Build a K-means model

Step 1: Imports

In [1]:
```
# Import standard operational packages.
import numpy as np
import pandas as pd

# Important tools for modeling and evaluation.
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler

# Import visualization packages.
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```

# Save the `pandas` DataFrame in variable `penguins`.


penguins = pd.read_csv("KWeQDUwtRQunkA1MLWULVA_62058830bf9445469b3ceb29f63

```

In [3]:
```
# Review the first 10 rows.

penguins.head(n = 10)
```

Out[3]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
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
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	male
6	Adelie	Torgersen	38.9	17.8	181.0	3625.0	female
7	Adelie	Torgersen	39.2	19.6	195.0	4675.0	male
8	Adelie	Torgersen	34.1	18.1	193.0	3475.0	NaN
9	Adelie	Torgersen	42.0	20.2	190.0	4250.0	NaN

Step 2: Data exploration

After loading the dataset, the next step is to prepare the data to be suitable for clustering. This includes:

Exploring data Checking for missing values Encoding data Dropping a column Scaling the features using StandardScaler

Explore data

To cluster penguins of multiple different species, determine how many different types of penguin species are in the dataset.

In [4]:
```
# Find out how many penguin types there are.

penguins['species'].unique()
```

Out[4]:
```
array(['Adelie', 'Chinstrap', 'Gentoo'], dtype=object)
```

In [5]:
```
# Find the count of each species type.

penguins['species'].value_counts(dropna = False)
```

Out[5]:
```
Adelie       152
Gentoo       124
Chinstrap     68
Name: species, dtype: int64
```

How many types of species are present in the dataset?

There are three types of species. Note the Chinstrap species is less common than the other species. This has a chance to affect K-means clustering as K-means performs best with similar sized groupings.

Question: Why is it helpful to determine the perfect number of clusters using K-means when you already know how many penguin species the dataset contains?

For purposes of clustering, pretend you don't know that there are three different types of species. Then, you can explore whether the algorithm can discover the different species. You might even find other relationships in the data.

Check for missing values¶

Check for missing values

An assumption of K-means is that there are no missing values. Check for missing values in the rows of the data.

In [6]:
```
# Check for missing values.

penguins.isnull().sum()
```

Out[6]:
```
species              0
island               0
bill_length_mm       2
bill_depth_mm        2
flipper_length_mm    2
body_mass_g          2
sex                 11
dtype: int64
```

In [7]:
```
# Drop rows with missing values.
# Save DataFrame in variable `penguins_subset`.

penguins_subset = penguins.dropna(axis=0).reset_index(drop = True)
```

In [8]:
```
# Check for missing values.

penguins_subset.isna().sum()
```

Out[8]:
```
species              0
island               0
bill_length_mm       0
bill_depth_mm        0
flipper_length_mm    0
body_mass_g          0
sex                  0
dtype: int64
```

In [9]:
```
# View first 10 rows.

penguins_subset.head(10)
```

Out[9]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
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	36.7	19.3	193.0	3450.0	female
4	Adelie	Torgersen	39.3	20.6	190.0	3650.0	male
5	Adelie	Torgersen	38.9	17.8	181.0	3625.0	female
6	Adelie	Torgersen	39.2	19.6	195.0	4675.0	male
7	Adelie	Torgersen	41.1	17.6	182.0	3200.0	female
8	Adelie	Torgersen	38.6	21.2	191.0	3800.0	male
9	Adelie	Torgersen	34.6	21.1	198.0	4400.0	male

Encode data

Some versions of the penguins dataset have values encoded in the sex column as 'Male' and 'Female' instead of 'MALE' and 'FEMALE'. The code below will make sure all values are ALL CAPS.

K-means needs numeric columns for clustering. Convert the categorical column 'sex' into numeric. There is no need to convert the 'species' column because it isn't being used as a feature in the clustering algorithm.

In [10]:
```
penguins_subset['sex'] = penguins_subset['sex'].str.upper()
```

In [11]:
```
# Convert `sex` column from categorical to numeric.

penguins_subset = pd.get_dummies(penguins_subset, drop_first = True, colum
```

Drop a column

Drop the categorical column island from the dataset. While it has value, this notebook is trying to confirm if penguins of the same species exhibit different physical characteristics based on sex. This doesn't include location.

Note that the 'species' column is not numeric. Don't drop the 'species' column for now. It could potentially be used to help understand the clusters later.

Scale the features

In [15]:
```
penguins_subset = penguins_subset.drop(['island'], axis=1)
```

Because K-means uses distance between observations as its measure of similarity, it's important to scale the data before modeling. Use a third-party tool, such as scikit-learn's StandardScaler function. StandardScaler scales each point x_i by subtracting the mean observed value for that feature and dividing by the standard deviation:

x-scaled = $(x_i - \text{mean}(X)) / \sigma$

This ensures that all variables have a mean of 0 and variance/standard deviation of 1.

Note: Because the species column isn't a feature, it doesn't need to be scaled.

First, copy all the features except the 'species' column to a DataFrame X.

In [16]:
```
#Exclude `species` variable from X
X = penguins_subset.drop(['species'], axis=1)
```

Scale the features in X using StandardScaler, and assign the scaled data to a new variable X_scaled.

In [17]:
```
#Scale the features.
#Assign the scaled data to variable `X_scaled`.

X_scaled = StandardScaler().fit_transform(X)
```

Step 3: Data modeling

Now, fit K-means and evaluate inertia for different values of k. Because you may not know how many clusters exist in the data, start by fitting K-means and examining the inertia values for different values of k. To do this, write a function called kmeans_inertia that takes in num_clusters and x_vals (X_scaled) and returns a list of each k-value's inertia.

When using K-means inside the function, set the random_state to 42. This way, others can reproduce your results.

In [ ]:
```
# Fit K-means and evaluate inertia for different values of k.

num_clusters = [i for i in range(2, 11)]

def kmeans_inertia(num_clusters, x_vals):
    """
    Accepts as arguments list of ints and data array.
    Fits a KMeans model where k = each value in the list of ints.
    Returns each k-value's inertia appended to a list.
    """
    inertia = []
    for num in num_clusters:
        kms = KMeans(n_clusters=num, random_state=42)
        kms.fit(x_vals)
        inertia.append(kms.inertia_)

    return inertia
```

Use the kmeans_inertia function to return a list of inertia for k=2 to 10.

In [ ]:
```
# Return a list of inertia for k=2 to 10.

inertia = kmeans_inertia(num_clusters, X_scaled)
inertia
```

Next, create a line plot that shows the relationship between num_clusters and inertia. Use either seaborn or matplotlib to visualize this relationship.

In [ ]:
```
# Create a line plot.
plot = sns.lineplot(x=num_clusters, y=inertia, marker = 'o')
plot.set_xlabel("Number of clusters");
plot.set_ylabel("Inertia");
```

Question: Where is the elbow in the plot?

The plot seems to depict an elbow at six clusters, but there isn't a clear method for confirming that a six-cluster model is optimal. Therefore, the silhouette scores should be checked.

Step 4: Results and evaluation

Now, evaluate the silhouette score using the silhouette_score() function. Silhouette scores are used to study the distance between clusters.

Then, compare the silhouette score of each value of k, from 2 through 10. To do this, write a function called kmeans_sil that takes in num_clusters and x_vals (X_scaled) and returns a list of each k-value's silhouette score.

In [ ]:
```
# Evaluate silhouette score.
# Write a function to return a list of each k-value's score.

def kmeans_sil(num_clusters, x_vals):
    """
    Accepts as arguments list of ints and data array.
    Fits a KMeans model where k = each value in the list of ints.
    Calculates a silhouette score for each k value.
    Returns each k-value's silhouette score appended to a list.
    """
    sil_score = []
    for num in num_clusters:
        kms = KMeans(n_clusters=num, random_state=42)
        kms.fit(x_vals)
        sil_score.append(silhouette_score(x_vals, kms.labels_))

    return sil_score


sil_score = kmeans_sil(num_clusters, X_scaled)
sil_score
```

Next, create a line plot that shows the relationship between num_clusters and sil_score. Use either seaborn or matplotlib to visualize this relationship.

In [ ]:
```
# Create a line plot.

plot = sns.lineplot(x=num_clusters, y=sil_score, marker = 'o')
plot.set_xlabel("# of clusters");
plot.set_ylabel("Silhouette Score");
```

Question: What does the graph show?

Silhouette scores near 1 indicate that samples are far away from neighboring clusters. Scores close to 0 indicate that samples are on or very close to the decision boundary between two neighboring clusters.

The plot indicates that the silhouette score is closest to 1 when the data is partitioned into six clusters, although five clusters also yield a relatively good silhouette score.

Optimal k-value

To decide on an optimal k-value, fit a six-cluster model to the dataset.

In [ ]:
```
# Fit a 6-cluster model.
kmeans6 = KMeans(n_clusters=6, random_state=42)
kmeans6.fit(X_scaled)
```

Print out the unique labels of the fit model.

In [ ]:
```
# Print unique labels

print('Unique labels:', np.unique(kmeans6.labels_))
```

Now, create a new column cluster that indicates cluster assignment in the DataFrame penguins_subset. It's important to understand the meaning of each cluster's labels, then decide whether the clustering makes sense.

Note: This task is done using penguins_subset because it is often easier to interpret unscaled data.

In [ ]:
```
# Create a new column `cluster
penguins_subset['cluster'] = kmeans6.labels_
penguins_subset.head()
```

Use groupby to verify if any 'cluster' can be differentiated by 'species'

In [ ]:
```
# Verify if any `cluster` can be differentiated by `species`.

penguins_subset.groupby(by=['cluster', 'species']).size()
```

Next, interpret the groupby outputs. Although the results of the groupby show that each `'cluster'` can be differentiated by `'species'`, it is useful to visualize these results. The graph shows that each `'cluster'` can be differentiated by `'species'`.

Note: The code for the graph below is outside the scope of this lab.

In [ ]:
```
penguins_subset.groupby(by=['cluster', 'species']).size().plot.bar(title='
                                                                   figsize
                                                                   ylabel=
                                                                   xlabel=
```

Use groupby to verify if each 'cluster' can be differentiated by 'species' AND 'sex_MALE'.

In [ ]:
```
# Verify if each `cluster` can be differentiated by `species` AND `sex_MAL

penguins_subset.groupby(by=['cluster','species', 'sex_MALE']).size().sort_
```

Question: Are the clusters differentiated by 'species' and 'sex_MALE'?

Even though clusters 1 and 3 weren't all one species or sex, the groupby indicates that the algorithm produced clusters mostly differentiated by species and sex.

Finally, interpret the groupby outputs and visualize these results. The graph shows that each 'cluster' can be differentiated by 'species' and 'sex_MALE'. Furthermore, each cluster is mostly comprised of one sex and one species.

In [ ]:
```
penguins_subset.groupby(by=['cluster','species','sex_MALE']).size().unstac



plt.legend(bbox_to_anchor=(1.3, 1.0))
```

```

The K-means clustering enabled this data to be effectively grouped. It
helped identify patterns that can educate team members about penguins.
The success of the cluster results suggests that the organization can
apply clustering to other projects and continue augmenting employee
education.


```

In [ ]:
```

```