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: Emilio Berber Maldonado
- ID: A01640603
- My carreer: ITC

Importing libraries

```
# Import the packages that we will be using
import pandas as pd
import seaborn as sns ## graficas avanzadas
import matplotlib.pyplot as plt

from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

PART 1

Use your assigned dataset

A1 Load data

```
# Define where you are running the code: colab or local
RunInColab          = True     # (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/MachineLearningWithPython/"

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

```
# url string that hosts our .csv file
```

```
url = "/content/drive/MyDrive/A01640603.csv"
# Read the .csv file and store it as a pandas Data Frame

data = pd.read_csv(url)
data
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/dri
```

	Unnamed: 0	x1	x2
0	0	-0.982189	-0.231799
1	1	-0.606211	0.481379
2	2	0.851685	-0.426067
3	3	-0.970791	-0.182957
4	4	-0.709407	0.185979
...	...	...	...
2043	2043	0.691960	0.812012
2044	2044	-0.018004	0.811456
2045	2045	0.778800	-0.409516
2046	2046	-0.808459	-0.694742
2047	2047	-0.626861	0.321339

2048 rows × 3 columns

A2 Data managment

Print the first 7 rows

```
data.loc[0:6, :]
```

	Unnamed: 0	x1	x2
0	0	-0.982189	-0.231799
1	1	-0.606211	0.481379
2	2	0.851685	-0.426067
3	3	-0.970791	-0.182957
4	4	-0.709407	0.185979
5	5	1.002839	-0.179887
6	6	-0.004695	-0.993603

Print the first 4 last rows

```
data.tail(4)
```

	Unnamed: 0	x1	x2
2044	2044	-0.018004	0.811456
2045	2045	0.778800	-0.409516
2046	2046	-0.808459	-0.694742
2047	2047	-0.626861	0.321339

How many rows and columns are in your data?

Use the `shape` method

```
data.shape
```

```
(2048, 3)
```

Print the name of all columns

Use the `columns` method

```
data.columns
```

```
Index(['Unnamed: 0', 'x1', 'x2'], dtype='object')
```

What is the data type in each column

Use the `dtypes` method

```
data.dtypes
```

```
Unnamed: 0      int64
x1            float64
x2            float64
dtype: object
```

What is the meaning of rows and columns?

```
# Your responses here

# 1)  x1 son valores decimales

# 2)  x2 son valores decimales

# 3) Unnamed: 0 es el número de fila en la que se encuentra

#...
```

Print a statistical summary of your columns

```
data.describe()
```

	Unnamed: 0	x1	x2
count	2048.000000	2048.000000	2048.000000
mean	1023.500000	0.002297	-0.001386
std	591.350996	0.642631	0.646921
min	0.000000	-1.210385	-1.190506
25%	511.750000	-0.607794	-0.617366
50%	1023.500000	-0.009975	0.000304
75%	1535.250000	0.621137	0.615514
max	2047.000000	1.206925	1.237435

```
# 1) What is the minumum and maximum values of each variable
  # [x1] Mínimo = -1.210385  |   Máximo = 1.206925
  # [x2] Mínimo = -1.190506  |   Máximo = 1.237435
  # [ID] Mínimo = 0          |   Máximo = 2047
# 2) What is the mean and standar deviation of each variable
  # [x1] Promedio = 0.002297   |   STD = 0.646921
  # [x2] Promedio = -0.001386  |   STD = 0.646921
  # [ID] Promedio = 1023.5     |   STD = 591.350996
# 3) What the 25%, 50% and 75% represent?
    #Son los cuartiles, se usan en estadística para analizar diferentes muestreos de los conjuntos de datos, se acomodan de mayor a menor y s
```

Rename the columns using the same name with capital letters

```
data = data.rename(columns={"Unnamed: 0": "ID"})
data = data.rename(columns={ "x1" : "X1"})
data = data.rename(columns={"x2" : "X2"})
data
```

	ID	X1	X2
0	0	-0.982189	-0.231799
1	1	-0.606211	0.481379
2	2	0.851685	-0.426067
3	3	-0.970791	-0.182957
4	4	-0.709407	0.185979
...	...	...	...
2043	2043	0.691960	0.812012
2044	2044	-0.018004	0.811456
2045	2045	0.778800	-0.409516
2046	2046	-0.808459	-0.694742
2047	2047	-0.626861	0.321339

2048 rows × 3 columns

Rename the columns to their original names

```
data = data.rename(columns={"ID": "Unnamed: 0"})
data = data.rename(columns={ "X1" : "x1"})
data = data.rename(columns={"X2" : "x2"})
data
```

	Unnamed: 0	x1	x2
0	0	-0.982189	-0.231799
1	1	-0.606211	0.481379
2	2	0.851685	-0.426067
3	3	-0.970791	-0.182957
4	4	-0.709407	0.185979
...	...	...	...
2043	2043	0.691960	0.812012
2044	2044	-0.018004	0.811456
2045	2045	0.778800	-0.409516
2046	2046	-0.808459	-0.694742
2047	2047	-0.626861	0.321339

2048 rows × 3 columns

Use two different alternatives to get one of the columns

```
data["x1"]
```

```
0      -0.982189
1      -0.606211
2       0.851685
3      -0.970791
4      -0.709407
          ...
2043    0.691960
2044   -0.018004
2045    0.778800
2046   -0.808459
2047   -0.626861
Name: x1, Length: 2048, dtype: float64
```

```
data.x1
```

```
0      -0.982189
1      -0.606211
2       0.851685
3      -0.970791
```

```
4      -0.709407
          ...
2043    0.691960
2044   -0.018004
2045    0.778800
2046   -0.808459
2047   -0.626861
Name: x1, Length: 2048, dtype: float64
```

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

```
data.loc[62 : 72, "x1" : "x2"]
```

	x1	x2
62	-0.287650	-1.107696
63	-0.507232	-0.814804
64	-0.851486	-0.390649
65	-0.632028	0.452651
66	-0.325903	-0.800738
67	-0.852220	0.287445
68	0.489039	-0.737605
69	-0.265573	-1.052599
70	0.092001	-0.906418
71	0.785526	0.126705
72	0.391479	0.602723

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

```
nc1 = data['x1'].isnull().sum()
nc2 = data['x2'].isnull().sum()

nnc1 = data['x1'].notnull().sum()
nnc2 = data['x2'].notnull().sum()

total_rows = len(data)

print("x1 total: " +str(nc1+nnc1))
print("x2 total: " +str(nc2+nnc2))
print("Registros: "+str(data.shape[0]))
```

```
x1 total: 2048
x2 total: 2048
Registros: 2048
```

Discard the last column

```
data.drop("Unnamed: 0", axis=1, inplace = True)
data
```

	x1	x2
0	-0.982189	-0.231799
1	-0.606211	0.481379

Questions

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

Your response: Son dos variables flotantes, x1 y x2 que van desde los negativos hasta los positivos -1.21 como mínimo y 1.23 como máximo. Y ahora sin el ID repetido en la primera columna.

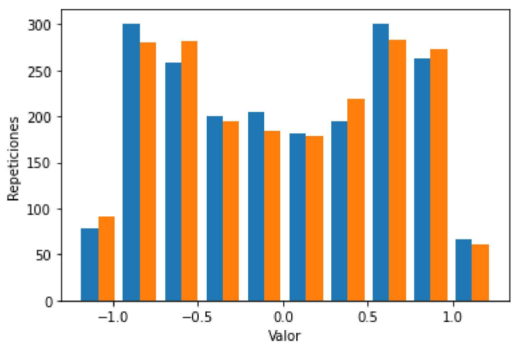
A3 Data visualization

2046	-0.808459	-0.694742

Plot in the same figure the histogram of the two variables

2048 rows × 2 columns

```
plt.hist([data["x1"],data["x2"]])
plt.xlabel("Valor")
plt.ylabel("Repeticiones")
plt.show()
```

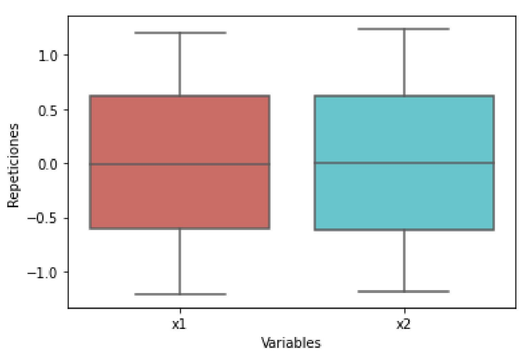


Based on this plots, provide a description of your data:

Your response here: Los datos siguen un patrón que forma una curva en los valores del centro, con extremos aparte con muchas menos repeticiones.

Plot in the same figure the boxplot of the two variables

```
x2bp  = sns.boxplot(data = data, orient = "v", palette ="hls")
plt.ylabel("Repeticiones")
plt.xlabel("Variables")
plt.show()
```

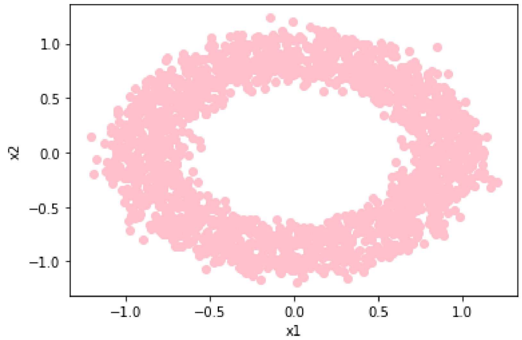


Scatter plot of the two variables

```
x = data.x1
y = data.x2
```

```
plt.scatter(x, y, c ="pink")
plt.xlabel("x1")
plt.ylabel("x2")
```

```
Text(0, 0.5, 'x2')
```



Questions

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

Your response: Vemos que los valores son muy similares con el boxplot y el scatterplot nos muestra que los datos graficados terminan formando una circunferencia.

A4 Kmeans

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

```
# Import sklearn KMeans
from sklearn.cluster import KMeans

# Define number of clusters
K  =  2 # Let's assume there are 2,3,4,5...? clusters/groups

# Create/initialize the Kmeans box/object
km = KMeans(n_clusters=K, n_init="auto")

# Do K-means clustering (assing each point in the dataset to a cluster)
yestimated = km.fit_predict(data)

# Print estimated cluster of each point in the dataset
yestimated
```

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

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

```
data["yestimated"] = yestimated
```

Print the number associated to each cluster

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

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

Print the centroids

```
km.cluster_centers_
```

```
array([[ 0.48726234,  0.29910133],
       [-0.49126837, -0.30720302]])
```

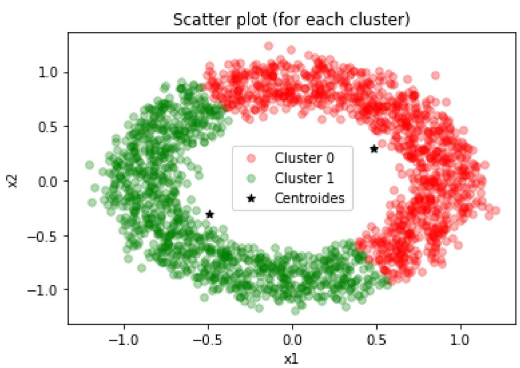
Print the intertia metric

```
km.inertia_
```

```
1023.5750191678404
```

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

```
# Get a dataframe with the data of each clsuter
df1 = data[data.yestimated == 0]
df2 = data[data.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 centroides
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1], color = "black", marker = '*', label = "Centroides")

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

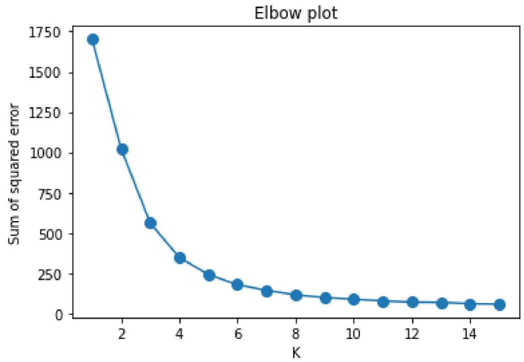
Your response: La gráfica muestra los datos dividos excatamente por la mitad, asignando a cada una un centroide distinto y por lo tanto un cluster diferente. Se ubican de manera aleatoria hasta que dejen de cambiar su posición y se asignan los vértices más cercanos a cada uno.

A5 Elbow plot

Compute the Elbow plot

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

# Define values of k
k_rng = range(1,16)

# 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(data[["x1", "x2"]])
  # Save see for each k
  sse.append(km.inertia_)


# 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")
```

```
Text(0, 0.5, 'Sum of squared error')
```



Questions

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

Your response: En los valores 8 o 9 se ve como el valor del error cuadrático deja de disminuir drásticamente, 8 clusters debe ser la mejor opción.

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

Your response: No, al ser una figura circular había muchas posibilidades para el clustering y decidí hacer grupos muy grandes con dos clusters.

PART 2

Load and do clustering using the "digits" dataset

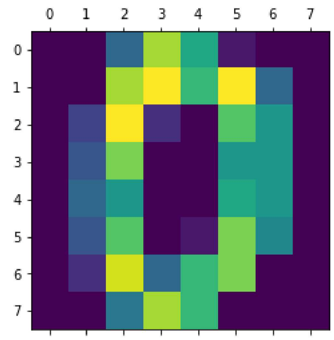
1) Load the dataset using the "load_digits()" function from "sklearn.datasets"

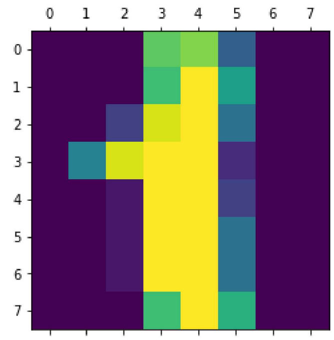
```
from sklearn.datasets import load_digits

digits = load_digits()
```

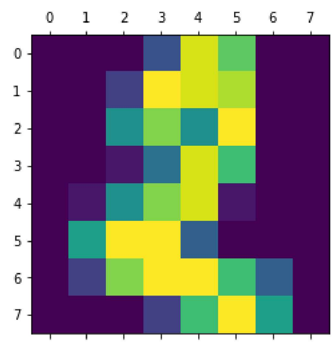
2) Plot some of the observations

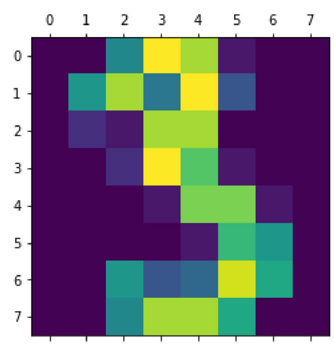
```
plt.viridis()
for i in range (10):
  plt.matshow(digits.images[i])
```

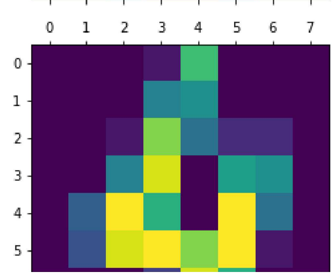
```
<Figure size 432x288 with 0 Axes>
```







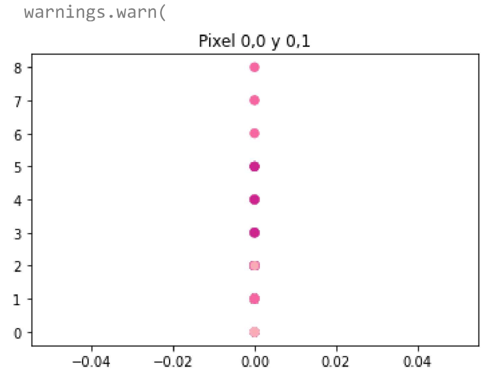




3) Do K means clustering

```
kmeans = KMeans(n_clusters=10)

# Fit the KMeans model to the data
kmeans.fit(digits.data)

# Get the predicted labels and cluster centers
yestimated = kmeans.predict(digits.data)
centers = kmeans.cluster_centers_
plt.scatter(digits.data[:, 0], digits.data[:, 1], c = labels, cmap="RdPu")
plt.title("Pixel 0,0 y 0,1")
plt.show()

digits["yestimated"] = yestimated
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value
  warnings.warn(
```



4) Verify your results in any of the observations

```
for i in range (10):
  print("Real: "+str(digits.target[i]))
  print("KMean: "+str(digits.yestimated[i]))
```