In [2]:
```
# Load the dataset and print its shape
from sklearn import datasets

# Load Iris dataset
iris = datasets.load_iris()

# Print shape of features (X) and target (y)
X = iris.data
y = iris.target

print(f"Shape of Features (X): {X.shape}")
print(f"Shape of Target (y): {y.shape}")
```

```
Shape of Features (X): (150, 4)
Shape of Target (y): (150,)
```

In [3]:
```
# Split the dataset into training and testing sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randc

print(f"Training set size (X_train): {X_train.shape}")
print(f"Testing set size (X_test): {X_test.shape}")
```

```
Training set size (X_train): (120, 4)
Testing set size (X_test): (30, 4)
```

In [4]:
```
# Train a logistic regression model and check accuracy
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Create model and fit
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Test Accuracy: {accuracy * 100:.2f}%")
```

```
Test Accuracy: 100.00%
```

In [5]:
```
# Calculate confusion matrix
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

print("Confusion Matrix:\n", cm)
```

```
Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

In [6]:
```
# Perform cross-validation
from sklearn.model_selection import cross_val_score

# 5-fold cross-validation
cv_scores = cross_val_score(model, X, y, cv=5)

print("Cross-validation accuracy for each fold:", cv_scores)
print(f"Average accuracy: {cv_scores.mean() * 100:.2f}%")
```

```
Cross-validation accuracy for each fold: [0.96666667 1.         0.93333333 0.
96666667 1.        ]
Average accuracy: 97.33%
```

In [7]:
```
# Hyperparameter tuning using Grid Search
from sklearn.model_selection import GridSearchCV

# Define hyperparameter grid
param_grid = {'C': [0.1, 1, 10]}

# Perform GridSearchCV
grid_search = GridSearchCV(LogisticRegression(max_iter=200), param_grid, cv=5)
grid_search.fit(X_train, y_train)

# Output the best hyperparameters
print("Best Hyperparameter (C):", grid_search.best_params_)
```

```
Best Hyperparameter (C): {'C': 1}
```

In [8]:
```
# Normalize the features and train a logistic regression model
from sklearn.preprocessing import MinMaxScaler

# Normalize the features
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train the model on the normalized data
model.fit(X_train_scaled, y_train)

# Predict and check accuracy
y_pred_scaled = model.predict(X_test_scaled)
accuracy_scaled = accuracy_score(y_test, y_pred_scaled)
print(f"Accuracy with normalized features: {accuracy_scaled * 100:.2f}%")
```

```
Accuracy with normalized features: 96.67%
```

In [ ]: