INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y TECNOLOGÍAS AVANZADAS

PRÁCTICA 1

Sistemas de clasificación basados en distancias

RECONOCIMIENTO DE PATRONES

Montes Martínez Itsy Teri

Ramirez Villegas Arturo Javier

Reyes Leal Leonardo

Velázquez Osorio Samara Ishtar

PROFESOR

Anzueto Ríos Álvaro

GRUPO 4BM3 **FECHA DE ENTREGA** 18/09/2024

Teoría de las métricas de distancias utilizadas

Distancia euclidiana

Es una medida de la distancia en línea recta entre dos puntos en un espacio euclidiano. La distancia euclidiana se utiliza para cuantificar la similitud o disimilitud entre puntos de datos, esto nos ayuda para diversas tareas como agrupación, clasificación y detección de anomalías.

La distancia euclidiana se utiliza como una métrica de similitud para comparar vectores de características. Un vector de características representa un punto de datos en un espacio dimensional, donde cada dimensión corresponde a una característica o atributo específico de los datos a comparar. Al calcular la distancia euclidiana entre los vectores de características, podemos determinar qué tan similares o diferentes son.

Para calcular la distancia euclidiana, utilizamos la siguiente fórmula:

$$d_e(P,Q) = \sqrt{\sum_{i=1}^{p}(P_i - Q_i)^2}$$

Donde *p* es el espacio dimensional de los datos y P y Q son los puntos en dicho espacio donde obtendremos su distancia entre ellos.

Distancia de Mahalanobis

La distancia de Mahalanobis es una herramienta estadística utilizada para cuantificar la separación entre dos puntos dentro de un conjunto de datos, considerando tanto la distancia entre estos como la covarianza entre sus variables. A diferencia de la distancia euclidiana, donde se asume una independencia entre las variables, la distancia de Mahalanobis ajusta su cálculo en función de la correlación entre los datos. Esto la convierte en una medida especialmente adecuada para datos en los que las variables están correlacionadas entre sí.

Dado que se considera la correlación entre los datos, la distancia de Mahalanobis es particularmente eficaz en la detección de datos atípicos. Al capturar relaciones complejas entre características, puede identificar patrones atípicos que podrían pasar desapercibidos con otras métricas de distancia. Esto la convierte en una herramienta indispensable en aplicaciones donde las anomalías están influenciadas por múltiples factores interrelacionados.

La distancia de Mahalanobis entre dos puntos P y Q se calcula de la siguiente manera:

$$d(P,Q) = \sqrt{(P-Q)^T \Sigma^{-1} (P-Q)}$$

Donde Σ es la matriz de covarianza del conjunto de datos.

Tal como se puede apreciar en la expresión, la fórmula utiliza la inversa de la matriz de covarianza para “normalizar” las diferencias entre las variables, compensando de esta forma las diferencias de escala entre las diferentes dimensiones y teniendo en cuenta las correlaciones entre ellas.

De la distancia de Mahalanobis podemos observar las siguientes propiedades:

- La distancia de Mahalanobis siempre será positiva.
- La distancia de Mahalanobis será cero si los dos puntos a analizar son idénticos.
- La distancia de Mahalanobis es simétrica, o bien, d(P,Q) = d(P,Q)

Distancia coseno

También llamado “similitud de cosenos”, es una medida de similitud que se calcula entre dos vectores distintos de cero dentro del espacio interno del producto que mide el coseno del ángulo entre ellos.

Así que este caso se trata de un cálculo que da origen a un juicio de orientación y no de magnitud. Este se encarga de conocer el ángulo entre dos vectores n-dimensionales en un espacio n-dimensional. Esto significa que el resultado que obtenemos es el producto escalar de los dos vectores dividido por el producto de las longitudes o magnitudes de los dos vectores. De:

$$cos\alpha = \frac{\vec{u} \cdot \vec{v}}{|\vec{u}| \cdot |\vec{v}|}$$

Donde u y v son los puntos en nuestro espacio dimensional a los que queremos conocer su ángulo entre ellos.

Distancia Manhattan

La distancia Manhattan es una métrica utilizada para determinar la distancia entre dos puntos de una trayectoria en forma de cuadrícula. A diferencia de la distancia euclidiana, que mide la línea más corta posible entre dos puntos, la distancia de Manhattan mide la suma de las diferencias absolutas entre las coordenadas de los puntos. Este método se denomina "distancia Manhattan" porque, como un taxi que circula por las calles cuadriculadas de Manhattan, debe recorrer las líneas de la cuadrícula.

Matemáticamente, la distancia Manhattan entre dos puntos de un espacio n-dimensional es la suma de las diferencias absolutas de sus coordenadas cartesianas.

Fórmula para vectores n dimensionales:

$$Manhattan\ distance = |x_1 - x_2| + |y_1 - y_2| + \cdots + |V_{n1} - V_{n2}|$$

La fórmula de la distancia Manhattan incorpora la función del valor absoluto, que simplemente convierte cualquier diferencia negativa en un valor positivo. Esto es crucial para calcular la distancia, ya que garantiza que todas las mediciones de distancia sean no negativas, reflejando la verdadera distancia escalar independientemente de la dirección de desplazamiento. Cabe destacar que esta distancia es más robusta a los valores atípicos que la distancia euclidiana, porque no cuadra las diferencias. Sin embargo, también ignora la correlación y la forma de las variables, y puede no reflejar la distancia real si la cuadrícula no está alineada con los ejes de las variables.

Base de datos utilizada "Palmer Penguins"

Los datos utilizados para esta práctica se obtuvieron del paper "*Ecological Sexual Dimorphism and Environmental Variability within a Community of Antarctic Penguins (Genus Pygoscelis)*", donde se analiza el dimorfismo sexual ecológico y su relación con la variabilidad ambiental dentro de una comunidad de pingüinos del género *Pygoscelis* (Adelie, barbijo y papúa) en la Península Antártica.

En este trabajo, los investigadores liderados por Kristen B. Gorman, estudiaron cómo las diferencias en el tamaño y el comportamiento alimenticio entre machos y hembras afectan el uso de recursos y el nicho ecológico de estas especies. Analizaron cómo la variabilidad ambiental, como las condiciones del hielo marino, influye en la competencia por los recursos alimenticios entre los sexos, específicamente antes de la temporada de cría.

Para analizar el dimorfismo sexual, los investigadores tomaron varias medidas morfométricas de los pingüinos:

- Longitud y profundidad del culmen (parte superior del pico)
- Longitud de las aletas (flippers)
- Masa corporal

Y las especies clasificadas fueron:

- Pingüino Adelie (*Pygoscelis adeliae*)
- Pingüino Chinstrap (*Pygoscelis antárctica*)
- Pingüino Gentoo (*Pygoscelis papua*)

Para el programa, se utilizaron 2 dataset, uno denominado “penguins_training.csv”, dicho archivo nos serviría para entrenar o definir los valores representativos de las clases para cada una de las distancias, después, dichos datos se utilizarían como referencia para compararlos contra los datos del archivo “penguins_testing.csv”, los datos de este último archivo serían los datos que presentamos comparando la base de datos original contra la clase a la que cada métrica de distancia utilizada relaciono dicha especie.

Explicación del código

Se definieron funciones para cada métrica de distancia para separar las clases, a estas se les proporcionan los centroides de la base de datos “training” para, de acuerdo con ellos, clasificar por especie tanto la misma base de datos “training” como posteriormente la base a los datos de “testing”. Las funciones se muestran a continuación:

```python
#-----------------------------------------------------------------------
#   FUNCIÓN PARA CÁLCULO DE DISTANCIA EUCLIDIANA
#-----------------------------------------------------------------------

def distancia_euclidiana(data,centros,num_clases):
    #Calcula la distancia euclidiana entre cada dato y cada centro
    distancias = np.zeros((data.shape[0],num_clases))
    distancias[:,0] = np.sqrt((data[:,0]-centros[0,0])**2+(data[:,1]-centros[0,1])**2+(data[:,2]-centros[0,2])**2)
    distancias[:,1] = np.sqrt((data[:,0]-centros[1,0])**2+(data[:,1]-centros[1,1])**2+(data[:,2]-centros[1,2])**2)
    distancias[:,2] = np.sqrt((data[:,0]-centros[2,0])**2+(data[:,1]-centros[2,1])**2+(data[:,2]-centros[2,2])**2)

    return distancias
```

```python
#-----------------------------------------------------------------------
#   FUNCIÓN PARA CÁLCULO DE DISTANCIA DE MAHALANOBIS
#-----------------------------------------------------------------------
def distancia_mahalanobis(data,centros):
    #la función np.cov() pide las variables(caracaterísticas) en filas
    matriz_cov = np.cov(data.T)# por lo que se obtiene la transpuesta para cumplir con este requisito
    inv_cov_matriz = np.linalg.inv(matriz_cov)
    distancias = np.zeros((data.shape[0],3))
    for k in range(0,3,1):
        for i in range(0,data.shape[0],1):
            #for j in range(0,3,1):
            delta = data[i,:] - centros[k,:]
            distancias[i,k] = np.sqrt(np.dot(np.dot(delta.T,inv_cov_matriz),delta))
    return distancias
```

```python
#-----------------------------------------------------------------------
#   FUNCIÓN PARA CÁLCULO DE DISTANCIA DE COSENO
#-----------------------------------------------------------------------
def distancia_coseno(data, centros):
    distancias = np.zeros((data.shape[0], centros.shape[0]))
    for i in range(data.shape[0]):
        for j in range(centros.shape[0]):
            # Producto punto entre el dato y el centroide
            numerador = np.dot(data[i], centros[j])
            # Normas de los vectores
            norma_data = np.linalg.norm(data[i])
            norma_centro = np.linalg.norm(centros[j])
            # Distancia coseno
            distancias[i, j] = 1 - (numerador / (norma_data * norma_centro))
    return distancias
#-----------------------------------------------------------------------
```

```python
#-----------------------------------------------------------------------
#   FUNCIÓN PARA CÁLCULO DE DISTANCIA MANHATTAN
#-----------------------------------------------------------------------

def distancia_manhattan(data,centros,num_clases):
    #Calcula la distancia manhattan entre cada dato y cada centro
    distancias = np.zeros((data.shape[0],num_clases))
    for i in range(data.shape[0]):
        distancias[i,0] = np.sum(np.abs(data[i,:] - centros[0,:]))
        distancias[i,1] = np.sum(np.abs(data[i,:] - centros[1,:]))

    return distancias
```

Para poder implementar las funciones, primero se desplegó en una figura la gráfica en 3D donde se muestra la distribución de los datos de “training” y las variables utilizadas para la separación de especies, que en este caso se utilizaron las variables “Culmen Length”, “Culmen Depth” y “Flipper Length”

Además, se obtuvieron los promedios de los valores de los datos para cada especie, estos promedios se utilizarán posteriormente para calcular los centroides de las clases.

```python
#diccionario para almacenar los promedios
promedios_clases = {}

fig = plt.figure(figsize=(10,8))
ax = fig.add_subplot(111,projection='3d')
for specie, color in species.items():
    subset = train_data[train_data['Species']==specie]
    ax.scatter(subset['Culmen Length (mm)'],subset['Culmen Depth (mm)'],subset['Flipper Length (mm)'],color=color,label=specie,alpha=0.6)

    #formato de la gráfica
    ax.set_xlabel('Culmen Length (mm)')
    ax.set_ylabel('Culmen Depth (mm)')
    ax.set_zlabel('Flipper Length (mm)')
    plt.title('TRAINING DATA (TODAS LAS MUESTRAS DE LAS ESPECIES)')
    plt.legend()

    # Obtiene los promedios de las características de cada especie
    promediosCL = np.mean(subset['Culmen Length (mm)'])
    promediosCD = np.mean(subset['Culmen Depth (mm)'])
    promediosFL = np.mean(subset['Flipper Length (mm)'])
    # Agregar los resultados al diccionario
    resultados['Species'].append(specie)
    resultados['Culmen Length (mm)'].append(promediosCL)
    resultados['Culmen Depth (mm)'].append(promediosCD)
    resultados['Flipper Length (mm)'].append(promediosFL)
    # Crear un array con los promedios
    promedio = np.array([promediosCL,promediosCD,promediosFL])

    # Guardar el array de promedios en el diccionario con el nombre de la especie
    promedios_clases[specie] = promedio
```

Después, se creó una variable “clases_mapeo” para asociar las clases a marcadores que nos ayudarán a realizan la relación de distancia mínima por cada métrica de distancia con su clase asociada. Además, se extrajeron los datos de la base de datos “testing” para utilizarla después.

```
#-----------------------------------------------------------------------
# #obtención de los promedios en un arreglo
centroides_promedio = np.zeros([3,3])

centroides_promedio[0,:] = promedios_clases['Adelie Penguin (Pygoscelis adeliae)']
centroides_promedio[1,:] = promedios_clases['Chinstrap penguin (Pygoscelis antarctica)']
centroides_promedio[2,:] = promedios_clases['Gentoo penguin (Pygoscelis papua)']

#Crear diccionario para mapear las clases a nombres de especies
clases_mapeo = {0: 'Adelie Penguin (Pygoscelis adeliae)',
                1: 'Chinstrap penguin (Pygoscelis antarctica)',
                2: 'Gentoo penguin (Pygoscelis papua)'}

#Extrae los datos del archivo test_data
longi = np.array(test_data['Culmen Length (mm)'])
prof = np.array(test_data['Culmen Depth (mm)'])
aleta = np.array(test_data['Flipper Length (mm)'])
#concatena los datos en una sola matriz
datos  = np.column_stack((longi,prof,aleta))
#-----------------------------------------------------------------------
```

Por último, se mandan a llamar las funciones que calculan las diferentes distancias de las métricas de distancia con los datos de la base de datos “testing”, para después asignar la clase con base en la distancia mínima calculada por las diferentes métricas de distancia. Finalmente, se guardan los datos en la variable “tabla_resultados”.

```
#Calcula la distancia EUCLIDIANA entre los datos de test y los centroides
dist_euclidiana = distancia_euclidiana(datos,centroides_promedio,3)
#encontrar la clase con la distancia mínima
asigna_euclidiana = np.argmin(dist_euclidiana,axis = 1)
#-----------------------------------------------------------------------
#Calcula la distancia de MAHALANOBIS entre los datos de test y los centroides
dist_mahalanobis = distancia_mahalanobis(datos, centroides_promedio)
#encontrar la clase con la distancia mínima
asigna_mahalanobis = np.argmin(dist_mahalanobis,axis = 1)
# Calcula la distancia de COSENO entre los datos de test y los centroides
dist_coseno = distancia_coseno(datos, centroides_promedio)
# encontrar la clase con la distancia mínima
asigna_coseno = np.argmin(dist_coseno, axis=1)
#Calcula la distancia MANHATTAN entre los datos de test y los centroides
dist_manhattan = distancia_manhattan(datos,centroides_promedio,3)
#encontrar la clase con la distancia mínima
asigna_manhattan = np.argmin(dist_manhattan,axis = 1)
#-----------------------------------------------------------------------
# Convertir las clases numéricas en nombres de especies
test_data['Clase Asignada (Dist euclidiana)'] = [clases_mapeo[clase] for clase in asigna_euclidiana]
test_data['Clase Asignada (Dist Mahalanobis)'] = [clases_mapeo[clase] for clase in asigna_mahalanobis]
test_data['Clase Asignada (Dist Coseno)'] = [clases_mapeo[clase] for clase in asigna_coseno]
test_data['Clase Asignada (Dist Manhattan)'] = [clases_mapeo[clase] for clase in asigna_manhattan]

# Mostrar una tabla con la clase real y la clase asignada
tabla_resultados = test_data[['Species', 'Clase Asignada (Dist euclidiana)', 'Clase Asignada (Dist Mahalanobis)',
                              'Clase Asignada (Dist Coseno)', 'Clase Asignada (Dist Manhattan)']]
print(tabla_resultados)
```

Para separar los datos de acuerdo con *Male/Female,* se utilizaron las mismas funciones de métricas de distancias y las mismas bases de datos “training” y “testing”. Para la clasificación por sexo de los datos, se utilizaron los parámetros “Culmen Depth”, “Culmen Length” y “Body mass”.

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Lectura de archivos
train_data = pd.read_csv(r'F:\ARCHIVOS HDD\Materias\7mo Semestre\Reconocimiento de Patrones\penguins_training.csv')
test_data = pd.read_csv(r'F:\ARCHIVOS HDD\Materias\7mo Semestre\Reconocimiento de Patrones\penguins_testing.csv')

# Especies a analizar
species = {'MALE': 'blue',
           'FEMALE': 'pink'}
```

```python
# Crear un diccionario para almacenar los promedios por especie
resultados = {
    'Sex': [],
    'Culmen Length (mm)': [],
    'Culmen Depth (mm)': [],
    'Body Mass (g)': []
}
```

De igual forma que para separar las especies por distancias, se calcularon los promedios de cada parámetro, dichos promedios se utilizan posteriormente para obtener los centroides de las clases que caracterizan a dichos parámetros.

```python
# Diccionario para almacenar los promedios
promedios_clases = {}

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
for specie, color in species.items():
    subset = train_data[train_data['Sex'] == specie]
    ax.scatter(subset['Culmen Length (mm)'], subset['Culmen Depth (mm)'], subset['Body Mass (g)'], color=color, label=specie, alpha=0.6)

    # Formato de la grafica
    ax.set_xlabel('Culmen Length (mm)')
    ax.set_ylabel('Culmen Depth (mm)')
    ax.set_zlabel('Body Mass (g)')
    plt.title('TRAINING DATA (MACHO Y HEMBRA)')
    plt.legend()

    # Obtiene los promedios de las caracteristicas de cada especie
    promediosCL = np.mean(subset['Culmen Length (mm)'])
    promediosCD = np.mean(subset['Culmen Depth (mm)'])
    promediosFL = np.mean(subset['Body Mass (g)'])

    # Agregar los resultados al diccionario
    resultados['Sex'].append(specie)
    resultados['Culmen Length (mm)'].append(promediosCL)
    resultados['Culmen Depth (mm)'].append(promediosCD)
    resultados['Body Mass (g)'].append(promediosFL)

    # Crear un array con los promedios
    promedio = np.array([promediosCL, promediosCD, promediosFL])

    # Guardar el array de promedios en el diccionario con el nombre de la especie
    promedios_clases[specie] = promedio
```

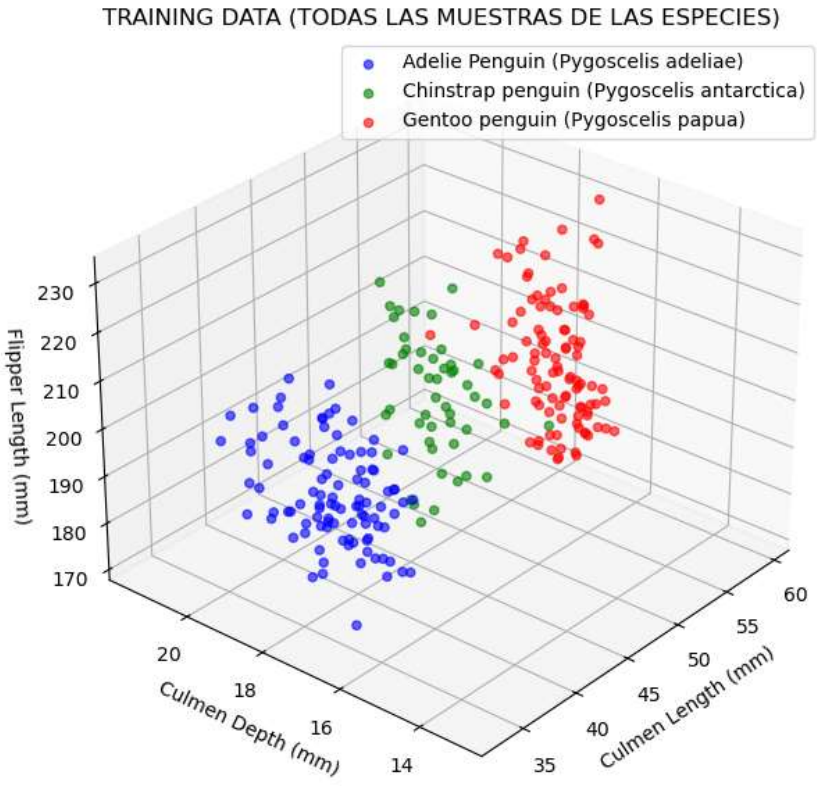
```python
#-----------------------------------------------------------------------
# Obtención de los promedios en un arreglo
centroides_promedio = np.zeros([2, 3])  # Dos clases, tres características

centroides_promedio[0, :] = promedios_clases['MALE']
centroides_promedio[1, :] = promedios_clases['FEMALE']

# Crear diccionario para mapear las clases a nombres de especies
clases_mapeo = {0: 'MALE',
                1: 'FEMALE'}

# Extrae los datos del archivo test_data
longi = np.array(test_data['Culmen Length (mm)'])
prof = np.array(test_data['Culmen Depth (mm)'])
masa = np.array(test_data['Body Mass (g)'])
# Concatena los datos en una sola matriz
datos = np.column_stack((longi, prof, masa))
```

Por último, nuevamente se mandaron a llamar las funciones que calculan las distancias con base en las diferentes métricas de distancia, además, de asignar cada distancia a un sexo de acuerdo con la distancia mínima obtenida por cada métrica. Por último, se muestran los resultados obtenidos de las asignaciones de cada métrica y se compara con el sexo original de la base de datos.

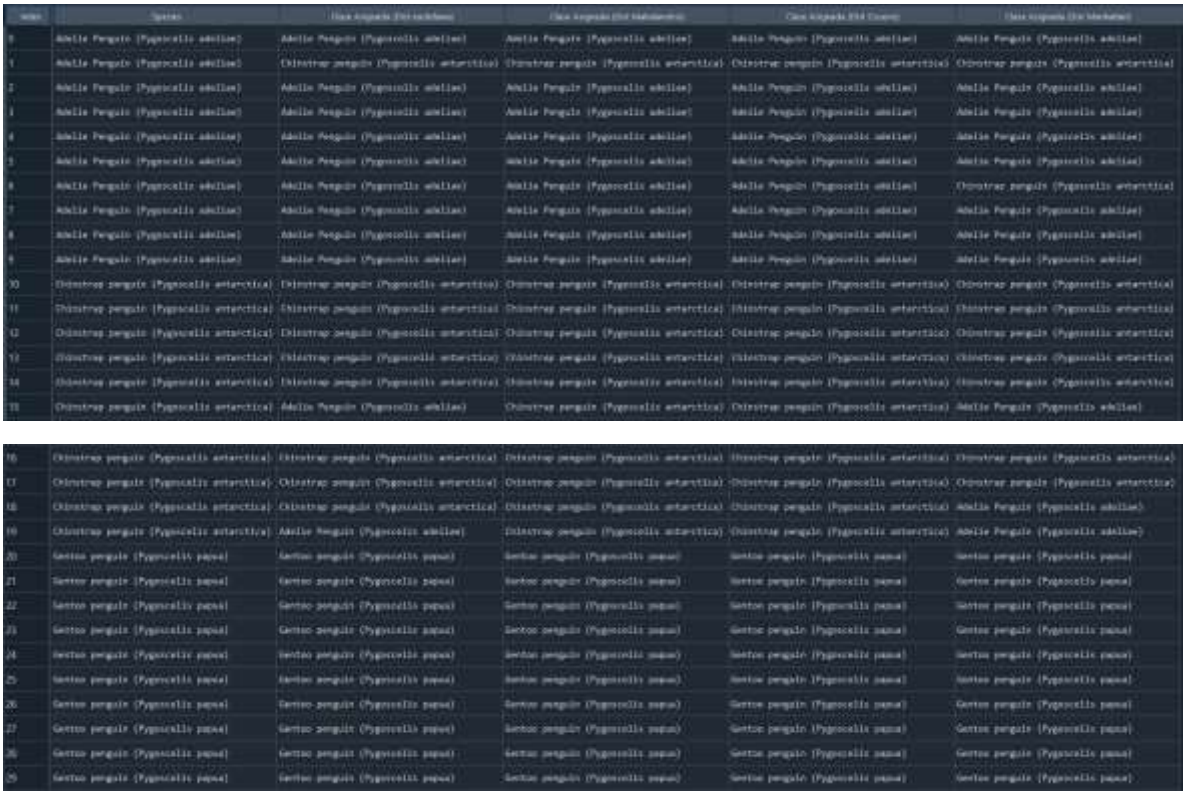
```
#-----------------------------------------------------------------
# Calcula la distancia EUCLIDIANA entre los datos de test y los centroides
dist_euclidiana = distancia_euclidiana(datos, centroides_promedio, 2)
asigna_euclidiana = np.argmin(dist_euclidiana, axis=1)
# Calcula la distancia de MAHALANOBIS entre los datos de test y los centroides
dist_mahalanobis = distancia_mahalanobis(datos, centroides_promedio)
asigna_mahalanobis = np.argmin(dist_mahalanobis, axis=1)
# Calcula la distancia de COSENO entre los datos de test y los centroides
dist_coseno = distancia_coseno(datos, centroides_promedio)
# encontrar la clase con la distancia minima
asigna_coseno = np.argmin(dist_coseno, axis=1)
#Calcula la distancia MANHATTAN entre los datos de test y los centroides
dist_manhattan = distancia_manhattan(datos,centroides_promedio,2)
#encontrar la clase con la distancia minima
asigna_manhattan = np.argmin(dist_manhattan,axis = 1)

#-----------------------------------------------------------------
# Convertir las clases numéricas en nombres de especies
test_data['Clase Asignada (Dist euclidiana)'] = [clases_mapeo[clase] for clase in asigna_euclidiana]
test_data['Clase Asignada (Dist Mahalanobis)'] = [clases_mapeo[clase] for clase in asigna_mahalanobis]
test_data['Clase Asignada (Dist Coseno)'] = [clases_mapeo[clase] for clase in asigna_coseno]
test_data['Clase Asignada (Dist Manhattan)'] = [clases_mapeo[clase] for clase in asigna_manhattan]

# Mostrar una tabla con la clase real y la clase asignada
tabla_resultados = test_data[['Species', 'Sex','Clase Asignada (Dist euclidiana)',
                              'Clase Asignada (Dist Mahalanobis)', 'Clase Asignada (Dist Coseno)', 'Clase Asignada (Dist Manhattan)']]
print(tabla_resultados)
```

Resultados

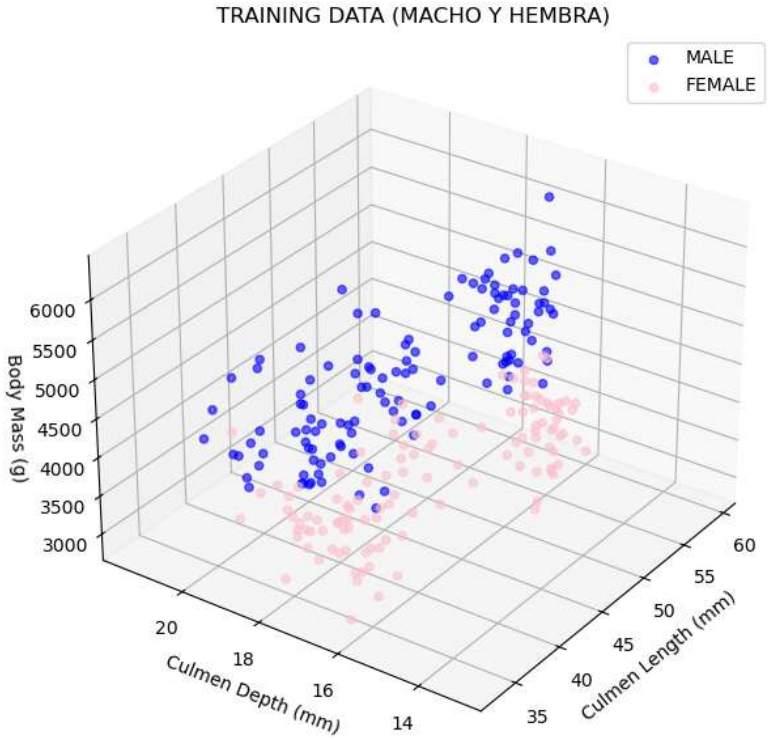
Para la separación de especies, primero se muestra el gráfico en 3D donde se muestra la separación de clase con base en los parámetros seleccionados.



A continuación, se muestra la tabla de las especies originales comparada con las especies asignadas por las diferentes métricas de distancia.



Ahora, se presenta la gráfica de la separación original de sexos obtenida de la base de datos “training”.



Por último, se muestra una tabla donde se comparan los sexos originales de la base de datos “testing” contra los sexos asociados por las diferentes distancias.

Index	Species	Sex	Clase Asignada (Dist euclidiana)	Clase Asignada (Dist Mahalanobis)	Clase Asignada (Dist Coseno)	Clase Asignada (Dist Manhattan)
0	Adelie Penguin (Pygoscelis adeliae)	MALE	FEMALE	MALE	MALE	FEMALE
1	Adelie Penguin (Pygoscelis adeliae)	MALE	FEMALE	MALE	FEMALE	FEMALE
2	Adelie Penguin (Pygoscelis adeliae)	MALE	FEMALE	MALE	MALE	FEMALE
3	Adelie Penguin (Pygoscelis adeliae)	FEMALE	FEMALE	FEMALE	MALE	FEMALE
4	Adelie Penguin (Pygoscelis adeliae)	FEMALE	FEMALE	FEMALE	FEMALE	FEMALE
5	Adelie Penguin (Pygoscelis adeliae)	FEMALE	FEMALE	FEMALE	FEMALE	FEMALE
6	Adelie Penguin (Pygoscelis adeliae)	MALE	FEMALE	MALE	FEMALE	FEMALE
7	Adelie Penguin (Pygoscelis adeliae)	MALE	FEMALE	FEMALE	FEMALE	FEMALE
8	Adelie Penguin (Pygoscelis adeliae)	FEMALE	FEMALE	FEMALE	FEMALE	FEMALE
9	Adelie Penguin (Pygoscelis adeliae)	MALE	FEMALE	MALE	MALE	FEMALE
10	Chinstrap penguin (Pygoscelis antarctica)	FEMALE	FEMALE	FEMALE	FEMALE	FEMALE
11	Chinstrap penguin (Pygoscelis antarctica)	MALE	FEMALE	MALE	FEMALE	FEMALE
12	Chinstrap penguin (Pygoscelis antarctica)	MALE	FEMALE	MALE	FEMALE	FEMALE
13	Chinstrap penguin (Pygoscelis antarctica)	MALE	FEMALE	MALE	FEMALE	FEMALE
14	Chinstrap penguin (Pygoscelis antarctica)	FEMALE	FEMALE	FEMALE	FEMALE	FEMALE
15	Chinstrap penguin (Pygoscelis antarctica)	FEMALE	FEMALE	FEMALE	FEMALE	FEMALE
16	Chinstrap penguin (Pygoscelis antarctica)	MALE	FEMALE	MALE	FEMALE	FEMALE
17	Chinstrap penguin (Pygoscelis antarctica)	FEMALE	FEMALE	FEMALE	FEMALE	FEMALE
18	Chinstrap penguin (Pygoscelis antarctica)	FEMALE	FEMALE	FEMALE	FEMALE	FEMALE
19	Chinstrap penguin (Pygoscelis antarctica)	FEMALE	FEMALE	FEMALE	FEMALE	FEMALE
20	Gentoo penguin (Pygoscelis papua)	FEMALE	MALE	FEMALE	MALE	MALE
21	Gentoo penguin (Pygoscelis papua)	FEMALE	MALE	FEMALE	MALE	MALE
22	Gentoo penguin (Pygoscelis papua)	MALE	MALE	MALE	MALE	MALE
23	Gentoo penguin (Pygoscelis papua)	MALE	MALE	MALE	MALE	MALE
24	Gentoo penguin (Pygoscelis papua)	MALE	MALE	MALE	MALE	MALE
25	Gentoo penguin (Pygoscelis papua)	FEMALE	FEMALE	FEMALE	FEMALE	FEMALE
26	Gentoo penguin (Pygoscelis papua)	FEMALE	MALE	FEMALE	MALE	MALE
27	Gentoo penguin (Pygoscelis papua)	FEMALE	MALE	MALE	MALE	MALE
28	Gentoo penguin (Pygoscelis papua)	FEMALE	MALE	FEMALE	MALE	MALE
29	Gentoo penguin (Pygoscelis papua)	MALE	MALE	MALE	MALE	MALE

A partir de ambas situaciones, la separación de especies y separación de sexos, la métrica de distancia de Mahalanobis resultó ser la mejor para clasificar ambas situaciones, fueron casos muy puntuales donde la distancia de Mahalanobis no fue precisa al momento de definir de forma correcta la clase que se le proporcionó.

Para la separación de especies, podemos observar que no hay muchas diferencias entre distancia Euclidiana, distancia Manhattan y distancia coseno, únicamente la distancia Euclidiana fallo un dato más que la distancia coseno y Manhattan una más que euclidiana, para el resto de los datos, ambas distancias cumplen una función de clasificación aceptable, con uno que otro fallo.

Por último, para la separación de sexos de las muestras tanto distancia Euclidiana, distancia de Manhattan como distancia cosenos varían mucho en acertar el dato

real, siendo la distancia de Mahalanobis la métrica más confiable para clasificar en este caso. Nótese que existe un dato en el que incluso distancia de Mahalanobis no detecta de forma correcta el sexo de la muestra, lo que nos puede indicar que dicha muestra es un dato muy atípico tanto que la mejor métrica para clasificar muestras falla.

Para concluir, se nota una clara ventaja de utilizar ventaja de Mahalanobis al observar que en ambos casos fue la métrica que mejor separó las distancias, sin embargo, si conocemos de antemano la distribución de las clases podemos usar cualquiera de las otras tres distancias para separar las clases con un alto grado de certeza; aunque podemos destacar que utilizar distancia de Mahalanobis es más robusta para clasificación de muestras con base en la distancia entre puntos y centroides representativos.

Extra-Distancia Manhattan

```python
#---------------------------------------------------------------------
#   FUNCIÓN PARA CÁLCULO DE DISTANCIA MANHATTAN
#---------------------------------------------------------------------

def distancia_manhattan(data,centros,num_clases):
    #Calcula la distancia euclidiana entre cada dato y cada centro
    distancias = np.zeros((data.shape[0],num_clases))
    for i in range(data.shape[0]):
        distancias[i,0] = np.sum(np.abs(data[i,:] - centros[0,:]))
        distancias[i,1] = np.sum(np.abs(data[i,:] - centros[1,:]))
        distancias[i,2] = np.sum(np.abs(data[i,:] - centros[2,:]))

    return distancias

#---------------------------------------------------------------------
```

	Species	...	Clase Asignada (Dist Manhattan)
0	Adelie Penguin (Pygoscelis adeliae)	...	Adelie Penguin (Pygoscelis adeliae)
1	Adelie Penguin (Pygoscelis adeliae)	...	Chinstrap penguin (Pygoscelis antarctica)
2	Adelie Penguin (Pygoscelis adeliae)	...	Adelie Penguin (Pygoscelis adeliae)
3	Adelie Penguin (Pygoscelis adeliae)	...	Adelie Penguin (Pygoscelis adeliae)
4	Adelie Penguin (Pygoscelis adeliae)	...	Adelie Penguin (Pygoscelis adeliae)
5	Adelie Penguin (Pygoscelis adeliae)	...	Adelie Penguin (Pygoscelis adeliae)
6	Adelie Penguin (Pygoscelis adeliae)	...	Chinstrap penguin (Pygoscelis antarctica)
7	Adelie Penguin (Pygoscelis adeliae)	...	Adelie Penguin (Pygoscelis adeliae)
8	Adelie Penguin (Pygoscelis adeliae)	...	Adelie Penguin (Pygoscelis adeliae)
9	Adelie Penguin (Pygoscelis adeliae)	...	Adelie Penguin (Pygoscelis adeliae)
10	Chinstrap penguin (Pygoscelis antarctica)	...	Chinstrap penguin (Pygoscelis antarctica)
11	Chinstrap penguin (Pygoscelis antarctica)	...	Chinstrap penguin (Pygoscelis antarctica)
12	Chinstrap penguin (Pygoscelis antarctica)	...	Chinstrap penguin (Pygoscelis antarctica)
13	Chinstrap penguin (Pygoscelis antarctica)	...	Chinstrap penguin (Pygoscelis antarctica)
14	Chinstrap penguin (Pygoscelis antarctica)	...	Chinstrap penguin (Pygoscelis antarctica)
15	Chinstrap penguin (Pygoscelis antarctica)	...	Adelie Penguin (Pygoscelis adeliae)
16	Chinstrap penguin (Pygoscelis antarctica)	...	Chinstrap penguin (Pygoscelis antarctica)
17	Chinstrap penguin (Pygoscelis antarctica)	...	Chinstrap penguin (Pygoscelis antarctica)
18	Chinstrap penguin (Pygoscelis antarctica)	...	Adelie Penguin (Pygoscelis adeliae)
19	Chinstrap penguin (Pygoscelis antarctica)	...	Adelie Penguin (Pygoscelis adeliae)
20	Gentoo penguin (Pygoscelis papua)	...	Gentoo penguin (Pygoscelis papua)
21	Gentoo penguin (Pygoscelis papua)	...	Gentoo penguin (Pygoscelis papua)
22	Gentoo penguin (Pygoscelis papua)	...	Gentoo penguin (Pygoscelis papua)
23	Gentoo penguin (Pygoscelis papua)	...	Gentoo penguin (Pygoscelis papua)
24	Gentoo penguin (Pygoscelis papua)	...	Gentoo penguin (Pygoscelis papua)
25	Gentoo penguin (Pygoscelis papua)	...	Gentoo penguin (Pygoscelis papua)
26	Gentoo penguin (Pygoscelis papua)	...	Gentoo penguin (Pygoscelis papua)
27	Gentoo penguin (Pygoscelis papua)	...	Gentoo penguin (Pygoscelis papua)
28	Gentoo penguin (Pygoscelis papua)	...	Gentoo penguin (Pygoscelis papua)
29	Gentoo penguin (Pygoscelis papua)	...	Gentoo penguin (Pygoscelis papua)

Conclusiones

Montes Martínez Itsy Teri

Es importante mencionar que esta práctica fue de mucha ayuda ya que permitió aterrizar los conocimientos de distancias de una forma numérica, en los ejemplos vistos en clase tomábamos los valores de algunas posiciones de las imágenes para poder realizar los promedios y cálculos necesarios, sin embargo, al tener una base de datos la información se vuelve más tangible a la percepción.

Fue de gran ayuda leer la investigación sobre los pingüinos ya que ahí se nos explicó que una de las características más funcionales para poder detectar el sexo del pingüino es la masa corporal y las otras dos características que ya habíamos ocupado anteriormente.

Como primer parte del programa es importante entender y extraer los valores de la base de datos con los cuales vamos a trabajar, para el primer programa seria ancho y largo del pico del pingüino y la longitud del ala del pingüino, posterior a eso se crearon las funciones de cada distancia, la euclidiana, la de mahalanobis y la de coseno para que de esta forma sea más fácil mandarlas a llamar, se calcula los promedios de la información obtenida de cada coordenada para que de esta forma se les asignes los valores más próximos.

Considero que esta práctica fue de mucha ayuda ya que nos permitió analizar las ecuaciones de los diferentes tipos de distancias y que a la hora de realizar la información con la distinción entre el sexo macho/hembra de los pingüinos tuviéramos que adaptar las matrices y distancias para recalcular la información.

Ramirez Villegas Arturo Javier

A lo largo de la realización de esta práctica se apreció la utilidad de conocer diferentes métricas de clasificación de muestras con base a la distancia, pues nos permitió comparar de forma práctica las ventajas y desventajas de cada distancia, por ejemplo, la complejidad o amplitud de conceptos va aumentando de acuerdo con la exactitud de cada método, nótese que para la distancia de Mahalanobis se tuvo que observar con mayor detenimiento la fórmula para la distancia, lo que nos permitió comprender de mejor manera la importancia de la correlación entre las características para esta métrica.

Otro punto que destacar para la realización de esta práctica fue la implementación de conocimientos adquiridos en otras unidades de aprendizaje a Python, en lo personal es un entorno de programación totalmente nuevo, es de utilidad conocer nuevas plataformas de programación y observar que el traslado de conocimientos entre unidades de aprendizaje o entornos de programación no es de gran

complejidad, destacando la importancia de conocer la sintaxis del lenguaje y las funciones que las diferentes librerías nos pueden ofrecer.

Por último, reconocer la utilidad de comprender el fin con el que se aplicaron las diferentes métricas para la clasificación de muestras en una base de datos, así, podremos extrapolar la utilidad de estas métricas hacia otros casos donde se busque clasificar diferentes clases con base en características o parámetros de un objeto a estudiar.

Reyes Leal Leonardo

El análisis de las distintas características de los pingüinos nos permitió poner en práctica cada una de las distancias: euclidiana, Mahalanobis, coseno y Manhattan. En general, Mahalanobis nos ayudó a tener una clasificación más certera tanto al separar las especies, como al separar los sexos, ya que toma en cuenta la covarianza entre las variables, lo que le permite ser menos sensible a valores atípicos en comparación con las otras distancias.

La selección de las características a considerar para hacer las agrupaciones fue un punto clave, ya que con esto pudimos observar qué era lo que hacían más diferentes a las especies. En nuestro caso, se compararon las medidas del pico y del ala para identificar a cada muestra con su respectiva especie. Posteriormente, para la clasificación por sexo, las características consideradas fueron las medidas del pico y la masa. Al trabajar con conjuntos de datos multidimensionales, es importante tener un punto de partida al cual podamos referirnos para verificar nuestro progreso. Cabe mencionar que pudieron utilizarse más de tres características para realizar la clasificación, sin embargo, al trabajar con más de tres dimensiones, el proceso se vuelve menos intuitivo y un tanto más complicado.

Velazquez Osorio Samara Ishtar

Clasificar datos resulta realmente útil cuando se trata de alguna investigación para simplificar grandes cantidades de información, para encontrar nuevas especies como lo fue para esta base de datos o aplicaciones donde a partir de clasificar un nuevo dato se puedan tomar las decisiones pertinentes, entre muchos otros ejemplos. En esta práctica abordamos cuatro métodos distintos para hacerlo, en donde, para todos, se inicia con una base de datos cuyas características conozcamos, obtenemos las características que nos permitan diferenciar cada una de las clases de nuestro interés para que esta base de datos conocida sirva como entrenamiento, para así, en cada clase se encuentre un vector representativo calculado con promedios y poder aplicar el método que hayamos elegido para calcular la distancia entre vectores y determinar a qué clase pertenece cada muestra.

Para nuestra aplicación, encontramos que el método que mejores resultados obtuvo fue el de Mahalanobis, esto dado los pesos que le da a las características que escogimos con respecto a las demás, esto lo pudimos observar principalmente en la clasificación por sexo, dado que fue la única que tuvo prácticamente aciertos en todas, mientras que, para clasificar especies, a pesar de que también fue la que más resaltó, no tuvo tanta diferencia con respecto a las demás. Por otro lado, encontramos que la distancia euclidiana y Manhattan son muy parecidas tanto en resultados como en algoritmo, sin embargo, Manhattan es menos sensible a los valores atípicos y la distancia del coseno es útil cuando se desea comparar la similitud de dos vectores en función de su dirección, en lugar de su longitud. Por lo tanto, se puede concluir que el mejor método dependerá del tipo de datos con el que contemos y la aplicación que requiramos.

Anexo/Códigos

Código para aplicar distancias y generar tabla de clasificación

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
#PRÁCTICA 1: DISTINCIÓN ENTRE ESPECIES DE PINGÜINOS


#lectura de archivos
train_data     =     pd.read_csv(r'F:\ARCHIVOS     HDD\Materias\7mo
Semestre\Reconocimiento de Patrones\penguins_training.csv')
test_data     =     pd.read_csv(r'F:\ARCHIVOS     HDD\Materias\7mo
Semestre\Reconocimiento de Patrones\penguins_testing.csv')

#Especies a analizar
species = {'Adelie Penguin (Pygoscelis adeliae)':'blue',
           'Chinstrap penguin (Pygoscelis antarctica)' : 'green',
           'Gentoo penguin (Pygoscelis papua)' : 'red'}

#-----------------------------------------------------------------
-----
#   FUNCIÓN PARA CÁLCULO DE DISTANCIA EUCLIDIANA
#-----------------------------------------------------------------
-----

def distancia_euclidiana(data,centros,num_clases):
    #Calcula la distancia euclidiana entre cada dato y cada centro
    distancias = np.zeros((data.shape[0],num_clases))
```

```
    distancias[:,0]              =              np.sqrt((data[:,0]-
centros[0,0])**2+(data[:,1]-centros[0,1])**2+(data[:,2]-
centros[0,2])**2)
    distancias[:,1]              =              np.sqrt((data[:,0]-
centros[1,0])**2+(data[:,1]-centros[1,1])**2+(data[:,2]-
centros[1,2])**2)
    distancias[:,2]              =              np.sqrt((data[:,0]-
centros[2,0])**2+(data[:,1]-centros[2,1])**2+(data[:,2]-
centros[2,2])**2)

    return distancias
#-------------------------------------------------------------------
-----
#   FUNCIÓN PARA CÁLCULO DE DISTANCIA DE MAHALANOBIS
#-------------------------------------------------------------------
-----
def distancia_mahalanobis(data,centros):
    #la función np.cov() pide las variables(caracaterísticas) en
filas
    matriz_cov = np.cov(data.T)# por lo que se obtiene la transpuesta
para cumplir con este requisito
    inv_cov_matriz = np.linalg.inv(matriz_cov)
    distancias = np.zeros((data.shape[0],3))
    for k in range(0,3,1):
        for i in range(0,data.shape[0],1):
            #for j in range(0,3,1):
            delta = data[i,:] - centros[k,:]
            distancias[i,k]                                          =
np.sqrt(np.dot(np.dot(delta.T,inv_cov_matriz),delta))
    return distancias


#-------------------------------------------------------------------
-----
#   FUNCIÓN PARA CÁLCULO DE DISTANCIA DE COSENO
#-------------------------------------------------------------------
-----
def distancia_coseno(data, centros):
    distancias = np.zeros((data.shape[0], centros.shape[0]))
    for i in range(data.shape[0]):
        for j in range(centros.shape[0]):
            # Producto punto entre el dato y el centroide
            numerador = np.dot(data[i], centros[j])
            # Normas de los vectores
            norma_data = np.linalg.norm(data[i])
            norma_centro = np.linalg.norm(centros[j])
            # Distancia coseno
```

```
            distancias[i, j] = 1 - (numerador / (norma_data *
norma_centro))
    return distancias


#-----------------------------------------------------------------
-----
#   FUNCIÓN PARA CÁLCULO DE DISTANCIA MANHATTAN
#-----------------------------------------------------------------
-----

def distancia_manhattan(data,centros,num_clases):
    #Calcula la distancia Manhattan entre cada dato y cada centro
    distancias = np.zeros((data.shape[0],num_clases))
    for i in range(data.shape[0]):
        distancias[i,0] = np.sum(np.abs(data[i,:] - centros[0,:]))
        distancias[i,1] = np.sum(np.abs(data[i,:] - centros[1,:]))
        distancias[i,2] = np.sum(np.abs(data[i,:] - centros[2,:]))

    return distancias

#-----------------------------------------------------------------
-----

# Crear un diccionario para almacenar los promedios por especie
resultados = {
    'Species': [],
    'Culmen Length (mm)': [],
    'Culmen Depth (mm)': [],
    'Flipper Length (mm)': []
}


#diccionario para almacenar los promedios
promedios_clases = {}

fig = plt.figure(figsize=(10,8))
ax = fig.add_subplot(111,projection='3d')
for specie, color in species.items():
    subset = train_data[train_data['Species']==specie]
    ax.scatter(subset['Culmen Length (mm)'],subset['Culmen Depth
(mm)'],subset['Flipper                                          Length
(mm)'],color=color,label=specie,alpha=0.6)

    #formato de la gráfica
    ax.set_xlabel('Culmen Length (mm)')
    ax.set_ylabel('Culmen Depth (mm)')
    ax.set_zlabel('Flipper Length (mm)')
```

```
    plt.title('TRAINING DATA (TODAS LAS MUESTRAS DE LAS ESPECIES)')
    plt.legend()

    # #obtiene los promedios de las características de cada especie
    promediosCL = np.mean(subset['Culmen Length (mm)'])
    promediosCD = np.mean(subset['Culmen Depth (mm)'])
    promediosFL = np.mean(subset['Flipper Length (mm)'])
    # Agregar los resultados al diccionario
    resultados['Species'].append(specie)
    resultados['Culmen Length (mm)'].append(promediosCL)
    resultados['Culmen Depth (mm)'].append(promediosCD)
    resultados['Flipper Length (mm)'].append(promediosFL)
    # Crear un array con los promedios
    promedio = np.array([promediosCL,promediosCD,promediosFL])

    # Guardar el array de promedios en el diccionario con el nombre
de la especie
    promedios_clases[specie] = promedio

#-----------------------------------------------------------------
-----
# #obtención de los promedios en un arreglo
centroides_promedio = np.zeros([3,3])

centroides_promedio[0,:]    =    promedios_clases['Adelie    Penguin
(Pygoscelis adeliae)']
centroides_promedio[1,:]   =   promedios_clases['Chinstrap   penguin
(Pygoscelis antarctica)']
centroides_promedio[2,:]    =    promedios_clases['Gentoo    penguin
(Pygoscelis papua)']

#Crear diccionario para mapear las clases a nombres de especies
clases_mapeo = {0: 'Adelie Penguin (Pygoscelis adeliae)',
                1: 'Chinstrap penguin (Pygoscelis antarctica)',
                2: 'Gentoo penguin (Pygoscelis papua)'}

#Extrae los datos del archivo test_data
longi = np.array(test_data['Culmen Length (mm)'])
prof = np.array(test_data['Culmen Depth (mm)'])
aleta = np.array(test_data['Flipper Length (mm)'])
#concatena los datos en una sola matriz
datos  = np.column_stack((longi,prof,aleta))
#-----------------------------------------------------------------
-----

#Calcula la distancia EUCLIDIANA entre los datos de test y los
centroides
dist_euclidiana = distancia_euclidiana(datos,centroides_promedio,3)
```

```
#encontrar la clase con la distancia mínima
asigna_euclidiana = np.argmin(dist_euclidiana,axis = 1)
#-----------------------------------------------------------------
-----
#Calcula la distancia de MAHALANOBIS entre los datos de test y los
centroides
dist_mahalanobis           =           distancia_mahalanobis(datos,
centroides_promedio)
#encontrar la clase con la distancia mínima
asigna_mahalanobis = np.argmin(dist_mahalanobis,axis = 1)
# Calcula la distancia de COSENO entre los datos de test y los
centroides
dist_coseno = distancia_coseno(datos, centroides_promedio)
# encontrar la clase con la distancia mínima
asigna_coseno = np.argmin(dist_coseno, axis=1)
#Calcula la distancia MANHATTAN entre los datos de test y los
centroides
dist_manhattan = distancia_manhattan(datos,centroides_promedio,3)
#encontrar la clase con la distancia mínima
asigna_manhattan = np.argmin(dist_manhattan,axis = 1)
#-----------------------------------------------------------------
-----
# Convertir las clases numéricas en nombres de especies
test_data['Clase Asignada (Dist euclidiana)'] = [clases_mapeo[clase]
for clase in asigna_euclidiana]
test_data['Clase      Asignada      (Dist      Mahalanobis)']      =
[clases_mapeo[clase] for clase in asigna_mahalanobis]
test_data['Clase Asignada (Dist Coseno)'] = [clases_mapeo[clase] for
clase in asigna_coseno]
test_data['Clase Asignada (Dist Manhattan)'] = [clases_mapeo[clase]
for clase in asigna_manhattan]

# Mostrar una tabla con la clase real y la clase asignada
tabla_resultados  =  test_data[['Species',  'Clase  Asignada  (Dist
euclidiana)', 'Clase Asignada (Dist Mahalanobis)','Clase Asignada
(Dist Coseno)', 'Clase Asignada (Dist Manhattan)']]
print(tabla_resultados)
```

Código para separar las muestras por *Male/Female*

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Lectura de archivos
```

```
train_data    =    pd.read_csv(r'C:\Users\ikerf\Desktop\Upiita\7mo
semestre\Patrones\penguins_training.csv')
test_data     =    pd.read_csv(r'C:\Users\ikerf\Desktop\Upiita\7mo
semestre\Patrones\penguins_testing.csv')

# Especies a analizar
species = {'MALE': 'blue',
           'FEMALE': 'pink'}

#-----------------------------------------------------------------
-----
# FUNCIÓN PARA CÁLCULO DE DISTANCIA EUCLIDIANA
#-----------------------------------------------------------------
-----
def distancia_euclidiana(data, centros, num_clases):
    # Calcula la distancia euclidiana entre cada dato y cada centro
    distancias = np.zeros((data.shape[0], num_clases))
    for k in range(num_clases):
        distancias[:, k] = np.sqrt(np.sum((data - centros[k, :]) **
2, axis=1))
    return distancias

#-----------------------------------------------------------------
-----
# FUNCIÓN PARA CÁLCULO DE DISTANCIA DE MAHALANOBIS
#-----------------------------------------------------------------
-----
def distancia_mahalanobis(data, centros):
    # La función np.cov() pide las variables (características) en
filas
    matriz_cov  =  np.cov(data.T)    # por lo que se obtiene la
transpuesta para cumplir con este requisito
    inv_cov_matriz = np.linalg.inv(matriz_cov)
    distancias = np.zeros((data.shape[0], centros.shape[0]))
    for k in range(centros.shape[0]):
        for i in range(data.shape[0]):
            delta = data[i, :] - centros[k, :]
            distancias[i,   k]   =   np.sqrt(np.dot(np.dot(delta.T,
inv_cov_matriz), delta))
    return distancias

#-----------------------------------------------------------------
-----
#   FUNCIÓN PARA CÁLCULO DE DISTANCIA DE COSENO
#-----------------------------------------------------------------
-----
def distancia_coseno(data, centros):
    distancias = np.zeros((data.shape[0], centros.shape[0]))
```

```
    for i in range(data.shape[0]):
        for j in range(centros.shape[0]):
            # Producto punto entre el dato y el centroide
            numerador = np.dot(data[i], centros[j])
            # Normas de los vectores
            norma_data = np.linalg.norm(data[i])
            norma_centro = np.linalg.norm(centros[j])
            # Distancia coseno
            distancias[i, j] = 1 - (numerador / (norma_data *
norma_centro))
    return distancias

#-----------------------------------------------------------------
-----
#   FUNCIÓN PARA CÁLCULO DE DISTANCIA MANHATTAN
#-----------------------------------------------------------------
-----

def distancia_manhattan(data,centros,num_clases):
    #Calcula la distancia manhattan entre cada dato y cada centro
    distancias = np.zeros((data.shape[0],num_clases))
    for i in range(data.shape[0]):
        distancias[i,0] = np.sum(np.abs(data[i,:] - centros[0,:]))
        distancias[i,1] = np.sum(np.abs(data[i,:] - centros[1,:]))

    return distancias

#-----------------------------------------------------------------
-----
# Crear un diccionario para almacenar los promedios por especie
resultados = {
    'Sex': [],
    'Culmen Length (mm)': [],
    'Culmen Depth (mm)': [],
    'Body Mass (g)': []
}

# Diccionario para almacenar los promedios
promedios_clases = {}

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
for specie, color in species.items():
    subset = train_data[train_data['Sex'] == specie]
    ax.scatter(subset['Culmen Length (mm)'], subset['Culmen Depth
(mm)'], subset['Body Mass (g)'], color=color, label=specie,
alpha=0.6)
```

```
    # Formato de la gráfica
    ax.set_xlabel('Culmen Length (mm)')
    ax.set_ylabel('Culmen Depth (mm)')
    ax.set_zlabel('Body Mass (g)')
    plt.title('TRAINING DATA (MACHO Y HEMBRA)')
    plt.legend()

    # Obtiene los promedios de las características de cada especie
    promediosCL = np.mean(subset['Culmen Length (mm)'])
    promediosCD = np.mean(subset['Culmen Depth (mm)'])
    promediosFL = np.mean(subset['Body Mass (g)'])

    # Agregar los resultados al diccionario
    resultados['Sex'].append(specie)
    resultados['Culmen Length (mm)'].append(promediosCL)
    resultados['Culmen Depth (mm)'].append(promediosCD)
    resultados['Body Mass (g)'].append(promediosFL)

    # Crear un array con los promedios
    promedio = np.array([promediosCL, promediosCD, promediosFL])

    # Guardar el array de promedios en el diccionario con el nombre
de la especie
    promedios_clases[specie] = promedio

#-----------------------------------------------------------------
-----
# Obtención de los promedios en un arreglo
centroides_promedio = np.zeros([2, 3])    # Dos clases, tres
características

centroides_promedio[0, :] = promedios_clases['MALE']
centroides_promedio[1, :] = promedios_clases['FEMALE']

# Crear diccionario para mapear las clases a nombres de especies
clases_mapeo = {0: 'MALE',
                1: 'FEMALE'}

# Extrae los datos del archivo test_data
longi = np.array(test_data['Culmen Length (mm)'])
prof = np.array(test_data['Culmen Depth (mm)'])
masa = np.array(test_data['Body Mass (g)'])
# Concatena los datos en una sola matriz
datos = np.column_stack((longi, prof, masa))

#-----------------------------------------------------------------
-----
```

```
# Calcula la distancia EUCLIDIANA entre los datos de test y los
centroides
dist_euclidiana = distancia_euclidiana(datos, centroides_promedio,
2)
asigna_euclidiana = np.argmin(dist_euclidiana, axis=1)
# Calcula la distancia de MAHALANOBIS entre los datos de test y los
centroides
dist_mahalanobis          =          distancia_mahalanobis(datos,
centroides_promedio)
asigna_mahalanobis = np.argmin(dist_mahalanobis, axis=1)
# Calcula la distancia de COSENO entre los datos de test y los
centroides
dist_coseno = distancia_coseno(datos, centroides_promedio)
# encontrar la clase con la distancia mínima
asigna_coseno = np.argmin(dist_coseno, axis=1)
#Calcula la distancia MANHATTAN entre los datos de test y los
centroides
dist_manhattan = distancia_manhattan(datos,centroides_promedio,2)
#encontrar la clase con la distancia mínima
asigna_manhattan = np.argmin(dist_manhattan,axis = 1)

#-----------------------------------------------------------------
-----
# Convertir las clases numéricas en nombres de especies
test_data['Clase Asignada (Dist euclidiana)'] = [clases_mapeo[clase]
for clase in asigna_euclidiana]
test_data['Clase     Asignada     (Dist     Mahalanobis)']     =
[clases_mapeo[clase] for clase in asigna_mahalanobis]
test_data['Clase Asignada (Dist Coseno)'] = [clases_mapeo[clase] for
clase in asigna_coseno]
test_data['Clase Asignada (Dist Manhattan)'] = [clases_mapeo[clase]
for clase in asigna_manhattan]

# Mostrar una tabla con la clase real y la clase asignada
tabla_resultados = test_data[['Species', 'Sex','Clase Asignada (Dist
euclidiana)', 'Clase Asignada (Dist Mahalanobis)', 'Clase Asignada
(Dist Coseno)', 'Clase Asignada (Dist Manhattan)']]
print(tabla_resultados)
```