```
#Choose a classification dataset and normalize features

import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from matplotlib.colors import ListedColormap

# Load the Iris dataset
iris = load_iris()
X = iris.data[:, :2]  # Take the first two features for visualization
y = iris.target

# Normalize features using StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Create KNeighborsClassifier with different values of K
k_values = [1, 3, 5, 7]
models = {}
for k in k_values:
    models[k] = KNeighborsClassifier(n_neighbors=k)
    models[k].fit(X_train, y_train)

# Evaluate models using accuracy and confusion matrix
for k in k_values:
    y_pred = models[k].predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    cm = confusion_matrix(y_test, y_pred)
    print(f"K = {k}")
    print(f"Accuracy: {accuracy}")
    print(f"Confusion Matrix:\n{cm}\n")

# Visualize decision boundaries
plt.figure(figsize=(12, 8))

# Meshgrid creation for decision boundary visualization
x_min, x_max = X_scaled[:, 0].min() - 1, X_scaled[:, 0].max() + 1
y_min, y_max = X_scaled[:, 1].min() - 1, X_scaled[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01),
                     np.arange(y_min, y_max, 0.01))

# Plot decision boundaries for each classifier
for idx, k in enumerate(k_values):
    Z = models[k].predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.subplot(2, 2, idx + 1)
    plt.contourf(xx, yy, Z, alpha=0.8, cmap=ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF']))
    plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=y, cmap=ListedColormap(['#FF0000', '#00FF00', '#0000FF']), edgecolors='k')
    plt.title(f'K = {k}')
    plt.xlabel('Sepal Length (scaled)')
    plt.ylabel('Sepal Width (scaled)')

plt.tight_layout()
plt.show()
```

```
K = 1
Accuracy: 0.8
Confusion Matrix:
[[10  0  0]
 [ 0  7  2]
 [ 0  4  7]]

K = 3
Accuracy: 0.8333333333333334
Confusion Matrix:
[[10  0  0]
 [ 0  6  3]
 [ 0  2  9]]

K = 5
Accuracy: 0.8333333333333334
Confusion Matrix:
[[10  0  0]
 [ 0  7  2]
 [ 0  3  8]]

K = 7
Accuracy: 0.8
Confusion Matrix:
[[10  0  0]
 [ 0  7  2]
 [ 0  4  7]]
```