In [2]:

```
# Kindly use the Jupyter Notebook to run this program.

import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, f1_score
```

In [4]:

```
# Load the Iris dataset
iris = load_iris()
X = iris.data  # Features
y = iris.target  # Labels

# Split the data: 70% train, 30% test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
```

In [6]:

```
# Function to evaluate k-NN for different values of k
def cls_knn(X_train, X_test, y_train, y_test, k_values, weighted=False):
    results = {}
    for k in k_values:
        knn = KNeighborsClassifier(n_neighbors=k, weights='distance' if weighted else 'uniform')
        knn.fit(X_train, y_train)
        y_pred = knn.predict(X_test)

        accuracy = accuracy_score(y_test, y_pred)
        f1 = f1_score(y_test, y_pred, average='weighted')  # Handle multi-class
        results[k] = {'accuracy': accuracy, 'f1_score': f1}
    return results
```

In [8]:

```
# Test for different values of k
k_values = [1, 3, 5]

# Evaluate regular k-NN
print("📊 Regular k-NN Results:")
regular_knn = cls_knn(X_train, X_test, y_train, y_test, k_values, weighted=False)
for k, metrics in regular_knn.items():
    print(f"k={k}: Accuracy = {metrics['accuracy']:.4f}, F1-Score = {metrics['f1_score']:.4f}")

# Evaluate weighted k-NN
print("\n📊 Weighted k-NN Results:")
weighted_knn = cls_knn(X_train, X_test, y_train, y_test, k_values, weighted=True)
for k, metrics in weighted_knn.items():
    print(f"k={k}: Accuracy = {metrics['accuracy']:.4f}, F1-Score = {metrics['f1_score']:.4f}")
```

```
📊 Regular k-NN Results:
k=1: Accuracy = 1.0000, F1-Score = 1.0000
k=3: Accuracy = 1.0000, F1-Score = 1.0000
k=5: Accuracy = 1.0000, F1-Score = 1.0000

📊 Weighted k-NN Results:
k=1: Accuracy = 1.0000, F1-Score = 1.0000
k=3: Accuracy = 1.0000, F1-Score = 1.0000
k=5: Accuracy = 1.0000, F1-Score = 1.0000
```

In [10]:

```
# Compare both versions side by side
print("\n🔍 Comparison of Regular vs Weighted k-NN Accuracy:")
for k in k_values:
    reg_acc = regular_knn[k]['accuracy']
    wt_acc = weighted_knn[k]['accuracy']
    print(f"k={k}: Regular = {reg_acc:.4f}, Weighted = {wt_acc:.4f}")
```

```

🔍 Comparison of Regular vs Weighted k-NN Accuracy:
k=1: Regular = 1.0000, Weighted = 1.0000
```

```
k=3: Regular = 1.0000, Weighted = 1.0000
k=5: Regular = 1.0000, Weighted = 1.0000
```