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:Raúl Romero Martínez
- ID: A01770083
- My carreer: ITC

Importing libraries

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

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/A01770083/"

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

```
Mounted at /content/drive
```

```
# Dataset url
url = Ruta + "datasets/Evidencia/A01770083.csv"

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

A2 Data managment

```
# Delete first column (unnecessary column)
df.drop("Unnamed: 0", axis=1, inplace = True)
```

Print the first 7 rows

```
df.head(7)
```

	x1	x2
0	-0.273650	-0.029398
1	-0.511670	0.795239
2	0.399517	-0.355576
3	-0.395029	-1.001311
4	0.304692	-1.414656
5	-0.449601	-2.119329
6	-0.172294	0.835551

Print the first 4 last rows

```
df.tail(4)
```

	x1	x2
1596	0.409286	-0.697285
1597	0.300306	-0.854893
1598	-0.350030	-1.076114
1599	-0.871391	-0.430677

How many rows and columns are in your data?

Use the `shape` method

```
df.shape
```

```
(1600, 2)
```

Print the name of all columns

Use the `columns` method

```
df.columns
```

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

What is the data type in each column

Use the `dtypes` method

```
df.dtypes
```

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

What is the meaning of rows and columns?

```
# Your responses here
```

```
# 1) Rows are the observations

# 2) Columns are the variables

# 3)

#...
```

Print a statistical summary of your columns

```
df.describe()
```

	x1	x2
count	1600.000000	1600.000000
mean	-0.250325	-0.501330
std	0.500185	0.867771
min	-1.196778	-2.225543
25%	-0.699918	-1.077413
50%	-0.248803	-0.487908
75%	0.209978	0.074860
max	0.786641	1.211092

```
# 1) What is the minumum and maximum values of each variable
#           x1          x2
#   MIN: -1.1967     -2.2255
#   MAX:  0.7866      1.2120

# 2) What is the mean and standar deviation of each variable
#            x1          x2
#   MEAN: -0.2503     -0.5013
#   STD :  0.5001      0.8677

# 3) What the 25%, 50% and 75% represent?
#   The percentiles, stadistics with different % of all the data
```

Rename the columns using the same name with capital letters

```
df2 = df.rename(columns={"x1": "X1","x2" : "X2"})
df2.head()
```

	X1	X2
0	-0.273650	-0.029398
1	-0.511670	0.795239
2	0.399517	-0.355576
3	-0.395029	-1.001311
4	0.304692	-1.414656

Rename the columns to their original names

```
df2·=·df.rename(columns={"X1":·"x1","X2"·:·"x2"})
df2.head()
```

Use two different alternatives to get one of the columns

```
#Alternative 1

# Create a column data
NewColumnData = df.x1/df.x1

# Insert that column in the data frame
df.insert(2, "ColumnInserted", NewColumnData, True)

df.head()
```

	x1	x2	ColumnInserted	ColumnInserted	ColumnInserted	ColumnInserted	ColumnInserted	ColumnInse
0	-0.273650	-0.029398	1.0	1.0	1.0	1.0	1.0	
1	-0.511670	0.795239	1.0	1.0	1.0	1.0	1.0	
2	0.399517	-0.355576	1.0	1.0	1.0	1.0	1.0	
3	-0.395029	-1.001311	1.0	1.0	1.0	1.0	1.0	
4	0.304692	-1.414656	1.0	1.0	1.0	1.0	1.0	

```
#Alternative 1

# Create a column data
#NewColumnData = df.x1/df.x1

# Insert that column in the data frame
df.assign()

df.head()
```

	x1	x2	ColumnInserted	ColumnInserted
0	-0.273650	-0.029398	1.0	1.0
1	-0.511670	0.795239	1.0	1.0
2	0.399517	-0.355576	1.0	1.0
3	-0.395029	-1.001311	1.0	1.0
4	0.304692	-1.414656	1.0	1.0

```
#Delete the new columns
df.drop("ColumnInserted", axis=1, inplace = True)
df.head(2)
```

	x1	x2
0	-0.27365	-0.029398
1	-0.51167	0.795239

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

```
df.iloc[62:73,:]
```

	x1	x2
62	0.228785	-0.388111
63	-0.705303	0.489000
64	0.265845	-1.535966

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

66	-0.252154	0.827089

```
#Returns the number of non-null values in each data frame column

df.count()
```

```
x1    1600
x2    1600
dtype: int64
```

71	-0.900811	0.293573

Discard the last column

Questions

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

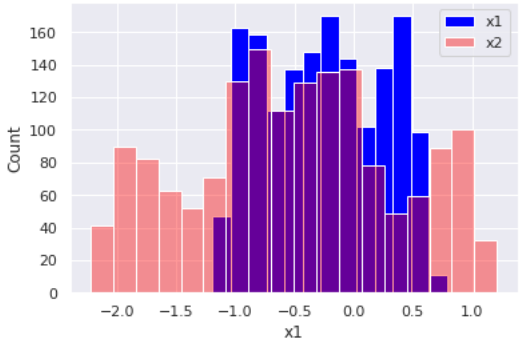
Your response:

El dataset que tengo contiene 2 columnas o variables que son "x1" y "x2" y cada una de ella cuenta con 1600 observaciones de las cuales todas contienen un valor numérico flotante positivo o negativo.

A3 Data visualization

Plot in the same figure the histogram of the two variables

```
sns.histplot(data=df, x="x1", color="blue", label="x1", kde=False, alpha = 1)
sns.histplot(data=df, x="x2", color="red", label="x2", kde=False, alpha = 0.4)

plt.legend()
plt.show()
```



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

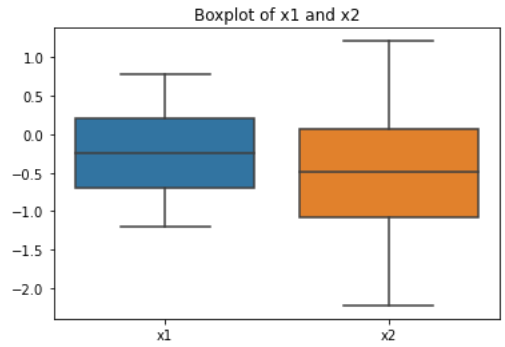
Your response here:

Segun las graficas obtenidas de primeras se observa que "x1" tiene más cantidad de valores grandes que "x2" , además,"x2" tiene más variedad en el valor de los números dentro de la variable y con valores más negativos y más positivos que "x1".

Plot in the same figure the boxplot of the two variables

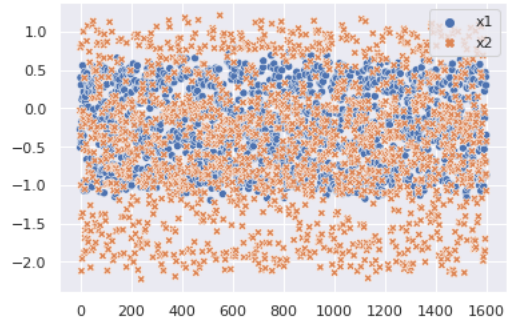
```
x = df.loc[:, ["x1","x2"]]

x2bp = sns.boxplot(data = x, orient = "V")
```

```
x2bp.set_title("Boxplot of x1 and x2")
plt.show()
```



Scatter plot of the two variables

```
sns.scatterplot(data = df)
plt.show()
```



Questions

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

Your response:

En el dataset se observa que los valores de "x2" se encuentran más dispersos que los de "x1" que muchos se concentran en el mismo rango de valores, y que en "x2" tenemos muchos valores que son más negativos que los de "x1".

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

# Creates 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(df[['x1','x2']])

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

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

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

```
df['yestimated'] = yestimated
df
```

	x1	x2	yestimated
0	-0.273650	-0.029398	0
1	-0.511670	0.795239	0
2	0.399517	-0.355576	1
3	-0.395029	-1.001311	1
4	0.304692	-1.414656	1
...	...	...	...
1595	-0.684115	-0.880091	1
1596	0.409286	-0.697285	1
1597	0.300306	-0.854893	1
1598	-0.350030	-1.076114	1
1599	-0.871391	-0.430677	0

1600 rows × 3 columns

Print the number associated to each cluster

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

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

Print the centroids

```
km.cluster_centers_
```

```
array([[-0.57940129,  0.19950037],
       [ 0.07466397, -1.19345523]])
```

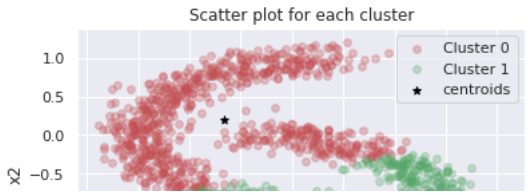
Print the intertia metric

```
km.inertia_
```

```
656.909760477346
```

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 = 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)

plt.scatter(km.cluster_centers_[:,0], km.cluster_centers_[:,1],color='black',marker ='*',label = 'centroids')

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

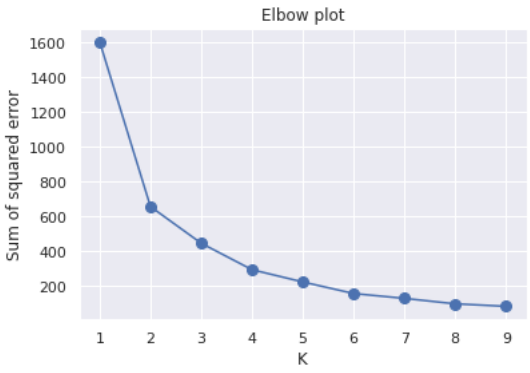
Your response:

Para esta seccion considere que hay solo 2 clusters, y viendo los resultados considero que esta cantidad de clusters arroja buenos y deseados resultados observando la ultima grafica

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(1,10)

# For each k
for k in k_rng:
  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, '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: El número ideal de clusters deberia ser entre 5 o 6 ya que en los anteriores se observa que de uno a otro hay un cambio notable que es un cluster por cada uno pero al llegar al 5 el cambio ya no es tanto y no es necesario agragar más clusters.

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

Your response:No, yo elegi como inicial 2 clusters basandome en que segun graficas que obtuve, habia una concentracion de muchos datos y otra con menor cantidad de datos.

PART 2

Load and do clustering using the "digits" dataset

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

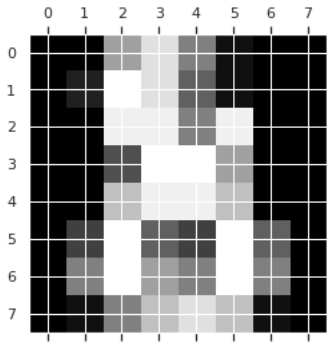
```
from sklearn.datasets import load_digits


dgts = load_digits()
print(dgts.data.shape)
import matplotlib.pyplot as plt
```

```
(1797, 64)
```

2) Plot some of the observations

```
plt.gray()
plt.matshow(dgts.images[1796])
plt.show()
```

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



3) Do K means clustering

```
digits = load_digits()
#digits

# Crear dataframe
dfx = pd.DataFrame(digits['data'])
#dfx['label'] = digits['target']
dfx.head(3)
```

	0	1	2	3	4	5	6	7	8	9	...	54	55	56	57	58	59	60
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	6.0	13.0	10.0
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	11.0	16.0
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	...	5.0	0.0	0.0	0.0	0.0	3.0	11.0

3 rows × 64 columns

```
# 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

# Creates 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(dfx)

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

```
yestimated
```

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

```
dfx['yestimated'] = yestimated
dfx.head(5)
```

	0	1	2	3	4	5	6	7	8	9	...	55	56	57	58	59	60	61	62	63	yestimated
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	6.0	13.0	10.0	0.0	0.0	0.0	0
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	11.0	16.0	10.0	0.0	0.0	1
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	3.0	11.0	16.0	9.0	0.0	1
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	...	0.0	0.0	0.0	7.0	13.0	13.0	9.0	0.0	0.0	1
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	2.0	16.0	4.0	0.0	0.0	0

5 rows × 65 columns

```
dfx.yestimated.unique()
```

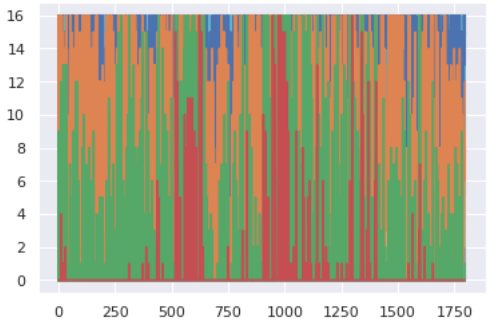
```
array([0, 1], dtype=int32)
```

```
km.cluster_centers_
```

```
km.inertia_
```

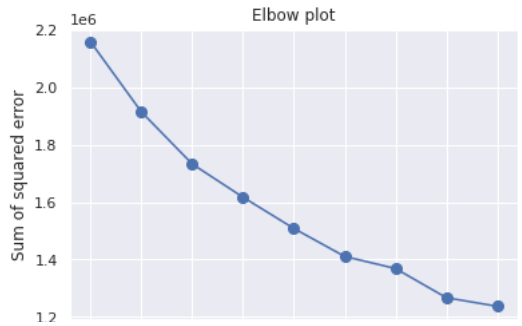
```
1914619.6175501016
```

```
plt.plot(dfx)
plt.show()
```



4) Verify your results in any of the observations

```
# 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:
  km = KMeans(n_clusters = k, n_init = "auto")
  km.fit_predict(dfx)
  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")

plt.show()
```



Questions

Provides a detailed description of your results.

Your response:Para esta parte lo primero que se hizo fue usar load_digits para cargar los datos y transformarlos en una imagen, obteniendo un nuevo dataframe de 1797 por 64, ya teniendo esa información transformamos de nuevo esos datos a pandas para poder leerlos y manejarlos, para hacerle el proceso de Kmeans y obtener nuevos clusters con esta nueva información.

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: Considero cumplir con esta competenecia, esta semana me ayudo a reforzar algunos conocimiento previos que he visto en la carrera, pero ahora creo que ya se de verdad como se interpretan y como manejarlos para generar el modelo necesario.

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 esta entrega final consider que cumplo de manera satisfactoria con esta subcompetencia, ya que gracias al trabajo de esta semana, pude desarrollar de manera satisfactoria con esta entrega final que cubre todos los aspectos que vimos a lo largo de la semana.

✓ 2 s completado a las 10:10