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: Ana Luisa González del Rosal
- ID: A01566927
- My carreer: ITC, Ingeniería en Tecnologías Computacionales

Importing libraries

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

PART 1

Use your assigned dataset

A1 Load data

In [ ]:
```
ruta= "C:/Users/Ana del Rosal/Documents/School/semanaTec/TC1002S/"
url = ruta + "NotebooksStudents/A01566927/A01566927_X.csv"
df = pd.read_csv(url, index_col= 0)
```

A2 Data managment

Print the first 7 rows

In [ ]:
```
# If we want the first 7 rows excluding the column names
df.head(8)

# First 7 rows considering columns names as a row
#df.head(7)
```

Out[ ]:

	x1	x2	x3	x4
0	-8.699798	9.082826	10.505440	-2.094418
1	7.910020	-1.769694	0.420732	3.343387
2	6.887981	-3.215543	10.732548	1.071340
3	-7.349416	8.678779	8.626893	-2.629711
4	-2.096514	-10.326640	-9.012192	-7.035618
5	10.166258	-0.715941	7.233119	6.804418
6	-0.846832	8.965758	10.888728	-3.153926
7	-1.023779	4.217112	10.984307	-0.076023

Print the last 4 rows

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

Out[ ]:

	x1	x2	x3	x4
343	5.410932	-0.565822	9.948077	0.898555
344	6.444991	3.471684	4.015452	5.557066
345	3.409736	-6.354255	-10.305898	-5.760450
346	7.633068	-3.001541	6.663707	3.039988

How many rows and columns are in your data?

Use the `shape` method

In [ ]:
```
print("Rows: ", df.shape[0])
print("Columns: ", df.shape[1])
```

```
Rows:  347
Columns:  4
```

Print the name of all columns

Use the `columns` method

In [ ]:
```
df.columns
```

Out[ ]:
```
Index(['x1', 'x2', 'x3', 'x4'], dtype='object')
```

What is the data type in each column

Use the `dtypes` method

In [ ]:
```
df.dtypes
```

Out[ ]:
```
x1    float64
x2    float64
x3    float64
x4    float64
dtype: object
```

What is the meaning of rows and columns?

In [ ]:
```
# Your responses here

#  Row: A row is a data entry which is identified by a unique value (index). Also k
#  A row has one or more variables.

#  Column: A column is a variable that is present in each row. It has its datatype
#  Even if its present in a row, its value can be null

# In this case, we have 4 columns: x1,x2,x3,x4. All of them are numerical, specific
# And we have 347 rows, each of them having all the variables.
```

Print a statistical summary of your columns

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

Out[ ]:

	x1	x2	x3	x4
count	347.000000	347.000000	347.000000	347.000000
mean	1.290063	0.467090	3.586111	-1.232981
std	6.129809	6.306259	7.854815	5.102685
min	-13.316015	-12.546350	-12.903834	-12.307747
25%	-3.194580	-4.011062	-1.636141	-5.703103
50%	1.901837	1.007800	6.392656	-1.482615
75%	6.119265	5.215560	9.638096	3.182624
max	13.622797	11.654640	14.198124	8.208970

In [ ]:

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

# x1:  min = -13.316015  max = 13.622797
# x2:  min = -12.546350  max = 11.654640
# x3:  min = -12.903834  max = 14.198124
# x4:  min = -12.307747  max = 8.208970

# We could also use the functions to obtain these values
print("== Min ==\n", df.min())
print("\n== Max ==\n", df.max())


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

# x1:  std = 6.129809  mean = 1.290063
# x2:  std = 6.306259  mean = 0.467090
# x3:  std = 7.854815  mean = 3.586111
# x4:  std = 5.102685  mean = -1.232981

# We could also use the functions to obtain these values

print("\n== std ==\n", df.std())
print("\n== mean ==\n", df.mean())

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

# They are the default percentiles the df.describe() function returns. A percentile
# First we order the values, and then return the value that is in the position
# for example if we have the values: [2, 3, 3, 4]
# for the 25% percentil we use the rule of 3: 25 * 4 / 100 = 1
# we return the value in position 1, value = 2  (we start counting in 1, not in zer
# for the 50% percentil we use the rule of 3: 50 * 4 / 100 = 2
# we return the value in position 2, value = 3
# for the 75% percentil we use the rule of 3: 75 * 4 / 100 = 3
# we return the value in position 3, value = 3
# In this example we can see that most of the data equals 3 as 2 percentiles have t

# Using our data we can analyze the behavior of each variable, for example
```

```
# for variable x3
# 0% (min) = -12.903834    25% = -1.636141        50% = 6.392656         75% = 9.63

# We can see that even though there is a bigger range between the  min value (0%) a
# there could be more positive values. At least half of them are positve, and there
# there is a big jump between the min (0%) and the 25% percentile, while the 50% an
```

```
== Min ==
 x1   -13.316015
x2   -12.546350
x3   -12.903834
x4   -12.307747
dtype: float64

== Max ==
 x1    13.622797
x2    11.654640
x3    14.198124
x4     8.208970
dtype: float64

== std ==
 x1    6.129809
x2    6.306259
x3    7.854815
x4    5.102685
dtype: float64

== mean ==
 x1    1.290063
x2    0.467090
x3    3.586111
x4   -1.232981
dtype: float64
```

Rename the columns using the same name with capital letters

In [ ]:
```
df = df.rename(columns={"x1": "X1", "x2": "X2", "x3": "X3", "x4": "X4"})

df.head(5)
```

Out[ ]:

	X1	X2	X3	X4
0	-8.699798	9.082826	10.505440	-2.094418
1	7.910020	-1.769694	0.420732	3.343387
2	6.887981	-3.215543	10.732548	1.071340
3	-7.349416	8.678779	8.626893	-2.629711
4	-2.096514	-10.326640	-9.012192	-7.035618

Rename the columns to their original names

In [ ]:
```
df = df.rename(columns={"X1": "x1", "X2": "x2", "X3": "x3", "X4": "x4"})
df.head(5)
```

Out[ ]:

	x1	x2	x3	x4
0	-8.699798	9.082826	10.505440	-2.094418
1	7.910020	-1.769694	0.420732	3.343387
2	6.887981	-3.215543	10.732548	1.071340
3	-7.349416	8.678779	8.626893	-2.629711
4	-2.096514	-10.326640	-9.012192	-7.035618

Use two different alternatives to get one of the columns

In [ ]:
```
x2_alt1 = df.x2
x2_alt2 = df["x2"]

print(x2_alt1)
print('\n===\n')
print(x2_alt2)
```

```
0       9.082826
1      -1.769694
2      -3.215543
3       8.678779
4     -10.326640
         ...
342     1.007800
343    -0.565822
344     3.471684
345    -6.354255
346    -3.001541
Name: x2, Length: 347, dtype: float64

===

0       9.082826
1      -1.769694
2      -3.215543
3       8.678779
4     -10.326640
         ...
342     1.007800
343    -0.565822
344     3.471684
345    -6.354255
346    -3.001541
Name: x2, Length: 347, dtype: float64
```

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

In [ ]:
```
df.loc[62:72, ["x2", "x3"]]
```

Out[ ]:

	x2	x3
62	1.451882	4.020712
63	6.535456	9.192824
64	10.876848	7.775449
65	3.735506	3.913879
66	0.842117	9.113710
67	-9.883563	-7.484494
68	-9.992748	-5.855158
69	-7.522563	-7.571163
70	-3.799281	7.224032
71	10.825537	9.834152
72	-3.024650	10.116247

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("Number of rows:")
print(df.shape[0])

print("\nNumber of null values:")
print(df[["x2","x3"]].isnull().sum())

print("\nNumber of not null values:")
print(df[["x2","x3"]].notnull().sum())

print("\n347 + 0 = 347")
```

```
Number of rows:
347

Number of null values:
x2    0
x3    0
dtype: int64

Number of not null values:
x2    347
x3    347
dtype: int64

347 + 0 = 347
```

Discard the last column

In [ ]:
```
df.drop(columns=['x4'], inplace = True)
df.head()
```

Out[ ]:

	x1	x2	x3
0	-8.699798	9.082826	10.505440
1	7.910020	-1.769694	0.420732
2	6.887981	-3.215543	10.732548
3	-7.349416	8.678779	8.626893
4	-2.096514	-10.326640	-9.012192

Questions

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

Your response:

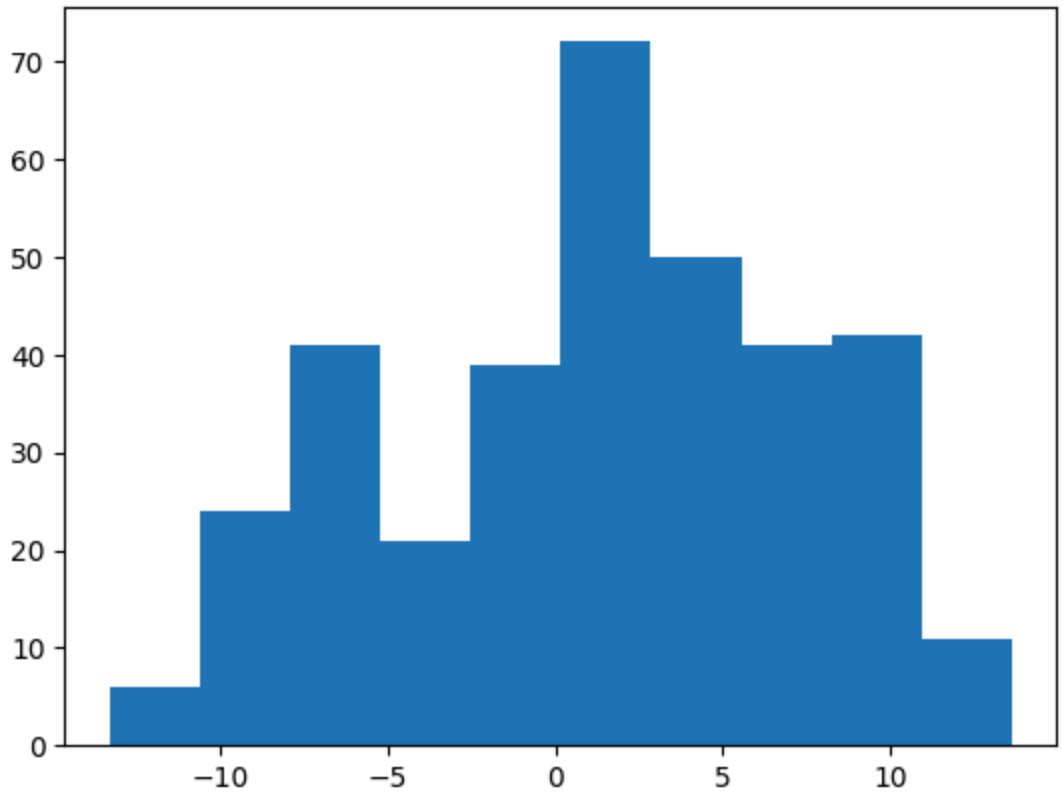
We have a dataset with *347 rows* and *4 variables*. The csv already contained an index, so we used that column as the default index for the DataFrame.

All variables are floats (decimals)and have a similar minimum value [-13.4, -12.3].

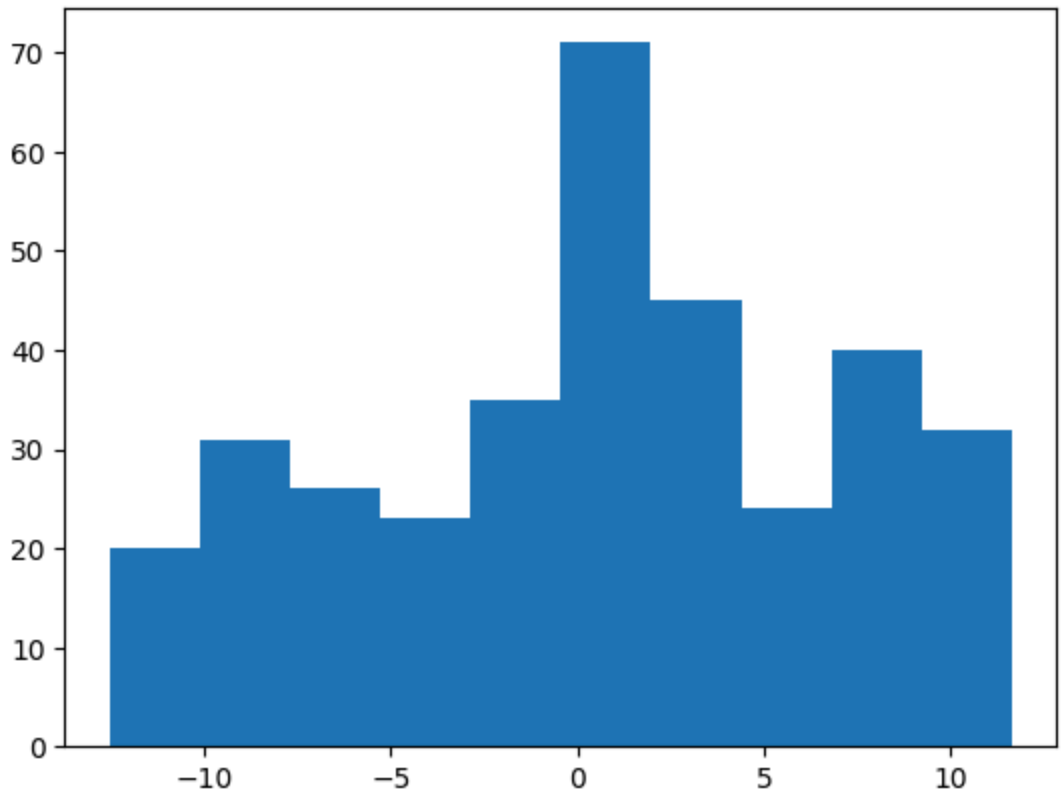
They have a range from 20-27

Analyze of the mean:

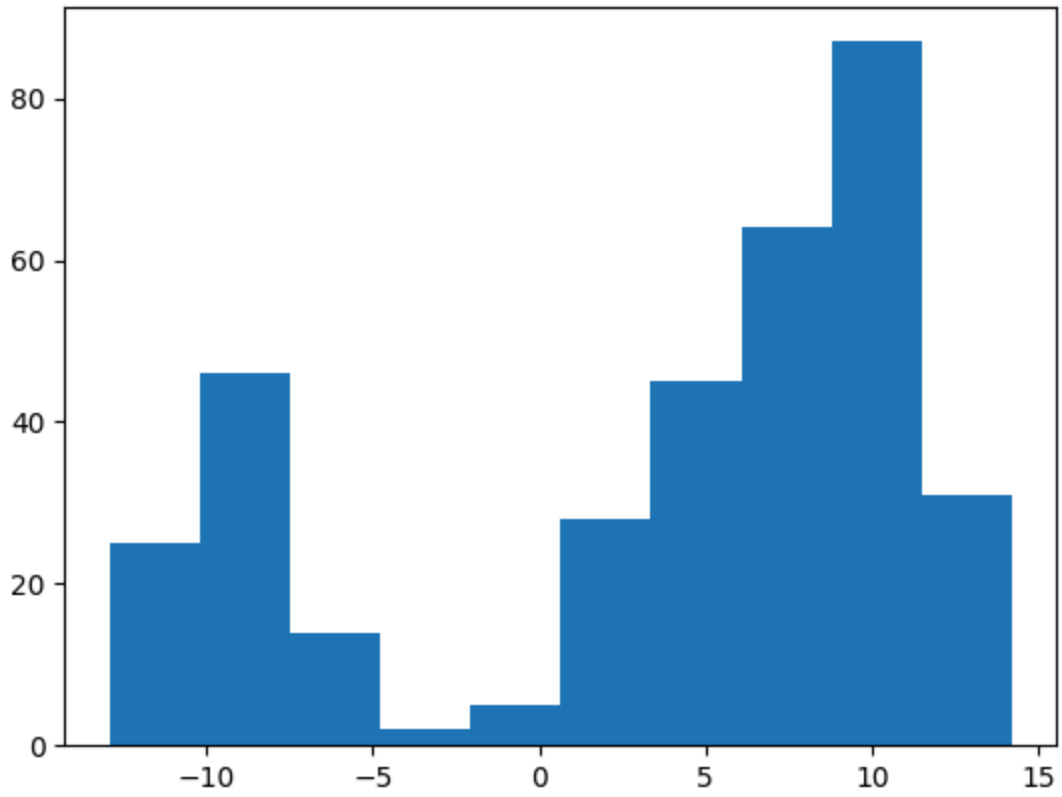
- x1: mean = 1.29 median = 1.90. these values are close, which tells us x1 has a symmetrical distribution, with a slight tendency to positive numbers.



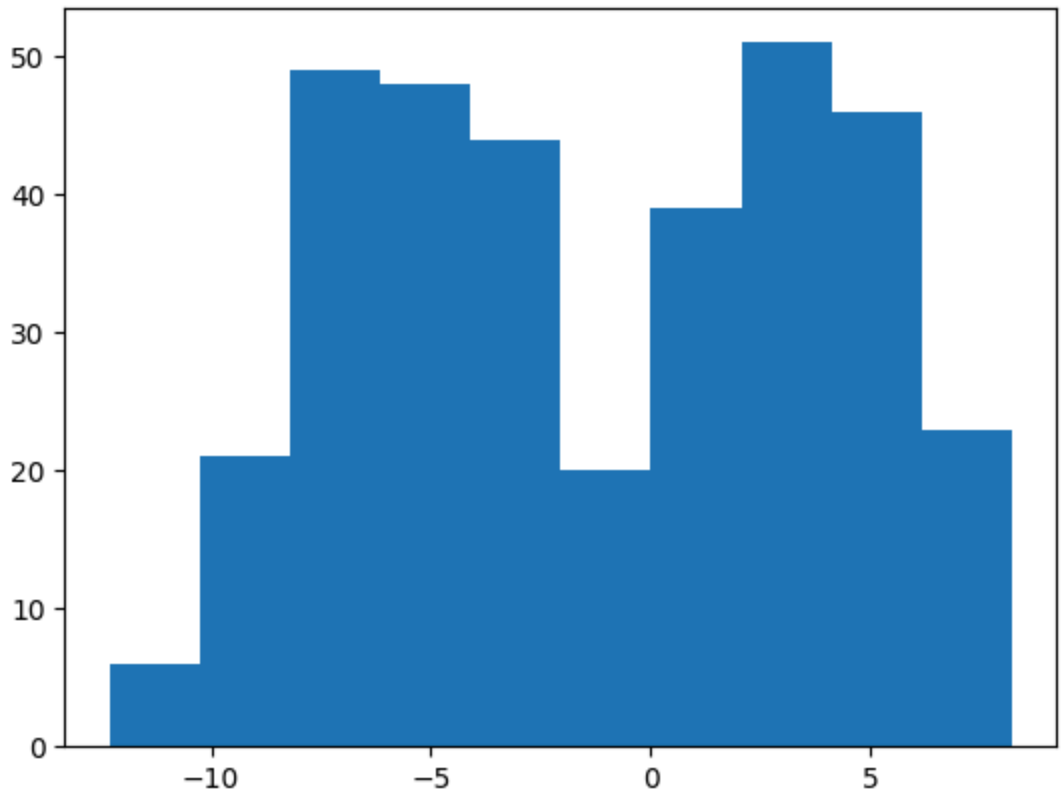
- x2: mean = 0.46 median = 1 these values are close considering the range, which tells us x2 has a symmetrical distribution.



- x3: mean = 3.58 median = 6.39 these values are not close considering the range, which tells us x3 has a distribution that tends to the right (skewed left)



- x4: mean = -1.23 median = -1.48 these values are very close considering the range, which tells us x4 has a symmetrical distribution.



Analyze of the percentiles: In most cases, there is a bigger jump from the 0% - 25% and 75% - 100% percentiles than 25% - 50% and 50% - 75%. This means that most numbers are

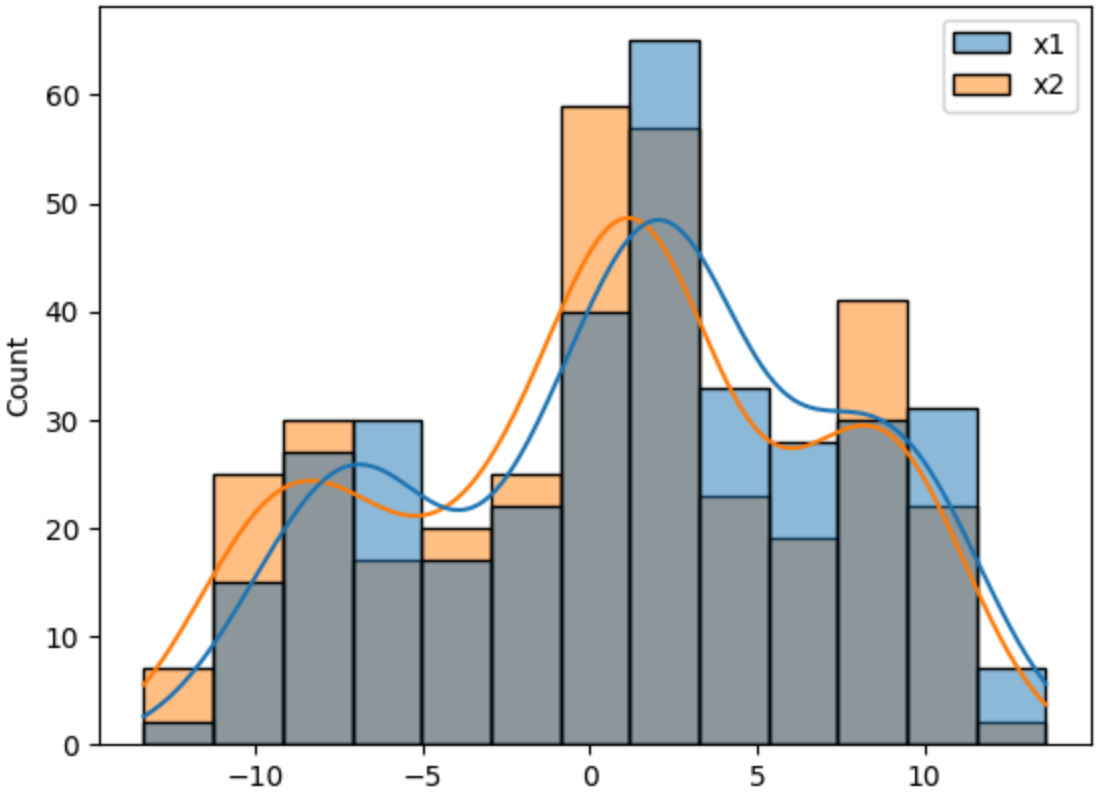
closer to the median than to the minimun and maximum value.

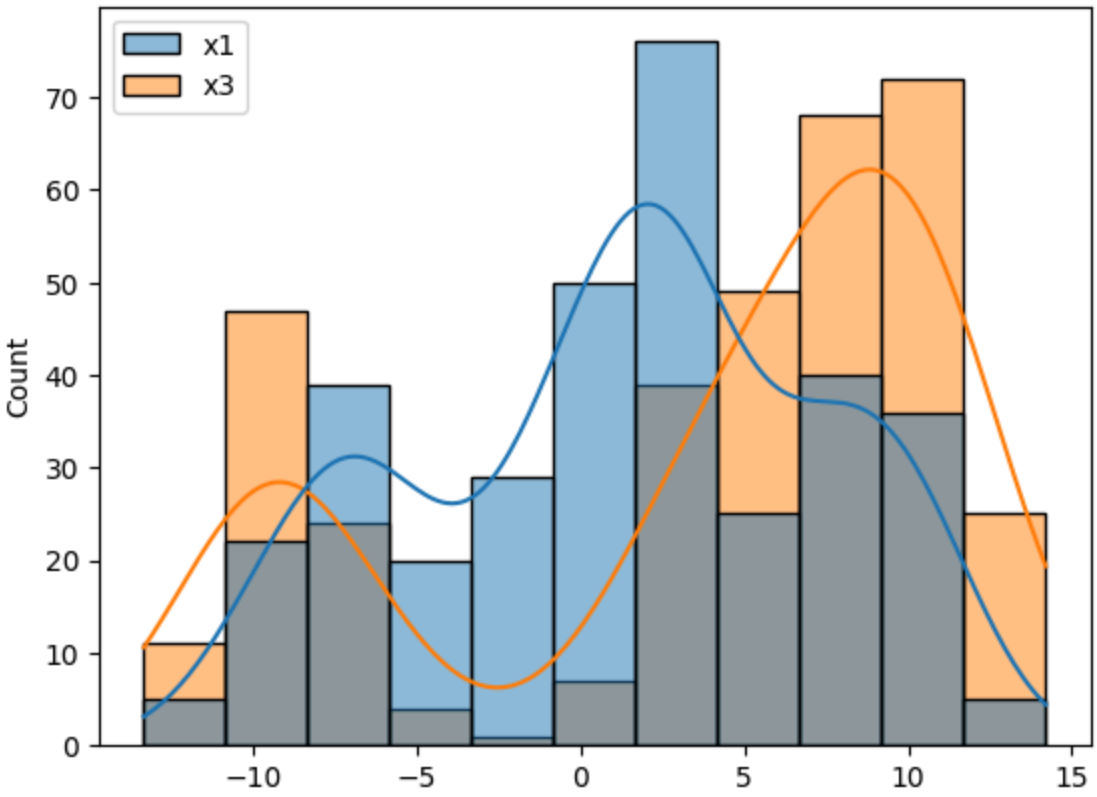
A3 Data visualization

Plot in the same figure the histogram of two variables

In [ ]:
```
df2plot = df[["x1", "x2"]]
sns.histplot(df2plot, kde = True)
plt.show()

df2plot = df[["x1", "x3"]]
sns.histplot(df2plot, kde = True)
plt.show()
```





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

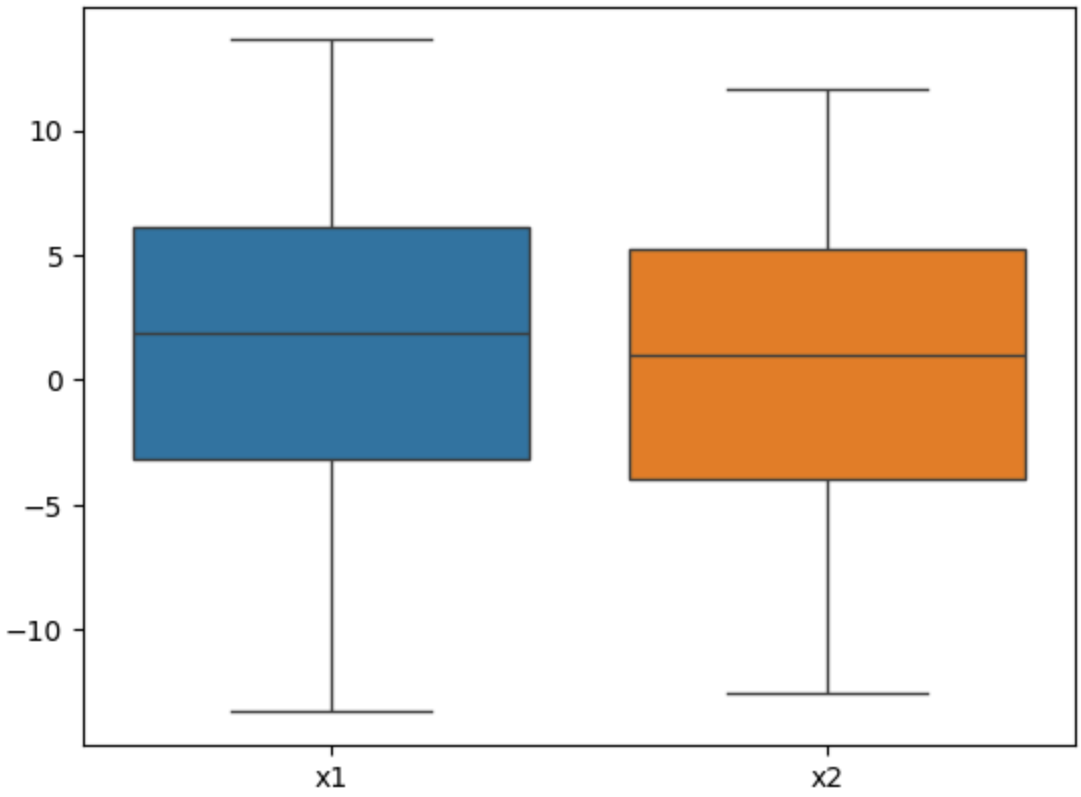
Your response here:

Both x1 and x2 have a symmetrical distribution, and they are almost the same, as we can see thanks to the kde. Their data is different but their behavior is alike.

In the second diagram we compare x1 and x3, and we can see very different behaviors. While x1 is symmetrical, x3 is skewed left (as mentioned in the last description).

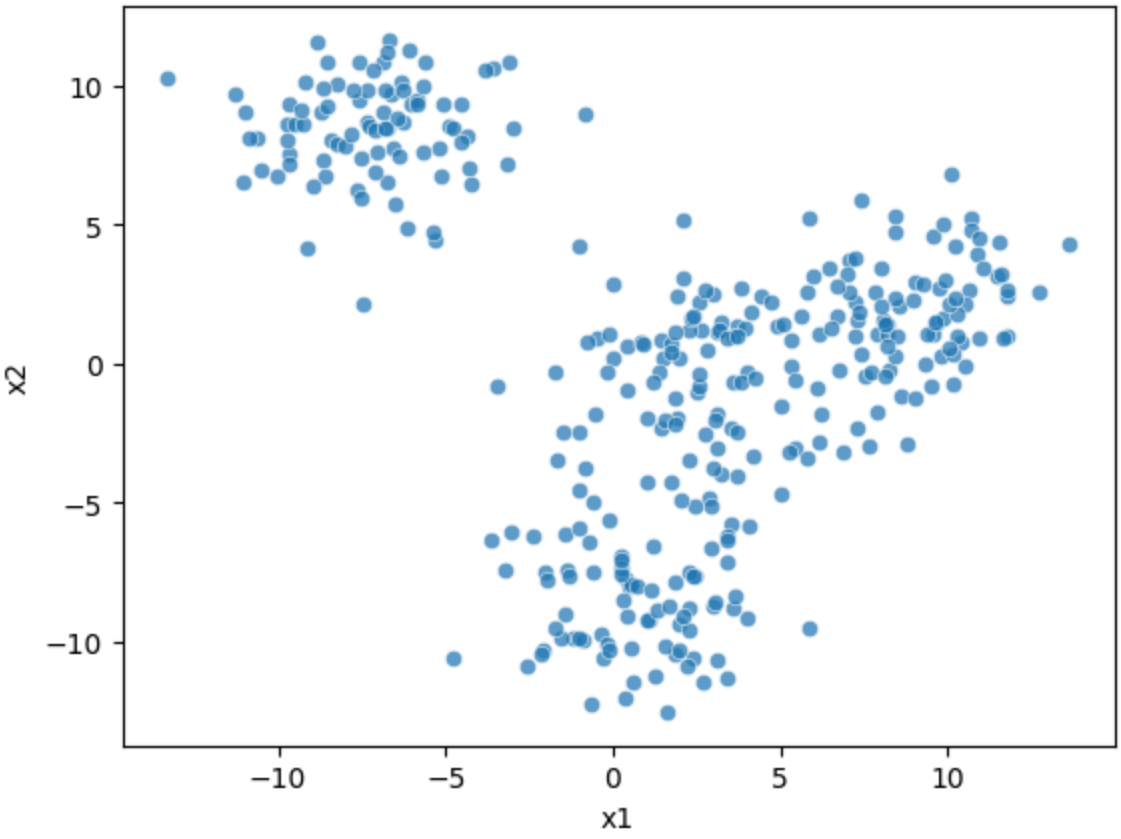
Plot in the same figure the boxplot of two variables

In [ ]:
```
sns.boxplot(data = df[["x1",'x2']])
plt.show()
```



Plot the scatter plot of two variables

In [ ]:
```
sns.scatterplot(data = df, x = "x1", y= "x2", alpha = 0.7)
plt.show()
```



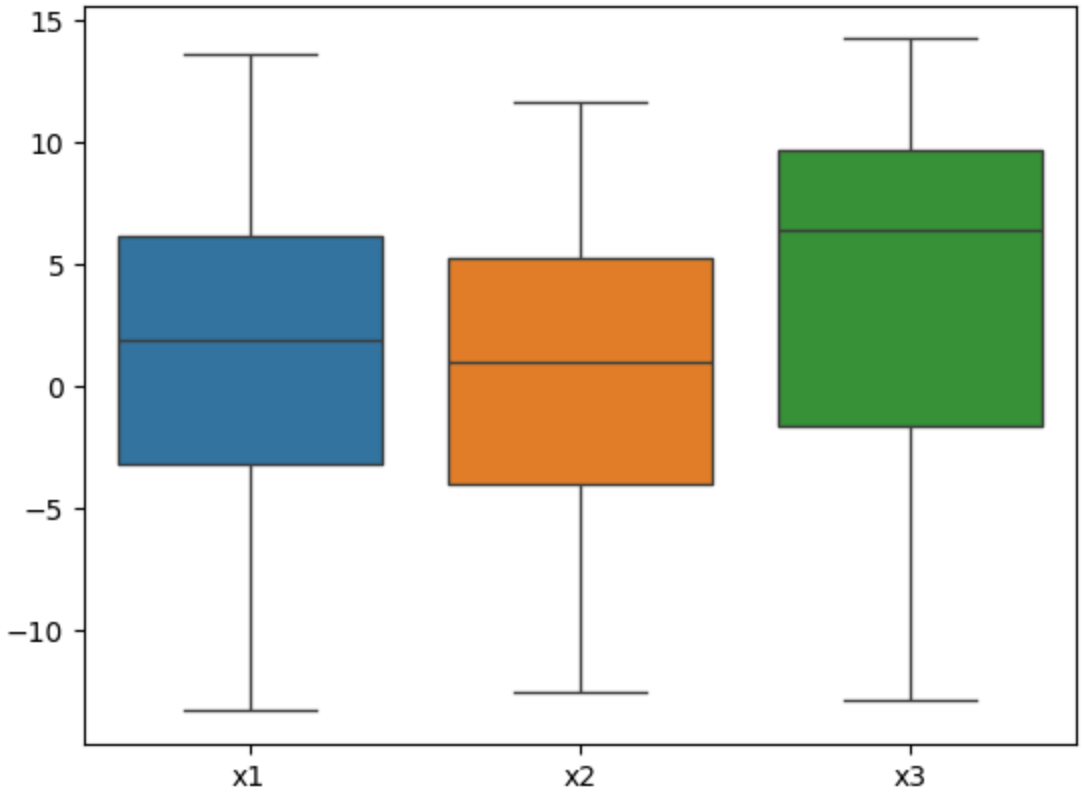
Questions

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

Your response:

In the first section of the lab we mentioned the quartiles and their behavior, stating that x1 was almost completely symmetrical and x2 had a slight tendency towards the positive numbers, and this is reflected in the boxplots. The orange box (x2) is localtes slightly upwards in its range, while the blue box (x1) is almost centered.

As an experiment, I added x3. We can see how the green box is not centered, nor its mean, showing us again, that x3 does not have a symmetrical distribution.



For the scatter plot we used x1 and x2, and got the following information:

- x2 has the highest values when x1 is negative
- when x1 is close to zero, x2 tends to have a value equal or lower than zero
- when x2 is close to zero, x1 tends to have a value equal or higher than zero

A4 Kmeans

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

In [ ]:
```
from sklearn.cluster import KMeans

K  = 3

km = KMeans(n_clusters=K, n_init="auto")

# We re-declare the DataFrame to do the KMeans Clustering considering all variables
# (in previous exercises we deleted x4)
df = pd.read_csv(url, index_col= 0)

yestimated = km.fit_predict(df)

yestimated
```

Out[ ]:
```
array([1, 2, 2, 1, 0, 2, 1, 1, 2, 1, 2, 1, 2, 2, 2, 1, 2, 2, 0, 2, 2, 1,
       2, 2, 2, 2, 0, 0, 2, 2, 2, 1, 1, 0, 1, 2, 2, 1, 2, 1, 2, 2, 1, 0,
       1, 1, 1, 2, 2, 2, 0, 2, 1, 2, 2, 2, 2, 1, 2, 1, 1, 0, 2, 1, 1, 2,
       2, 0, 0, 0, 2, 1, 2, 2, 2, 2, 1, 1, 1, 1, 0, 2, 0, 1, 2, 2, 1, 0,
       0, 2, 0, 1, 2, 0, 0, 2, 0, 2, 1, 2, 0, 2, 2, 2, 1, 2, 2, 2, 0, 2,
       2, 2, 2, 0, 2, 2, 2, 1, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2,
       1, 0, 1, 1, 2, 1, 0, 2, 2, 1, 0, 2, 2, 0, 2, 0, 2, 2, 2, 0, 2, 0,
       1, 0, 2, 1, 0, 0, 2, 0, 2, 1, 1, 2, 0, 0, 0, 0, 1, 1, 1, 2, 1, 0,
       1, 2, 2, 2, 2, 1, 0, 2, 2, 2, 0, 0, 2, 2, 0, 2, 2, 1, 0, 0, 2, 1,
       1, 2, 1, 0, 1, 0, 1, 1, 2, 0, 2, 2, 0, 2, 0, 1, 0, 0, 2, 2, 2, 0,
       2, 2, 1, 1, 1, 2, 1, 1, 2, 2, 2, 0, 0, 2, 0, 0, 2, 2, 2, 2, 1, 0,
       0, 1, 2, 1, 0, 0, 1, 2, 2, 2, 0, 0, 2, 0, 1, 2, 2, 2, 2, 0, 0, 0,
       0, 1, 1, 2, 1, 2, 2, 2, 1, 2, 1, 2, 1, 1, 2, 2, 1, 1, 0, 2, 2, 0,
       2, 1, 2, 1, 0, 2, 0, 1, 2, 0, 0, 2, 1, 1, 1, 0, 0, 0, 2, 2, 0, 2,
       2, 0, 2, 2, 0, 2, 2, 2, 2, 1, 2, 1, 2, 0, 2, 2, 0, 2, 2, 0, 2, 0,
       2, 2, 1, 2, 1, 1, 2, 0, 2, 1, 2, 2, 2, 2, 2, 0, 2])
```

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

In [ ]:
```
df['yestimated'] = yestimated
```

Print the number associated to each cluster

In [ ]:
```
df.yestimated.unique()
```

Out[ ]:
```
array([1, 2, 0])
```

Print the centroids

In [ ]:
```
km.cluster_centers_
```

Out[ ]:
```
array([[ 0.84734754, -8.24296729, -8.97104597, -6.93061653],
       [-7.11513023,  8.40545069,  9.28123803, -4.39332447],
       [ 5.81432702,  0.81127208,  7.02391021,  3.26588304]])
```

Print the intertia metric

In [ ]:
```
km.inertia_
```

Out[ ]: 9406.650107502028

Plot a scatter plot of your data using different color for each cluster. Also plot the centroids

In [ ]:
```
# Get a dataframe with the data of each clsuter
df1 = df[df.yestimated==0]
df2 = df[df.yestimated==1]
df3 = df[df.yestimated==2]


plt.scatter(df1.x1, df1.x2, label='Cluster 0', c='r', marker='o', s=32, alpha=0.3)
plt.scatter(df2.x1, df2.x2, label='Cluster 1', c='g', marker='o', s=32, alpha=0.3)
plt.scatter(df3.x1, df3.x2, label='Cluster 2', c='b', marker='o', s=32, alpha=0.3)


# Plot centrodides
plt.scatter(km.cluster_centers_[:,0], km.cluster_centers_[:,1], color='black', mark

plt.title('Scatter plot (for each cluster)')
plt.xlabel('x1')
plt.ylabel('x2')
plt.legend()
plt.show()
```

Questions

Provides a detailed description of your results

Your response:

I decided to use 3 clusters to do the KMeans Clustering, and the results were excellent. Each centroide clearly represents their cluster. The 2nd cluster (blue) has a wide range in the x1 axis, but I would not recommend adding a 4th cluster, I believe these number of K is optimal for aggrupations.

A5 Elbow plot

Compute the Elbow plot

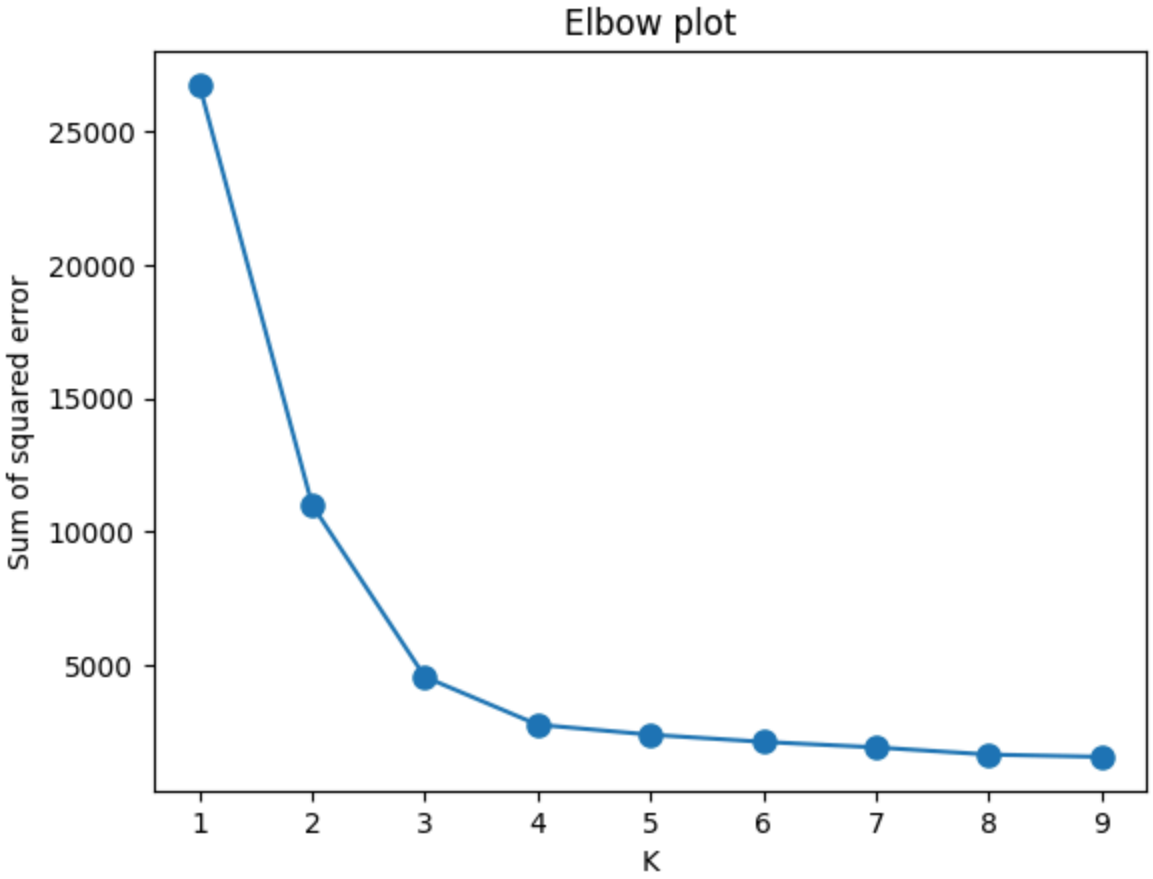
In [ ]:
```
# Intialize a list to hold sum of squared error (sse)
sse = []

# Define values of k
k_rng = range(1,10)

# For each k
for k in k_rng:
    # Create model
    km = KMeans(n_clusters=k, n_init="auto")
    # Do K-means clustering
    km.fit_predict(df[['x1','x2']])
    # Save sse for each k
    sse.append(km.inertia_)
```

In [ ]:
```
# Plot sse versus k
plt.plot(k_rng,sse, 'o-', markersize=8)

plt.title('Elbow plot')
plt.xlabel('K')
plt.ylabel('Sum of squared error')
plt.show()
```



Questions

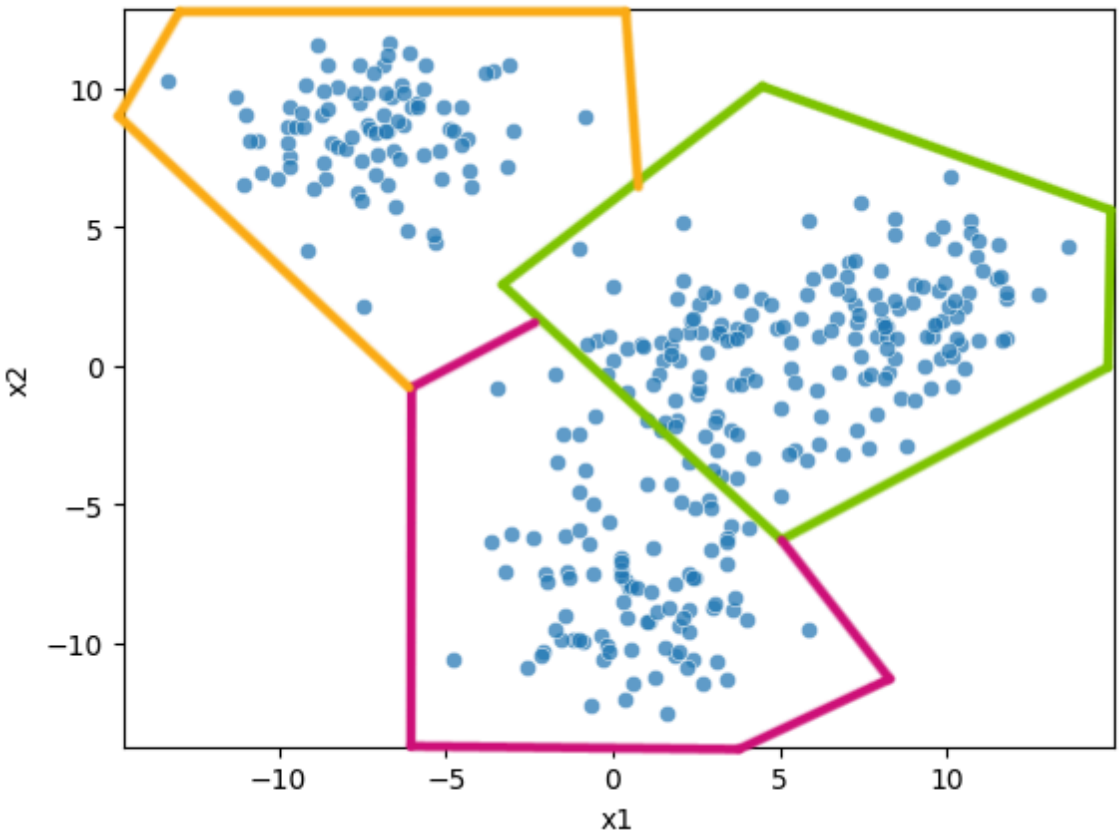
What is the best number of clusters K? (argue your response)

Your response: 3

Before 3 the squared error is too big, and after three the disminution of sse is minimal. And we must find the a balance between a small sse and the least clusters possible. We can see that the point of inflection is at 3.

Does this number of clusters agree with your inital guess? (argue your response, no problem at all if they do not agree)

Your response: It does. I selected this value after seeing the scatter plot. I considered the following groups:



Even though there were two clear groupings in the plot, the second one (lower right) had a big range of values for both x1 and x2. That's why I decided to separate it into 2.

In the end the groups ended up being similar to the actual clusters.

PART 2

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:

Creo que logré un nivel destacado en esta subcompetencia. Al ser una estudiante de programación he tenido la oportunidad de desarrollar esta habilidad durante toda mi

carrera, por lo que me siento segura de decir que domino esta subcompentencia. En la semana tec tuvimos la oportunidad de hacer ejercicios prácticos, lo que me permitió comprender las interacciones, pues era sencillo hacer pequeños cambios y ver como afectaba a los laboratorios.

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:

De la misma manera creo que logré un nivel destacado en esta subcompetencia. Para la evidencia final no tuve nínguna dificultad para comprender el funcionamiento de los modelos que utilizamos, y pude hacer un análisis profundo sobre cada paso, que significaba, sus consecuencias y su justificación. Sinceramente creo que también dominé esta habilidad.