PROBLEM 2: DBSCAN and Spectral Clustering

Dataset: https://www.kaggle.com/datasets/taweilo/fish-species-sampling-weight-and-height-data

It is a synthetic data generated and inspired from the paper: Length-weight relationships of nine fish species from the Tetulia River, southern Bangladesh.

We will preprocess the dataset by dropping unnecessary columns and selecting two features for clustering. experiment with DBSCAN using different values for eps and min_samples, and compare DBSCAN results with spectral clustering.
First, the dataset is cleaned by removing irrelevant columns (w_l_ratio) and dropping specific classes (Sillaginopsis panijus, Setipinna taty, Puntius lateristriga) that may add noise or complexity.
The numerical columns (length and weight) are selected as the features for clustering.
Standardization is performed to scale the features to a uniform range, ensuring fair distance calculations for DBSCAN.

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv('C:\\Users\\lynet\\OneDrive\\Documents\\2024fall\\Part2\\fish_data.csv')

# Inspect the dataset
print(data.info())
print(data.head())

data.drop(columns=['w_l_ratio'], inplace=True)


# Remove certain classes
classes_to_drop = ['Sillaginopsis panijus', 'Setipinna taty', 'Puntius lateristriga']
data = data[~data['species'].isin(classes_to_drop)]


# Select numerical columns for clustering
numeric_columns = data.select_dtypes('float64').columns
X = data[numeric_columns].values
```

```
species  length  weight
0       Anabas testudineus   10.66    3.45
1       Anabas testudineus    6.91    3.27
2       Anabas testudineus    8.38    3.46
3       Anabas testudineus    7.57    3.36
4       Anabas testudineus   10.83    3.38
...                     ...     ...     ...
2722  Polynemus paradiseus   11.61    4.02
2723  Polynemus paradiseus   12.96    4.08
2724  Polynemus paradiseus    9.75    4.04
2725  Polynemus paradiseus   11.49    3.96
2726  Polynemus paradiseus   15.06    3.95

[2727 rows x 3 columns]
```

For DBSCAN parameters: DBSCAN uses two main parameters: eps (epsilon) and min_samples.

a) Make 2 - 3 experiments with DBSCAN using different values for the two parameters

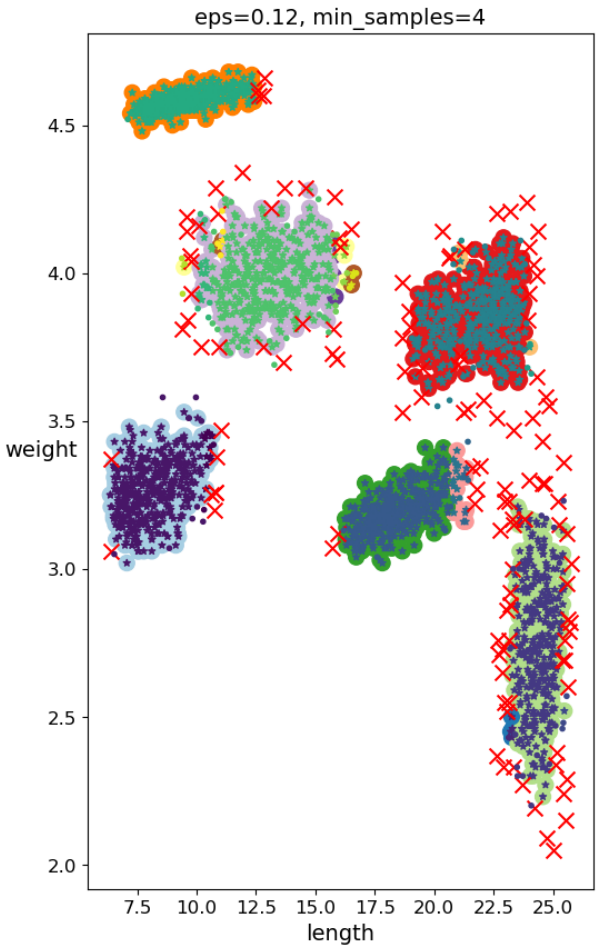
The experiments use three different parameter sets to observe their impact:

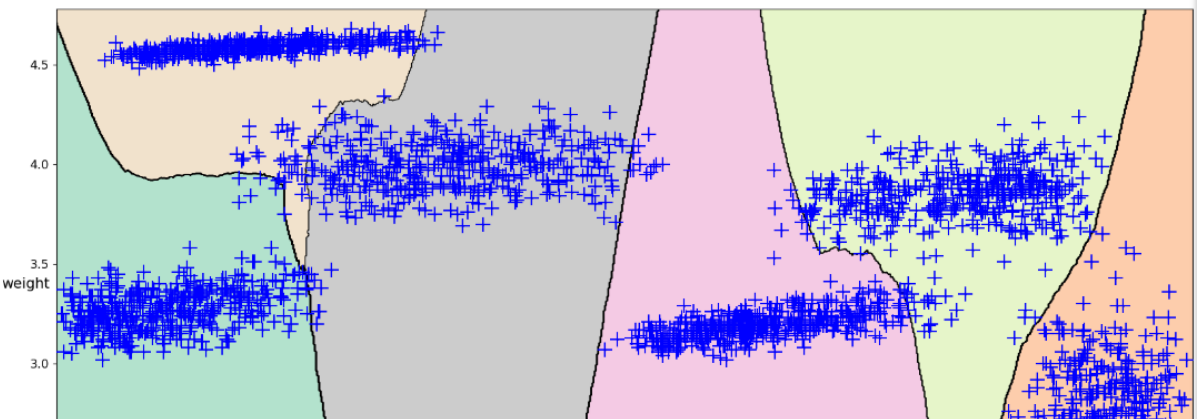
a. eps=0.12, min_samples=4
b. eps=0.15, min_samples=5
c. eps=0.19, min_samples=6

```
=== DBSCAN 1 Parameters ===
DBSCAN(eps=0.12, min_samples=4)
==========================
Labels: [ 0  1  1  1 -1  1  1  1  1  1]
Length of core sample indices: 2497
Core Sample Indices: [ 0  1  2  3  5  6  7  8  9 10]
Components:
[[10.66  3.45]
 [ 6.91  3.27]
 [ 8.38  3.46]]
Unique Labels:
[-1  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18]


KNeighborsClassifier(n_neighbors=50)
[13  1  1 ... 11 13 13]
[[0.24 0.22 0.   ... 0.   0.   0.  ]
 [0.   1.   0.   ... 0.   0.   0.  ]
 [0.   1.   0.   ... 0.   0.   0.  ]
 ...
 [0.   0.   0.   ... 0.02 0.   0.  ]
 [0.   0.   0.   ... 0.   0.   0.  ]
 [0.   0.   0.   ... 0.   0.   0.  ]]


[ 0  1  1 ... -1 13 13]
=== END OF DBSCAN 1 ===
```

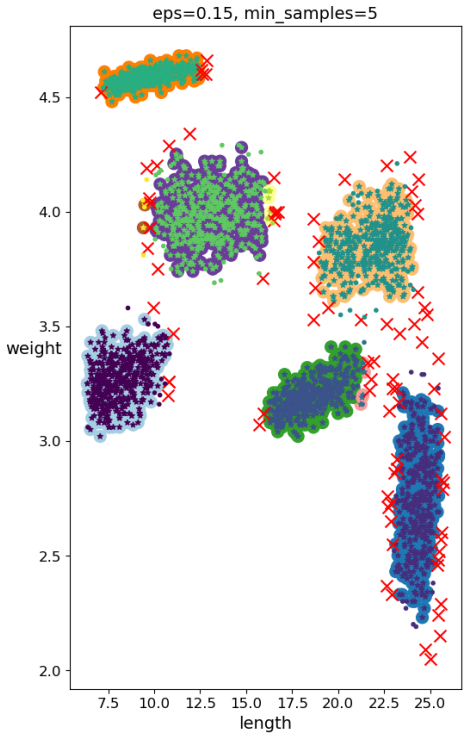


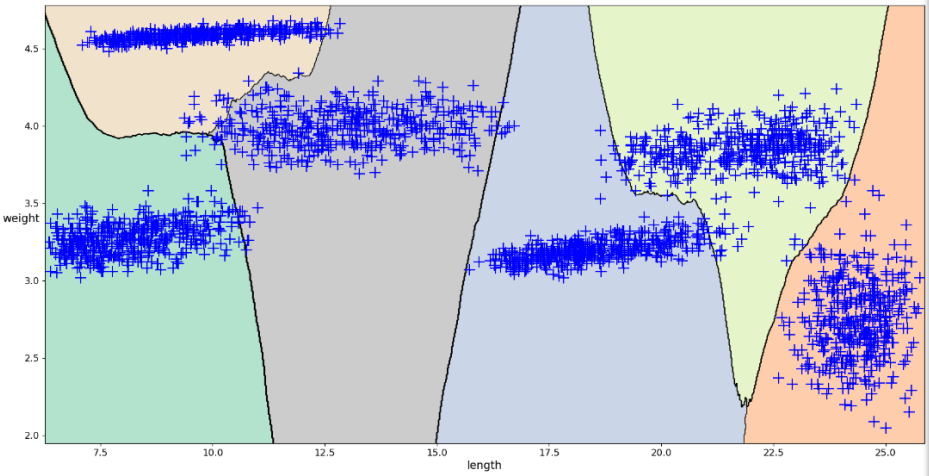


```
=== DBSCAN 2 Parameters ===
DBSCAN(eps=0.15)
==========================
Labels: [0 0 0 0 0 0 0 0 0 0]
Length of core sample indices: 2546
Core Sample Indices: [ 0  1  2  3  5  6  7  8  9 10]
Components:
[[10.66  3.45]
 [ 6.91  3.27]
 [ 8.38  3.46]]
Unique Labels:
[-1  0  1  2  3  4  5  6  7  8]


KNeighborsClassifier(n_neighbors=50)
[6 0 0 ... 5 6 6]
[[0.42 0.   0.   ... 0.58 0.   0.  ]
 [1.   0.   0.   ... 0.   0.   0.  ]
 [1.   0.   0.   ... 0.   0.   0.  ]
 ...
 [0.   0.   0.   ... 0.16 0.   0.04]
 [0.   0.   0.   ... 1.   0.   0.  ]
 [0.   0.   0.   ... 1.   0.   0.  ]]


[ 0  0  0 ... -1  6  6]
=== END OF DBSCAN 2 ===
```

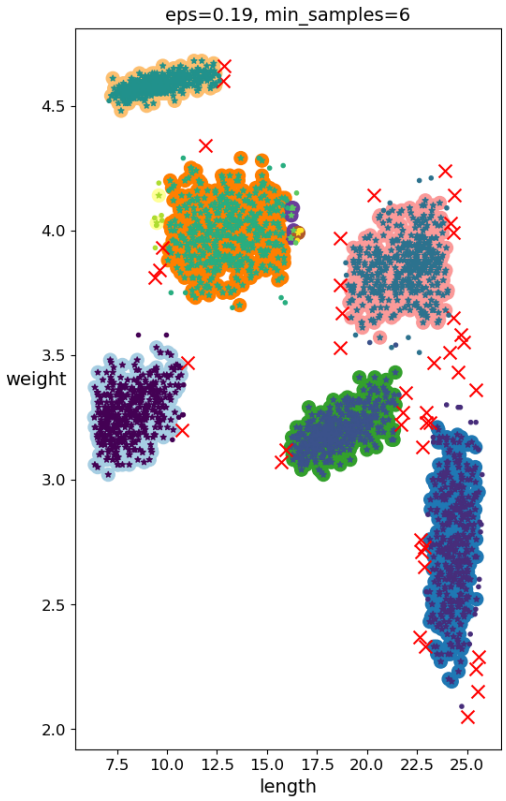


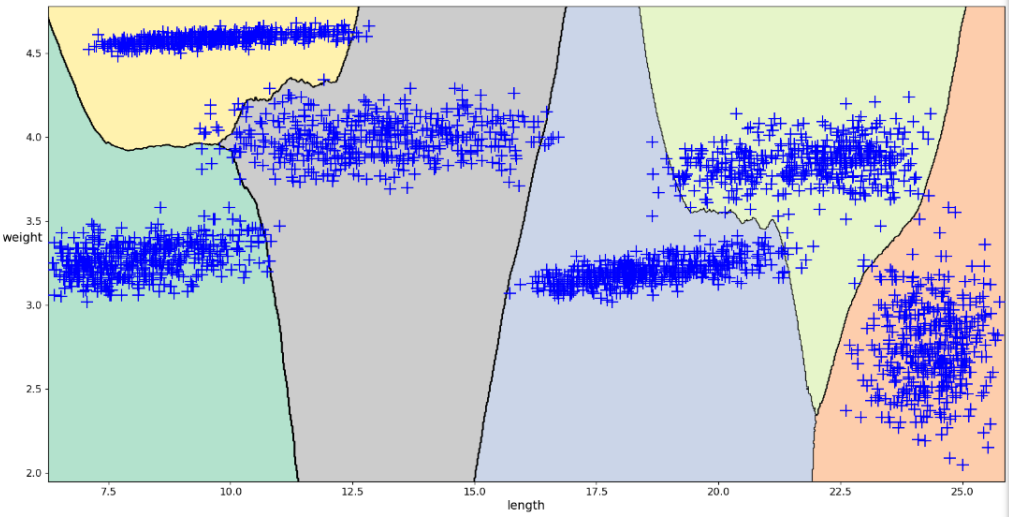


```
=== DBSCAN 3 Parameters ===
DBSCAN(eps=0.19, min_samples=6)
========================
Labels: [0 0 0 0 0 0 0 0 0 0]
Length of core sample indices: 2613
Core Sample Indices: [ 0  1  2  3  5  6  7  8  9 10]
Components:
[[10.66  3.45]
 [ 6.91  3.27]
 [ 8.38  3.46]]
Unique Labels:
[-1  0  1  2  3  4  5  6  7  8]


KNeighborsClassifier(n_neighbors=50)
[0 0 0 ... 4 5 5]
[[0.5  0.   0.   ... 0.   0.   0.  ]
 [1.   0.   0.   ... 0.   0.   0.  ]
 [1.   0.   0.   ... 0.   0.   0.  ]
 ...
 [0.04 0.   0.   ... 0.   0.04 0.  ]
 [0.   0.   0.   ... 0.   0.   0.  ]
 [0.   0.   0.   ... 0.   0.   0.  ]]


[0 0 0 ... 7 5 5]
=== END OF DBSCAN 3 ===
```



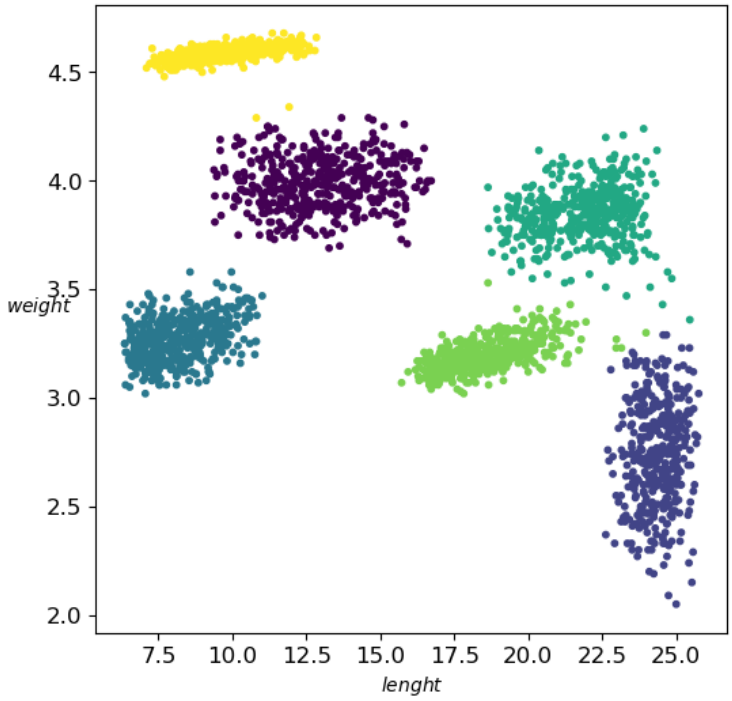


- I find that DBSCAN is particularly suitable for the Fish Dataset since it handles non-linear cluster boundaries effectively (e.g., clusters shaped by length-weight relationships).

- Compare the results with K-Means for the same dataset using the number of clusters you received from DBSCAN

I checked the Silhouette Score to inspect how well the clusters are formed. I checked a range between 4-10, and got 7 clusters would be the best option. You can see it from the code.

```
73]: num_data=data[numericColumns]
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_data = scaler.fit_transform(num_data[numericColumns])
kmeans = KMeans(n_clusters=6, random_state=6)
y = kmeans.fit_predict(scaled_data)
def plot_clusters(num_data, y=None):
    plt.scatter(num_data.iloc[:, 0], num_data.iloc[:, 1], c=y, s=10)
    plt.xlabel("$lenght$", fontsize=10)
    plt.ylabel("$weight$", fontsize=10, rotation=0)
plt.figure(figsize=(6, 6))
plot_clusters(num_data[numericColumns], y)
plt.show()
```



K-means method we chose 1 parameter and DBSCAN we chose 2 parameters.

b) Make 2-3 experiments with Spectral Clustering using different values for RBF gamma parameter.

The gamma parameter in the RBF kernel controls the spread of the Gaussian function used in calculating similarity between data points. We will test 2-3 different gamma values to observe their impact on the clustering results.

I noticed that as gamma increases, the clusters become more tightly bound around core points, but more fragmentation if the gamma is too large.

For gamma=1.0 is more balanced cluster shapes. As the number of clusters I see are more correspond to different species or biological categories of fish.

You can tell from below that more well-defined clusters with reasonable separation when gamma = 1.

```
81]: sc1 = SpectralClustering(n_clusters=6, affinity='nearest_neighbors',
     n_neighbors=10, gamma=1, random_state=42)
     sc1.fit(scaled_data)
```

```
C:\Users\lynet\anaconda3\Lib\site-packages\sklearn\manifold\_spectral_embedding.py:329: UserWarning: Graph is not fully connected, spectral embedding
not work as expected.
  warnings.warn(
```

```
81]: SpectralClustering(affinity='nearest_neighbors', gamma=1, n_clusters=6,
                        random_state=42)
```

```
83]: sc2 = SpectralClustering(n_clusters=6, affinity='nearest_neighbors',
     n_neighbors=13, gamma=100, random_state=42)
     sc2.fit(scaled_data)
```

```
C:\Users\lynet\anaconda3\Lib\site-packages\sklearn\manifold\_spectral_embedding.py:329: UserWarning: Graph is not fully connected, spectral embedding
not work as expected.
  warnings.warn(
```

```
83]: SpectralClustering(affinity='nearest_neighbors', gamma=100, n_clusters=6,
                        n_neighbors=13, random_state=42)
```

```
85]: SpectralClustering(affinity='nearest_neighbors', gamma=1, n_clusters=2,
                        random_state=42)

     np.percentile(sc1.affinity_matrix_, 95)
```

```
85]: <2727x2727 sparse matrix of type '<class 'numpy.float64'>'
        with 32457 stored elements in Compressed Sparse Row format>
```

```
09]: def plot_spectral_clustering(sc, X, size, alpha, show_xlabels=True,
     show_ylabels=True):
         plt.scatter(X[:, 0], X[:, 1], marker='x', s=size, c='green',
     cmap="Paired", alpha=alpha)
         plt.scatter(X[:, 0], X[:, 1], marker='x', s=30, c='w')
         plt.scatter(X[:, 0], X[:, 1], marker='.', s=10, c=sc.labels_,
     cmap="Paired")

         if show_xlabels:
             plt.xlabel(numericColumns[0], fontsize=14)
         else:
             plt.tick_params(labelbottom=False)
         if show_ylabels:
             plt.ylabel(numericColumns[1], fontsize=14, rotation=0)
         else:
             plt.tick_params(labelleft=False)
         plt.title("RBF gamma={}".format(sc.gamma), fontsize=14)
```

```
11]: plt.figure(figsize=(9, 3.2))

     plt.subplot(121)
     plot_spectral_clustering(sc1, X, size=500, alpha=0.1)

     plt.subplot(122)
     plot_spectral_clustering(sc2, X, size=4000, alpha=0.01, show_ylabels=False)

     plt.show()
```

```
C:\Users\lynet\AppData\Local\Temp\ipykernel_4108\1802152998.py:3: UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will be ig
ed
  plt.scatter(X[:, 0], X[:, 1], marker='x', s=size, c='green',
```