

Máster Big Data, Data Science & Inteligencia Artificial 2024-2025 - UCM

Profesor: Dr Daniel Martin

Alumno: Ing Paez Sheridan, Pablo Santiago

Materia: Minería de datos y modelización predictiva II

Tarea de Minería de datos y modelización predictiva II

Minería de Datos y Modelización Predictiva II - Tarea

Analisis de Componentes Principales (PCA)

Liberias usadas:

```
In [38]: import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import scipy.cluster.hierarchy as sch

from sklearn.discriminant_analysis import StandardScaler
from sklearn.decomposition import PCA
from scipy.spatial import distance
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples

import warnings
warnings.filterwarnings("ignore")
```

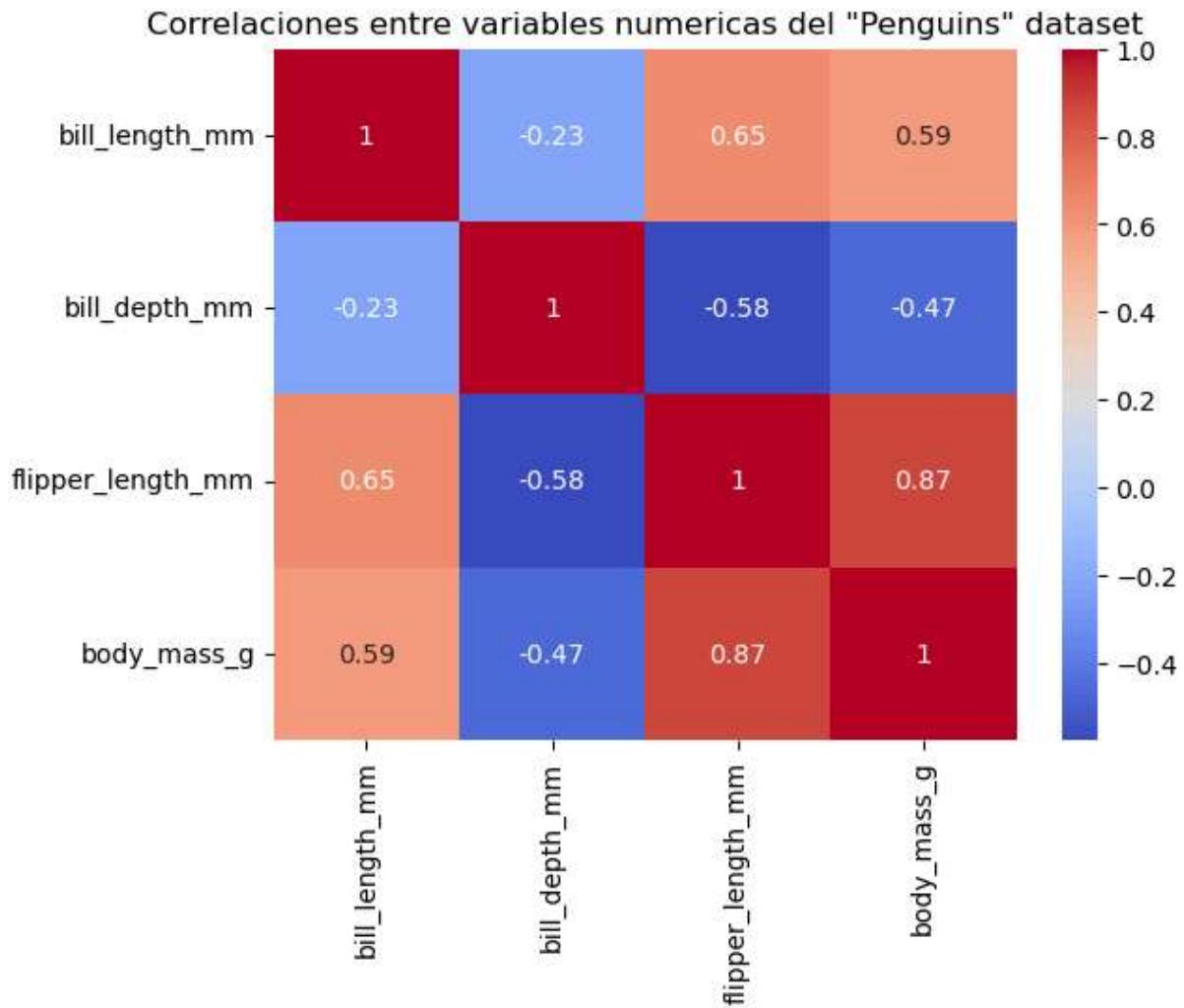
1 - Matriz de correlaciones entre las variables del conjunto de datos

```
In [39]: penguins = sns.load_dataset('penguins')

penguins = penguins.dropna()

numeric_cols = penguins.select_dtypes(include=['number']).columns
penguins_numeric_data = penguins[numeric_cols]
correlation_matrix = penguins_numeric_data.corr()

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlaciones entre variables numericas del "Penguins" dataset')
plt.show()
```



Las variables más correlacionadas son body_mass_g con flipper_length_mm, las otras relaciones se encuentran escasamente correlacionadas.

2 - Análisis de componentes principales

primero hacemos la estandarización de los datos para que todas las variables estén en la misma escala

```
In [40]: data_standardized = StandardScaler().fit_transform(penguins_numeric_data)
```

Ahora hacemos la primera selección de componentes principales, tomando como número de componentes el total del número de variables, para retener el máximo de componentes principales y analizar la pérdida de explicación de la variabilidad del conjunto de datos con respecto a un menor número de componentes.

```
In [41]: pca = PCA(n_components=4)
fit = pca.fit(data_standardized)
```

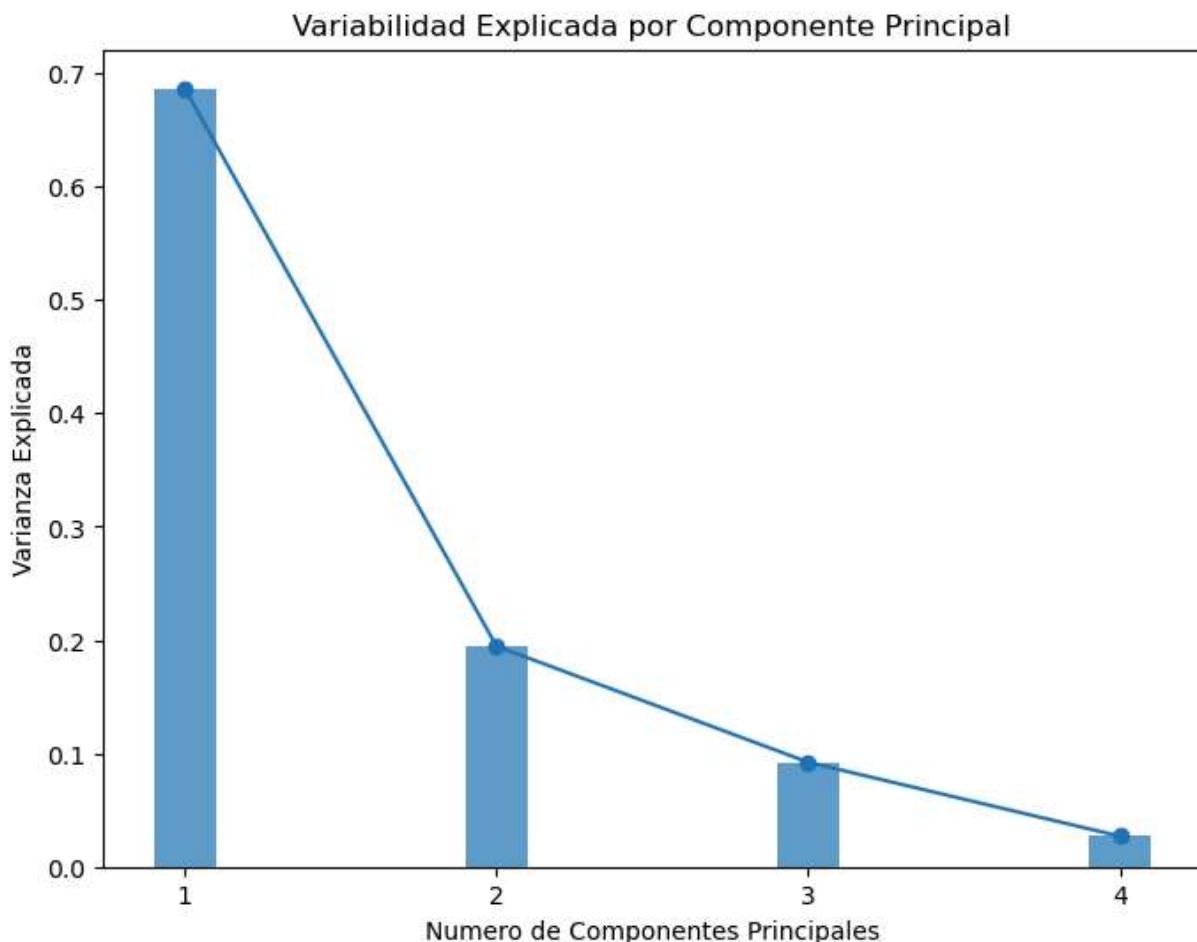
Calculamos la variabilidad explicada de cada componente principal calculado. La mostramos para cada componente junto con su autovalor y la variabilidad acumulada.

```
In [42]: 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)])
print(tabla)
```

	Autovalores	Variabilidad Explicada	Variabilidad Acumulada
Componente 1	2.753625	0.686339	0.686339
Componente 2	0.780461	0.194529	0.880868
Componente 3	0.369753	0.092161	0.973029
Componente 4	0.108210	0.026971	1.000000

Si bien, al ser tan pocas variables, no hace falta mucho más que un cuadro comparativo para decidir el número adecuado de componentes, a fines de ser rigurosos vamos a hacer un gráfico de la variabilidad explicada de cada componente, ordenada de mayor a menor, para decidir el número de componentes en base a la 'regla del codo'.

```
In [43]: def plot_varianza_explicada(var_explicada, n_components):
    num_componentes_range = np.arange(1, n_components + 1)
    plt.figure(figsize=(8, 6))
    plt.plot(num_componentes_range, var_explicada, marker='o')
    plt.xlabel('Número de Componentes Principales')
    plt.ylabel('Varianza Explicada')
    plt.title('Variabilidad Explicada por Componente Principal')
    plt.xticks(num_componentes_range)
    plt.bar(num_componentes_range, var_explicada, width=0.2, align='center', alpha=0.5)
    plt.show()
plot_varianza_explicada(var_explicada, fit.n_components_)
```



El número de componentes principales que elegimos como indicado es 2 componentes principales, ya que podemos observar que al incorporar el 3er componente principal, ya no incorporamos proporcionalmente mucha explicación de la variabilidad, y el 88% de la explicación de la variabilidad que tenemos con 2 componentes es suficientemente significativo como para conformarnos con ese porcentaje y con dos componentes principales.

3 - Análisis de Componentes principales sobre la matriz de correlaciones

Ya que hemos decidido quedarnos con dos componentes principales procedemos a calcular los autovectores para cada uno

```
In [44]: pca = PCA(n_components=2)
fit = pca.fit(data_standardized)
autovectores = pd.DataFrame(pca.components_.T,
columns = ['Autovector {}'.format(i) for i in
range(1, pca.n_components_+1)],
index = ['{}_z'.format(variable) for variable
in list(penguins_numeric_data.columns)])
print(autovectores)
```

	Autovector 1	Autovector 2
bill_length_mm_z	0.453753	0.600195
bill_depth_mm_z	-0.399047	0.796170
flipper_length_mm_z	0.576825	0.005788
body_mass_g_z	0.549675	0.076464

Ahora vamos a calcular los valores de los componentes principales para cada una de las observaciones de nuestro conjunto de datos final. Para facilitar la legibilidad, vamos a mostrar solo la primera observación con los CP calculados, pero nos vamos a guardar el nuevo dataset con los CP añadidos a él (podríamos eliminar las variables originales para cada observación, reteniendo el 88% de la variabilidad del conjunto de datos original, como ya enunciamos antes).

```
In [45]: data_standardized_df = pd.DataFrame(data_standardized,
                                             columns=penguins_numeric_data.columns,
                                             index=penguins_numeric_data.index)

resultados_pca = pd.DataFrame(fit.transform(data_standardized_df),
                               columns=['Componente {}'.format(i) for i in
                                         range(1, fit.n_components_+1)],
                               index=data_standardized_df.index)
data_penguins_z_cp = pd.concat([data_standardized_df, resultados_pca], axis=1)

print(data_penguins_z_cp.head(1))

    bill_length_mm   bill_depth_mm   flipper_length_mm   body_mass_g \
0      -0.896042        0.780732         -1.426752       -0.568475

    Componente 1   Componente 2
0      -1.853593        0.032069
```

Inciso A

Analicemos cuál es el cálculo matemático que se hace para asignar los valores a los CP (Componentes Principales) para la primera observación. (El mismo procedimiento se debe hacer para todas las demás).

Cada componente principal se calcula usando los autovectores de cada uno, así como los datos de la observación, quedando así la fórmula para los autovectores generados en el anteúltimo código:

$$\begin{aligned} CP1 &= 0.453753 * \text{bill_length_mm_z} - 0.399047 * \text{bill_depth_mm_z} + 0.576825 * \\ &\quad \text{flipper_length_mm_z} + 0.549675 * \text{body_mass_g_z} \\ CP2 &= 0.600195 * \text{bill_length_mm_z} + 0.796170 * \text{bill_depth_mm_z} + 0.005788 * \\ &\quad \text{flipper_length_mm_z} + 0.076464 * \text{body_mass_g_z} \end{aligned}$$

En el caso de la primera observación (correspondiente a la última salida de código), el cálculo quedaría de la siguiente manera:

$$\begin{aligned} CP1 &= 0.453753 * (-0.896042) - 0.399047 * 0.780732 + 0.576825 * (-1.426752) + 0.549675 * \\ &\quad (-0.568475) \\ CP1 &= -1.85359322606 \end{aligned}$$

$$CP2 = 0.600195 * (-0.896042) + 0.796170 * 0.780732 + 0.005788 * (-1.426752) + 0.076464 * (-0.568475)$$

$$CP2 = 0.03206955527$$

Como se puede ver, los CPs calculados coinciden con los valores calculados por el código que se muestran en la última salida.

Inciso B e Inciso D

Vamos a analizar ahora cuando fuerte o débil es la relación entre los componentes principales seleccionados con cada una de las variables originales

```
In [46]: variables_cp = data_penguins_z_cp.columns
n_variables = fit.n_features_in_
covarianzas_var_comp = data_penguins_z_cp.cov()
covarianzas_var_comp = covarianzas_var_comp.iloc[:,fit.n_features_in_, fit.n_features_in_]
correlaciones = data_penguins_z_cp.corr()
correlaciones = correlaciones.iloc[:,fit.n_features_in_, fit.n_features_in_ :]

print(correlaciones)
```

	Componente 1	Componente 2
bill_length_mm	0.751829	0.529438
bill_depth_mm	-0.661186	0.702309
flipper_length_mm	0.955748	0.005106
body_mass_g	0.910762	0.067449

Es curioso ver cómo flipper_length_mm y body_mass_g están altamente relacionadas positivamente con el componente 1, pero no tienen casi injerencia en el componente 2. En el caso de bill_length_mm y bill_depth_mm, es distinto, ya que ambas tienen una correlación mixta con ambos componentes, teniendo bill_depth_mm una correlación negativa con el componente 1.

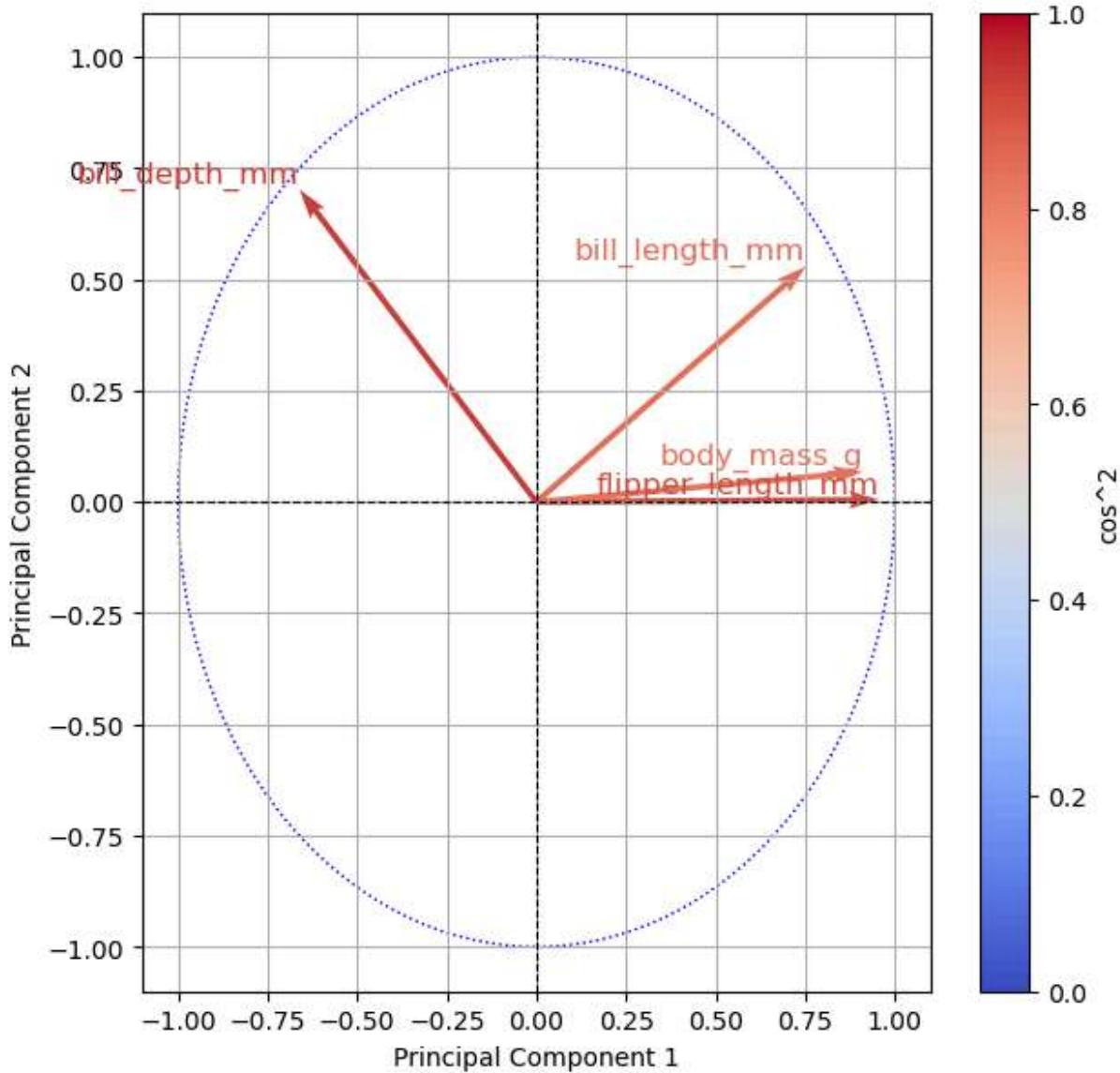
Vamos a hacer un gráfico para poder visualizar mejor la correlación de los CP con las variables originales.

```
In [47]: def plot_corr_cos(correlations_data_with_cp):
    cmap = plt.get_cmap('coolwarm')
    _, ax = plt.subplots(figsize=(7, 7)) # Tamaño reducido
    i = 0
    j = 1
    sum_cos2 = correlations_data_with_cp.iloc[:, i] ** 2 + correlations_data_with_cp.iloc[:, j] ** 2
    circle = plt.Circle((0, 0), 1, fill=False, color='b', linestyle='dotted')
    ax.add_patch(circle)
    for k, var_name in enumerate(correlations_data_with_cp.index):
        x = correlations_data_with_cp.iloc[k, i]
        y = correlations_data_with_cp.iloc[k, j]
        color = cmap(sum_cos2.iloc[k])
        ax.quiver(0, 0, x, y, angles='xy', scale_units='xy', scale=1, color=color)
        ax.text(x, y, var_name, color=color, fontsize=12, ha='right', va='bottom')
        ax.axhline(0, color='black', linestyle='--', linewidth=0.8)
        ax.axvline(0, color='black', linestyle='--', linewidth=0.8)
    ax.set_xlabel(f'Principal Component {i + 1}')
```

```

ax.set_ylabel(f'Principal Component {j + 1}')
ax.set_xlim(-1.1, 1.1)
ax.set_ylim(-1.1, 1.1)
sm = plt.cm.ScalarMappable(cmap=cmap)
sm.set_array([])
plt.colorbar(sm, ax=ax, orientation='vertical', label='cos^2')
plt.grid()
plt.show()
plot_corr_cos(correlaciones)

```



El incremento tanto de la longitud del pico como de la masa corporal, y principalmente de la longitud de las aletas, están relacionados con el incremento del valor del CP1. Por otro lado, el incremento de la profundidad del pico está más bien relacionado con el decremento del CP1.

Para el caso del CP2, el incremento de la profundidad del pico está altamente relacionado con el incremento del valor del CP2. En menor medida, sucede lo mismo con el aumento de la longitud del pico. Con respecto a la masa corporal y la longitud de las aletas, no parecen tener un efecto significativo en el valor del CP2.

Inciso C

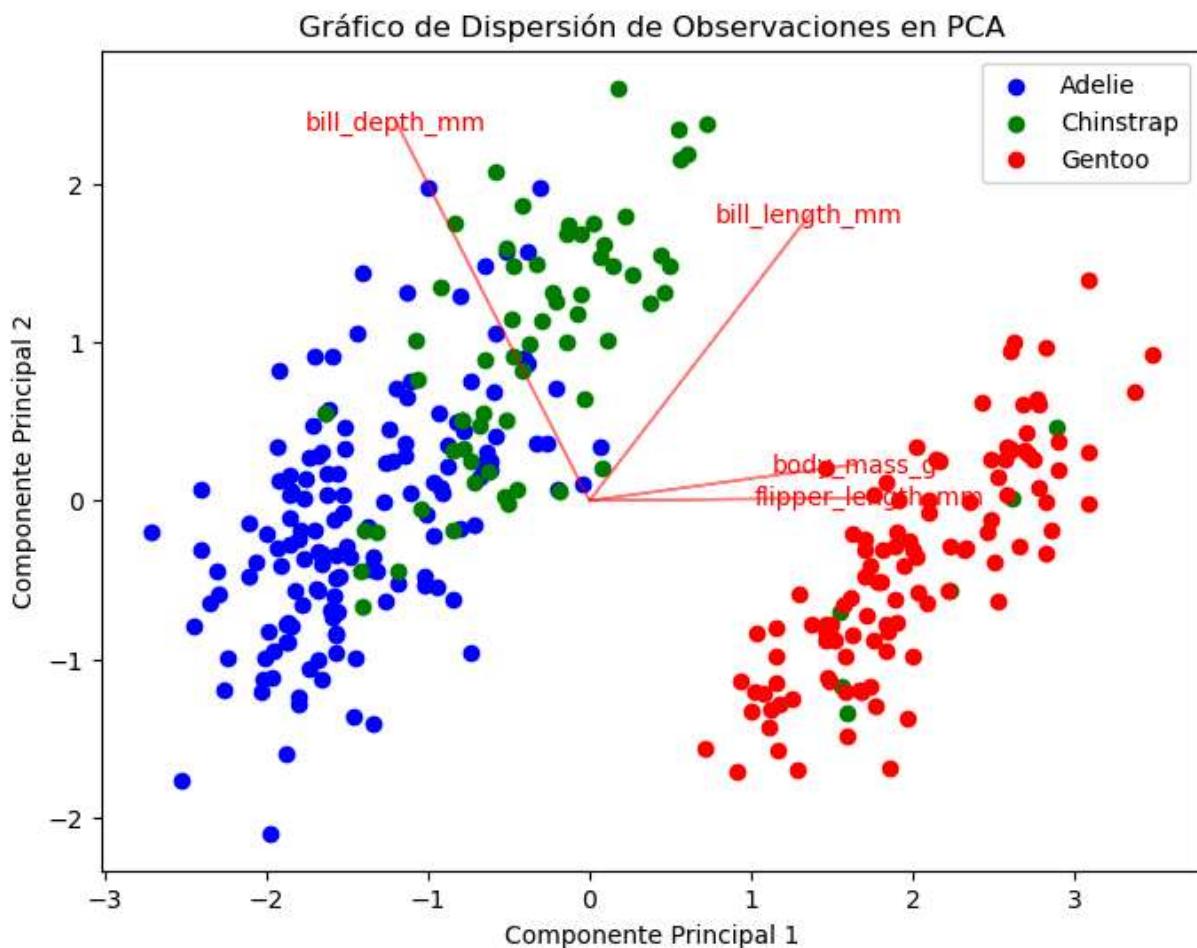
```
In [48]: def plot_pca_scatter(pca, datos_estandarizados_df, n_components):
    componentes_principales = pca.transform(datos_estandarizados_df)
    penguins = sns.load_dataset('penguins')
    penguins = penguins.dropna()
    especies = penguins['species'].loc[datos_estandarizados_df.index]

    for i in range(n_components):
        for j in range(i + 1, n_components):
            plt.figure(figsize=(8, 6))
            colores = {'Adelie': 'blue', 'Chinstrap': 'green', 'Gentoo': 'red'}
            for especie in especies.unique():
                indices = datos_estandarizados_df.index[especies == especie]
                indices = indices[indices < componentes_principales.shape[0]]
                plt.scatter(componentes_principales[indices, i], componentes_principales[indices, j],
                            color=colores[especie], label=especie)

    # Añadir vectores de variables originales
    coeff = np.transpose(pca.components_)
    scaled_coeff = 3 * coeff
    for var_idx in range(scaled_coeff.shape[0]):
        plt.arrow(0, 0, scaled_coeff[var_idx, i], scaled_coeff[var_idx, j],
                  plt.text(scaled_coeff[var_idx, i], scaled_coeff[var_idx, j], datos.columns[i]))

    plt.xlabel(f'Componente Principal {i + 1}')
    plt.ylabel(f'Componente Principal {j + 1}')
    plt.title('Gráfico de Dispersion de Observaciones en PCA')
    plt.legend()
    plt.show()

plot_pca_scatter(pca, data_standardized_df, pca.n_components)
```



En el gráfico podemos visualizar con claridad que hay dos grupos bien diferenciados (Adelie y Chinstrap por un lado, y Gentoo por el otro), donde Gentoo se caracteriza por tener una masa corporal más grande y la longitud de las aletas también superior (podemos inferir que la especie Gentoo está conformada por ejemplares de mayor tamaño). Las otras dos razas están algo más entremezcladas al ser algo más parecidas entre sí que con la especie Gentoo. Podemos inferir que, si bien ambas especies tienen ejemplares de menor tamaño que la Gentoo, la Chinstrap, por lo general, destaca por tener picos más grandes, tanto en longitud como en profundidad, que la especie Adelie.

Hay un par de observaciones para la especie Chinstrap que se ubican en el agrupamiento de la Gentoo. Esto es algo extraño y valdría la pena analizarlas para ver si deberían ser tratadas como outliers o si es que hubo un error en la recolección de datos. También se puede deber a la pérdida de variabilidad explicada cuando reducimos la dimensionalidad del conjunto de datos de 4 a 2 componentes principales.

Clustering

1 - Calcular la Matriz de Distancias

Calculemos una matriz que represente la distancia entre cada una de las observaciones usando la distancia euclidiana. Para facilitar la visualización, generamos una pequeña matriz también con las primeras 5 observaciones y sus distancias redondeadas. Como input usaremos nuestro conjunto de datos sin nulos y con las variables estandarizadas."

```
In [49]: distance_matrix = distance.cdist(data_standardized_df, data_standardized_df, 'euclidean')
distance_small = distance_matrix[:5, :5]
distance_small = pd.DataFrame(distance_small, index=data_standardized_df.index[:5],
                               columns=data_standardized_df.columns[:5])
distance_small_rounded = distance_small.round(2)
print("Matriz de Distancias Euclidianas (Primeras 5 filas/columnas):")
print(distance_small_rounded.to_string(index_names=False, header=True))
```

Matriz de Distancias Euclidianas (Primeras 5 filas/columnas):

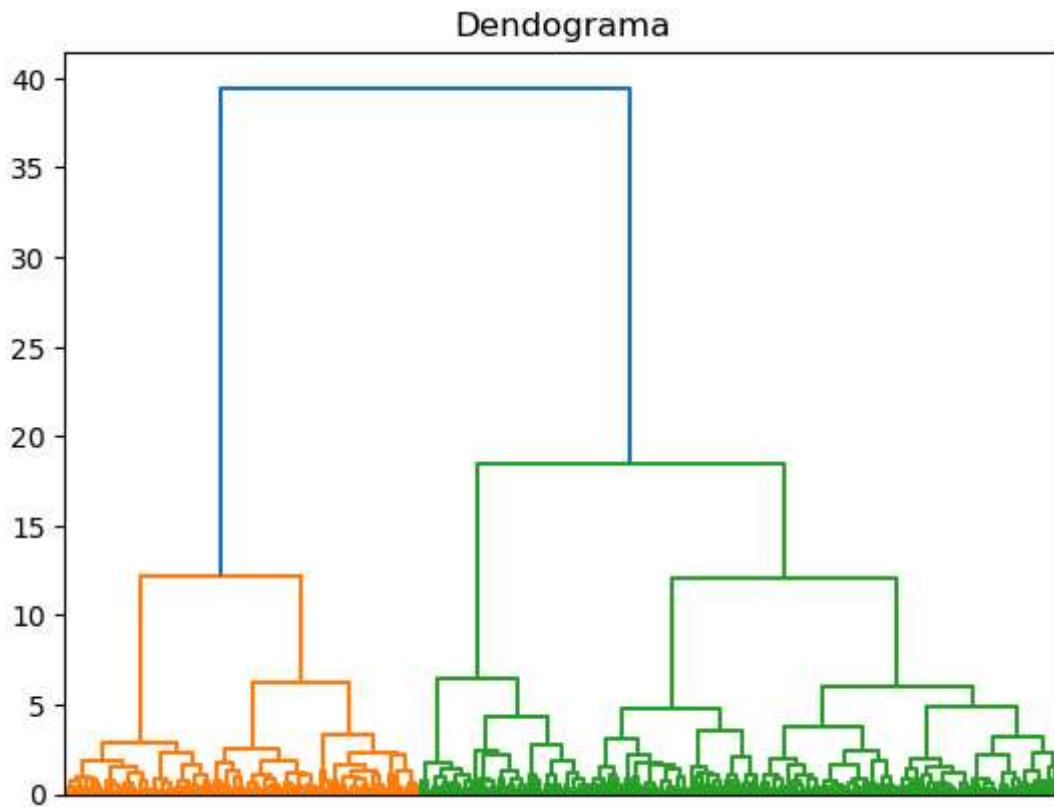
	0	1	2	4	5
0	0.00	0.76	1.25	1.08	1.17
1	0.76	0.00	1.00	1.28	1.66
2	1.25	1.00	0.00	0.98	1.47
4	1.08	1.28	0.98	0.00	0.88
5	1.17	1.66	1.47	0.88	0.00

2 - Determinación del Número de Clústeres- Análisis Jerárquico

Primero procedamos a realizar un dendograma para visualizar los clusteres que son identificados en nuestras observaciones haciendo un análisis jerárquico

```
In [50]: df_std_distance = pd.DataFrame(distance_matrix, index=data_standardized_df.index,
                                      columns=data_standardized_df.columns)

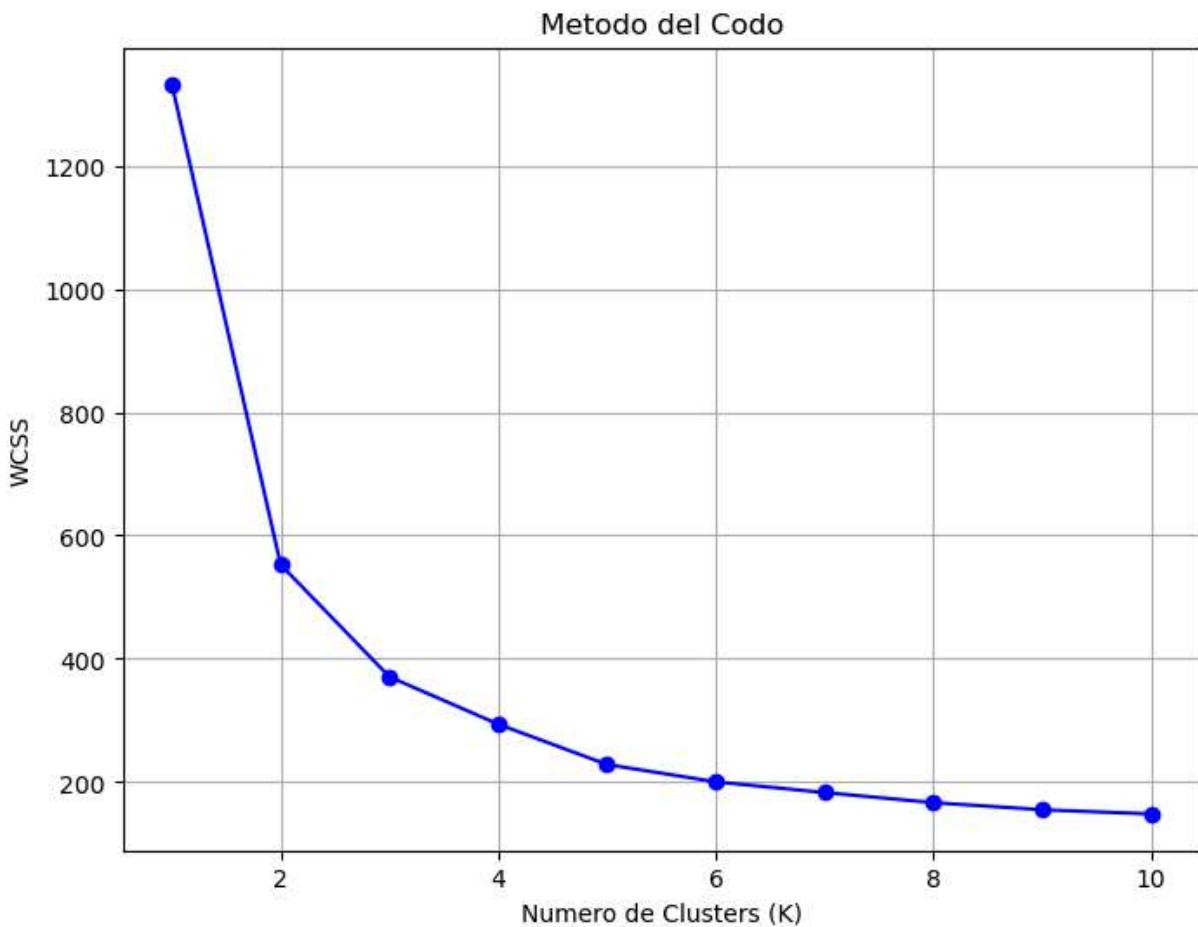
linkage_matrix = sch.linkage(df_std_distance, method='ward')
dendrogram = sch.dendrogram(linkage_matrix, leaf_rotation=90)
plt.xticks([])
plt.title('Dendograma')
plt.show()
```



El dendrograma no solo nos indica la jerarquía de los clústeres, sino también, mediante la distancia de las uniones, nos indica qué tan distantes son en términos de la distancia euclídea que existe entre las diversas agrupaciones. Frente a la decisión, esta podría estar reñida entre quedarnos con 3 clústeres o bien quedarnos con 5. Tres sería una buena opción para maximizar la distancia entre las agrupaciones, sin embargo, 5 no resta demasiada distancia y nos ofrece un mayor grado de especificidad.

Para poder decidirnos entre estos dos, vamos a hacer el método del codo, que ya usamos con PCA, utilizando como medida de rendimiento el WCSS, el cual mide la variabilidad dentro de cada grupo.

```
In [51]: wcss = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=0, n_init=10)
    kmeans.fit(data_standardized_df)
    wcss.append(kmeans.inertia_)
plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), wcss, marker='o', linestyle='-', color='b')
plt.title('Metodo del Codo')
plt.xlabel('Numero de Clusters (K)')
plt.ylabel('WCSS')
plt.grid(True)
plt.show()
```



Como observamos en la gráfica, tener dos clústeres es donde la variabilidad entre los dos grupos es mayor. Si aumentamos a 4, la variabilidad se reduce bastante y, a partir de ahí, no se reduce mucho más, por lo que el punto que representa nuestro punto del codo sería 4 clústeres para tener un equilibrio entre clústeres representativos, pero con una variabilidad alta (mientras más pequeña es la variabilidad, más compacto es el grupo, siendo el caso extremo un clúster por observación).

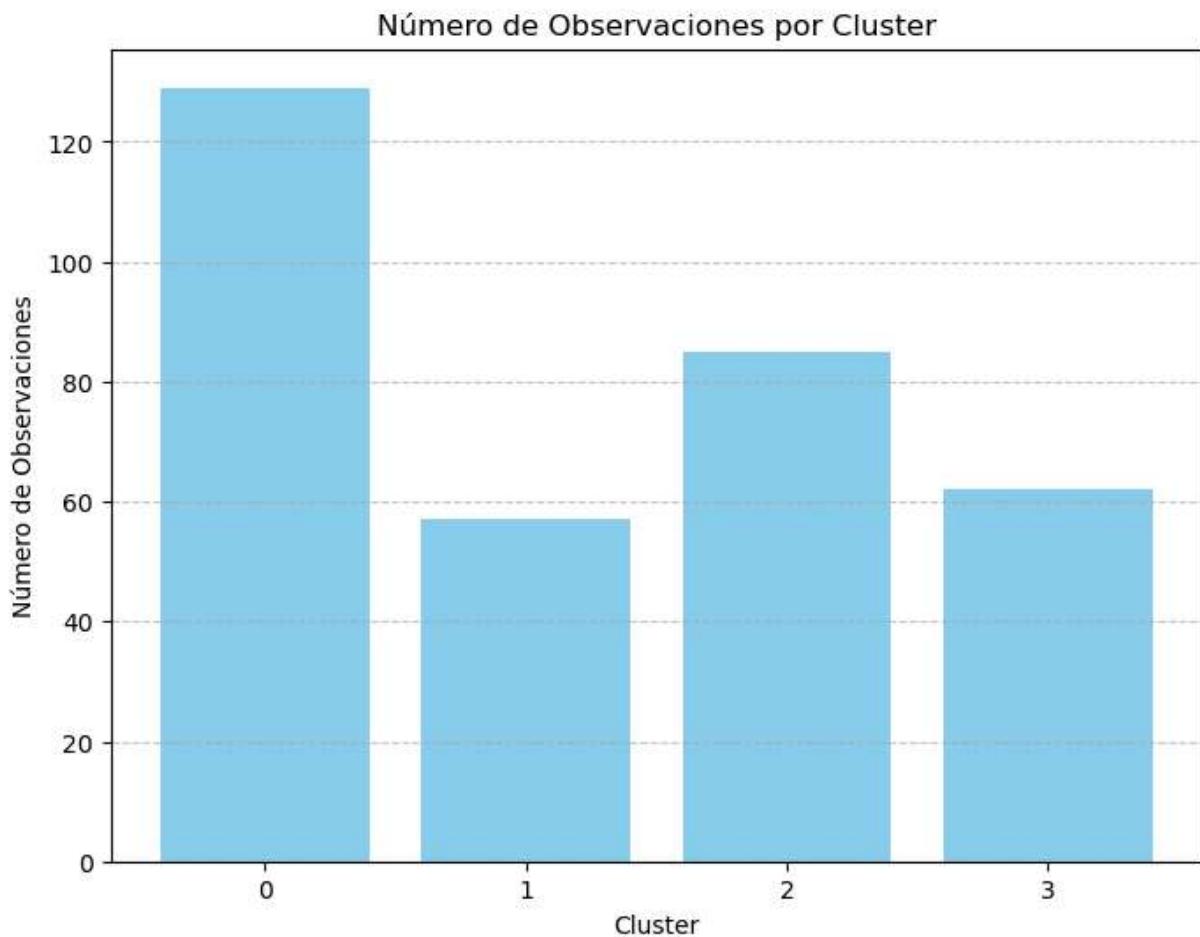
3 - Análisis de Cluster No Jerarquico

Procedemos a hacer un análisis no jerárquico de los clústeres con el método K-Medias, siendo nuestra $k = 4$ (número de clústeres). A cada observación le asignamos el clúster al que pertenece en la columna 'Clúster4'. También hacemos un gráfico para que se pueda visualizar el número de observaciones que tiene cada clúster.

```
In [52]: k = 4
kmeans = KMeans(n_clusters=k, random_state=0, n_init=10)
kmeans.fit(data_standardized_df)
kmeans_cluster_labels = kmeans.labels_
data_standardized_df['Cluster4'] = kmeans_cluster_labels

cluster_counts = data_standardized_df['Cluster4'].value_counts().sort_index()
plt.figure(figsize=(8, 6))
```

```
plt.bar(cluster_counts.index, cluster_counts.values, color='skyblue')
plt.title('Número de Observaciones por Cluster')
plt.xlabel('Cluster')
plt.ylabel('Número de Observaciones')
plt.xticks(range(k))
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



4 - Evaluacion de la calidad de las Agrupaciones

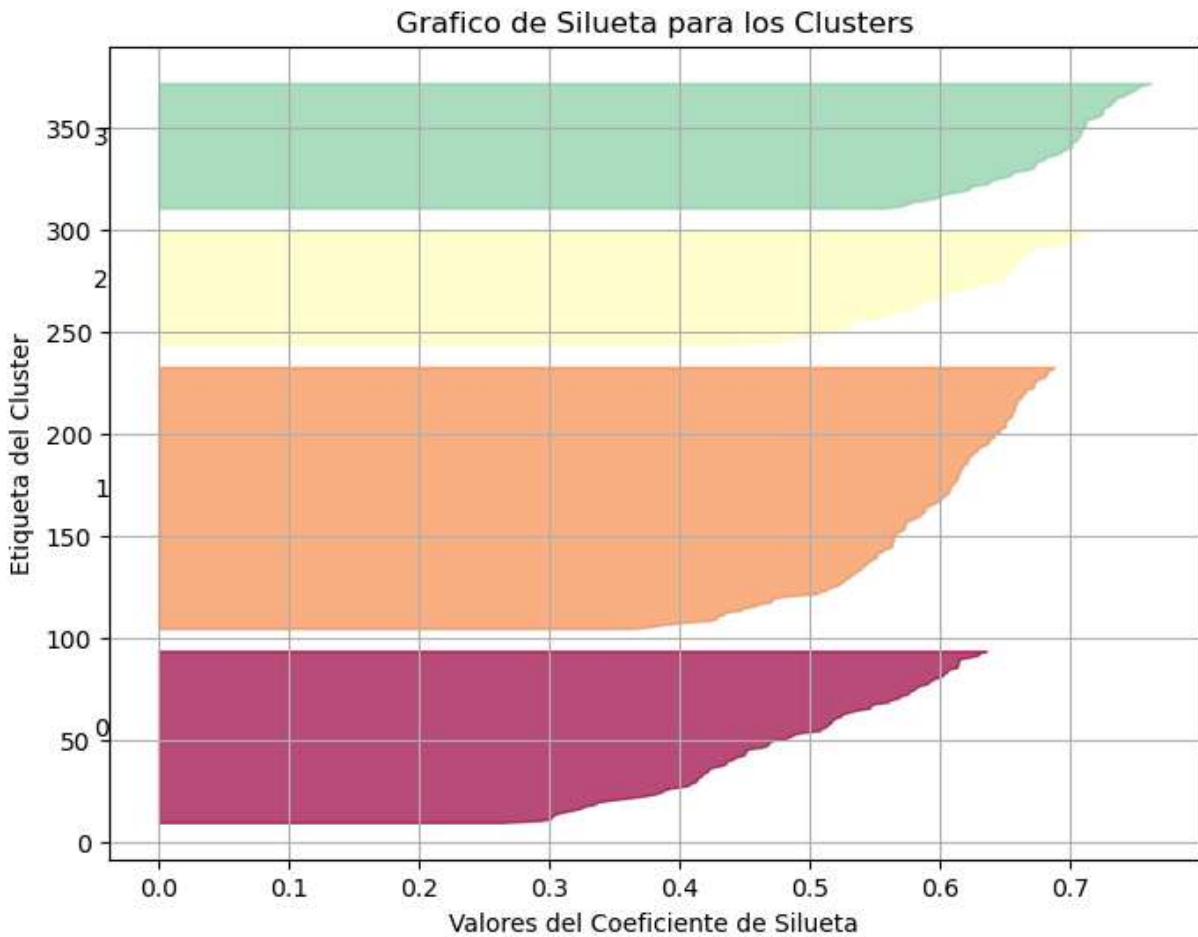
Procedemos a hacer un grafico que represente la calidad de cada una de las observaciones en terminos de que tan bien agrupada esta y que tan lejos esta de otros clusters. Esta tecnica se llama la tecnica de la silueta y nos sirve para medir la cohesion de los clusters observacion por observacion.

```
In [53]: kmeans = KMeans(n_clusters=4, random_state=0)
kmeans.fit(data_standardized_df)
labels = kmeans.labels_
silhouette_values = silhouette_samples(data_standardized_df, labels)
plt.figure(figsize=(8, 6))
y_lower = 10
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
color = plt.cm.get_cmap("Spectral")(float(i) / 4)
plt.fill_betweenx(np.arange(y_lower, y_upper),
                 0, ith_cluster_silhouette_values,
                 facecolor=color, 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("Grafico de Silueta para los Clusters")
plt.xlabel("Valores del Coeficiente de Silueta")
plt.ylabel("Etiqueta del Cluster")
plt.grid(True)
plt.show()

```



El gráfico nos muestra que los grupos están excelentemente agrupados en todos los casos. Las observaciones que peor agrupadas están superan con creces el 0.4 y, en todos los casos, las agrupaciones mejor agrupadas son superiores a 0.7. Cabe destacar que para que una observación se encuentre mal agrupada, debe ser menor a 0.

5 - Variables supplementarias

Añadimos una nueva variable a nuestro análisis: las islas. Las observaciones están etiquetadas por la isla en la que fueron registradas, por lo que procedemos a representar los centroides de cada una de las especies, así como los centroides de los grupos correspondientes a las observaciones registradas en cada una de las islas.

```
In [54]: data_standarized_df['species'] = penguins['species']
data_standarized_df['island'] = penguins['island']

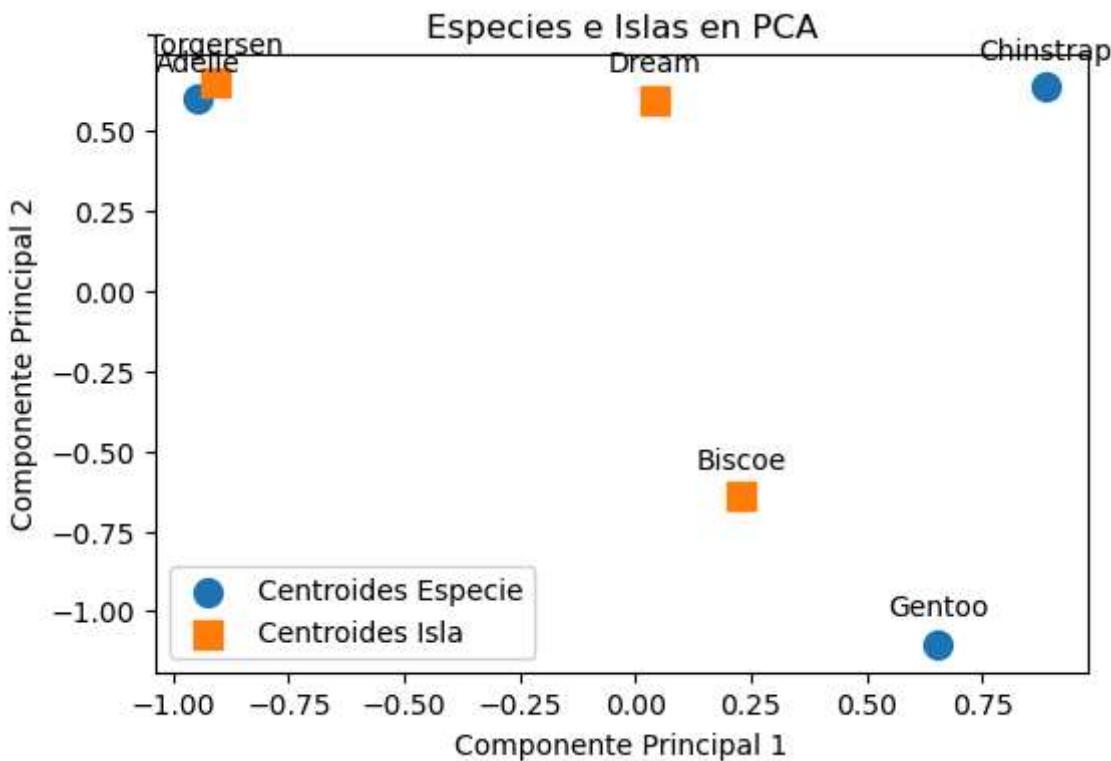
centroides_especie = data_standarized_df.groupby('species').mean()
centroides_isla = data_standarized_df.groupby('island').mean()

plt.figure(figsize=(6, 4))

plt.scatter(centroides_especie.iloc[:, 0], centroides_especie.iloc[:, 1], marker='o')
for especie, centroide in centroides_especie.iterrows():
    plt.annotate(especie, (centroide.iloc[0], centroide.iloc[1]), textcoords="offset points", xytext=(5, 10), color='black')

plt.scatter(centroides_isla.iloc[:, 0], centroides_isla.iloc[:, 1], marker='s', s=100)
for isla, centroide in centroides_isla.iterrows():
    plt.annotate(isla, (centroide.iloc[0], centroide.iloc[1]), textcoords="offset points", xytext=(-15, -15), color='black')

plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.title('Especies e Islas en PCA')
plt.legend()
plt.show()
```



Como podemos observar, los centroides de las islas tienen cierta correspondencia con cada una de las especies. Donde esto es más evidente es en el caso de las islas Torgersen y la especie de pingüinos Adelie (los pequeños de pico corto y poco profundo). La especie con pingüinos de mayor porte se encuentra próxima al centroide de la isla Biscoe. La isla Dream se encuentra a medio camino entre la especie Chinstrap y Adelie, eso es interesante porque ambas especies eran las que más se parecían. Quizás en dicha isla convivan ambas especies

de manera tal que haya algún entrecruzamiento entre las mismas, reduciendo las distancias genéticas.

6 - Caracterización de los Clústeres

```
In [55]: def estadisticos_descriptivos_por_cluster(data_clustered, variables):
    for cluster in sorted(data_clustered['Cluster4'].unique()):
        print(f"\nEstadísticos Descriptivos para Cluster {cluster+1}:\n")
        cluster_data = data_clustered[data_clustered['Cluster4'] == cluster][variables]
        estadisticos = cluster_data.agg(['mean', 'std', 'min', 'max']).T.round(3)
        print(estadisticos)

# Añadir columna 'Cluster4' a Los datos originales
penguins_numeric_data['Cluster4'] = kmeans_cluster_labels
variables = numeric_cols
estadisticos_descriptivos_por_cluster(penguins_numeric_data, variables)
```

Estadísticos Descriptivos para Cluster 1:

	mean	std	min	max
bill_length_mm	38.277	2.259	32.1	43.2
bill_depth_mm	18.122	1.164	15.5	21.2
flipper_length_mm	188.628	5.721	172.0	202.0
body_mass_g	3593.798	407.486	2850.0	4675.0

Estadísticos Descriptivos para Cluster 2:

	mean	std	min	max
bill_length_mm	49.793	2.588	44.4	59.6
bill_depth_mm	15.739	0.761	14.1	17.3
flipper_length_mm	221.912	5.658	208.0	231.0
body_mass_g	5519.737	291.193	4925.0	6300.0

Estadísticos Descriptivos para Cluster 3:

	mean	std	min	max
bill_length_mm	47.662	3.868	40.8	58.0
bill_depth_mm	18.748	1.137	16.4	21.5
flipper_length_mm	196.918	6.346	178.0	212.0
body_mass_g	3898.235	414.413	2700.0	4800.0

Estadísticos Descriptivos para Cluster 4:

	mean	std	min	max
bill_length_mm	45.523	1.907	40.9	49.6
bill_depth_mm	14.315	0.602	13.1	15.8
flipper_length_mm	212.935	3.908	203.0	222.0
body_mass_g	4699.597	284.882	3950.0	5200.0

Los estadísticos descriptivos nos ofrecen una buena mirada rápida de los aspectos físicos representativos de las agrupaciones seleccionadas.

- Clúster 1: pingüinos más pequeños, con una media apenas superior a los 3.5 kg y registrando los picos más cortos, así como las aletas más cortas de todos los clústeres.
- Clúster 2: pingüinos más grandes en términos de peso, con una media de 5.5 kg y con las aletas más grandes, superiores a los 20 cm, y los picos más largos (casi 5 cm), aunque no los más anchos.
- Clúster 3: pingüinos apenas más grandes que el clúster 1 (3.9 kg aprox.), pero con unos picos casi tan largos como los del clúster 2 (casi 4.8 cm) y algo más anchos que el clúster 2, por lo que podemos decir que se caracterizan por picos grandes y tamaño pequeño.
- Clúster 4: segundo grupo más grande en términos de tamaño, con una media superior a 4.5 kg de masa. La longitud de las aletas también es apenas más chica que la del clúster 1. Los picos son relativamente largos y los más angostos de entre los clústeres. Podríamos decir que este grupo se destaca por pingüinos comparativamente con picos largos y finos y relativamente grandes en términos de masa corporal.