UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE ESTUDIOS ESTADÍSTICOS

TAREA EVALUABLE

MINERÍA DE DATOS Y MODELIZACIÓN PREDICTIVA

ANÁLISIS DE COMPONENTES PRINCIPALES Y TÉCNICAS DE CLUSTERING

Jorge González Perea 51553561G

Máster en Big Data, Data Science & Inteligencia Artificial

Curso académico 2024-2025

Índice

1. Introducción.

En esta práctica se va a tratar un conjunto de datos que contiene información sobre ciertas características físicas y demográficas de un grupo de pingüinos. En el siguiente listado se encuentra el código para la obtención de los datos a partir de la librería *Seaborn*.

```
df = sns.load_dataset("penguins")
df.columns = ['Especie', 'Isla', 'Longitud del pico', 'Profundidad del pico',
              'Longitud de la aleta', 'Masa', 'Sexo']
display(df.head())
variables = df.columns.tolist()
v_numericas = df.select_dtypes(include = ['int', 'float']).columns.tolist()
v_categoricas = [var for var in v_numericas if var not in v_numericas]
```

Listing 1: Lectura de datos.

Nota: todas las librerías y funciones importadas se pueden encontrar en el primer listado del anexo.

La tabla de datos tiene la siguiente forma:

	Especie	Isla	Longitud del pico	Profundidad del pico	Longitud de la aleta	Masa	Sexo
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female

Figura 1: Cabecera de la tabla de datos.

Donde las variables son las siguientes:

- **Especie**: variable categórica que indica la especie de cada pingüino. Contiene tres valores posibles: '*Adelie*', '*Chinstrap*' o '*Gentoo*'.
- **Isla**: variable categórica que indica la isla donde habita cada pingüino. Puede tener los valores '*Torgersen*', '*Biscoe*' o '*Dream*'.
- **Longitud del pico**: variable numérica que indica la longitud del pico de cada pingüino en milímetros.
- **Profundidad del pico**: variable numérica que indica la profundidad del pico de cada pingüino en mm.
- **Longitud de la aleta**: variable numérica que indica la longitud de la aleta de cada pingüino en mm.
- **Masa**: variable numérica que indica la masa de cada pingüino en gramos.
- **Sexo**: variable categórica que indica el sexo de cada pingüino. Puede tener dos valores: '*Male*' o '*Female*'.

Además de sus valores correspondientes, algunas de estas variables presentan valores perdidos o **NaN** que habrá que tener en cuenta a la hora de trabajar con la base de datos.

1.1. Medidas estadísticas básicas.

Una visualización inicial del conjunto de datos es indispensable para poder comprobar si hay correlación aparente entre las variables. Para ello se construye una tabla de medidas estadísticas básicas (medias, desviaciones, etc.) para las variables numéricas. De esta forma se puede comprobar de antemano si existe algún problema con los datos, como por ejemplo un gran número de datos perdidos o un rango de valores sin sentido aparente. Los cálculos son los siguientes:

	Mínimo	Q1	Mediana	Q3	Media	Máximo	Desv. típica	Varianza	Coef. de Variación	Missing
Longitud del pico	32.1	39.22	44.45	48.5	43.92	59.6	5.46	29.81	0.12	2
Profundidad del pico	13.1	15.60	17.30	18.7	17.15	21.5	1.97	3.90	0.12	2
Longitud de la aleta	172.0	190.00	197.00	213.0	200.92	231.0	14.06	197.73	0.07	2
Masa	2700.0	3550.00	4050.00	4750.0	4201.75	6300.0	801.95	643131.08	0.19	2

Figura 2: Tabla de estadísticos.

1.2. Gráficos de dispersión.

Otra herramienta útil es la representación visual de los datos. Con la librería *Seaborn* se puede generar una matriz de gráficos que permite ver la correlación inicial entre los datos:

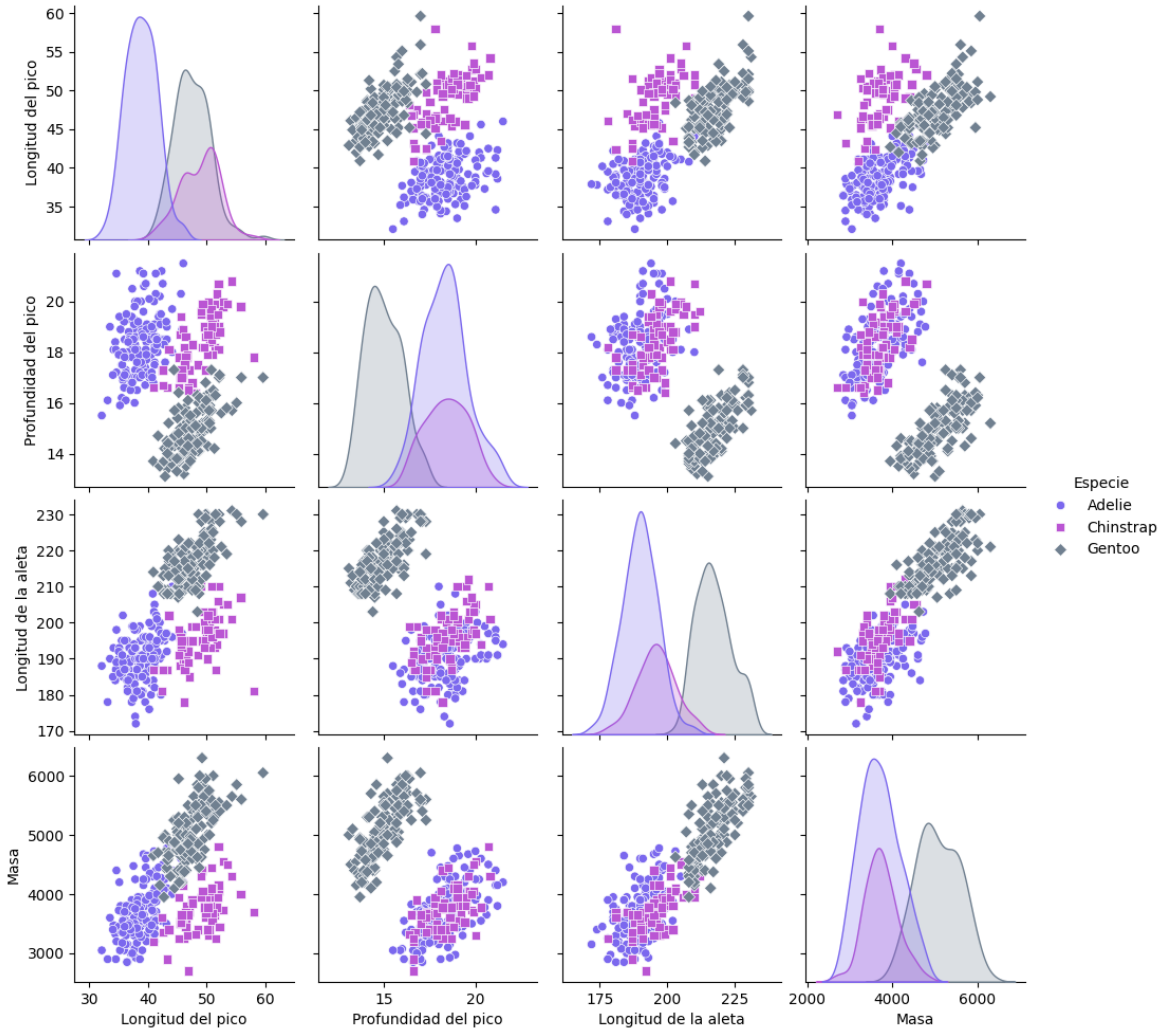

Figura 3: Gráficos de dispersión.

Como se puede observar en la figura 3, existe una clara distinción entre las medidas para cada especie (sobre todo para la especie '*Gentoo*', ya que sus puntos suelen encontrarse separados del resto en los gráficos de dispersión). Los histogramas indican cómo se distribuyen los datos entre las especies: se puede observar que hay un mayor número de pingüinos de especie '*Adelie*' que de las otras dos por separado; así como similitudes entre especies, como por ejemplo las distribuciones de masa para '*Adelie*' y '*Chinstrap*'. Otra observación interesante puede ser la fuerte correlación entre la longitud de la aleta o del pico y la masa de los pingüinos respectivamente, ya que presentan un cierto comportamiento lineal.

2. Matriz de correlación.

La matriz de correlación permite ver cómo se relacionan las variables visual y numéricamente al mismo tiempo.

```
R = df[v_numericas].corr()
plt.figure(figsize = (10, 8))
sns.heatmap(R, annot = True, cmap = 'Purples', fmt = '.2f', linewidths = 0.5)
plt.title('Matriz de correlaci n')
plt.show()
```

Listing 2: Cálculo de la matriz de correlación.

Y la representación gráfica de la matriz se puede ver en la imagen siguiente. Tal y como se observa, se tienen coeficientes de correlación positivos (correlación positiva) y negativos (correlación inversa):

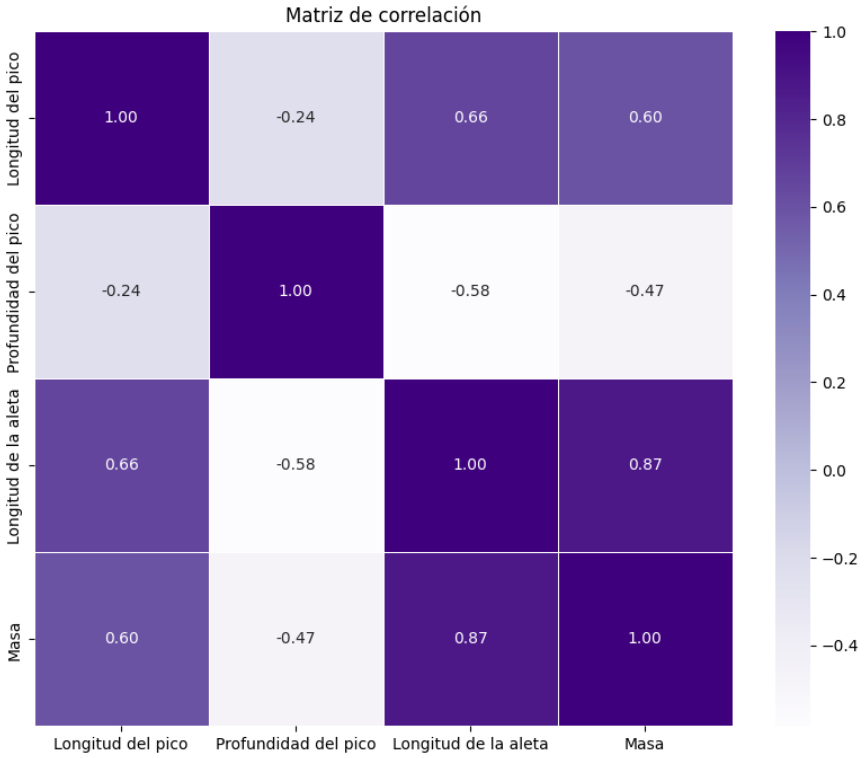

Figura 4: Matriz de correlación.

Donde es evidente que se tiene una fuerte correlación positiva entre la longitud de la aleta y la masa de los pingüinos (coeficiente con valor +0,87), aunque tanto estas dos variables y la longitud del pico presentan correlación entre ellas. Por otro lado, la única variable que presenta correlación inversa con los demás es la profundidad del pico, siendo notable sobre todo con la longitud de la aleta (−0,58).

Es decir, a primera vista, los pingüinos más pesados presentan las aletas y los picos más largos, así como los picos menos profundos.

3. Análisis de componentes principales.

El análisis de componentes principales (ACP) permite reducir la dimensionalidad del conjunto de datos perdiendo la menor información posible mediante la eliminación de ruido. Mediante este análisis se consigue que las variables, que están correlacionadas en cierta medida, dejen de estarlo (ya que se trabaja con los componentes principales, que son ortogonales). Un análisis inicial de componentes principales viene incluido en el siguiente listado:

```
df_std = pd.DataFrame(
    StandardScaler().fit_transform(df[v_numericas]),
    columns=['{}_z'.format(variable) for variable in v_numericas]).dropna()
display(df_std.head())
pca = PCA(n_components = 4)
fit = pca.fit(df_std)
autovalores = fit.explained_variance_
var_explicada = fit.explained_variance_ratio_
var_acumulada = np.cumsum(var_explicada)
data = {'Autovalores': autovalores,
        'Variabilidad Explicada': var_explicada,
        'Variabilidad Acumulada': var_acumulada}
tabla = pd.DataFrame(data, index=['Componente {}'.format(i) for i in range(1, fit.
n_components_+1)])
plot_varianza_explicada(var_explicada, fit.n_components_)
```

Listing 3: Análisis de componentes principales.

Estas instrucciones llevan a cabo el siguiente proceso: se normalizan los datos con la función *StandarScaler* para que tengan media $\mu = 0$ y desviación estándar $\sigma = 1$, y se ajusta la normalización a los datos con el método *fit_ transform*. Posteriormente se crea una instancia de PCA con el número máximo de componentes posibles (número de variables numéricas) gracias a la función *PCA* y se aplica el análisis a los datos con el método *fit*. Por último, se calculan de forma sencilla los autovalores y la varianza explicada. La representación del ACP se puede ver en la siguiente imagen:

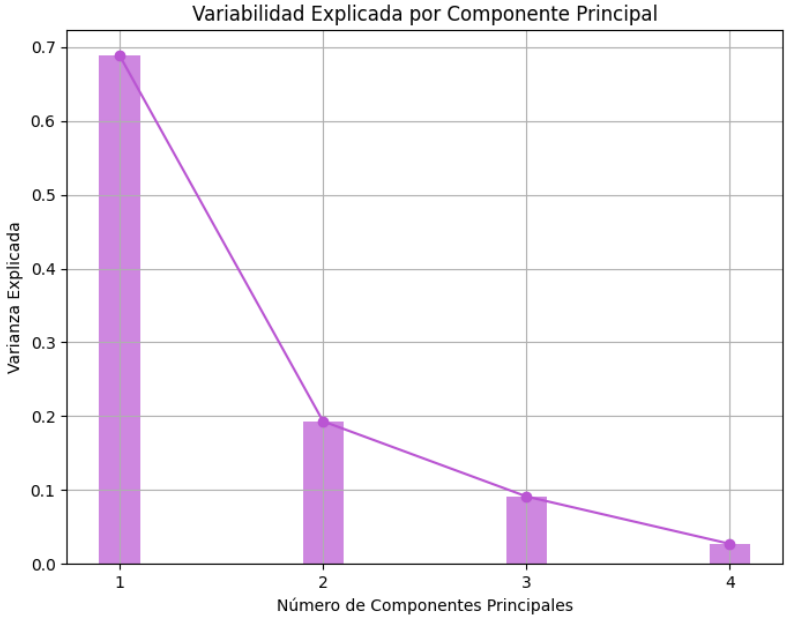

Figura 5: Análisis de componentes principales.

Tal y como se puede observar, con tan sólo los dos primeros componentes principales se explica cerca del 90 % de la varianza de los datos (en concreto un 88,1 %), por lo que son suficientes para el estudio.

3.1. ACP con los componentes seleccionados.

A continuación se crea otra instancia de ACP con únicamente los dos componentes de interés.

```
pca = PCA(n_components = 2)
fit = pca.fit(df_std)
autovectores = pca.components_
autovectores = pd.DataFrame(pca.components_.T,
    columns = ['Autovector {}'.format(i) for i in range(1, fit.n_components_+1)],
    index = ['{}_z'.format(variable) for variable in v_numericas])
resultados_pca = pd.DataFrame(fit.transform(df_std),
    columns=['Componente {}'.format(i) for i in range(1, fit.n_components_+1)],
    index=df_std.index)
df_z_cp = pd.concat([df_std, resultados_pca], axis=1)

```

Listing 4: ACP con 2 componentes.

Una vez obtenidos los autovectores de cada componente para cada una de las cuatro variables numéricas, se pueden representar de multitud de formas diferentes.

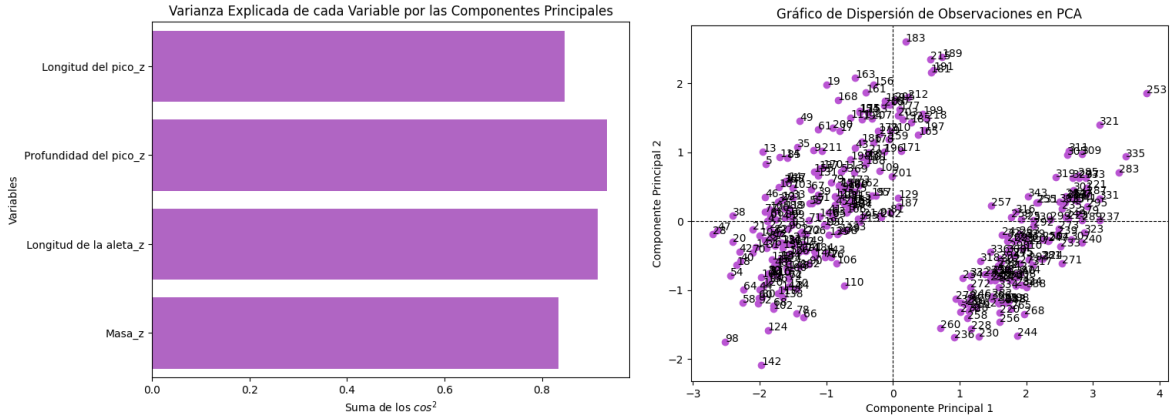

Figura 6: **Izq**: varianza explicada por los componentes principales para cada variable. **Dch**: gráfico de dispersión de los componentes principales.

En estas imágenes se puede ver la proporción de la varianza que explica cada variable, así como la separación de las componentes principales en dos grandes grupos. Por otro lado, los autovectores son los siguientes:

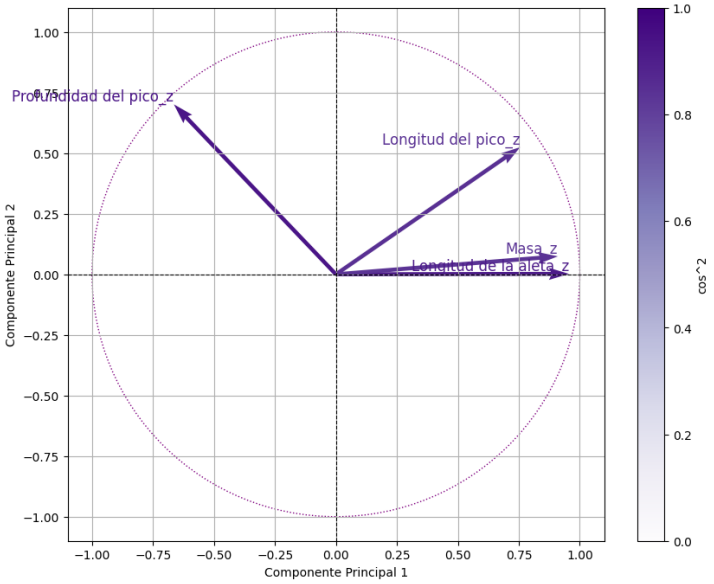

Figura 7: Autovectores del análisis de componentes principales.

Otro gráfico útil es el que se ve en la figura 8, ya que muestra la contribución de cada variable a los

componentes principales. Se puede observar que, por ejemplo, la longitud de la aleta apenas contribuye al valor de la segunda componente.

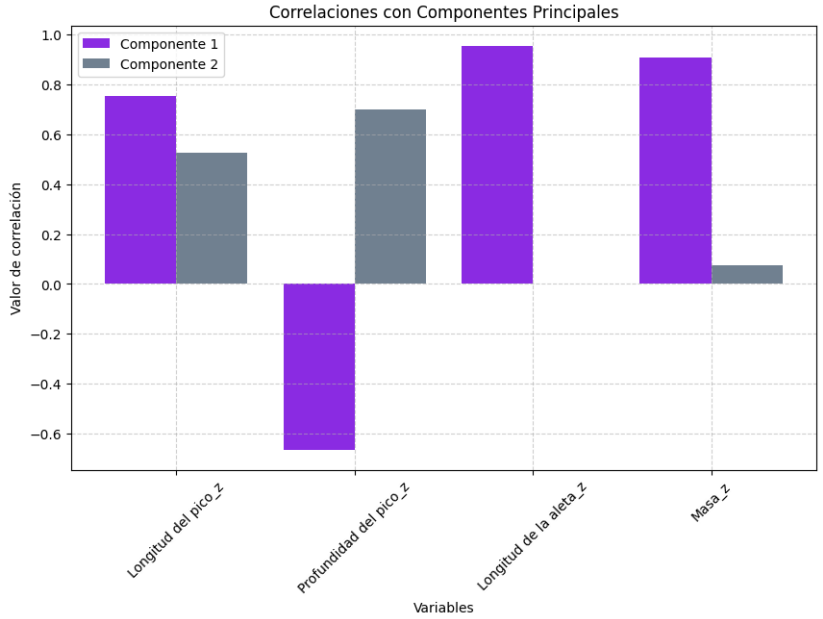

Figura 8: Carga de los componentes principales en función de las variables.

La conclusión que se puede sacar de estos resultados es la siguiente:

- La componente principal 1 tiene relación sobre todo con el tamaño físico de los pingüinos, ya que contribuye en gran medida a la correlación de las longitudes de aletas y picos, profundidad de estos y a la masa de cada individuo.
- Asímismo, la componente principal 2 explica las características físicas de los picos únicamente. Tal y como se puede ver en la imagen 8, no tiene correlación notable con las aletas o la masa en comparación con la anterior componente.

Una diferenciación entre especies en el segundo gráfico de la figura 6 permite ver patrones estadísticos:

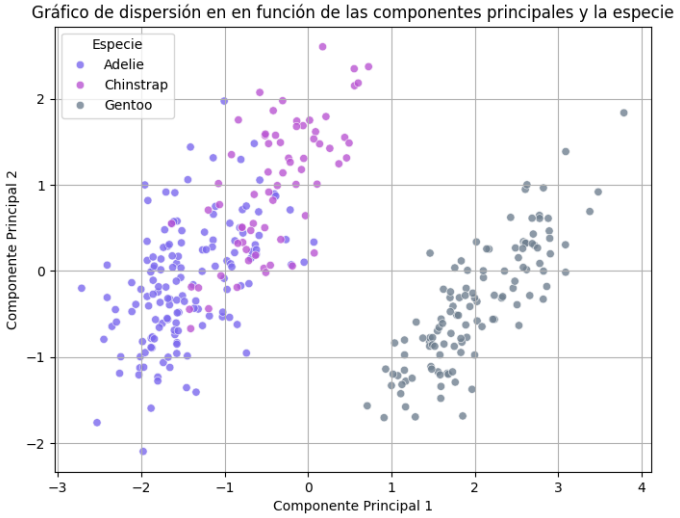

Figura 9: Dispersión en el espacio de las componentes principales en función de la especie.

Donde es evidente que los pingüinos de la especie *Gentoo* contribuyen más a la primera componente y por lo tanto tendrán características físicas más pronunciadas en cuanto a tamaño, longitud del pico y masa.

Además, tienen picos mucho menos profundos. Por el contrario, las especies *Adelie* o *Chinstrap* representan picos más característicos, ya que contribuyen más a la segunda componente principal, incluyendo una ligera diferencia en cuanto a dimensiones de longitud, pero una diferencia notable en cuanto a masa.

3.2. Índices de combinación lineal.

Para evaluar de forma conjunta las características físicas de los pingüinos se puede calcular un índice resultado de la combinación lineal de las dos componentes principales (que viene siendo una combinación lineal de todas las variables ya que estas representan casi toda la varianza de los datos). Para ello se puede distinguir por especies y utilizar el promedio. Además, no se ha tenido en cuenta que ninguna de las dos componentes principales tenga más importancia que la otra:

```
df = df.dropna()
df_std = StandardScaler().fit_transform(df[v_numericas])
pca = PCA(n_components = 2)

fit = pca.fit_transform(df_std)
df_pca = pd.DataFrame(fit, columns=['PC1', 'PC2'])
df_pca['Especie'] = df['Especie'].values

coef = [1, 1]
df_pca[' ndice  PCA'] = np.dot(df_pca[['PC1', 'PC2']], coef)

index_adelie_pca = df_pca[df_pca['Especie'] == 'Adelie'][' ndice  PCA'].mean()
index_chinstrap_pca = df_pca[df_pca['Especie'] == 'Chinstrap'][' ndice  PCA'].mean()
index_gentoo_pca = df_pca[df_pca['Especie'] == 'Gentoo'][' ndice  PCA'].mean()

print(f"Valor del  ndice  PCA para la especie 'Adelie': {index_adelie_pca}")
print(f"Valor del  ndice  PCA para la especie 'Chinstrap': {index_chinstrap_pca}")
print(f"Valor del  ndice  PCA para la especie 'Gentoo': {index_gentoo_pca}")
```

Listing 5: Coeficientes de combinación lineal.

Y se obtienen los siguientes índices para cada especie:

Especie	**Índice PCA**
Adelie	-1.6012
Chinstrap	0.6048
Gentoo	1.6189

Tabla 1: Índice PCA por especie de pingüino

Es decir, y tal y como se ha estado comprobando hasta ahora, los pingüinos de la especie *Adelie* o *Gentoo* tendrán características físicas más pronunciadas y distintivas que los pingüinos de la especie *Chinstrap*.

4. Análisis de *Clustering*.

Para llevar a cabo un análisis de *Clustering* es necesario aplicar primero un modelo jerárquico que permita identificar el número de grupos o *clusters* con los que se va a trabajar. Para ello, se crea en el siguiente listado un dendrograma con todos los posibles grupos.

```
distance_std = distance.cdist(df_std, df_std,"euclidean")
df_std_distance = pd.DataFrame(distance_std, index = df_std.index, columns = df.index)
linkage_matrix = sch.linkage(df_std_distance, method='ward')
dendrogram = sch.dendrogram(linkage_matrix, labels=df.index, leaf_font_size=9,
leaf_rotation=90)
plt.title('Dendrograma de las observaciones')
plt.xlabel('Individuos')
plt.ylabel('Distancia euc dea')
plt.xticks([], [])
plt.grid()
plt.show()
```

Listing 6: Código para generar el dendrograma

El dendrograma queda tal y como se observa en la figura siguiente, donde se puede ver que, a primera vista y según el modelo jerárquico, el número óptimo de grupos sería 3. Además, este número de *clusters* cuadra con los datos empíricos de especies vistos en los gráficos anteriores, ya que se tienen 3 especies de pingüinos.

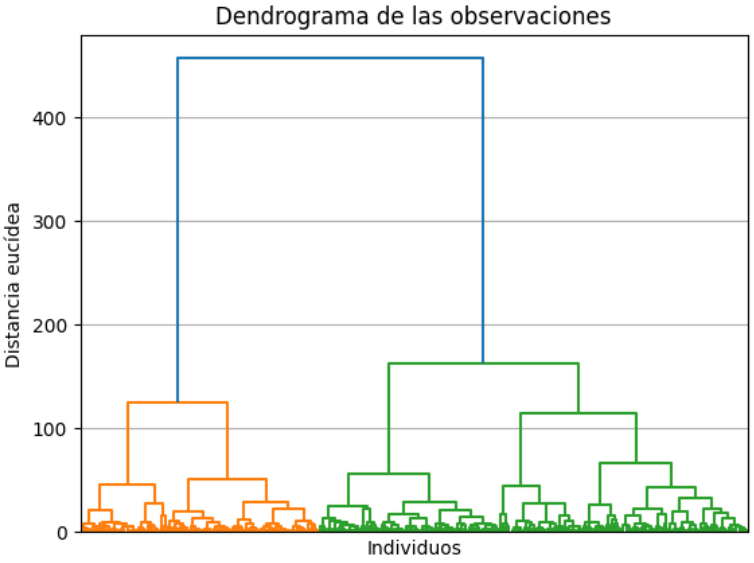

Figura 10: Dendograma de los datos.

4.1. Agrupamiento *K-means*.

El método K-Means es un algoritmo de clustering que agrupa a los pingüinos según sus características físicas sin usar etiquetas de especie. Funciona asignando cada pingüino a uno de K grupos con centroides que representan el centro de cada clúster. En esta práctica, K-Means puede identificar patrones ocultos en los datos, agrupando especies similares basándose en variables como longitud del pico, profundidad del pico, longitud de la aleta y masa. Esto permite comparar los clústeres obtenidos con las especies reales y evaluar la separación entre ellas.

```
k = 3
kmeans = KMeans(n_clusters=k, random_state=0)
kmeans.fit(df_std)
kmeans_cluster_labels = kmeans.labels_
plt.figure(figsize=(10, 6))

```

```
    for cluster in np.unique(kmeans_cluster_labels):
        plt.scatter(df_pca.loc[kmeans_cluster_labels == cluster, 'PC1'],
                    df_pca.loc[kmeans_cluster_labels == cluster, 'PC2'],
                    label=f'Cluster {cluster}')
    for i, row in df_pca.iterrows():
        plt.text(row['PC1'], row['PC2'], str(df.index[i]), fontsize=8)

    plt.title("Gr fico de ACP con asignaciones K-means")
    plt.xlabel("Componente Principal 1")
    plt.ylabel("Componente Principal 2")
    plt.legend()
    plt.grid()
    plt.show()
```

Listing 7: Código para generar un gráfico de dispersión diferenciando por clusters.

Tal y como se observa, los 3 grupos elegidos se ajustan muy bien (a primera vista) a los datos diferenciados por especies del gráfico de la figura 9.

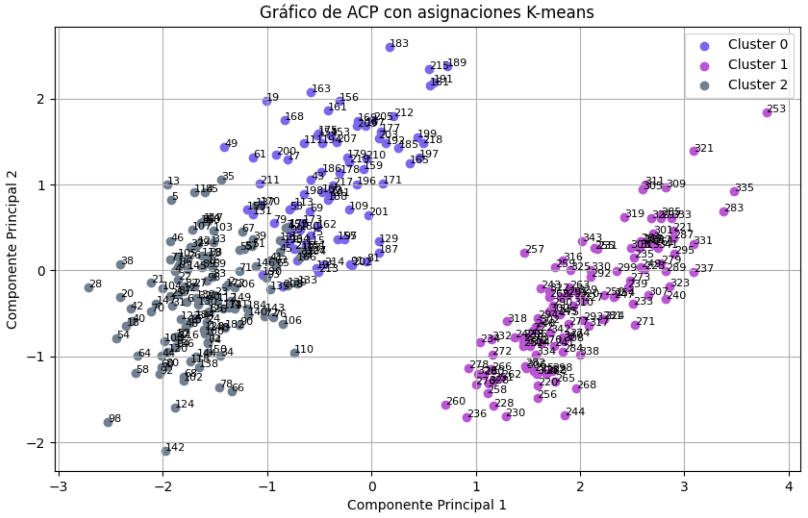

Figura 11: Scatter de ACP por método K-Means para k = 3 grupos.

Otro valor óptimo para el número de *clusters* sería $k = 2$, ya que diferencia los dos grupos grandes en el gráfico de dispersión que se ven desde el principio y sin necesidad de ningún tipo de análisis de los datos:

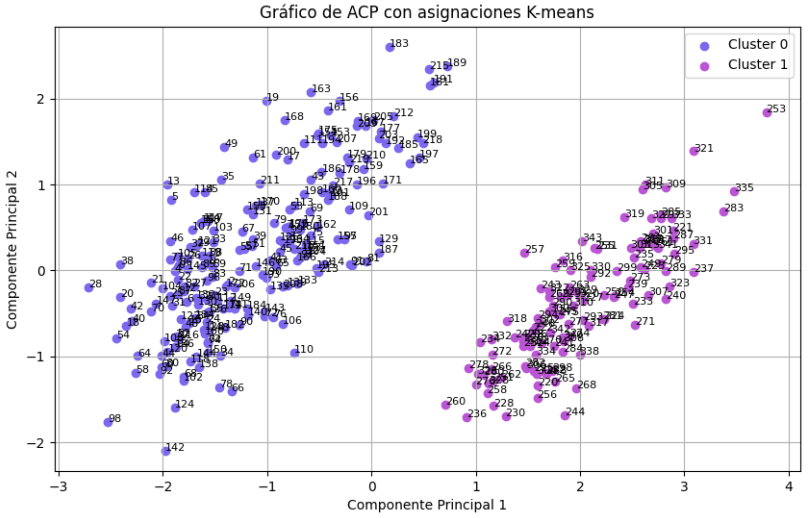

Figura 12: Scatter de ACP por método K-Means para k = 2 grupos.

4.2. Método del codo.

El método del codo es una otra técnica utilizada para determinar el número óptimo de clusters. Se basa en calcular la suma de cuadrados dentro de los grupos (WCSS) para distintos valores de K y trazar una curva. A medida que K aumenta, WCSS disminuye, pero llega un punto donde la reducción es mínima, formando un çodo", que indica la K óptima. En esta práctica sobre pingüinos, este método ayuda a definir cuántos grupos naturales existen en los datos según sus características físicas. Aplicándolo, se puede elegir el número adecuado de clusters para que la agrupación sea representativa sin sobreajustar los datos.

```
    wcss = []
    for k in range(1, 11):
        kmeans = KMeans(n_clusters=k, random_state=0)
        kmeans.fit(df_std)
        wcss.append(kmeans.inertia_)
    plt.figure(figsize=(8, 6))
    plt.plot(range(1, 11), wcss, marker='o', linestyle='-', color='mediumorchid')
    plt.title('M todo del Codo')
    plt.xlabel('N mero de Clusters (K)')
    plt.ylabel('WCSS')
    plt.grid()
    plt.show()
```

Listing 8: Código para generar una curva mediante el método del codo.

Se puede observar que la curva se dobla (la posición del codo está) en $k = 3$, por lo que este es el número óptimo de *clusters*, aunque $k = 2$ también sería una opción válida.

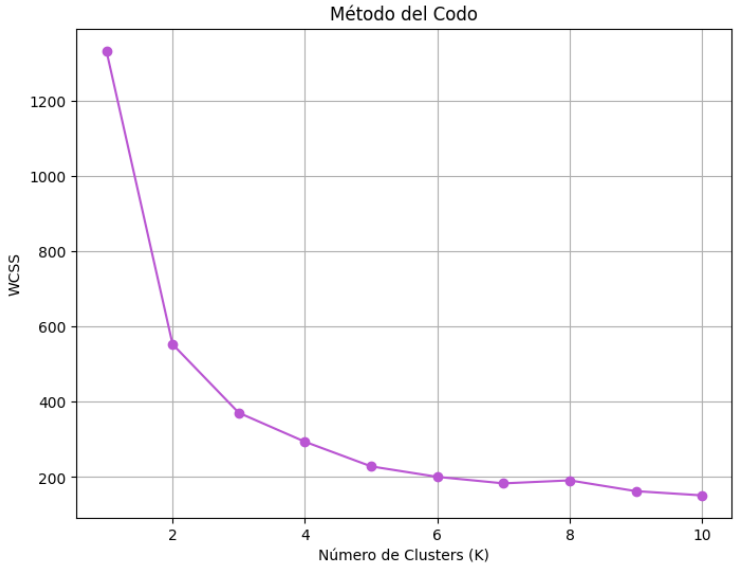

Figura 13: Curva del método del codo.

4.3. Método de la silueta.

El método de la silueta es otra técnica para determinar el número óptimo de *clusters* en un análisis de agrupamiento. Se basa en calcular, para cada observación, la distancia promedio a los puntos dentro de su mismo grupo y la distancia al *cluster* más cercano. Con estos valores, se obtiene una puntuación de silueta que varía entre -1 y 1: valores cercanos a 1 indican una buena agrupación, mientras que valores cercanos a 0 o negativos sugieren superposición o una mala asignación. En esta práctica, este método permite evaluar qué cantidad de *clusters* logra la mejor separación entre especies de pingüino basándose en sus características físicas. El listado siguiente contiene el código para generar los gráficos que permiten visualizar este método.

```
    from sklearn.metrics import silhouette_score
    silhouette_scores = []
    for k in range(2, 11):
```

```
        kmeans = KMeans(n_clusters=k, random_state=0)
        kmeans.fit(df_std)
        labels = kmeans.labels_
        silhouette_avg = silhouette_score(df_std, labels)
        silhouette_scores.append(silhouette_avg)
    plt.figure(figsize=(8, 6))
    plt.plot(range(2, 11), silhouette_scores, marker='o', linestyle ='-', color='
    mediumorchid')
    plt.title('M todo de la Silueta')
    plt.xlabel('Number of Clusters (K)')
    plt.ylabel('')
    plt.grid(True)
    plt.show()

    kmeans = KMeans(n_clusters=3, random_state=0)
    kmeans.fit(df_std)
    labels = kmeans.labels_
    silhouette_values = silhouette_samples(df_std, labels)
    plt.figure(figsize=(8, 6))
    y_lower = 10
    color = ['#7B68EE', '#BA55D3', '#708090', '#708090']
    for i in range(4):
        ith_cluster_silhouette_values = silhouette_values[labels == i]
        ith_cluster_silhouette_values.sort()
        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i
        plt.fill_betweenx(np.arange(y_lower, y_upper),
                          0, ith_cluster_silhouette_values,
                          facecolor=color[i], edgecolor=color, alpha=0.7)

        plt.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))
        y_lower = y_upper + 10
    plt.title("Gr fico de siluetas para los grupos")
    plt.xlabel("Coeficientes de siluetas")
    plt.ylabel("Cluster")
    plt.grid(True)
    plt.show()
```

Listing 9: Código para generar una curva mediante el método del codo.

Se generan de esta forma los dos gráficos de la imagen siguiente. Tal y como se observa, el número óptimo de grupos es $k = 2$, ya que es el valor que maximiza la curva del método (ver figura 14 izquierda). Sin embargo, el valor $k = 3$ sigue siendo un número adecuado de *clusters* ya que no se aleja mucho. Además, para $k = 3$ *clusters*, el grupo mejor definido es el que tiene la etiqueta 1, con una mayoría de observaciones por encima del umbral de 0,4 para los coeficientes; y el peor definido el que tiene la etiqueta 0 por los valores tan dispares de los coeficientes de las observaciones (que pueden indicar superposición de grupos).

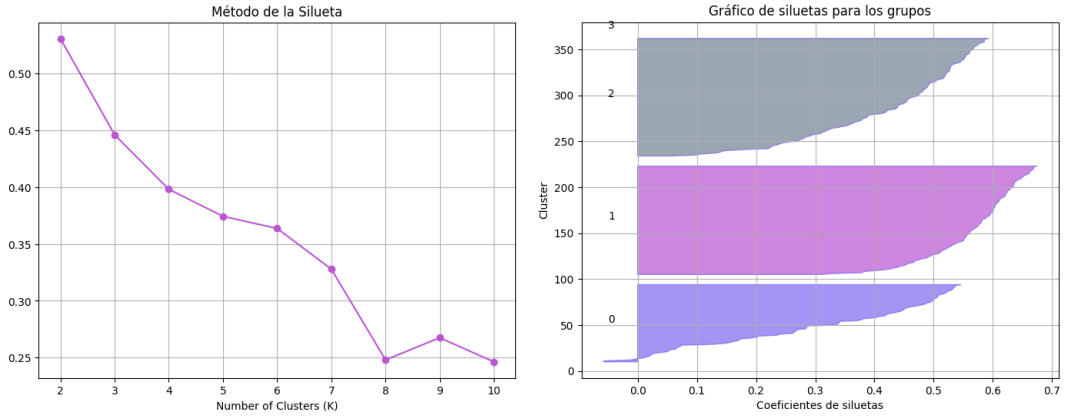

Figura 14: Representación del método de las siluetas.

En resumen, como la mayoría de coeficientes tienen un valor positivo y mayor que 0,4, se puede concluir que el agrupamiento tiene una calidad mejorable, pero es aceptable.

4.4. Comparación de métodos.

De los resultados obtenidos se puede concluir que el número óptimo de conglomerados es $k = 3$, en parte debido a la diferenciación entre especies. Los resultados de ambos métodos de *clustering* (jerárquico y *k-means*) son compatibles. La principal diferencia es que el método jerárquico puede estar abierto a interpretación del dendrograma, ya que las distancias euclídeas entre grupos pueden variar mucho dentro de la misma matriz de distancias. Por otro lado, los métodos de *k-means* proporcionan resultados matemáticamente más riguroso y exactos si se asume el riesgo de la aleatoriedad en la asignación de los centroides.

La conclusión principal es que aplicar ambos métodos en conjunto permite optimizar el trabajo a la hora de agrupar datos mediante clustering, ya que el método jerárquico permite la visualización de los grupos, y el método no jerárquico proporciona la elección de k más exacta.

4.5. Interpretación de los grupos.

Los tres grupos detectados mediante *clustering* reflejan la diferenciación natural entre las especies de pingüinos en función de sus características físicas.

- **Grupo 1 (*Gentoo*)**:
  - Representa los pingüinos con mayor tamaño, ya que presentan picos más largos, aletas más grandes y mayor masa.
  - Se encuentran muy separados en el análisis de componentes principales (ACP), indicando que sus características físicas los distinguen claramente de las demás especies.
- **Grupo 2 (*Adelie*)**:
  - Incluye individuos con tamaños intermedios, con un equilibrio entre la longitud del pico, la aleta y la masa.
  - Aunque se diferencian de los *Gentoo*, pueden solaparse con los *Chinstrap* debido a similitudes en el tamaño y estructura del pico.
- **Grupo 3 (*Chinstrap*)**:
  - Se caracteriza por picos más profundos pero no necesariamente más largos.
  - Es el grupo más difícil de separar de los *Adelie*, lo que indica que estas dos especies comparten más características en común.

Además, se observan las siguientes tendencias en los datos:

- Los pingüinos *Gentoo* se diferencian claramente de las otras dos especies debido a su tamaño y morfología más grande.
- *Adelie* y *Chinstrap* presentan mayor solapamiento, lo que sugiere que sus características físicas no son tan distintas como en el caso de los *Gentoo*.
- El método del codo y la silueta validan la existencia de estos tres grupos, lo que indica que la clasificación biológica se refleja en los datos.

En conclusión, el *clustering* revela que la estructura de las especies está bien representada en los datos, aunque la diferenciación entre Adelie y *Chinstrap* es más difusa.

5. Conclusiones.

El análisis realizado sobre los datos de pingüinos ha permitido identificar patrones en sus características físicas mediante *Análisis de Componentes Principales* (ACP) y técnicas de *clustering*.

- El ACP muestra que las dos primeras componentes principales explican cerca del 88 % de la varianza, lo que permite representar los datos en un espacio reducido sin perder demasiada información.
- El índice PCA construido a partir de estas componentes revela que los pingüinos *Gentoo* tienen características físicas más pronunciadas, mientras que *Adelie* y *Chinstrap* son más similares entre sí.
- Los métodos del codo y de la silueta sugieren que el número óptimo de clusters es $k = 3$, lo que coincide con las tres especies de pingüinos presentes en los datos.
- El *clustering* con *K-Means* y jerárquico dio resultados coherentes con la clasificación biológica real, aunque el método jerárquico dejó mayor margen de interpretación.

5.1. Limitaciones.

- La limitación principal es la superposición de especies: aunque el *clustering* con $k = 3$ parece adecuado, existe solapamiento entre las especies Adelie y Chinstrap, lo que dificulta su diferenciación clara.
- Sensibilidad a la normalización: los resultados dependen de la estandarización de los datos, lo que puede influir en la asignación de los clusters.
- Incertidumbre de los métodos: tanto el método del codo como el de silueta proporcionan sugerencias pero no siempre una respuesta definitiva, por lo que pueden ser necesarios otros criterios para validar los resultados.

En conclusión, el análisis confirma que las especies de pingüinos presentan diferencias en sus características físicas que pueden capturarse mediante ACP y *clustering*, aunque existen desafíos en la separación de ciertas especies.

6. Anexo.

En esta sección se encuentran tablas, listados, gráficas o imágenes relativas a la práctica pero de menor relevancia.

```
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    from matplotlib import cm
    import seaborn as sns
    from sklearn.decomposition import PCA
    from sklearn.preprocessing import StandardScaler
    import scipy.cluster.hierarchy as sch
    from scipy.spatial import distance
    from sklearn.cluster import KMeans
    from sklearn.metrics import silhouette_samples
    from FuncionesMineria2 import (plot_varianza_explicada, plot_cos2_heatmap, plot_corr_cos
    , plot_cos2_bars, plot_contribuciones_proporcionales, plot_pca_scatter,
    plot_pca_scatter_with_vectors, plot_pca_scatter_with_categories)
```

Listing 10: Librerías y funciones.

```
estadisticos = pd.DataFrame({
    'M nimo': df[v_numericas].min(),
    'Q1': df[v_numericas].quantile(0.25),
    'Mediana': df[v_numericas].median(),
    'Q3': df[v_numericas].quantile(0.75),
    'Media': df[v_numericas].mean(),
    'M ximo': df[v_numericas].max(),
    'Desv. t pica': df[v_numericas].std(),
    'Varianza': df[v_numericas].var(),
    'Coef. de Variaci n': (df[v_numericas].std() / df[v_numericas].mean()),
    'Missing': df[v_numericas].isna().sum()})
estadisticos = estadisticos.round(decimals = 2)
display(estadisticos)
```

Listing 11: Tabla de estadísticos.

```
sns.pairplot(df,
             vars = v_numericas,
             hue = 'Especie',
             diag_kind = 'kde',
             markers = ['o', 's', 'D'],
             palette = ['#7B68EE', '#BA55D3', '#708090'])
plt.show()
```

Listing 12: código para generar la figura 3.

```
    variables_cp = df_z_cp.columns
    correlacion = pd.DataFrame(np.corrcoef(df_std.T, resultados_pca.T),
                               index = variables_cp, columns = variables_cp)
    n_variables = fit.n_features_in_
    correlaciones_df_con_cp = correlacion.iloc[:fit.n_features_in_, fit.n_features_in_:]
    display(correlaciones_df_con_cp)
    cos2 = correlaciones_df_con_cp**2
    display(cos2)
    plot_cos2_heatmap(cos2)

    plot_corr_cos(fit.n_components, correlaciones_df_con_cp)
    plot_cos2_bars(cos2)
    plot_pca_scatter(pca, df_std, fit.n_components)
```

Listing 13: Código para generar las figuras 6 y 7.

```
    fig, ax = plt.subplots(figsize=(10, 6))
    indices = np.arange(len(correlaciones_df_con_cp))
    ancho = 0.4
    ax.bar(indices - ancho/2, correlaciones_df_con_cp['Componente 1'], width=ancho, color='
    #8A2BE2', label='Componente 1')  # Morado
```

```
    ax.bar(indices + ancho/2, correlaciones_df_con_cp['Componente 2'], width=ancho, color='
    #708090', label='Componente 2')  # Gris
    ax.set_xticks(indices)
    ax.set_xticklabels(correlaciones_df_con_cp.index)
    ax.set_ylabel('Valor de correlaci n')
    ax.set_xlabel('Variables')
    ax.set_title('Correlaciones con Componentes Principales')
    ax.legend()
    plt.grid(True, linestyle='--', alpha=0.6, zorder = 0)
    plt.xticks(rotation=45)
    plt.show()
```

Listing 14: Código para generar la figura 8.

```
    df = sns.load_dataset('penguins')
    df.columns = ['Especie', 'Isla', 'Longitud del pico', 'Profundidad del pico',
                  'Longitud de la aleta', 'Masa', 'Sexo']
    df = df.dropna()
    X = df[['Longitud del pico', 'Profundidad del pico', 'Longitud de la aleta', 'Masa']]
    scaler = StandardScaler()
    X_std = scaler.fit_transform(X)
    pca = PCA(n_components=2)
    X_pca = pca.fit_transform(X_std)
    df_pca = pd.DataFrame(X_pca, columns=['PC1', 'PC2'])
    df_pca['Especie'] = df['Especie'].values
    plt.figure(figsize=(8, 6))
    sns.scatterplot(x='PC1', y='PC2', hue='Especie', data=df_pca, palette = ['#7B68EE', '#
    BA55D3', '#708090'], alpha=0.8)
    plt.xlabel('Componente Principal 1')
    plt.ylabel('Componente Principal 2')
    plt.title('Gr fico de dispersi n en en funci n de las componentes principales y la
    especie')
    plt.legend(title='Especie')
    plt.grid()
    plt.show()
```

Listing 15: Código para generar la figura 9.