TC1002S Herramientas computacionales: el arte de la analítica

This is a notebook with all your work for the final evidence of this course

Niveles de dominio a demostrar con la evidencia

SING0202A

Interpreta interacciones entre variables relevantes en un problema, como base para la construcción de modelos bivariados basados en datos de un fenómeno investigado que le permita reproducir la respuesta del mismo. Es capaz de construir modelos bivariados que expliquen el comportamiento de un fenómeno.

Student information

- Name: Alan Alfredo Onofre Chávez
- ID: A01632858
- My carreer: Engineer Robotics and Digital Systems

Importing libraries

In [ ]:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
```

PART 1

Use your assigned dataset

A1 Load data

In [ ]:
```
df = pd.read_csv(r"C:\Users\alana\Documents\TC1002S\Evidencia\A01632858.csv")
df.drop(columns=["Unnamed: 0"], inplace=True)
```

A2 Data managment

Print the first 7 rows

In [ ]:
```
df.head(7)
```

Out[ ]:

	x1	x2
0	0.676261	-0.781409
1	0.909555	-0.179328
2	1.837036	-0.169800
3	0.685107	0.308597
4	-0.455524	-0.839300
5	0.634601	0.543943
6	0.902844	-0.543820

Print the first 4 last rows

In [ ]:
```
df.tail(4)
```

Out[ ]:

	x1	x2
1896	0.903400	-0.169171
1897	1.618596	0.238065
1898	0.945071	-0.724298
1899	0.584162	0.456375

How many rows and columns are in your data?

Use the `shape` method

In [ ]:
```
df.shape
```

Out[ ]:
```
(1900, 2)
```

Print the name of all columns

Use the `columns` method

In [ ]:

```
df.columns
```

Out[ ]:

```
Index(['x1', 'x2'], dtype='object')
```

What is the data type in each column

Use the `dtypes` method

In [ ]:

```
df.dtypes
```

Out[ ]:

```
x1    float64
x2    float64
dtype: object
```

What is the meaning of rows and columns?

In [ ]:

```
# Your responses here

# 1) In the first column we have integer values

# 2) In the second column we have floating point values

# 3) In the third column we have float values
```

Print a statistical summary of your columns

In [ ]:

```
print("Mean")
print(df.mean(numeric_only=True))

print("Standard Desviation")
print(df.std(numeric_only=True))

print("Median")
print(df.median(numeric_only=True))

print("Minimum")
print(df.min(numeric_only=True))

print("Maximum")
print(df.max(numeric_only=True))
```

```
Mean
x1    0.497284
x2   -0.247859
dtype: float64
Standard Desviation
x1    0.868894
x2    0.499814
dtype: float64
Median
x1    0.490602
x2   -0.247406
dtype: float64
Minimum
x1   -1.159740
x2   -1.190545
dtype: float64
Maximum
x1    2.186159
x2    0.692885
dtype: float64
```

In [ ]:

```
qV = df.quantile(q=0.75)
print(qV)

qV2 = df.quantile(q=0.50)
print(qV2)

qV3 = df.quantile(q=0.25)
print(qV3)
```

```
x1    1.062271
x2    0.202280
Name: 0.75, dtype: float64
x1    0.490602
x2   -0.247406
Name: 0.5, dtype: float64
x1   -0.047609
x2   -0.700743
Name: 0.25, dtype: float64
```

In [ ]:

```
# 1) What is the minumum and maximum values of each variable

#Maximum x2: 0.6928, x1: 2.1861. Minimum x2: -1.1905, x1 -1.1597

# 2) What is the mean and standar deviation of each variable

#Mean x2: -0.2478, x1: 0.4972. Standar deivation x2: 0.4998, x1: 0.8688

# 3) What the 25%, 50% and 75% represent?

#25 x2: -0.7007 x1: -0.04760, 50 x2: -0.2474 x1: 0.4906, 75 x1: 1.0622 x2: 0.2022
```

Rename the columns using the same name with capital letters

In [ ]:

```
df.rename(columns={"x1":"X1"}, inplace=True)
df.rename(columns={"x2":"X2"}, inplace=True)
df.head()
```

Out[ ]:

	X1	X2
0	0.676261	-0.781409
1	0.909555	-0.179328
2	1.837036	-0.169800
3	0.685107	0.308597
4	-0.455524	-0.839300

Rename the columns to their original names

In [ ]:

```
df.rename(columns={"X1":"x1"}, inplace=True)
df.rename(columns={"X2":"x2"}, inplace=True)
df.head()
```

Out[ ]:

	x1	x2
0	0.676261	-0.781409
1	0.909555	-0.179328
2	1.837036	-0.169800
3	0.685107	0.308597
4	-0.455524	-0.839300

Use two different alternatives to get one of the columns

In [ ]:

```
a = df.loc[:,"x1"]
b = df.loc[:,"x2"]
print(b)
```

```
0      -0.781409
1      -0.179328
2      -0.169800
3       0.308597
4      -0.839300
          ...
1895    0.541994
1896   -0.169171
1897    0.238065
1898   -0.724298
1899    0.456375
Name: x2, Length: 1900, dtype: float64
```

Get a slice of your data set: second and thrid columns and rows from 62 to 72

In [ ]:

```
print(df[62:73])
```

```
          x1        x2
62 -0.557061 -0.715460
63  1.810587  0.165126
64  0.990451  0.077964
65 -0.262957 -1.021611
66  0.750779 -0.534411
67  0.688207 -0.834643
68  0.974666 -0.433689
69  1.149298  0.574834
70  0.226195 -1.046441
71  1.681878  0.046262
72  0.377614  0.233084
```

For the second and thrid columns, calculate the number of null and not null values and verify that their sum equals the total number of rows

```
In [ ]: print(df.notnull().sum())
        print(df.isnull().sum())
```

```
x1    1900
x2    1900
dtype: int64
x1    0
x2    0
dtype: int64
```

Questions

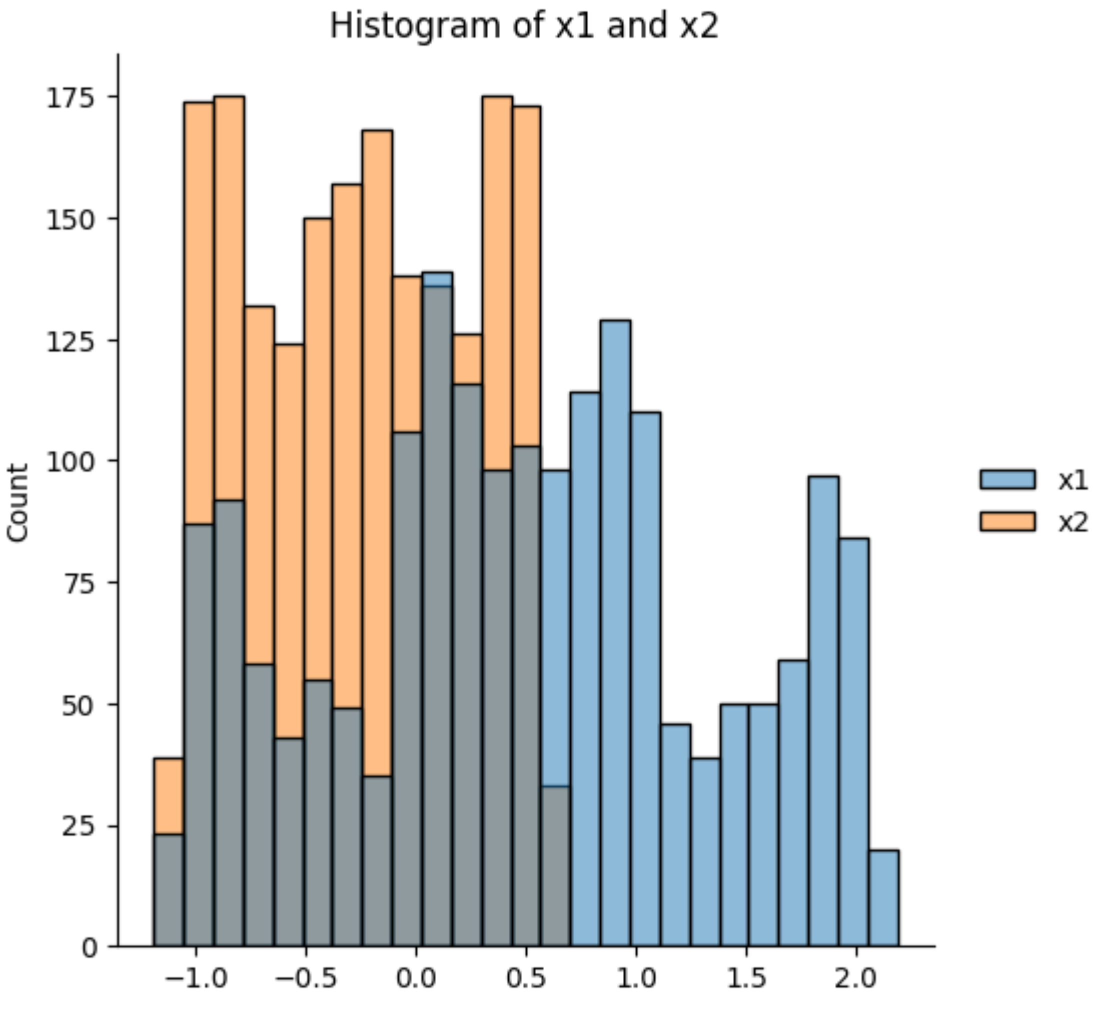
Based on the previos results, provide a description of yout dataset

Your response: In the data set presented we don't have null data, also there are 1900 rows and 3 columns with a mean of 0.4972 and -0.2478. Finally, the columns of each one are x1 and x2

A3 Data visualization

Plot in the same figure the histogram of the two variables

```
In [ ]: sns.displot(df, kde = False)
        plt.title("Histogram of x1 and x2")
        plt.ylabel("Count")
        plt.show()
```



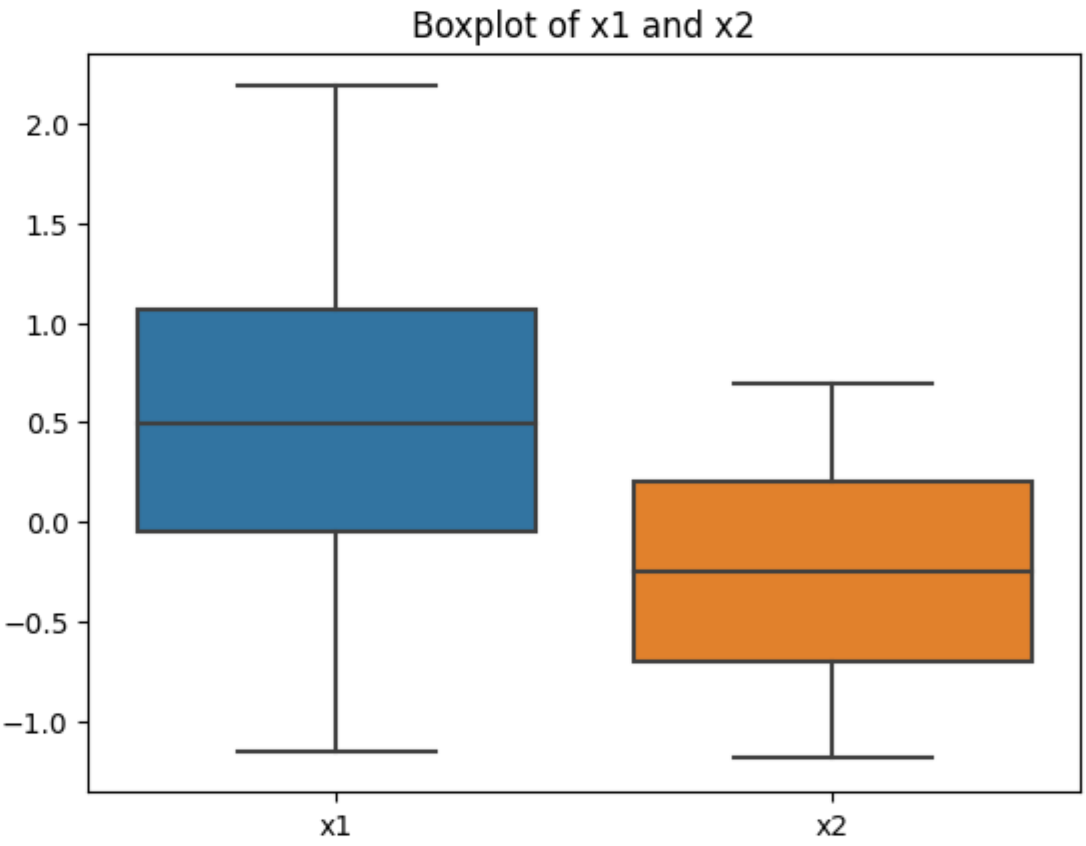
Based on this plots, provide a description of your data:

Your response here: The values of x2 are much higher than x1 and also the the values of x2 have more approach to negative values

Plot in the same figure the boxplot of the two variables

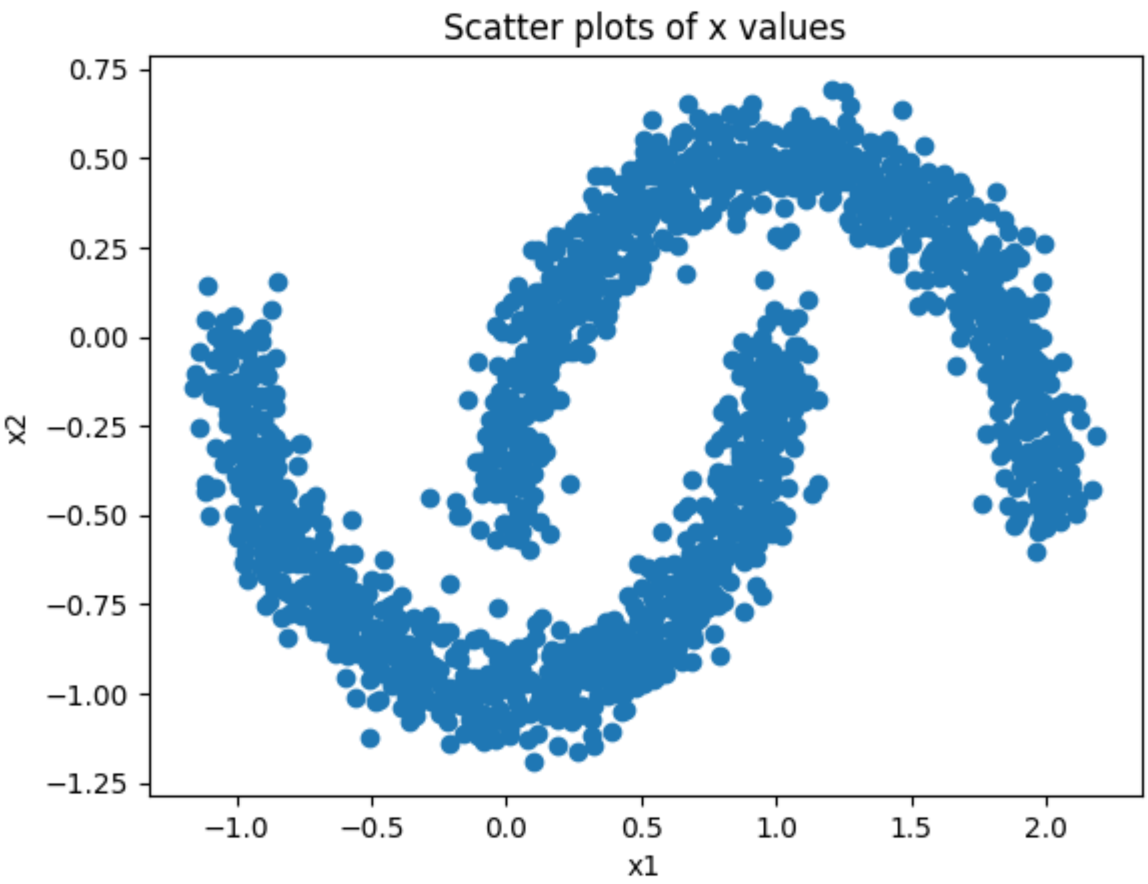
In [ ]:

```
sns.boxplot(df)
plt.title('Boxplot of x1 and x2')
plt.show()
```



Scatter plot of the two variables

```
In [ ]: plt.scatter(x=df['x1'], y=df['x2'])
        plt.title('Scatter plots of x values')
        plt.xlabel('x1')
        plt.ylabel('x2')
        plt.show()
```



Questions

Based on the previos plots, provide a description of yout dataset

Your response: Thamks to the boxplot we can see that the plot has relation with the statistical summary also in the scatter plot we can look that the values of each column form a parabola

A4 Kmeans

Do Kmeans clustering assuming a number of clusters accorging to your scatter plot

```
In [ ]: from sklearn.cluster import KMeans
        k = 2

        #Initialize the Kmeans box/object
        km = KMeans(n_clusters=k,n_init="auto")
```

Add to your dataset a column with the assihned cluster to each data point

```
In [ ]: #Do K-means clustering
        yestimated = km.fit_predict(df)
        df['yestimated'] = yestimated
        df
```

Out[ ]:

	x1	x2	yestimated
0	0.676261	-0.781409	0
1	0.909555	-0.179328	1
2	1.837036	-0.169800	1
3	0.685107	0.308597	1
4	-0.455524	-0.839300	0
...	...	...	...
1895	1.361944	0.541994	1
1896	0.903400	-0.169171	1
1897	1.618596	0.238065	1
1898	0.945071	-0.724298	1
1899	0.584162	0.456375	1

1900 rows × 3 columns

Print the number associated to each cluster

In [ ]:

```
df.yestimated.unique()
```

Out[ ]:

```
array([0, 1])
```

Print the centroids

In [ ]:

```
km.cluster_centers_
```

Out[ ]:

```
array([[-0.18329595, -0.56720841],
       [ 1.21465204,  0.08875181]])
```

Print the intertia metric

In [ ]:

```
km.inertia_
```

Out[ ]:

```
776.2209219794131
```

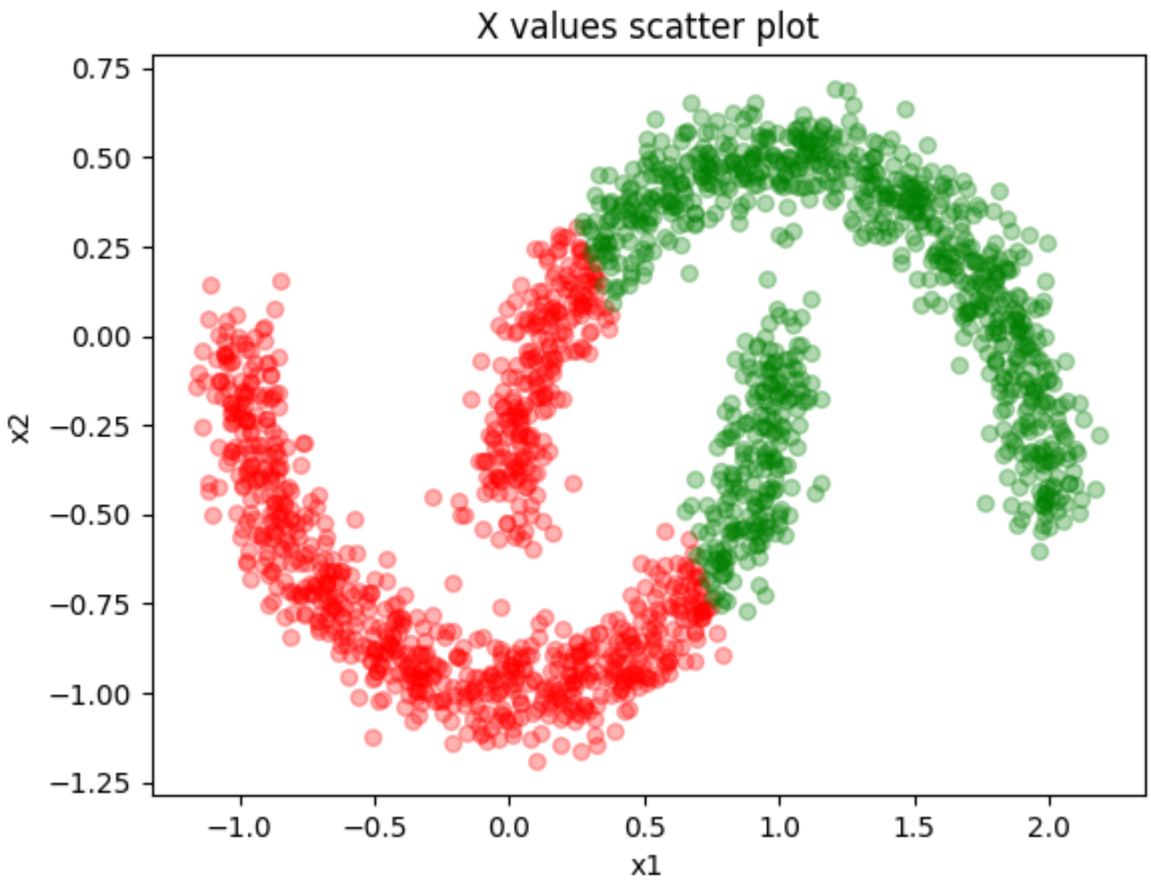
Plot a scatter plot of your data assigned to each cluster. Also plot the centroids

In [ ]:

```
#Get a dataframe with the data of each cluster
df1 = df[df.yestimated==0]
df2 = df[df.yestimated==1]

#Scatter plot of each cluster
plt.scatter(df1.x1, df1.x2, c='r', marker='o', s=32, alpha=0.3)
plt.scatter(df2.x1, df2.x2, c='g', marker='o', s=32, alpha=0.3)

plt.title('X values scatter plot')
plt.xlabel('x1')
plt.ylabel('x2')
plt.show()
```



Questions

Provides a detailed description of your results

Your response:

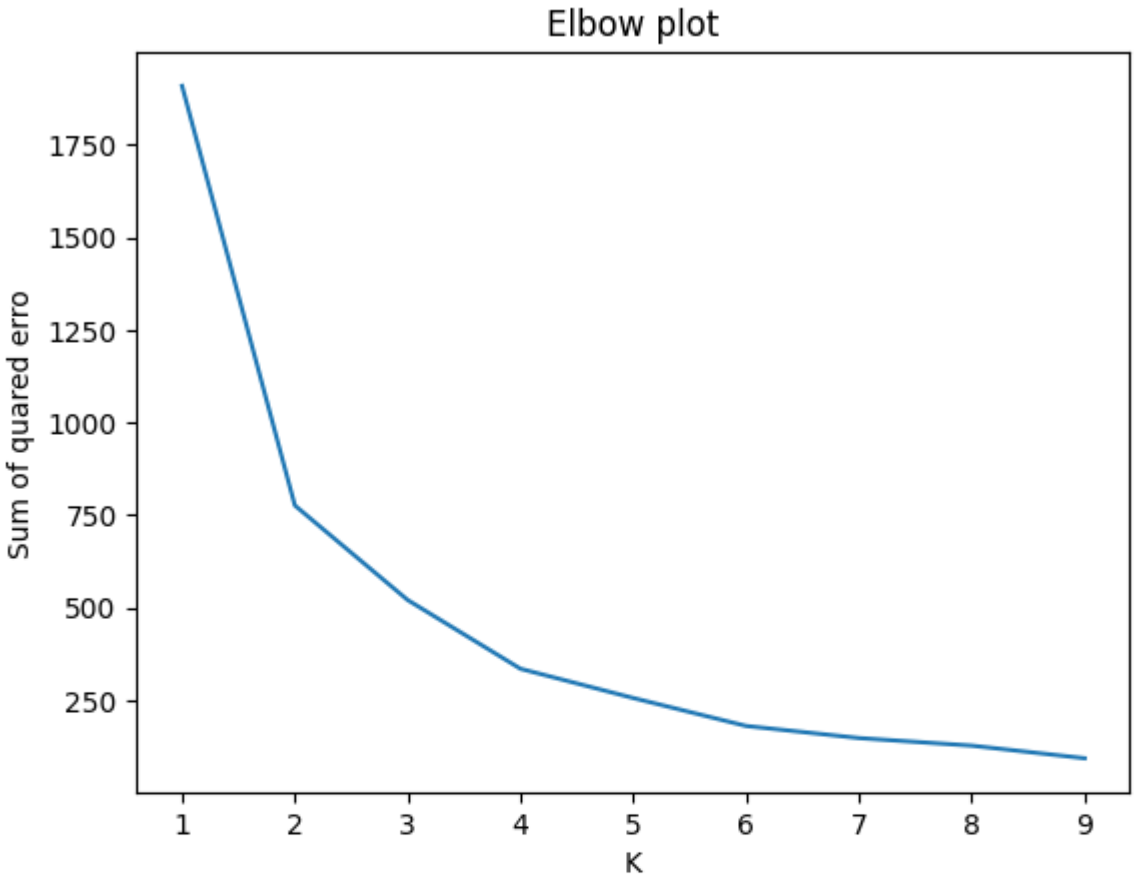
A5 Elbow plot

Compute the Elbow plot

```
In [ ]: #Initialize a list to hold sum of squarred error
        sse = []

        k_rng = range(1, 10)
        for k in k_rng:
            #Create model
            km = KMeans(n_clusters=k, n_init='auto')
            km.fit_predict(df[['x1', 'x2']])
            sse.append(km.inertia_)

        #Plot sse versus k
        plt.plot(k_rng, sse, markersize=8)
        plt.title('Elbow plot')
        plt.xlabel('K')
        plt.ylabel('Sum of quared erro')
        plt.show()
```



Questions

What is the best number of clusters K? (argue your response)

Your response: Two because we have only two values which are x1 and x2

Does this number of clusters agree with your inital guess? (argue your response)

Your response: No, because I was expected to have just two clusters for each parabola, but in the response of the algorithm it mixed it

PART 2

Create a dataset and do clustering

1. Generate some data using the "make_blobs" function from "sklearn.datasets"

- The number of observations is equal to the three last digits in your ID (if this number is lower than 99, then multiply it by ten)
- 3 variables
- 4 clusters
- Standar deviation of each cluster of 1.5

In [ ]:

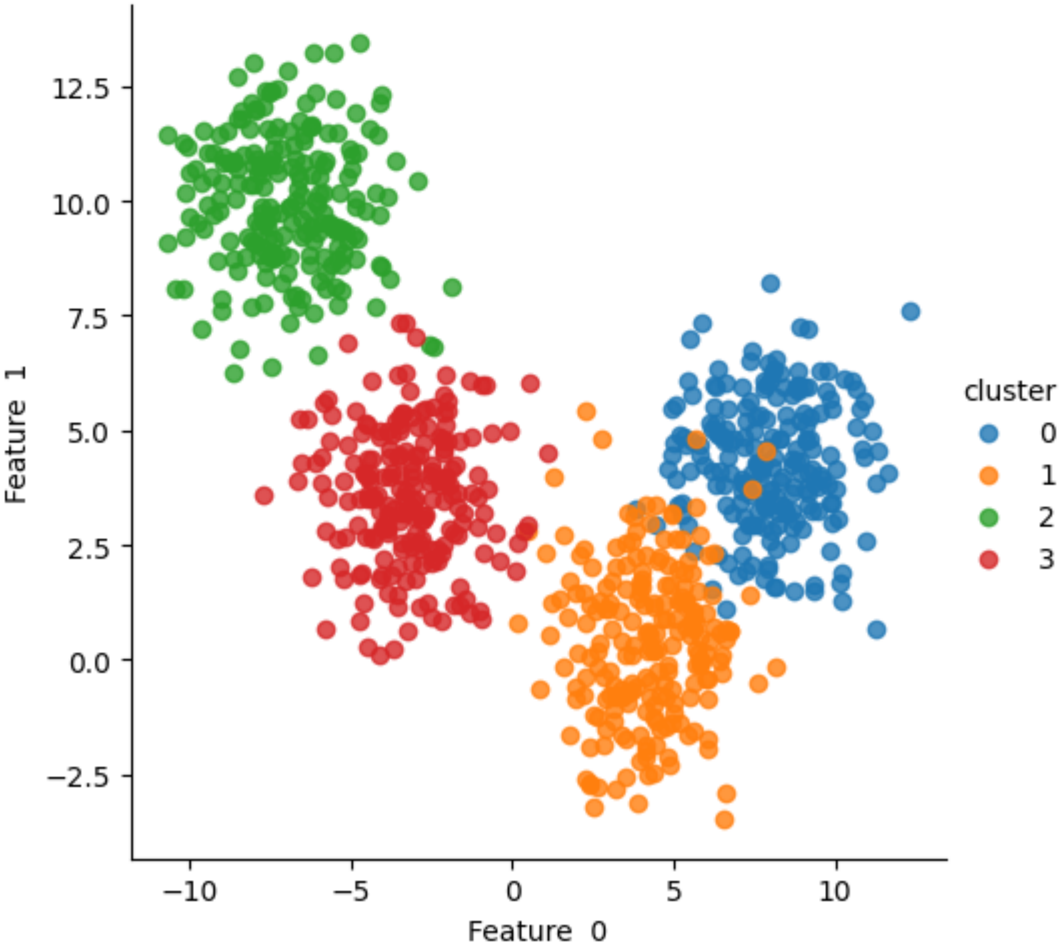
```
import sklearn.datasets
from itertools import combinations
x, y = sklearn.datasets.make_blobs(n_samples=858, n_features=3, centers=4, cluster_
data = pd.DataFrame(x, columns=['Feature {:2d}'.format(i)
                                for i in range(3)])
```

2. Plot the scatter plot of your data using the real cluster labels

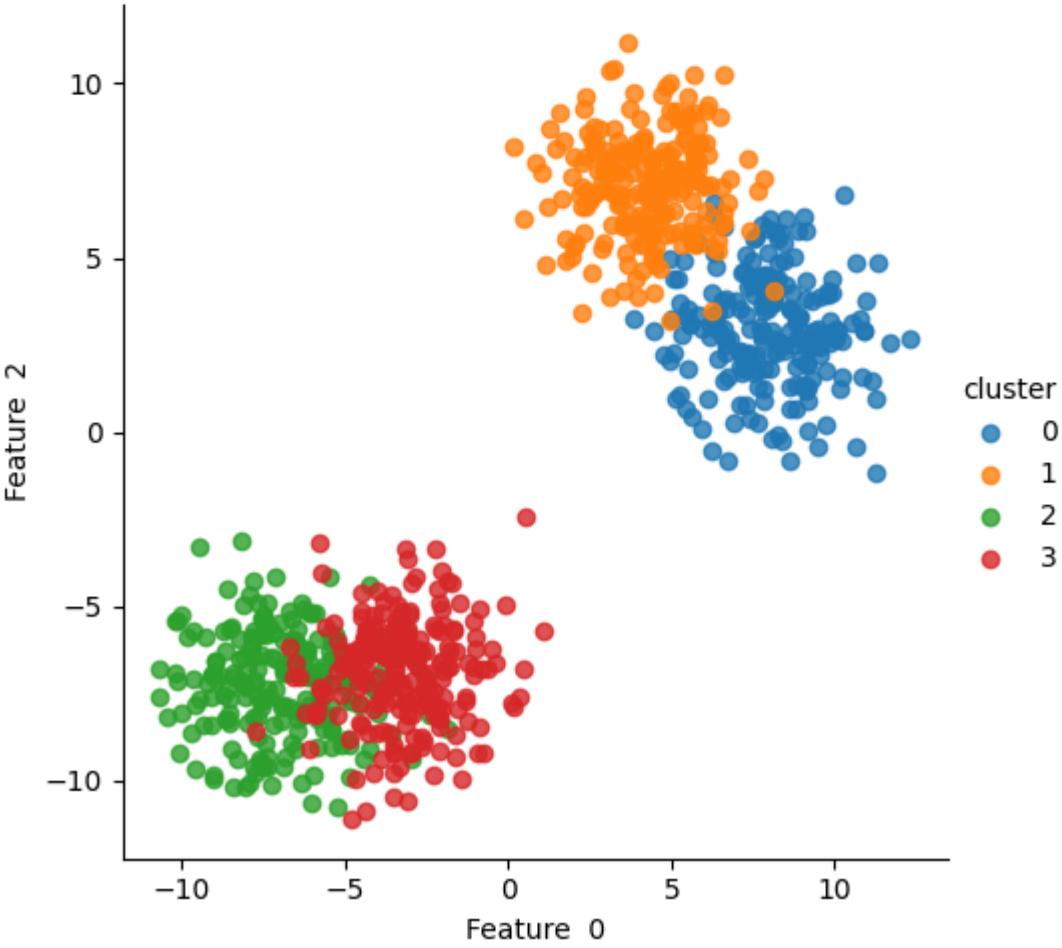
In [ ]:

```
data['cluster'] = y
for i, j in combinations(range(3), 2):
  print('Plotting Feature {:2d} vs Feature {:2d}'.format(i, j))
  sns.lmplot(x='Feature {:2d}'.format(i),
             y='Feature {:2d}'.format(j),
             hue='cluster',
             data=data,
             fit_reg=False
            )
  plt.show()
```

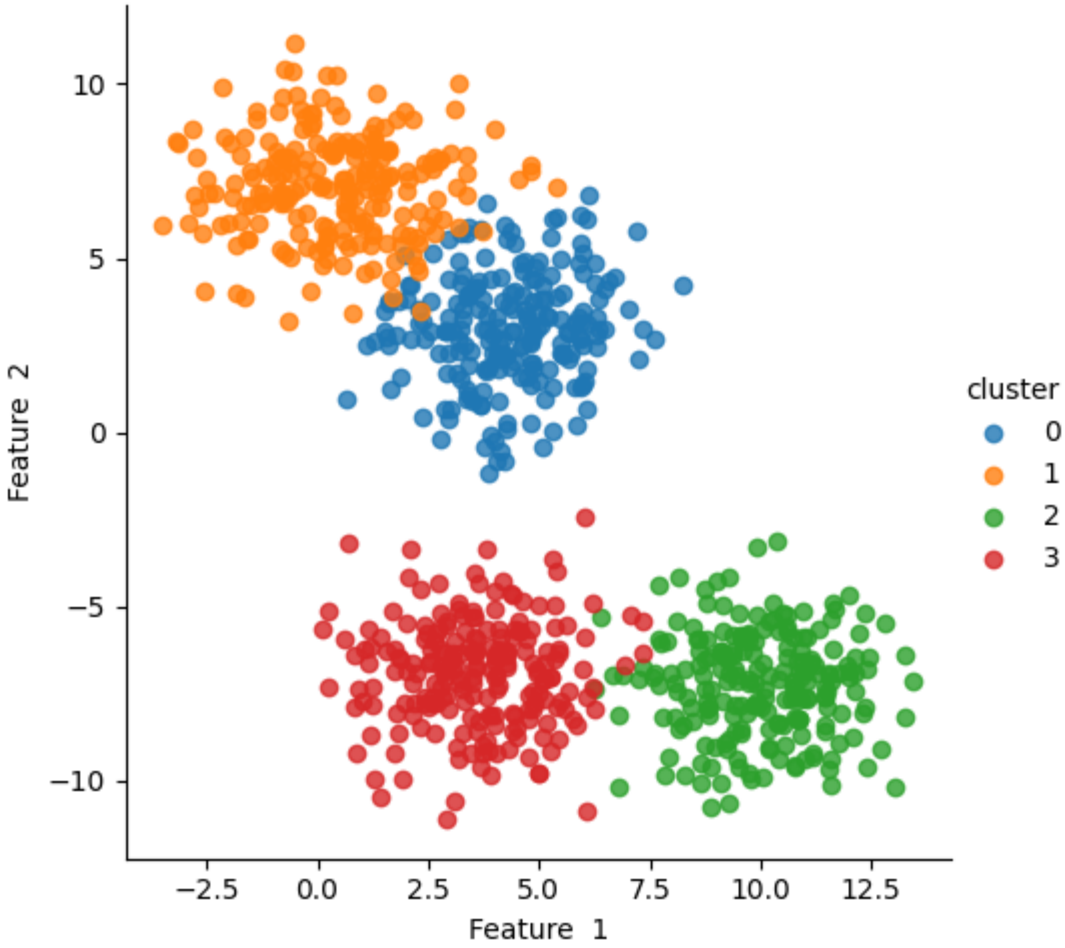
```
Plotting Feature  0 vs Feature  1
```



```
Plotting Feature  0 vs Feature  2
```



```
Plotting Feature  1 vs Feature  2
```



3. Do K means clustering

In [ ]:

```
k = 4

#Initialize the Kmeans box/object
km = KMeans(n_clusters=k,n_init="auto")

#Do K-means clustering
yestimated = km.fit_predict(data)
data['yestimated'] = yestimated
data.rename(columns={'Feature  0':'Feature0'}, inplace=True)
data.rename(columns={'Feature  1':'Feature1'}, inplace=True)
data.rename(columns={'Feature  2':'Feature2'}, inplace=True)
data
```

Out[ ]:

	Feature0	Feature1	Feature2	cluster	yestimated
0	9.269478	4.173959	3.751249	0	0
1	6.315161	5.175954	3.384628	0	0
2	7.070939	3.437533	2.360079	0	0
3	-1.249653	4.837821	-6.604721	3	2
4	-8.843084	11.546311	-6.933791	2	1
...	...	...	...	...	...
853	-4.379767	4.384758	-5.871486	3	2
854	-1.624923	4.777235	-9.302702	3	2
855	6.317874	5.950851	6.233895	0	0
856	-3.975010	4.002873	-4.566472	3	2
857	-7.796763	10.573301	-5.672855	2	1

858 rows × 5 columns

In [ ]:

```
data.yestimated.unique()
```

Out[ ]:

```
array([0, 2, 1, 3])
```

In [ ]:

```
data.columns
```

Out[ ]:

```
Index(['Feature0', 'Feature1', 'Feature2', 'cluster', 'yestimated'], dtype='object
')
```

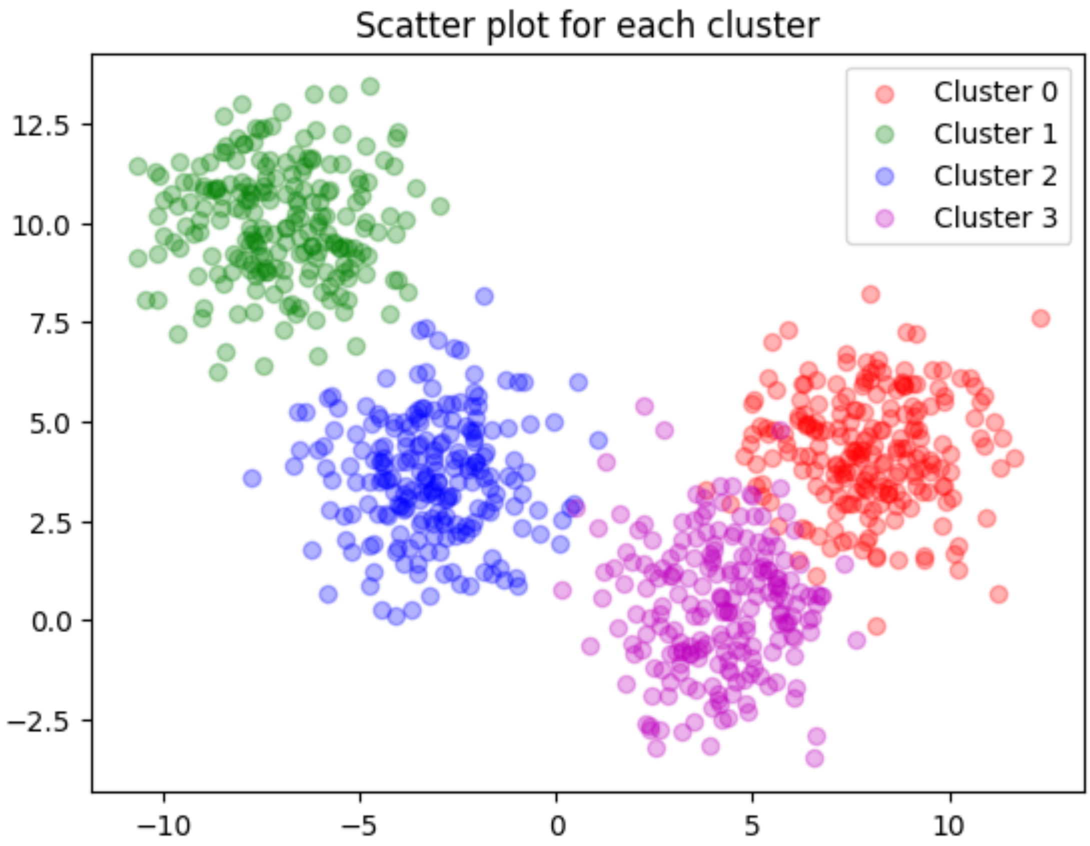
4. Plot the scatter plot of your data using the estimated cluster labels

In [ ]:

```
#Get a dataframe with the data of each cluster
data1 = data[data.yestimated==0]
data2 = data[data.yestimated==1]
data3 = data[data.yestimated==2]
data4 = data[data.yestimated==3]


#Scatter plot of each cluster
plt.scatter(data1.Feature0, data1.Feature1, label='Cluster 0', c='r', marker='o', a
plt.scatter(data2.Feature0, data2.Feature1, label='Cluster 1', c='g', marker='o', a
plt.scatter(data3.Feature0, data3.Feature1, label='Cluster 2', c='b', marker='o', a
plt.scatter(data4.Feature0, data4.Feature1, label='Cluster 3', c='m', marker='o', a

plt.title('Scatter plot for each cluster')
plt.legend()
plt.show()
```



Questions

Provides a detailed description of your results.

Your response: We obtain a new data set of three features with a shape of 858 times 4, also we have 4 different groups of clusters which are the values that approaches to the centers

PART 3

Descipcion de tu percepcion del nivel de desarrollo de la subcompetencia

SING0202A Interpretación de variables

Escribe tu description del nivel de logro del siguiente criterio de la subcompetencia

Interpreta interacciones. Interpreta interacciones entre variables relevantes en un problema, como base para la construcción de modelos bivariados basados en datos de un fenómeno investigado que le permita reproducir la respuesta del mismo.

Tu respuesta: Si lo pude lograr gracias a las librerias las cuales pude explora durante el curso además de aprender nuevamente cuales son los datos que tengo que tomar en cuenta a la hora de analizar datos y el como poder usarlos para implementar machine learning

Escribe tu description del nivel de logro del siguiente criterio de la subcompetencia

Construcción de modelos. Es capaz de construir modelos bivariados que expliquen el comportamiento de un fenómeno.

Tu respuesta: Lo pude dominar gracias a la practica que tuve durante el curso además de aprender a ver más a fondo con respecto a las graficas que obtuve al analizar datos aparte de tener un mejor entendimiento acerca de lo que significa klustering y el como esta se puede aplcar apara realizar inteligencia artificial