ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

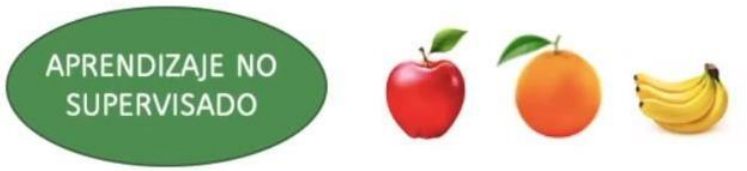
INTELIGENCIA ARTIFICIAL

ASIGNATURA: Inteligencia Artificial
PROFESOR:
PERÍODO ACADÉMICO:

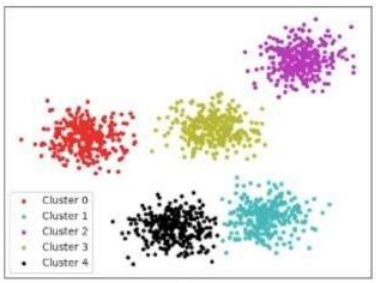
Tarea - 2

TÍTULO:

APRENDIZAJE NO SUPERVISADO



No se conoce la variable target



PROPÓSITO DE LA PRÁCTICA

Familiarizar al estudiante con el aprendizaje no supervisado (cluster Kmeans).

OBJETIVO GENERAL

Analizar, procesar y clasificar la data proporcionada para distinguir vinos acorde a características de los mismos.

OBJETIVOS ESPECÍFICOS

- Uso de librerías Python para aprendizaje no supervisado.
- Analizar la data disponible
- Desarrollar el código clasificación mediante aprendizaje no supervisado.
- Visualizar y sacar conclusiones de los resultados obtenidos.

DESAROLLO

1. Importación de librerías necesarias:
    - import numpy as np
    - import pandas as pd
    - from matplotlib import pyplot as plt
    - from sklearn.cluster import KMeans

2. Cargar la data *vinos = pd.read_csv("caracteristicas_vinos.csv")*

3. Analizar data se la que se dispone:
    - .info()
    - .describe()

4. Limpiar data
    - Eliminar columna “Vino”
    - Realizar normalización de datos “vinos_norm”, luego convertir la data en DataFrame.

5. Seleccionar la cantidad optima de clúster a formar con la técnica de “codo de jambu”. Para ello, se debe tomar en cuenta que: donde deja de disminuir de forma drástica es la cantidad de clusters, es una buena opción para el número de clúster a formar.

   Acorde a la **Figura 1.** Código codo de jambu**Figura 1**, se formaran 10 clusters, la gráfica resultante define cual es el mejor número de clúster a realizar.

```
wcss=[]

for i in range(1,11):
    kmeans = KMeans(n_clusters=i, max_iter=300)
    kmeans.fit(vinos_norm)
    wcss.append(kmeans.inertia_)


plt.plot(range(1,11),wcss)
plt.title("Cododе jambu")
plt.xlabel("# de clusters")
plt.ylabel("WCSS")
plt.show()
```

Figura 1. *Código codo de jambu*

6. En la **Figura 2**, se observa la creación, entrenamiento y arquitectura del modelo KMeans.

```
# Método kmeans
cluster = KMeans(n_clusters=?, max_iter=300) #crea el modelo
cluster.fit(vinos_norm) #entrena el modelo
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=?, n_init=10, n_jobs=None, precompute_distances='auto',
        random_state=None, tol=0.0001, verbose=0)#arquitectura
```

Figura 2. *Creación del modelo KMeans*

7. Agregando clasificación a la data original "vinos". Los resultados del clúster se guardan en ***labels_*** dentro del modelo. Donde, la data a mostrar a más de las características iniciales tiene la columna clúster asignada por el algoritmo KMeans, como se muestra en **Figura 3**.

```
# Agregando clasificación
vinos['Cluster'] = cluster.labels_
vinos.head()
```

	Alcohol	Malic	Ash	Alcalinity	Magnesium	Phenols	Flavanoids	Nonflavanoids	Proanthocyanins	Color	Hue	Dilution	Proline	Cluster
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065	1
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050	1
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185	1
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480	1
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735	1

Figura 3. *Agregando clarificación KMeans*

8. La data tiene en total 13 características y lo que se requiere es analizar las características más importantes, las que más sobresalen y esto no es posible realizarlo en un solo gráfico, por tanto, se aplica la técnica de "Análisis de Componentes Principales" – PCA.

   PCA reduce la cantidad de variables a analizar/visualizar, creando una cantidad menor de nuevas variables que representen lo mejor posible a las variables originales. En la **Figura 4**, se muestra el código de implementación de PCA.

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)# puede cambiear el # de componentes
pca_vinos = pca.fit_transform(vinos_norm)
pca_vinos_df = pd.DataFrame(data=pca_vinos, columns=['Componente1','Componente2'])
pca_nombres_vinos = pd.concat([pca_vinos_df, vinos[['Cluster']]], axis=1)
```

Figura 4. *Implementación de PCA*

pca_vinos_df contiene los componentes / características principales que representan mayormente la data original, a la misma se le añade con concat la columna Cluster la cual contienen los clusters que asigno KMeans a cada uno de los vinos. En la **Figura 5**, se muestra una porción de la data resultante.

```
1 pca_nombres_vinos
```

	Componente1	Componente2	Cluster
0	-0.706336	-0.253193	1
1	-0.484977	-0.008823	1
2	-0.521172	-0.189187	1
3	-0.821644	-0.580906	1
4	-0.202546	-0.059467	1

Figura 5. *Componentes y clúster de la data.*

9. En la **Figura 6**, se muestra el código utilizado para la visualización de los clusters creados, se muestran Componente1, Componente2 y los componentes principales. Donde el parámetro *S* es el tamaño de los puntos a graficar.

```
# Visualización
fig = plt.figure(figsize=(6,6))

ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Componente 1', fontsize =  15)
ax.set_ylabel('Componente 2', fontsize =  15)
ax.set_title('PCA', fontsize =  20)

color_theme = np.array(["blue","green","orange"])
ax.scatter(x = pca_nombres_vinos.Componente1, y = pca_nombres_vinos.Componente2,
           c= color_theme[pca_nombres_vinos.Cluster], s=50)
plt.show()
```

Figura 6. *Visualización de clusters*

10. Finalmente, se guarda la data con los clúster asignados a cada vino.
vinos.to_csv("caracteristicas_vinos_KMeans.csv")

11. En este punto el archivo *caracteristicas_vinos_KMeans.csv* a más de las características de los datos tiene su etiqueta otorgada por KMeans. Pasando de ser un problema de aprendizaje no supervisado, a aprendizaje supervisado.

12. Aplicar 3 algoritmos de aprendizaje supervisado revisado en clases anteriores. Colocar conclusiones del mejor modelo.