Arctic Penguin Exploration: Unraveling Clusters in the Icy Domain with K-means clustering

Alt text source: @allison_horst https://github.com/allisonhorst/penguins

You have been asked to support a team of researchers who have been collecting data about penguins in Antartica!

Origin of this data : Data were collected and made available by Dr. Kristen Gorman and the Palmer Station, Antarctica LTER, a member of the Long Term Ecological Research Network.

The dataset consists of 5 columns.

- culmen_length_mm: culmen length (mm)
- culmen_depth_mm: culmen depth (mm)
- flipper_length_mm: flipper length (mm)
- body_mass_g: body mass (g)
- sex: penguin sex

Unfortunately, they have not been able to record the species of penguin, but they know that there are three species that are native to the region: **Adelie**, **Chinstrap**, and **Gentoo**, so your task is to apply your data science skills to help them identify groups in the dataset!

In [ ]:
```
# Import Required Packages
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Loading and examining the dataset
penguins_df = pd.read_csv("data/penguins.csv")
```

In [ ]:
```
# view data head
penguins_df.head(10)
```

Out[ ]:

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	39.1	18.7	181.0	3750.0	MALE
1	39.5	17.4	186.0	3800.0	FEMALE
2	40.3	18.0	195.0	3250.0	FEMALE
3	NaN	NaN	NaN	NaN	NaN
4	36.7	19.3	193.0	3450.0	FEMALE
5	39.3	20.6	190.0	3650.0	MALE
6	38.9	17.8	181.0	3625.0	FEMALE
7	39.2	19.6	195.0	4675.0	MALE
8	34.1	18.1	193.0	3475.0	NaN
9	42.0	20.2	5000.0	4250.0	MALE

In [ ]:

```
penguins_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   culmen_length_mm   342 non-null    float64
 1   culmen_depth_mm    342 non-null    float64
 2   flipper_length_mm  342 non-null    float64
 3   body_mass_g        342 non-null    float64
 4   sex                335 non-null    object 
dtypes: float64(4), object(1)
memory usage: 13.6+ KB
```

Clean data by removing null values and outliers. Check for missing values, then use boxplot to detect outliers

In [ ]:

```
# Check for missing value
print(penguins_df.isna().sum())

# remove na
penguin_drop = penguins_df.dropna()

print("\n\nThe dataframe after dropping missing values\n")
# check new df
print(penguin_drop.isna().sum())
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
culmen_length_mm     2
culmen_depth_mm      2
flipper_length_mm    2
body_mass_g          2
sex                  9
dtype: int64


The dataframe after dropping missing values

culmen_length_mm     0
culmen_depth_mm      0
flipper_length_mm    0
body_mass_g          0
sex                  0
dtype: int64
```

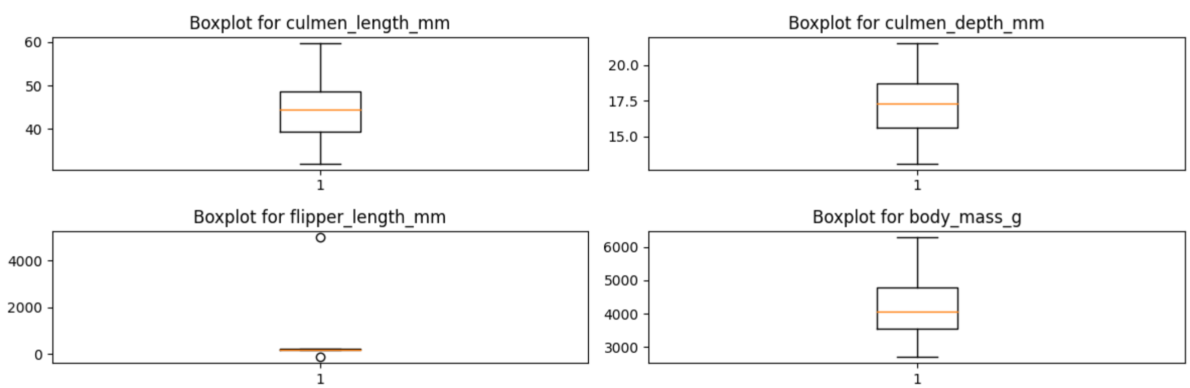
In [ ]:

```
# create a subplots
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12,4))

# loop through the columns
for i, col in enumerate(penguin_drop.columns[:4]):
    axes[i // 2, i % 2].boxplot(penguin_drop[col])
    axes[i // 2, i % 2].set_title(f"Boxplot for {col}")

# Use the tight layout
plt.tight_layout()

# show plot
plt.show()
```



It appears that the Filppers length column has two outliers

In [ ]:

```
# remove the outliers in the Flippers length column
# Identify outliers in flipper_length_mm
column_name = 'flipper_length_mm'
Q1 = penguin_drop[column_name].quantile(0.25)
Q3 = penguin_drop[column_name].quantile(0.75)
IQR = Q3 - Q1

# Define the outlier bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
# Identify rows with outliers
outliers = (penguin_drop[column_name] < lower_bound) | (penguin_drop[column_

# Remove rows with outliers
penguins_clean = penguin_drop[~outliers]

print(lower_bound, upper_bound)
print(len(outliers))
print(sum(outliers))
print(len(penguins_clean))
```

```
155.5 247.5
335
2
333
```

In [ ]:
```
penguins_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 333 entries, 0 to 343
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   culmen_length_mm   333 non-null    float64
 1   culmen_depth_mm    333 non-null    float64
 2   flipper_length_mm  333 non-null    float64
 3   body_mass_g        333 non-null    float64
 4   sex                333 non-null    object
dtypes: float64(4), object(1)
memory usage: 15.6+ KB
```

Preprocess data using standard scaler and one-hot encoding to add dummy variables to the categorical column.

In [ ]:
```
# encode the sex column
penguin_encoded = pd.get_dummies(penguins_clean, columns=["sex"])

# remove the sex column
penguin_encoded.drop("sex_.", axis=1, inplace=True)

# check the data head
penguin_encoded.head()
```

Out[ ]:

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex_FEMALE	sex_
0	39.1	18.7	181.0	3750.0	0	
1	39.5	17.4	186.0	3800.0	1	
2	40.3	18.0	195.0	3250.0	1	
4	36.7	19.3	193.0	3450.0	1	
5	39.3	20.6	190.0	3650.0	0	

In [ ]:
```
# instatiate a standard scaler object
```

```python
penguins_preprocessed = pd.DataFrame(scaler.fit_transform(penguin_encoded))

penguins_preprocessed.columns = penguin_encoded.columns

penguins_preprocessed.head()
```

Out[ ]:

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex_FEMALE	sex_I
0	-0.905520	0.793126	-1.428125	-0.569709	-0.991031	0.9
1	-0.831938	0.128503	-1.071522	-0.507579	1.009050	-1.0
2	-0.684775	0.435252	-0.429637	-1.191006	1.009050	-1.0
3	-1.347011	1.099875	-0.572278	-0.942487	1.009050	-1.0
4	-0.868729	1.764498	-0.786240	-0.693968	-0.991031	0.9

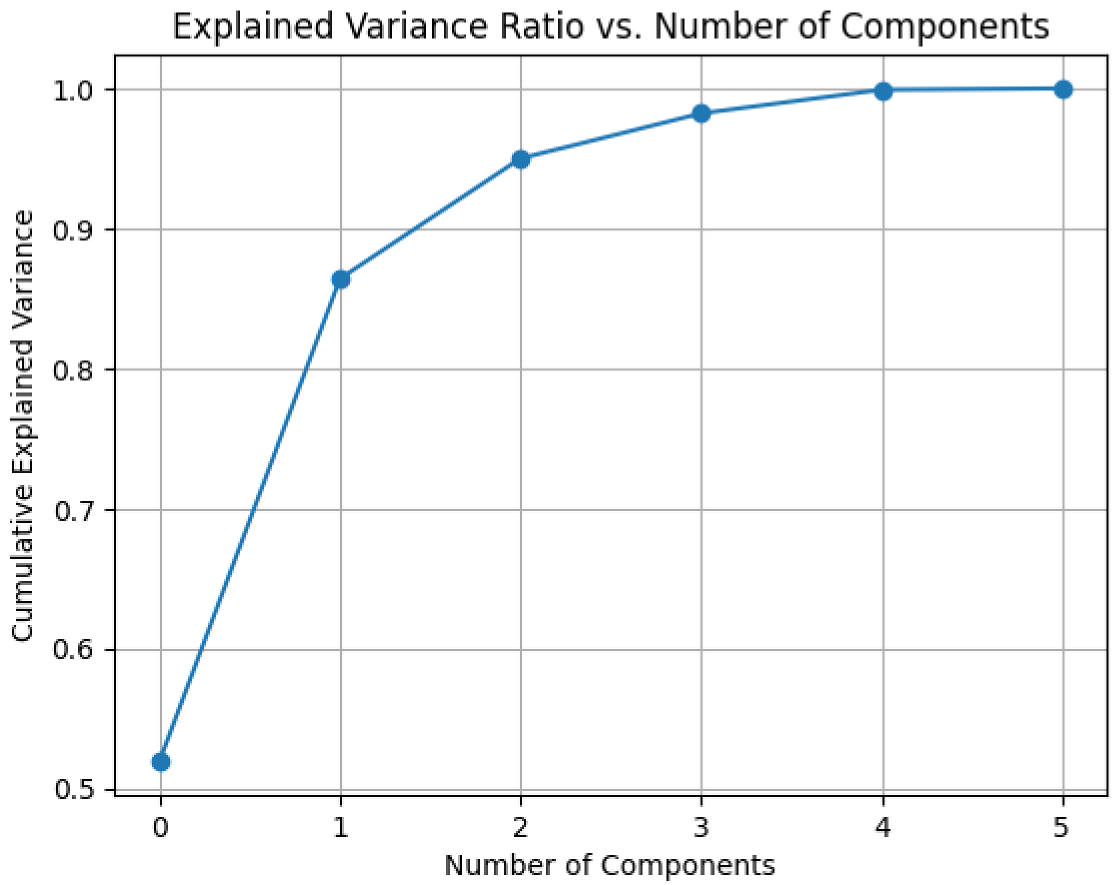
In [ ]:

```python
# Initiate a PCA object
pca = PCA(n_components=None)

# fit data to pca
pca.fit(penguins_preprocessed)
# Plotting the explained variance ratio
explained_variance_ratio = pca.explained_variance_ratio_
n_components=sum(dfx_pca.explained_variance_ratio_>0.1)

plt.plot(cumulative_explained_variance, marker='o')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('Explained Variance Ratio vs. Number of Components')
plt.grid(True)
plt.show()
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



In [ ]:

```
penguins_PCA = PCA(n_components=n_components)

penguins_PCA = pca.fit_transform(penguins_preprocessed)
```

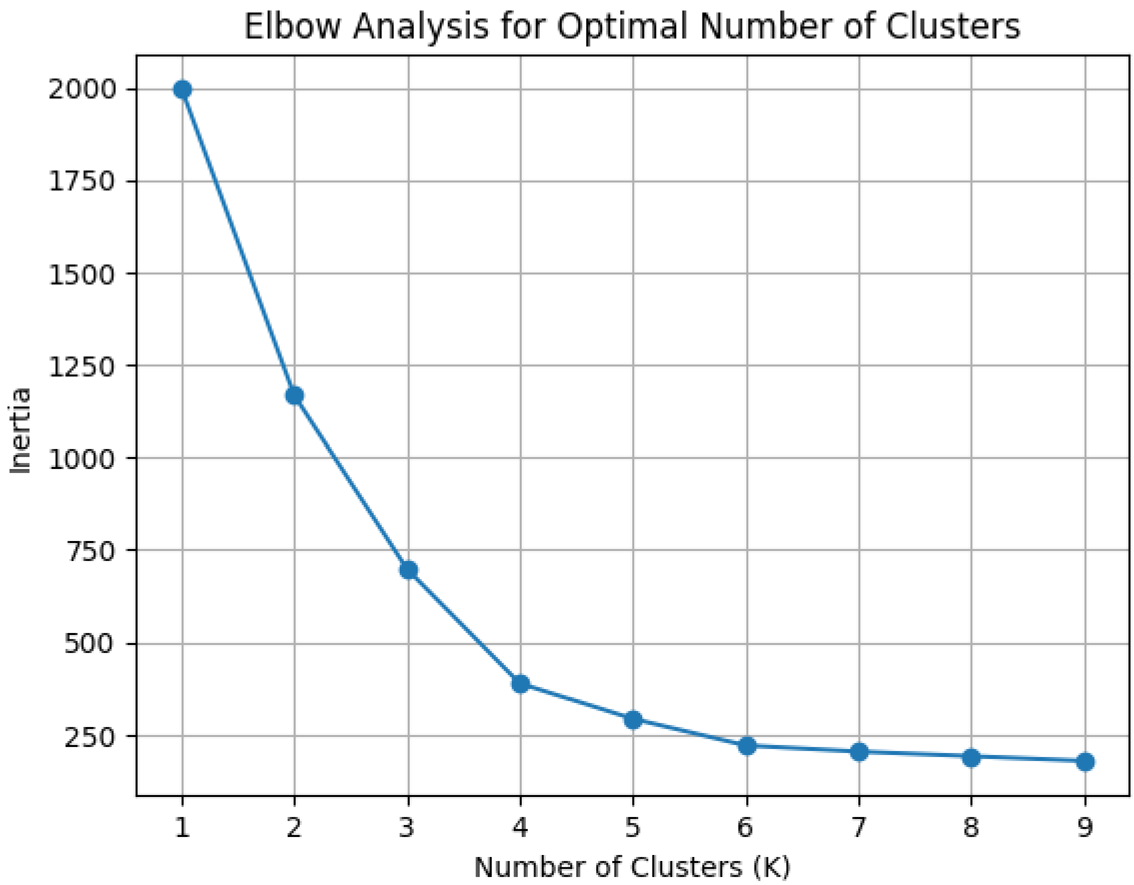
In [ ]:

```
# Perform Elbow analysis
inertia = []

for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, random_state=42).fit(penguins_PCA)
    inertia.append(kmeans.inertia_)

# Visualize the inertia values
plt.plot(range(1, 10), inertia, marker='o')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia')
plt.title('Elbow Analysis for Optimal Number of Clusters')
plt.grid(True)
plt.show()

n_cluster = 4
```



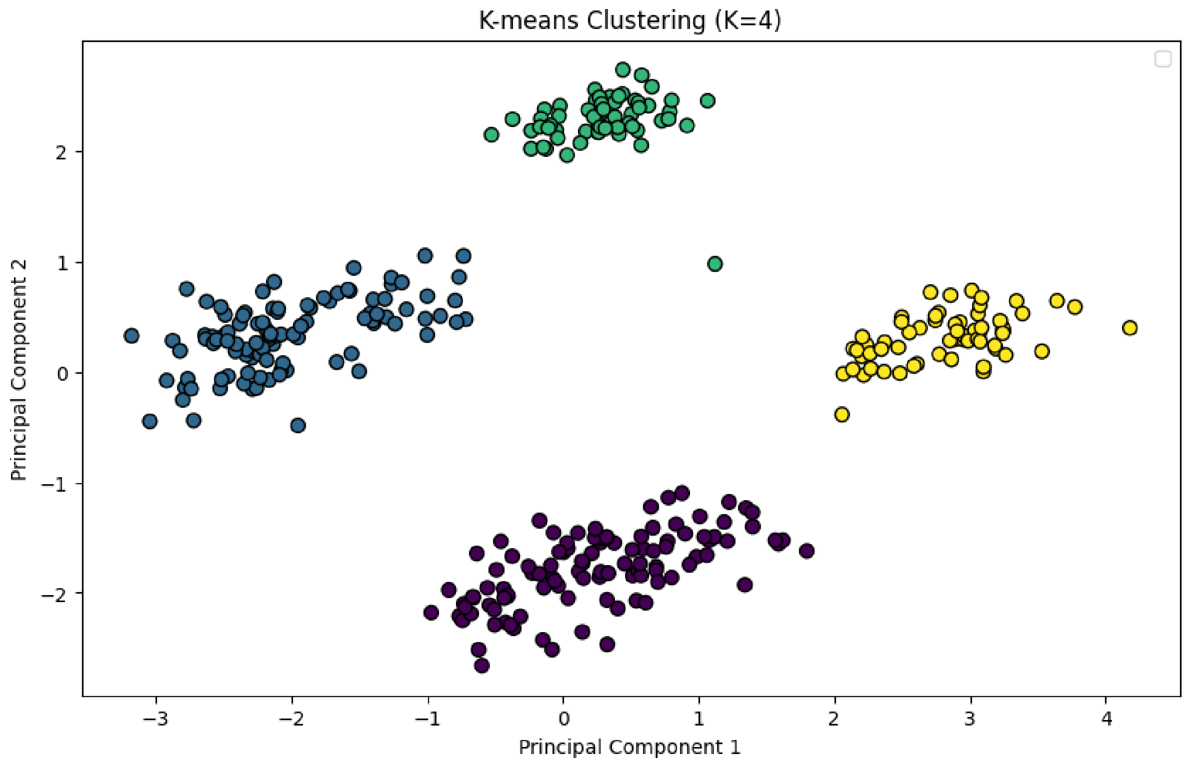
In [ ]:
```
kmeans = KMeans(n_clusters=n_cluster, random_state=42)
kmeans.fit(penguins_preprocessed)
```

Out[ ]:
```
▼ KMeans
KMeans(n_clusters=4, random_state=42)
```

In [ ]:
```
# Visualize the clusters using scatter plot
plt.figure(figsize=(10, 6))

kmeans = KMeans(n_clusters=n_clusters, random_state=42).fit(penguins_PCA)

plt.scatter( penguins_PCA[:, 0],  penguins_PCA[:, 1], c=kmeans.labels_, cmap
plt.title(f'K-means Clustering (K={n_clusters})')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.show()
```



In [ ]:
```
# Add a new column named 'label' to the penguins_clean dataset
penguins_clean['label'] = kmeans.labels_

# Create a list containing the names of the numeric columns of penguins_clea
numeric_columns = penguins_clean.select_dtypes(include=['float64', 'int64'])

# Create a final characteristic DataFrame
stat_penguins = penguins_clean.groupby('label')[numeric_columns].mean()

# Display the final characteristic DataFrame
print(stat_penguins)
```

```
       culmen_length_mm  culmen_depth_mm  flipper_length_mm  body_mass_g
label
0             43.878302        19.111321         194.764151  4006.603774
1             40.217757        17.611215         189.046729  3419.158879
2             45.545763        14.262712         212.779661  4683.050847
3             49.473770        15.718033         221.540984  5484.836066
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js