv('ny_temps.txt', sep=' ')
yahoo = pd.read_csv('yahoo_data.txt', sep = '    ')
```

In [282]:
```
x_values1=google['Modified Julian Date']
y_values1=google['Stock Value']

x_values3=yahoo['Modified Julian Date']
y_values3=yahoo['Stock Value']

x_values2=nyt['Modified Julian Date']
y_values2=nyt['Max Temperature']


fig=plt.figure()
ax=fig.add_subplot(111, label="1")
ax2=fig.add_subplot(111, label="2", frame_on=False)

x, = ax.plot(x_values1, y_values1, color="g")
y, = ax.plot(x_values3, y_values3, color="tab:purple")
ax.set_xlabel("Date (MJD)")
ax.set_ylabel("Value (Dollars)", color='tab:purple')
ax.tick_params(axis='x')
ax.tick_params(axis='y')
ax.set_xlim(49000,56000)
ax.tick_params(axis='y', colors="g")

z, = ax2.plot(x_values2, y_values2, linestyle='--' )
ax2.yaxis.tick_right()
ax2.set_ylabel('Max Temprature (F)', color='B')
ax2.yaxis.set_label_position('right')
ax2.tick_params(axis='x', colors="C1")
ax2.tick_params(axis='y', colors="B")
ax2.set_xticks([])
ax2.set_ylim(bottom=-150, top=100)


ax.legend(handles=[x, y, z], labels=['Google Stock Value', 'Yahoo! Sock
 Value', 'NY Mon. High Temp'],
              loc='center left',frameon=False,numpoints=3);

plt.title('New York Temprature, Google and Yahoo!')

fig.set_size_inches(8,5)

plt.show()
```


New York Temprature, Google and Yahoo!
Google Stock Value
Yahoo! Sock Value
NY Mon. High Temp
Value (Dollars)
Max Temprature (F)
Date (MJD)
0
100
200
300
400
500
600
700
−150
−100
−50
0
50
100
49000
50000
51000
52000
53000
54000
55000
56000

In [281]:

In [ ]: