Assignment 1

Author: Shweta Sampath Kumar

Created: 1/15/2023

MScA 31010 Linear and Non Linear Models

In [1]:
```
import pandas as pd
import numpy as np
import plotly.express as px
import math
```

-- QUESTION 1 ---

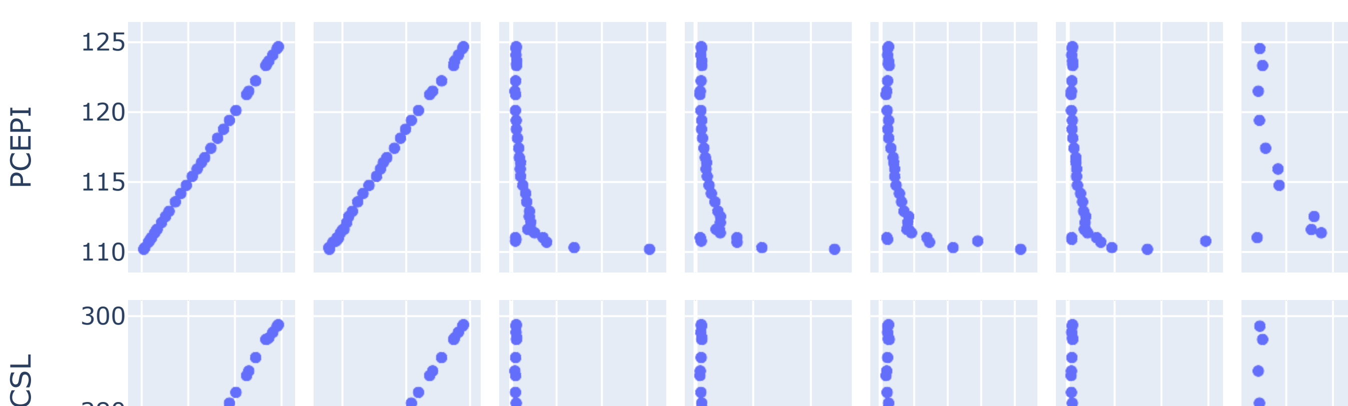
In [2]:
```
econ = pd.read_csv("Economy_2020_to_2022.csv")
econ.head()
```

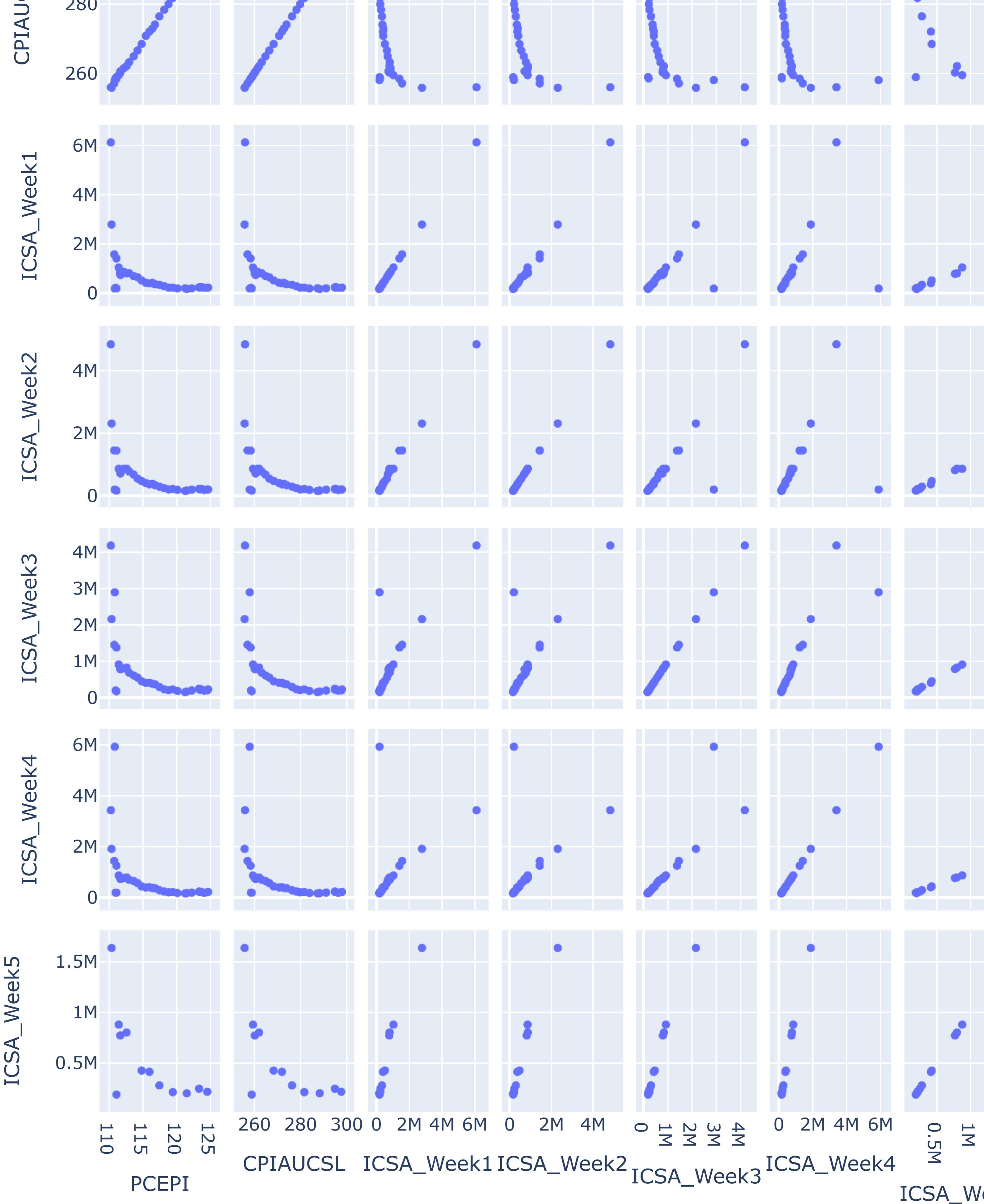
Out[2]:

	Year	Month	N_Week	PCEPI	CPIAUCSL	ICSA_Week1	ICSA_Week2	ICSA_Week3	ICSA_Week4	ICSA_Week5
0	2020	1	4	110.944	258.682	217000	203000	211000	200000	NaN
1	2020	2	5	111.070	259.007	191000	186000	190000	196000	190000.0
2	2020	3	4	110.824	258.165	186000	221000	2914000	5946000	NaN
3	2020	4	4	110.237	256.094	6137000	4869000	4201000	3446000	NaN
4	2020	5	5	110.353	255.944	2796000	2335000	2176000	1921000	1639000.0

(a) Generate a matrix of scatter plot (SPLOM) of these seven features: PCEPI, CPIAUCSL, ICSA_Week1, ICSA_Week2, ICSA_Week3, ICSA_Week4, and ICSA_Week5. You mut properly label the axes and add grid lines to all the scatter plots.

In [3]:
```
fig = px.scatter_matrix(econ,
                        width = 800,
                        height = 1100,
                        dimensions = ['PCEPI', 'CPIAUCSL', 'ICSA_Week1', 'ICSA_Week2', '
fig.show()
```





(b) Calculate the Pearson correlations for each pair of the seven features. Display your result up to four decimal places appropriately as a matrix.

In [4]:

```
cols = ['PCEPI', 'CPIAUCSL', 'ICSA_Week1', 'ICSA_Week2', 'ICSA_Week3', 'ICSA_Week4', 'IC
pearsoncorr = round(econ[cols].corr(method='pearson'), 4)
pearsoncorr
```

Out[4]:

	PCEPI	CPIAUCSL	ICSA_Week1	ICSA_Week2	ICSA_Week3	ICSA_Week4	ICSA_Week5
PCEPI	1.0000	0.9993	-0.4692	-0.5079	-0.5786	-0.4934	-0.6655

CPIAUCSL	0.9993	1.0000	-0.4669	-0.5056	-0.5720	-0.4853	-0.6669
ICSA_Week1	-0.4692	-0.4669	1.0000	0.9961	0.8436	0.4960	0.9710
ICSA_Week2	-0.5079	-0.5056	0.9961	1.0000	0.8483	0.5003	0.9854
ICSA_Week3	-0.5786	-0.5720	0.8436	0.8483	1.0000	0.8824	0.9907
ICSA_Week4	-0.4934	-0.4853	0.4960	0.5003	0.8824	1.0000	0.9952
ICSA_Week5	-0.6655	-0.6669	0.9710	0.9854	0.9907	0.9952	1.0000

(c) Calculate the Spearman rank-order correlations for each pair of the seven features. Display your result up to four decimal places appropriately as a matrix.

In [5]:

```
spearman_corr = round(econ[cols].corr('spearman'), 4)
spearman_corr
```

Out[5]:

	PCEPI	CPIAUCSL	ICSA_Week1	ICSA_Week2	ICSA_Week3	ICSA_Week4	ICSA_Week5
PCEPI	1.0000	0.9989	-0.5947	-0.6143	-0.7333	-0.7392	-0.5524
CPIAUCSL	0.9989	1.0000	-0.6094	-0.6302	-0.7474	-0.7532	-0.5524
ICSA_Week1	-0.5947	-0.6094	1.0000	0.9744	0.8412	0.8340	0.9860
ICSA_Week2	-0.6143	-0.6302	0.9744	1.0000	0.8986	0.8892	0.9842
ICSA_Week3	-0.7333	-0.7474	0.8412	0.8986	1.0000	0.9940	0.9860
ICSA_Week4	-0.7392	-0.7532	0.8340	0.8892	0.9940	1.0000	0.9860
ICSA_Week5	-0.5524	-0.5524	0.9860	0.9842	0.9860	0.9860	1.0000

(d) Calculate the Kendall's Tau-b correlations for each pair of the seven features. Display your result up to four decimal places appropriately as a matrix.

In [ ]:

```
kendalls_corr = round(econ[cols].corr('kendall'), 4)
kendalls_corr
```

Out[ ]:

	PCEPI	CPIAUCSL	ICSA_Week1	ICSA_Week2	ICSA_Week3	ICSA_Week4	ICSA_Week5
PCEPI	1.0000	0.9899	-0.5652	-0.5786	-0.6650	-0.6672	-0.5455
CPIAUCSL	0.9899	1.0000	-0.5685	-0.5820	-0.6684	-0.6706	-0.5455
ICSA_Week1	-0.5652	-0.5685	1.0000	0.8998	0.8226	0.8213	0.9394
ICSA_Week2	-0.5786	-0.5820	0.8998	1.0000	0.8544	0.8403	0.9313
ICSA_Week3	-0.6650	-0.6684	0.8226	0.8544	1.0000	0.9539	0.9394
ICSA_Week4	-0.6672	-0.6706	0.8213	0.8403	0.9539	1.0000	0.9394
ICSA_Week5	-0.5455	-0.5455	0.9394	0.9313	0.9394	0.9394	1.0000

(e) Calculate the Distance correlations for each pair of the seven features. Display your result up to four decimal places appropriately as a matrix.

In [ ]:
```
def empirical_distance(M):
    m = []
    m_mean = []
    for x in M:
        l = []
        for i in M:
            l.append(abs(x-i))
        m.append(l)
        m_mean.append(sum(l)/len(l))

    m = np.matrix(m)
    m_adjusted = []
    total_mean = sum(m_mean)/len(m_mean)
    c = m.shape[1]
    s = 0
    for i in enumerate(m):
        l = []
        for j in range(c):
            x = m.item(i[0], j) - m_mean[i[0]] - m_mean[j] + total_mean
            s = s + (x*x)
            l.append(x)
        m_adjusted.append(l)
    vn = s/(c*c)
    return vn, np.matrix(m_adjusted)

def distance_correlation(A, B):
    vn_A, s1 = empirical_distance(A)
    vn_B, s2 = empirical_distance(B)
    s = 0
    for i in enumerate(s1):
        for j in range(s1.shape[1]):
            s = s + (s1.item(i[0], j) * s2.item(i[0], j))

    vn_AB = s/(len(A) * len(B))

    R_squared = vn_AB/(math.sqrt(vn_A * vn_B))
    R = math.sqrt(R_squared)

    return R

cols = ['PCEPI', 'CPIAUCSL', 'ICSA_Week1', 'ICSA_Week2', 'ICSA_Week3', 'ICSA_Week4', 'IC
d = []
for x in cols:
    d_row = []
    for y in cols:
        df = pd.concat([econ[x], econ[y]], axis = 1)
        df = df.dropna()
        distance_corr = distance_correlation(df.iloc[:,0], df.iloc[:,1])
        d_row.append(round(distance_corr, 4))
    d.append(d_row)

distancecorr = pd.DataFrame(np.matrix(d), columns = cols, index = cols, dtype = 'float32
distancecorr
```

-- QUESTION 2 ---

(a) What is the first derivative of the function $f(x) = x^2 - a$ with respect of x?

$$f'(x) = 2x$$

(b) You will use the Newton-Raphson method to solve the equation $f(x) = x^2 - a = 0$. What is the formula for updating the estimate?

Formula to update the estimate:

$$x_{n+1} = x_n - f(x_n)/f'(x_n)$$

Solving for $f(x) = x^2 - a$

$f(x) = x^2 - a$
$f'(x) = 2x$
Let $a = 2$, $n = 0$
Thus, $x_0 = 1$
$x_1 = x_0 - f(x_0)/f'(x_0) = 1 - ((1)^2 - 2)/2(1) = 1.5$
$x_2 = x_1 - f(x_1)/f'(x_1) = 1.5 - ((1.5)^2 - 2)/2(1.5) = 1.4166666666666667$
$x_3 = x_2 - f(x_2)/f'(x_2) = 1.4166666666666667 - ((1.4166666666666667)^2 - 2)$
$/2(1.4166666666666667) = 1.41421568627451$

$x_4 = x_3 - f(x_3)/f'(x_3) = 1.41421568627451 - ((1.41421568627451)^2 - 2)$
$/2(1.41421568627451) = 1.41421356237469$

$x_5 = x_4 - f(x_4)/f'(x_4) = 1.41421356237469 - ((1.41421356237469)^2 - 2)$
$/2(1.41421356237469) = 1.414213562373095$

$x_6 = x_5 - f(x_5)/f'(x_5) = 1.414213562373095 - ((1.414213562373095)^2 - 2)$
$/2(1.414213562373095) = 1.414213562373095$

The root for this is 1.414213562373095

In [ ]:
```python
#Using Python to show the above
def func (x, a):
   y = x * (x) - a
   return (y)

def dfunc(x):
   dy = 2 * x
   return (dy)

def newton_raphson (init_x, a, max_iter, eps_conv, q_history):
   i_iter = 0
   q_continue = True
   reason = 0
   x_curr = init_x

   if (q_history):
      history = []
   while (q_continue):
      f_curr = func(x_curr, a)
```

```python
        dfunc_curr = dfunc(x_curr)
        if (q_history):
            history.append([i_iter, x_curr, f_curr, dfunc_curr])
        if (f_curr != 0.0):
            if (dfunc_curr != 0.0):
                i_iter = i_iter + 1
                x_next = x_curr - f_curr / dfunc_curr
                if (abs(x_next - x_curr) <= eps_conv):
                    q_continue = False
                    reason = 1                # Successful convergence
                elif (i_iter >= max_iter):
                    q_continue = False
                    reason = 2                # Exceeded maximum number of iterations
                else:
                    x_curr = x_next
            else:
                q_continue = False
                reason = 3                    # Zero derivative
        else:
            q_continue = False
            reason = 4                        # Zero function value

    if(q_history):
        print(pd.DataFrame(history, columns = ['Iteration', 'Estimate', 'Function', 'Deriv

    if reason == 1:
        r = "Successful convergance"
    elif reason == 2:
        r = "Exceeded maximum number of iterations"
    elif reason == 3:
        r = "Zero derivative"
    elif reason == 4:
        r = "Zero function value"

    return (x_curr, r)

x_solution, reason = newton_raphson (init_x = 1, a = 2, max_iter = 100, eps_conv = 1e-14
print("\nThe root of this equation is: " + str(x_solution) + "\nReason: " + reason)
```

(c) Suppose $a = 9$ and the initial estimate is $x_0 = 1$. The iteration will converge if $|x_{k+1} - x_k| \leq 10^{-13}$. Please show the iteration history.

In [ ]:
```python
x_solution, reason = newton_raphson (init_x = 1, a = 9, max_iter = 100, eps_conv = 1e-13
print("\nThe root of this equation is: " + str(x_solution) + "\nReason: " + reason)
```

(d) Suppose $a = 9000$ and the initial estimate is $x_0 = 1$. The iteration will converge if $|x_{k+1} - x_k| \leq 10^{-13}$. Please show the iteration history.

In [ ]:
```python
x_solution, reason = newton_raphson (init_x = 1, a = 9000, max_iter = 100, eps_conv = 1e
print("\nThe root of this equation is: " + str(x_solution) + "\nReason: " + str(reason))
```

(e) Suppose $a = 0.0000009$ and the initial estimate is $x_0 = 1$. The iteration will converge if $|x_{k+1} - x_k| \leq 10^{-13}$. Please show the iteration history.

In [ ]:
```python
x_solution, reason = newton_raphson (init_x = 1, a = 0.0000009, max_iter = 100, eps_conv
print("\nThe root of this equation is: " + str(x_solution) + "\nReason: " + str(reason))
```