```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.mixture import GaussianMixture
```

```
# Step 1: Load the Iris dataset
iris = datasets.load_iris()
```

```
import pandas as pd

# Create a DataFrame from the dataset
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)

# Display the first few rows of the DataFrame
display(iris_df.head())
```

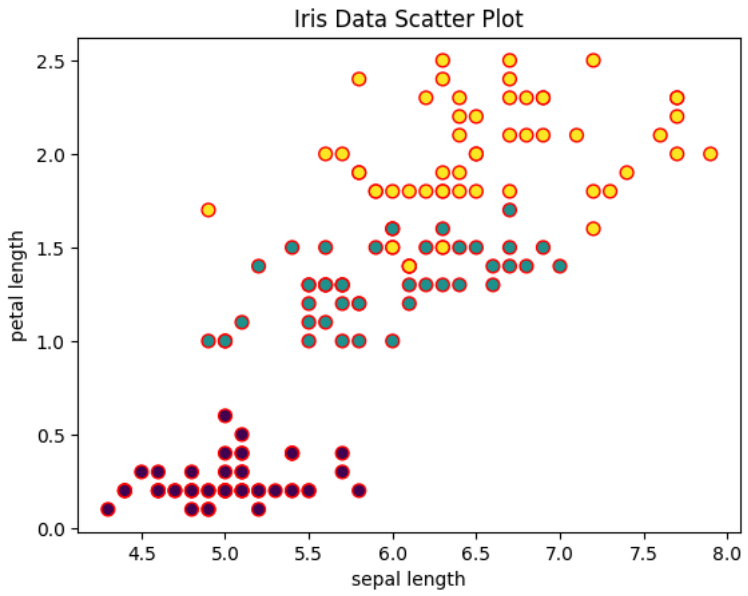
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
X = iris.data  # Features
y = iris.target  # True labels (species)
iris_species = iris.target_names
iris_species
```

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

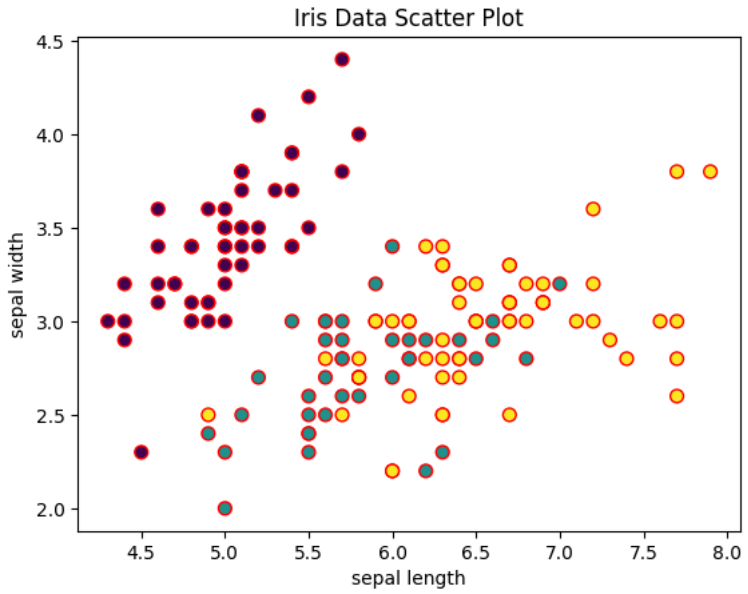
```
# Step 2: Scatter plot (example using petal length and petal width)
plt.scatter(X[:, 0], X[:, 3], c=y, cmap='viridis', edgecolor='r', s=50)

#Color abbreviations (e.g., 'k' for black, 'r' for red)
#You can try other colormaps like 'plasma', 'inferno', 'magma', or 'cividis' depending on your preference.
plt.xlabel('sepal length')
plt.ylabel('petal width')
plt.title('Iris Data Scatter Plot')
plt.show()
```



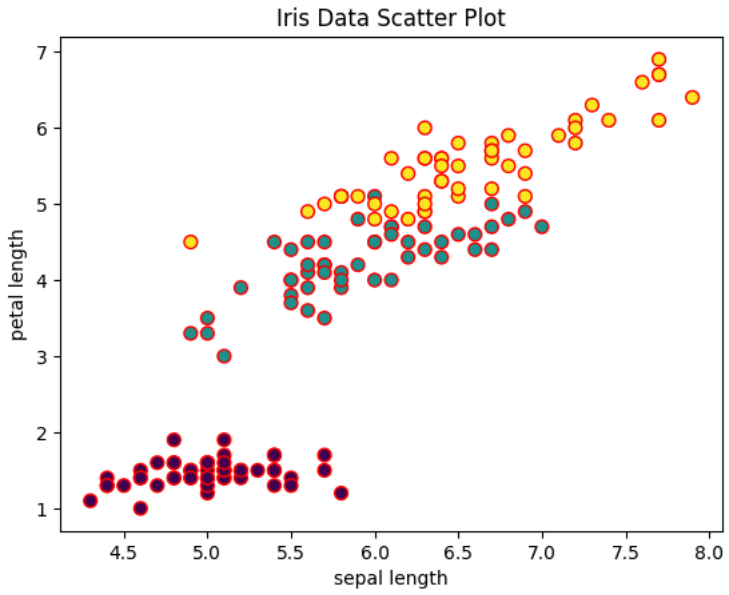
```
# Step 2: Scatter plot (example using petal length and petal width)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap='viridis', edgecolor='r', s=50)

#Color abbreviations (e.g., 'k' for black, 'r' for red)
#You can try other colormaps like 'plasma', 'inferno', 'magma', or 'cividis' depending on your preference.
plt.xlabel('sepal length')
plt.ylabel('sepal width')
plt.title('Iris Data Scatter Plot')
plt.show()
```



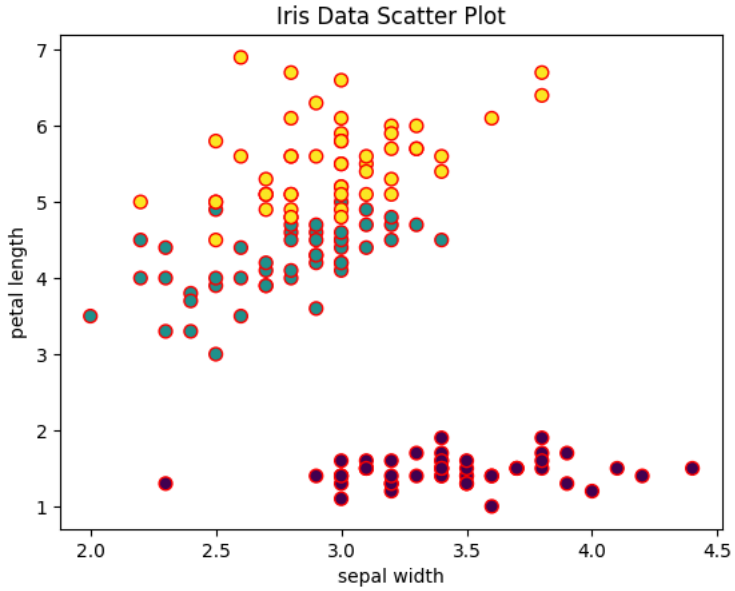

```
# Step 2: Scatter plot (example using petal length and petal width)
plt.scatter(X[:, 0], X[:, 2], c=y, cmap='viridis', edgecolor='r', s=50)

#Color abbreviations (e.g., 'k' for black, 'r' for red)
#You can try other colormaps like 'plasma', 'inferno', 'magma', or 'cividis' depending on your preference.
plt.xlabel('sepal length')
plt.ylabel('petal length')
plt.title('Iris Data Scatter Plot')
plt.show()
```



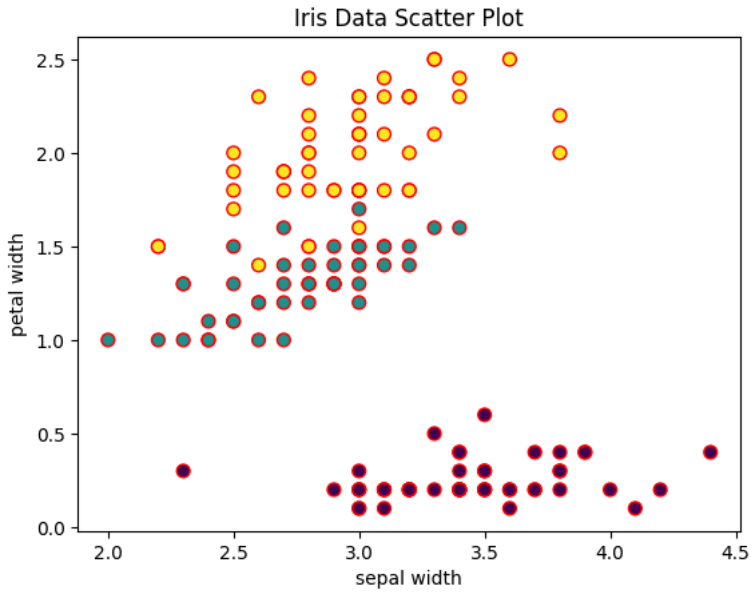

```
# Step 2: Scatter plot (example using petal length and petal width)
plt.scatter(X[:, 1], X[:, 2], c=y, cmap='viridis', edgecolor='r', s=50)

#Color abbreviations (e.g., 'k' for black, 'r' for red)
#You can try other colormaps like 'plasma', 'inferno', 'magma', or 'cividis' depending on your preference.
plt.xlabel('sepal width')
plt.ylabel('petal length')
plt.title('Iris Data Scatter Plot')
plt.show()
```



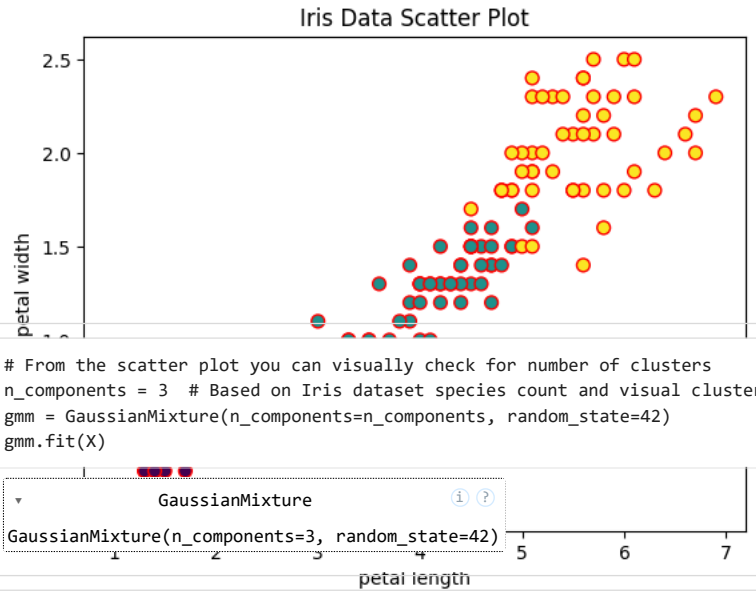

```
# Step 2: Scatter plot (example using petal length and petal width)
plt.scatter(X[:, 1], X[:, 3], c=y, cmap='viridis', edgecolor='r', s=50)

#Color abbreviations (e.g., 'k' for black, 'r' for red)
#You can try other colormaps like 'plasma', 'inferno', 'magma', or 'cividis' depending on your preference.
plt.xlabel('sepal width')
plt.ylabel('petal width')
plt.title('Iris Data Scatter Plot')
plt.show()
```




```
# Step 2: Scatter plot (example using petal length and petal width)
plt.scatter(X[:, 2], X[:, 3], c=y, cmap='viridis', edgecolor='r', s=50)

#Color abbreviations (e.g., 'k' for black, 'r' for red)
#You can try other colormaps like 'plasma', 'inferno', 'magma', or 'cividis' depending on your preference.
plt.xlabel('petal length')
plt.ylabel('petal width')
plt.title('Iris Data Scatter Plot')
plt.show()
```




```
# From the scatter plot you can visually check for number of clusters
n_components = 3  # Based on Iris dataset species count and visual clusters
gmm = GaussianMixture(n_components=n_components, random_state=42)
gmm.fit(X)
```

```
GaussianMixture(n_components=3, random_state=42)
```

```
# Cluster assignment (hard clustering)
cluster_labels = gmm.predict(X)
cluster_labels
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 0, 2, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
iris_df['Cluster'] = cluster_labels
iris_df
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Cluster
0	5.1	3.5	1.4	0.2	1
1	4.9	3.0	1.4	0.2	1
2	4.7	3.2	1.3	0.2	1
3	4.6	3.1	1.5	0.2	1
4	5.0	3.6	1.4	0.2	1
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	0
146	6.3	2.5	5.0	1.9	0
147	6.5	3.0	5.2	2.0	0
148	6.2	3.4	5.4	2.3	0
149	5.9	3.0	5.1	1.8	0

150 rows × 5 columns

**Bayesian Information Criterion (BIC), **

Akaike Information Criterion (AIC)**

```
# Cluster membership probabilities (soft clustering)
cluster_probabilities = gmm.predict_proba(X)

# Print log-likelihood, BIC, and AIC for evaluation
print(f"Log Likelihood: {gmm.score(X) * X.shape[0]:.2f}")
print(f"BIC: {gmm.bic(X):.2f}")
print(f"AIC: {gmm.aic(X):.2f}")
```

```
Log Likelihood: -180.20
BIC: 580.86
AIC: 448.39
```

The **log likelihood** tells you how well the model fits the data.It is the **logarithm of the probability of the data** given the model parameters.

Because likelihoods are values between 0 and 1, their logarithms are usually negative.

If the log likelihood is much lower (e.g., -300), it signifies a poorer fit.

BIC and AIC help in selecting the number of Gaussian components by penalizing overly complex models.

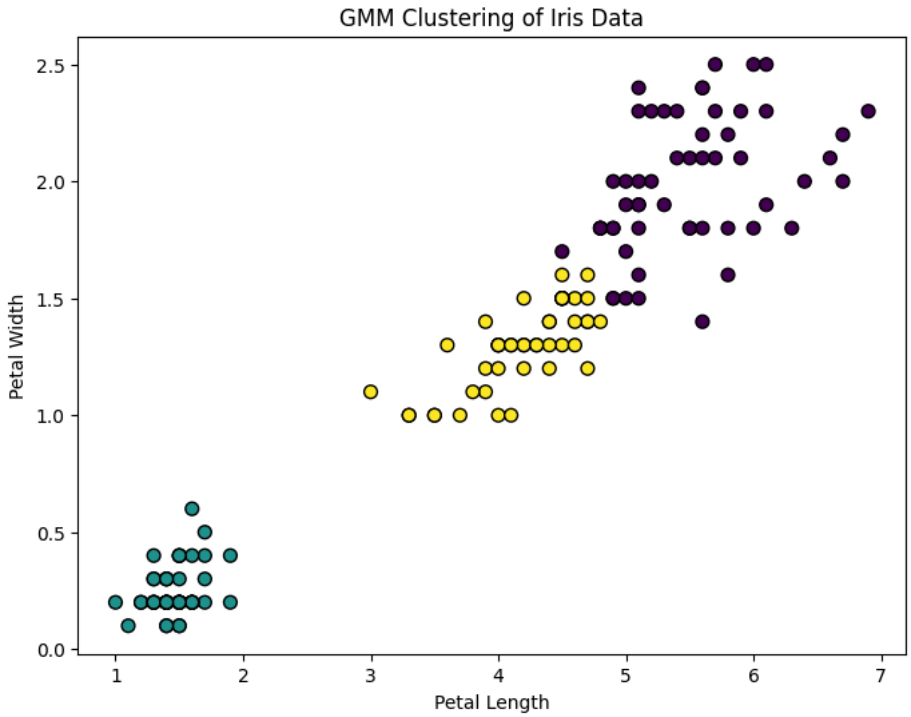
Both BIC and AIC are useful model selection criteria.

Your reported values (580.86 for BIC and 448.39 for AIC) numerically describe the trade-off between model complexity and fit for your GMM.

To judge if they are "good," compare with BIC and AIC values from GMMs of other component numbers—the model with the lowest BIC and AIC is generally preferable for balance between accuracy and simplicity.

```
# Visualization (scatter plot with clusters on petal length and width)
plt.figure(figsize=(8, 6))
scatter = plt.scatter(X[:, 2], X[:, 3], c=cluster_labels, cmap='viridis', edgecolor='k', s=50)
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.title('GMM Clustering of Iris Data')
```

```
Text(0.5, 1.0, 'GMM Clustering of Iris Data')
```




```
# Step 4: Model parameters - means and covariances
print("Model Means:")
print(gmm.means_)
```

```
Model Means:
[[6.54639415 2.94946365 5.48364578 1.98726565]
 [5.006      3.428      1.462      0.246     ]
 [5.9170732  2.77804839 4.20540364 1.29848217]]
```

Component	Sepal Length	Sepal Width	Petal Length	Petal Width
1	6.55	2.95	5.48	1.99
2	5.01	3.43	1.46	0.25
3	5.92	2.78	4.21	1.30

```
print("\nModel Covariances:")
print(gmm.covariances_)
```

```
Model Covariances:
[[[0.38744093 0.09223276 0.30244302 0.06087397]
  [0.09223276 0.11040914 0.08385112 0.05574334]
  [0.30244302 0.08385112 0.32589574 0.07276776]
  [0.06087397 0.05574334 0.07276776 0.08484505]]

 [[0.121765   0.097232   0.016028   0.010124  ]
  [0.097232   0.140817   0.011464   0.009112  ]
  [0.016028   0.011464   0.029557   0.005948  ]
  [0.010124   0.009112   0.005948   0.010885  ]]

 [[0.2755171  0.09662295 0.18547072 0.05478901]
  [0.09662295 0.09255152 0.09103431 0.04299899]
  [0.18547072 0.09103431 0.20235849 0.06171383]
  [0.05478901 0.04299899 0.06171383 0.03233775]]]
```