```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
ConfusionMatrixDisplay
from matplotlib.colors import ListedColormap

# 1. Load dataset and normalize features
df = pd.read_csv("/content/iris")

# Drop 'Id' and extract features and target
X = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
y = df['Species']

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

# 2. Split dataset and use KNeighborsClassifier
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3,
random_state=42)

# 3. Train & test with different values of K
k_values = [1, 3, 5, 7]
for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)

    # 4. Evaluate using accuracy and confusion matrix
    acc = accuracy_score(y_test, y_pred)
    print(f"K = {k}, Accuracy = {acc:.2f}")
    cm = confusion_matrix(y_test, y_pred, labels=knn.classes_)
    ConfusionMatrixDisplay(cm, display_labels=knn.classes_).plot()
    plt.title(f"Confusion Matrix (K = {k})")
    plt.show()

# 5. Visualize decision boundaries using only 2D (PetalLengthCm & PetalWidthCm)
X2 = df[['PetalLengthCm', 'PetalWidthCm']].values  # Convert to NumPy array
X2_scaled = scaler.fit_transform(X2).astype(float)  # Ensure float type

X2_train, X2_test, y2_train, y2_test = train_test_split(X2_scaled, y,
test_size=0.3, random_state=42)
knn_2d = KNeighborsClassifier(n_neighbors=3)
knn_2d.fit(X2_train, y2_train)
```

```
# Convert categorical labels to numeric
y_numeric = pd.factorize(y)[0].astype(int)

# Plot decision boundary
def plot_decision_boundary(knn_model, X, y_labels, title):
    h = .02
    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, h),
                         np.arange(y_min, y_max, h))

    # Predict for each mesh point
    Z = knn_model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = pd.factorize(Z)[0].reshape(xx.shape)  # Convert class labels to numeric
for plotting

    cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
    cmap_bold = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])

    plt.figure(figsize=(8, 6))
    plt.contourf(xx, yy, Z, cmap=cmap_light)
    plt.scatter(X[:, 0], X[:, 1], c=np.array(y_labels), cmap=cmap_bold,
edgecolor='k', s=30)
    plt.xlabel("PetalLengthCm (normalized)")
    plt.ylabel("PetalWidthCm (normalized)")
    plt.title(title)
    plt.show()

plot_decision_boundary(knn_2d, X2_scaled, y_numeric, "Decision Boundary (K=3)
using PetalLengthCm & PetalWidthCm")
```