d1-evidencia

March 22, 2024

Enrique Mora Navarro A01635459 ITC

[1]:
```python
# Define where you are running the code: colab or local
RunInColab          = False     # (False: no  | True: yes)

# If running in colab:
if RunInColab:
    # Mount your google drive in google colab
    from google.colab import drive
    drive.mount('/content/drive')

    # Find location
    #!pwd
    #!ls
    #!ls "/content/drive/My Drive/Colab Notebooks/MachineLearningWithPython/"

    # Define path del proyecto
    Ruta            = "/content/drive/My Drive/Colab Notebooks/"

else:
    # Define path del proyecto
    Ruta            = ""
```

[3]:
```python
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns


# Dataset url
url = Ruta + "/content/drive/MyDrive/R16/A01635459_X.csv"

# Load the dataset
df = pd.read_csv(url )
```

[4]:
```python
df.head(7)
```

[4]:
```
   Unnamed: 0        x1        x2        x3        x4
0           0 -5.866726 -7.236182 -3.662846  2.577765
1           1 -5.141411 -6.795414 -6.740861  1.242635
2           2 -2.242557 -6.920024  8.512946 -1.242033
3           3 -4.843926 -2.166504 -4.323611 -7.121348
4           4 -9.819970  5.065100 -8.529641 -5.482053
5           5 -5.838892 -0.606592 -5.103343 -5.714944
6           6 -4.008639 -3.879040 -3.265582 -5.878765
```

[5]:
```
df.tail(4)
```

[5]:
```
     Unnamed: 0        x1        x2        x3        x4
307         307 -4.047820 -8.558871 -7.274390  0.243761
308         308 -0.768518 -5.053981  7.761334 -1.707613
309         309 -1.800416 -2.493509  7.908761 -0.439710
310         310 -2.790776 -4.434742  5.917225 -1.528839
```

There are 311 rows and 5 columns on my df.

[6]:
```
df.shape
```

[6]:
```
(311, 5)
```

[7]:
```
print("Column names:")
for column_name in df.columns:
    print(column_name)
```

```
Column names:
Unnamed: 0
x1
x2
x3
x4
```

The data type of all columns is float64. The one that counts rows is an int64.

[11]:
```
df.dtypes
```

[11]:
```
Unnamed: 0      int64
x1            float64
x2            float64
x3            float64
x4            float64
dtype: object
```

The first column of my df is the same as an index column and the next four are floating quantitaive variables that are mainly negative numbers. I do not have more information about those variables.

[14]:
```
df.describe()
```

[14]:

	Unnamed: 0	x1	x2	x3	x4
count	311.000000	311.000000	311.000000	311.000000	311.000000
mean	155.000000	-4.284677	-3.143498	-3.353750	-3.519411
std	89.922189	2.736657	5.092951	5.740161	4.326610
min	0.000000	-12.941964	-11.969691	-12.993382	-13.484136
25%	77.500000	-6.258082	-7.223202	-7.487837	-7.106296
50%	155.000000	-4.355409	-4.434742	-5.217559	-3.692099
75%	232.500000	-2.312403	0.847994	0.299911	0.192239
max	310.000000	2.284891	10.179153	10.001712	7.311529

The minimum and maximum values of each column are:

- For x1: Min: -12.941964 Max: 2.284891
- For x2: Min: -11.969691 Max: 10.179153
- For x3: Min: -12.993382 Max: 10.001712
- For x4: Min: -11.969691 Max: 7.311529

The mean and std for each variable:

- For x1: Mean: -4.284677 std: 2.736657
- For x2: Mean: -3.143498 std: 5.092951
- For x3: Mean: -3.353750 std: 5.740161
- For x4: Mean: -3.519411 std: 4.326610

The 25%, 50%, and 75% are used to showcase the quartiles in statistics, they are used to determine the trends and location in data. Each percentage means that that number of the data is within that threshold. For example: the 50% quartile in x1 is -4.355 which mean 50% of that data lies below that number.

[15]:
```
df = df.rename(columns={"x1": "X1"})
df = df.rename(columns={"x2": "X2"})
df = df.rename(columns={"x3": "X3"})
df = df.rename(columns={"x4": "X4"})

df.head()
```

[15]:

	Unnamed: 0	X1	X2	X3	X4
0	0	-5.866726	-7.236182	-3.662846	2.577765
1	1	-5.141411	-6.795414	-6.740861	1.242635
2	2	-2.242557	-6.920024	8.512946	-1.242033
3	3	-4.843926	-2.166504	-4.323611	-7.121348
4	4	-9.819970	5.065100	-8.529641	-5.482053

[16]:
```
df = df.rename(columns={"X1": "x1"})
df = df.rename(columns={"X2": "x2"})
df = df.rename(columns={"X3": "x3"})
df = df.rename(columns={"X4": "x4"})
```

```
df.head()
```

[16]:

	Unnamed: 0	x1	x2	x3	x4
0	0	-5.866726	-7.236182	-3.662846	2.577765
1	1	-5.141411	-6.795414	-6.740861	1.242635
2	2	-2.242557	-6.920024	8.512946	-1.242033
3	3	-4.843926	-2.166504	-4.323611	-7.121348
4	4	-9.819970	5.065100	-8.529641	-5.482053

[18]:
```
column_data = df['x1']
column_data
```

[18]:
```
0     -5.866726
1     -5.141411
2     -2.242557
3     -4.843926
4     -9.819970
         …
306   -6.236235
307   -4.047820
308   -0.768518
309   -1.800416
310   -2.790776
Name: x1, Length: 311, dtype: float64
```

[19]:
```
column_data = df.x2
column_data
```

[19]:
```
0     -7.236182
1     -6.795414
2     -6.920024
3     -2.166504
4      5.065100
         …
306   -4.434358
307   -8.558871
308   -5.053981
309   -2.493509
310   -4.434742
Name: x2, Length: 311, dtype: float64
```

[21]:
```
slice_of_data = df.iloc[61:73, 1:3]
slice_of_data
```

[21]:

	x1	x2
61	-4.541766	4.069373

```
62 -5.901486  5.045851
63 -1.060076 -3.629479
64 -8.448120  0.239896
65 -7.294902  5.297757
66 -0.448759 -7.035900
67 -5.498900 -6.843808
68 -6.168159  2.034998
69 -2.771498 -9.714856
70 -0.118291 -3.542894
71 -4.438987 -4.670491
72 -7.280456  5.602641
```

[27]:
```
selected_columns = df.iloc[:, 1:3]

null_count = selected_columns.isnull().sum()
not_null_count = selected_columns.notnull().sum()

total_rows = len(df)
total_null_count = null_count.sum()
total_not_null_count = not_null_count.sum()
total_final = total_null_count + total_not_null_count

print(total_not_null_count)
print(total_null_count)
print(total_rows)
print(total_final)
```

```
622
0
311
622
```

We can see that there are 622 not null values.

[ ]:
```
df = df.drop(columns=['x4'])
```

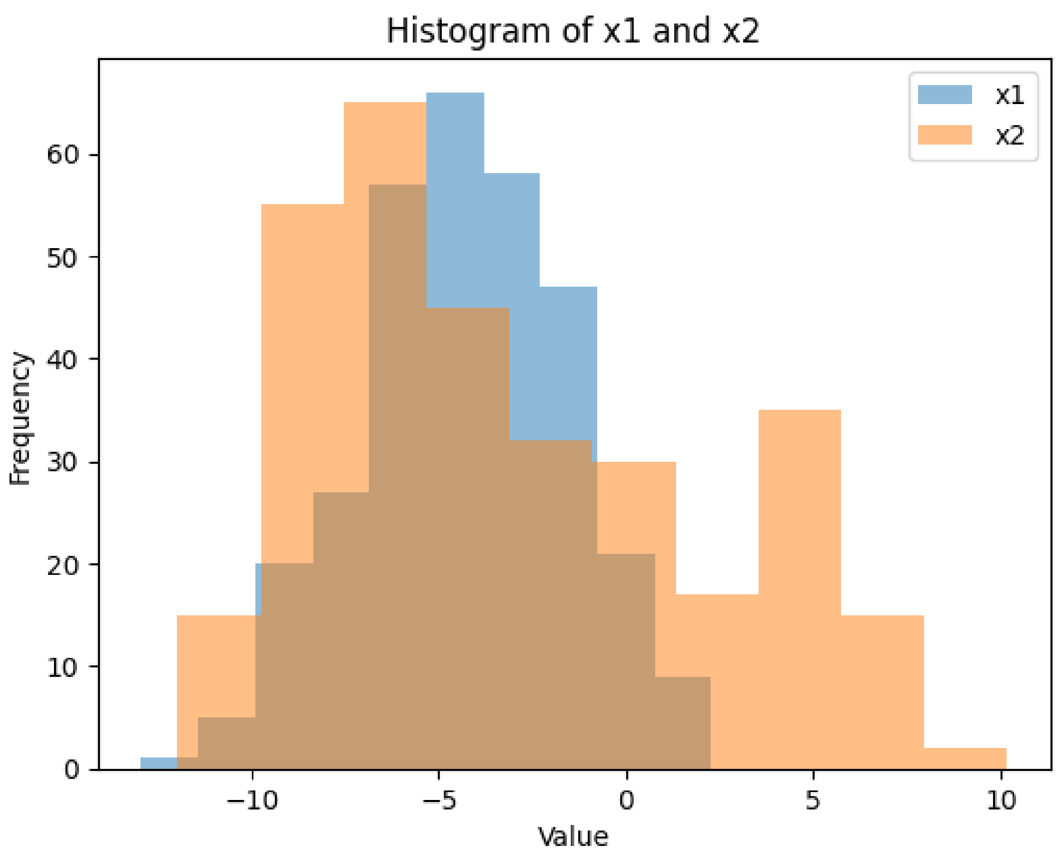
[38]:
```
df #we can see the database without the x4 column
```

[38]:
```
     Unnamed: 0        x1        x2        x3
0             0 -5.866726 -7.236182 -3.662846
1             1 -5.141411 -6.795414 -6.740861
2             2 -2.242557 -6.920024  8.512946
3             3 -4.843926 -2.166504 -4.323611
4             4 -9.819970  5.065100 -8.529641
..          ...       ...       ...       ...
306         306 -6.236235 -4.434358 -5.099410
307         307 -4.047820 -8.558871 -7.274390
308         308 -0.768518 -5.053981  7.761334
```

```
309         309 -1.800416 -2.493509  7.908761
310         310 -2.790776 -4.434742  5.917225

[311 rows x 4 columns]
```

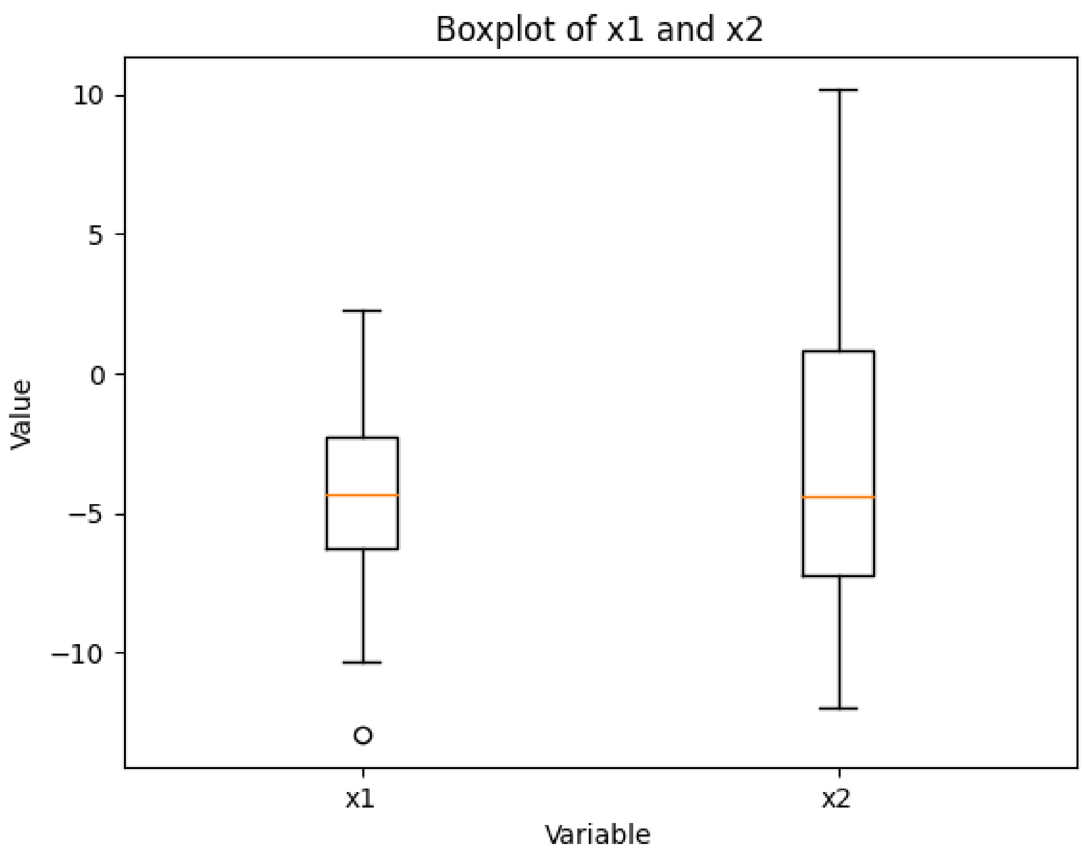
I have a database of 311 rows x 5 columns in the beginning. Each of the 4 variables: x1, x2, x3 and x4 are floats with mostly negative numbers conforming them. In the end, we have discarded the last column, which is x4.

[39]:
```
plt.hist(df['x1'], bins=10, alpha=0.5, label='x1')
plt.hist(df['x2'], bins=10, alpha=0.5, label='x2')

# Add labels and title
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Histogram of x1 and x2')

# Add legend
plt.legend()

# Show the plot
plt.show()
```



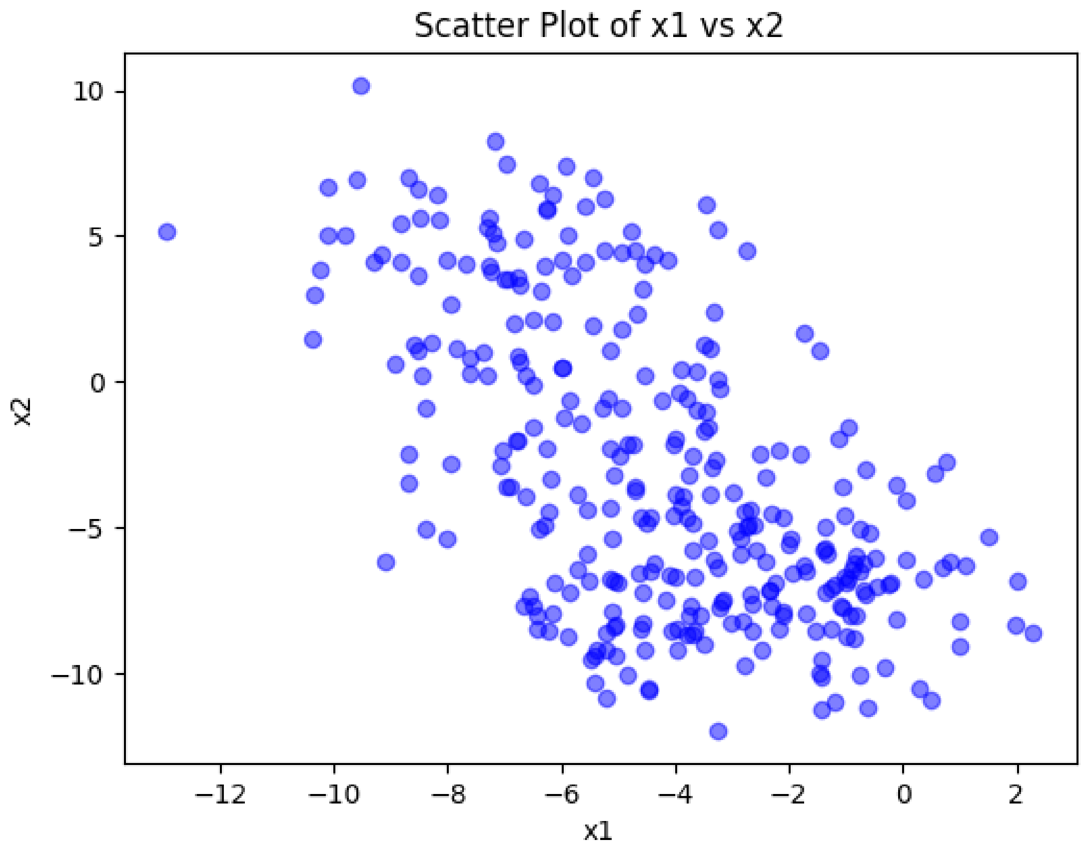
While both variables have similar minimum values and a high concentration of entries with a value near -5, x2 has a more extended distribution in terms of value. The maximum value of x2 could be more than 10 and the max value for x1 does not reach 3.

[40]:
```
plt.boxplot([df['x1'], df['x2']], labels=['x1', 'x2'])

plt.xlabel('Variable')
plt.ylabel('Value')
plt.title('Boxplot of x1 and x2')

plt.show()
```



[41]:
```
plt.scatter(df['x1'], df['x2'], c='blue', alpha=0.5)

plt.xlabel('x1')
plt.ylabel('x2')
plt.title('Scatter Plot of x1 vs x2')
```

```
plt.show()
```



The conclusion remains the same as the one with the histogram: While both variables have similar minimum values and a high concentration of entries with a value near -5, x2 has a more extended distribution in terms of value. The maximum value of x2 could be more than 10 and the max value for x1 does not reach 3.

[42]:
```
# Import sklearn KMeans
from sklearn.cluster import KMeans

# Define number of clusters
K  = 2

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

yestimated = km.fit_predict(df[['x1','x2']] )

yestimated
```

[42]:
```
array([0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1,
       0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 0, 0], dtype=int32)
```

[52]:
```python
df['yestimated'] = yestimated
df
```

[52]:
```
     Unnamed: 0        x1        x2        x3  clusterInfo  yestimated
0             0 -5.866726 -7.236182 -3.662846            0           0
1             1 -5.141411 -6.795414 -6.740861            0           0
2             2 -2.242557 -6.920024  8.512946            0           0
3             3 -4.843926 -2.166504 -4.323611            0           0
4             4 -9.819970  5.065100 -8.529641            1           1
..          …         …         …         …          …           …
306         306 -6.236235 -4.434358 -5.099410            0           0
307         307 -4.047820 -8.558871 -7.274390            0           0
308         308 -0.768518 -5.053981  7.761334            0           0
309         309 -1.800416 -2.493509  7.908761            0           0
310         310 -2.790776 -4.434742  5.917225            0           0

[311 rows x 6 columns]
```

[53]:
```python
df.yestimated.unique()
```

[53]:
```
array([0, 1], dtype=int32)
```

[48]:
```python
km.cluster_centers_
```

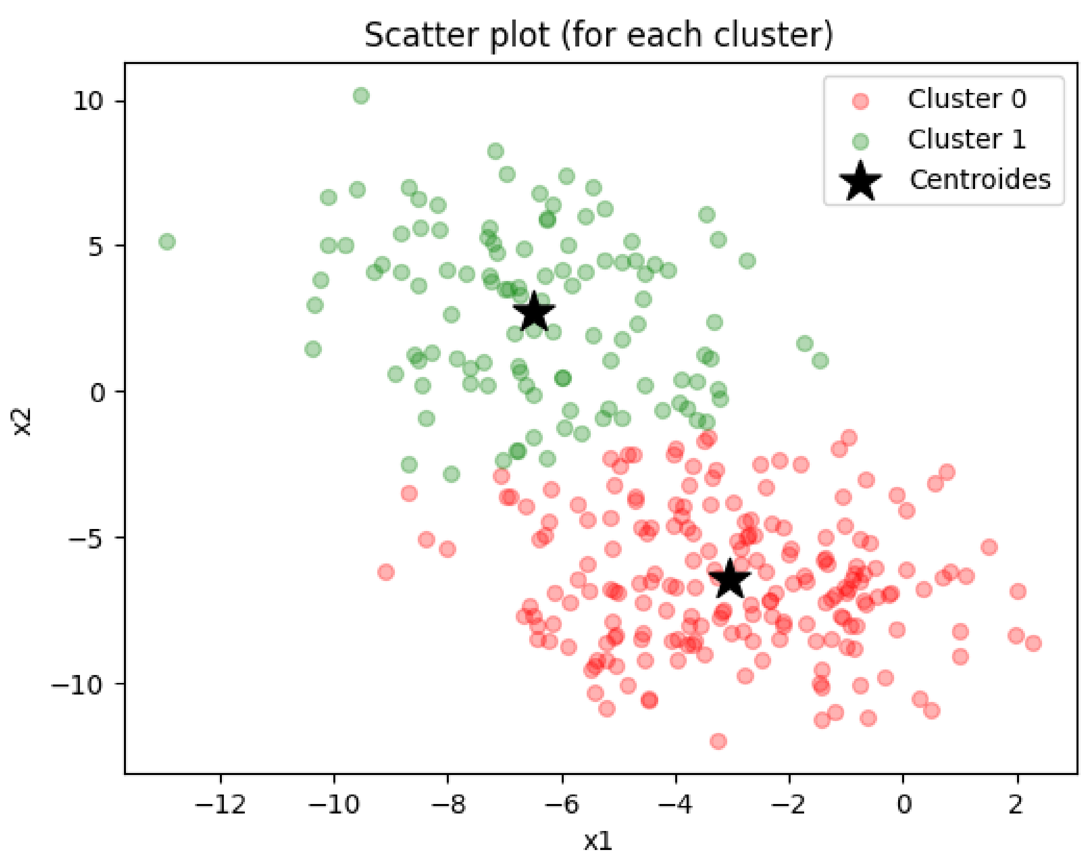
[48]:
```
array([[-3.0482815 , -6.4556123 ],
       [-6.48148662,  2.74141871]])
```

[49]:
```python
km.inertia_
```

[49]:
```
3455.929842270082
```

[54]:

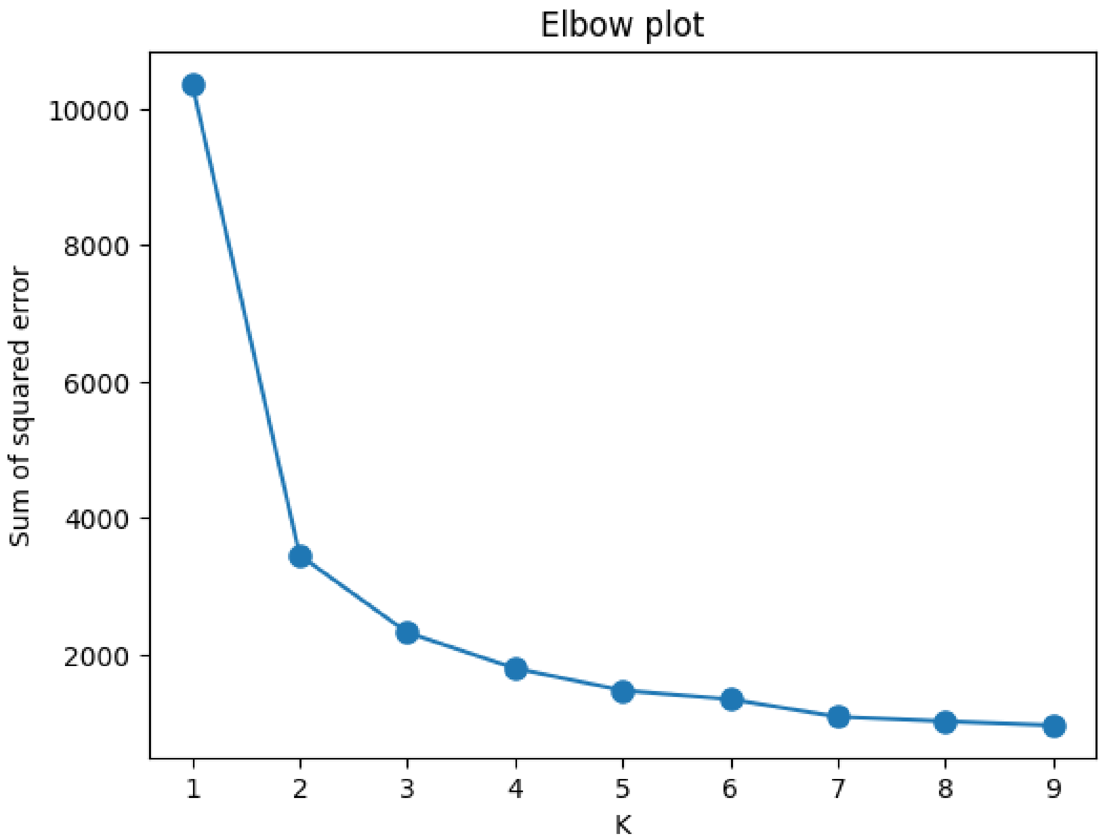
```python
# Get a dataframe with the data of each clsuter
df1 = df[df.yestimated==0]
df2 = df[df.yestimated==1]


# Scatter plot of each cluster
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)

# Plot centrodides
plt.scatter(km.cluster_centers_[:,0], km.cluster_centers_[:,1], color='black',␣
↪marker='*', label='Centroides', s=256)

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



In this scatter plot of my kmeans clustering, we can see we have correctly divided our dataset in 2 clusters and have included the centroids which seem to be accurate.

[55]:
```
# 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_)
```

[56]:
```
# 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()
```



As we can see in our elbow plot, the elbow point lies on 2, which means our best option for number o k = clusters is 2.

This cluster number agrees with my initial guess which was obviously 2. I selected this number of clusters as we were working with only 2 variables with a very similar distribution.

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: Después de esta semana tec, soy capaz de aplicar herramientas de análisis de datos para observar la relación entre las variables que se me presentan. Esto utilizando métodos de clustering como el kmeans.

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: Con las competencias que desarrollé a lo largo de la semana tec y mis competencias previas en programación utilizando python fui capaz de construir los modelos solicitados de clustering utilizando las bases de datos otorgadas y herramientas como Google Colab.