In [1]:

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler, LabelEncoder

file_path = 'Iris.csv'
df = pd.read_csv(file_path)

X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values

label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

def evaluate_knn(X_train, X_test, y_train, y_test, k_values, weighted=False):
    results = {}
    for k in k_values:
        if weighted:
            knn = KNeighborsClassifier(n_neighbors=k, weights='distance')
        else:
            knn = KNeighborsClassifier(n_neighbors=k)

        knn.fit(X_train, y_train)
        # Make predictions
        y_pred = knn.predict(X_test)

        accuracy = accuracy_score(y_test, y_pred)
        f1 = f1_score(y_test, y_pred, average='weighted')
        conf_matrix = confusion_matrix(y_test, y_pred)
        results[k] = {'accuracy': accuracy, 'f1_score': f1, 'conf_matrix': conf_matrix}
    return results

k_values = [1, 3, 5]

regular_knn_results = evaluate_knn(X_train, X_test, y_train, y_test, k_values, weighted=False)
print("Regular k-NN Results:")
for k, metrics in regular_knn_results.items():
    print(f"\nk={k}: Accuracy={metrics['accuracy']:.4f}, F1-Score={metrics['f1_score']:.4f}")
    print("Confusion Matrix:")
    print(metrics['conf_matrix'])

weighted_knn_results = evaluate_knn(X_train, X_test, y_train, y_test, k_values, weighted=True)
print("\nWeighted k-NN Results:")
for k, metrics in weighted_knn_results.items():
    print(f"\nk={k}: Accuracy={metrics['accuracy']:.4f}, F1-Score={metrics['f1_score']:.4f}")
    print("Confusion Matrix:")
    print(metrics['conf_matrix'])
```

```
Regular k-NN Results:

k=1: Accuracy=1.0000, F1-Score=1.0000
Confusion Matrix:
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]

k=3: Accuracy=1.0000, F1-Score=1.0000
Confusion Matrix:
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]

k=5: Accuracy=1.0000, F1-Score=1.0000
Confusion Matrix:
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]

Weighted k-NN Results:

k=1: Accuracy=1.0000, F1-Score=1.0000
Confusion Matrix:
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]

k=3: Accuracy=1.0000, F1-Score=1.0000
Confusion Matrix:
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]

k=5: Accuracy=1.0000, F1-Score=1.0000
Confusion Matrix:
[[19  0  0]
```

```
 [ 0 13  0]
 [ 0  0 13]]
```

In [ ]:

```
```