```
import numpy as np
import matplotlib.pyplot as plt
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
from matplotlib.colors import ListedColormap

# Load the dataset (Iris dataset for this example)
data = load_iris()
X = data.data  # Features
y = data.target  # Labels

# Select only the first two features for easy plotting
X = X[:, :2]
feature_names = data.feature_names[:2]

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Initialize the KNN classifier with k=5
k = 7
knn = KNeighborsClassifier(n_neighbors=k)

# Train the model on the training data
knn.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = knn.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of KNN classifier with k={k}: {accuracy:.2f}")

# Plotting
# Create a mesh grid for plotting decision boundaries
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01),
                     np.arange(y_min, y_max, 0.01))

# Predict on each point in the mesh grid
Z = knn.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Create a color map
cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
```

```
cmap_bold = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])

# Plot decision boundaries
plt.figure(figsize=(10, 6))
plt.contourf(xx, yy, Z, alpha=0.3, cmap=cmap_light)

# Plot training points
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cmap_bold,
edgecolor='k', marker='o', label='Training data')

# Plot testing points
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cmap_bold,
edgecolor='k', marker='x', label='Test data')

plt.xlabel(feature_names[0])
plt.ylabel(feature_names[1])
plt.title(f'KNN Decision Boundaries (k={k})')
plt.legend()
plt.show()
```