SVM Homework Overview

HW3 Project Summary

This project covers the development of an SVM evaluation tool for different types of datasets in Streamlit, spanning HW3-1, HW3-2, and HW3-3. Each part involved incremental development, including prompt guidance, Python code implementations, and subsequent changes to refine the functionality.

HW3-1: 1D Comparison Between Logistic Regression and SVM

Prompt Summary for HW3-1

1. **Initial Requirement**:
   - Create a **1D dataset** where an SVM classifier achieves significantly higher accuracy than Logistic Regression.
   - Design a dataset with two regions where the classes are separated by a nonlinear boundary to highlight SVM's advantage.
2. **Implementation of HW3-1**:
   - Generated a simple 1D dataset where `y=1` for `x<a` or `x>b`, and `y=0` for values between `a` and `b`.
   - Trained both **SVM (RBF kernel)** and **Logistic Regression** models on this dataset and plotted their decision boundaries.

Python Code for HW3-1 (Complete)

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Generate 1D dataset
np.random.seed(0)
x = np.linspace(-10, 10, 200)
y = np.where((x < -3) | (x > 3), 1, 0)
x += np.random.normal(0, 0.5, x.shape)

# Split into train and test sets
x_train, x_test, y_train, y_test = train_test_split(x.reshape(-1, 1), y, test_size=0.3, ra
ndom_state=42)

# Train SVM and Logistic Regression
svm_model = SVC(kernel='rbf', gamma='scale')
logistic_model = LogisticRegression()
svm_model.fit(x_train, y_train)
logistic_model.fit(x_train, y_train)

# Evaluate and visualize
y_pred_svm = svm_model.predict(x_test)
y_pred_logistic = logistic_model.predict(x_test)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
accuracy_logistic = accuracy_score(y_test, y_pred_logistic)

# Plotting
x_plot = np.linspace(-10, 10, 500).reshape(-1, 1)
y_plot_svm = svm_model.decision_function(x_plot)
y_plot_logistic = logistic_model.decision_function(x_plot)

plt.figure(figsize=(10, 6))
plt.scatter(x_train, y_train, color='blue', label='Training data')
plt.scatter(x_test, y_test, color='green', label='Test data')
plt.plot(x_plot, y_plot_svm, color='red', linestyle='--', label='SVM decision function')
plt.plot(x_plot, y_plot_logistic, color='purple', linestyle='-.', label='Logistic decision
function')
plt.axhline(0, color='grey', linestyle=':')
plt.title(f"SVM Accuracy: {accuracy_svm:.2f} | Logistic Regression Accuracy: {accuracy_log
istic:.2f}")
plt.xlabel("Feature (x)")
plt.ylabel("Decision Function Value")
plt.legend()
plt.show()
```

HW3-2: 2D SVM with Streamlit Deployment and 3D Visualization

Prompt Summary for HW3-2

1. **Initial Requirement**:
   - Develop a **Streamlit app** with SVM on a 2D circular dataset, visualized in 3D.
   - Allow users to adjust dataset and SVM parameters.
2. **First Code Implementation**:
   - Created a Streamlit app for a circular dataset with adjustable parameters (`n_samples`, `factor`, `noise`) and **SVM** parameters (`kernel`, `gamma`, `C`).
   - Displayed the SVM decision boundary in 3D.

Python Code for HW3-2 (Complete)

```python
import streamlit as st
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.svm import SVC
from sklearn.datasets import make_circles
from sklearn.metrics import accuracy_score

# Streamlit layout and dataset generation
st.title("2D SVM Classifier with Adjustable Parameters and 3D Visualization")
n_samples = st.sidebar.slider("Number of Samples", 100, 1000, 300, 50)
factor = st.sidebar.slider("Factor (Circle Separation)", 0.1, 1.0, 0.5, 0.1)
noise = st.sidebar.slider("Noise Level", 0.0, 0.3, 0.05, 0.01)
X, y = make_circles(n_samples=n_samples, factor=factor, noise=noise, random_state=42)

# SVM parameters
kernel = st.sidebar.selectbox("Kernel Type", ["rbf", "linear", "poly"])
gamma = st.sidebar.selectbox("Gamma (for RBF kernel)", ["scale", "auto"])
C = st.sidebar.slider("Regularization (C)", 0.1, 10.0, 1.0, 0.1)
svm_model = SVC(kernel=kernel, gamma=gamma, C=C)
svm_model.fit(X, y)
accuracy = accuracy_score(y, svm_model.predict(X))
st.write(f"Model Accuracy: {accuracy:.2f}")

# 3D plot of decision boundary
x_range = np.linspace(X[:, 0].min() - 0.5, X[:, 0].max() + 0.5, 100)
y_range = np.linspace(X[:, 1].min() - 0.5, X[:, 1].max() + 0.5, 100)
xx, yy = np.meshgrid(x_range, y_range)
z = svm_model.decision_function(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X[y == 0][:, 0], X[y == 0][:, 1], 0, color='blue', label='Class 0', alpha=0.6)
ax.scatter(X[y == 1][:, 0], X[y == 1][:, 1], 0, color='orange', label='Class 1', alpha=0.
6)
ax.plot_surface(xx, yy, z, rstride=1, cstride=1, color='green', alpha=0.3, edgecolor='non
e')
ax.set_xlabel("Feature 1")
ax.set_ylabel("Feature 2")
ax.set_zlabel("Decision Function")
ax.legend()
st.pyplot(fig)
```

HW3-3: Non-Circular 2D Dataset and Streamlit Modifications

Prompt Summary for HW3-3

1. **Dataset Expansion and Selector**:
   - Added **Moons, Spiral, and Elliptical Blobs** datasets, each with unique parameters.
   - Updated the dataset selector and parameter display in Streamlit to switch between Circular, Simple 1D, Moons, Spiral, and Elliptical Blobs datasets.

   Code Modification:

   ```python
   dataset_type = st.sidebar.selectbox("Choose Dataset Type", ["Circular (2D)", "Simple
   1D", "Moons", "Spiral", "Elliptical Blobs"])
   if dataset_type == "Moons":
       X, y = make_moons(n_samples=n_samples, noise=noise, random_state=42)
   elif dataset_type == "Spiral":
       X, y = generate_spiral(n_samples=n_samples, revolutions=revolutions, noise=noise)
   elif dataset_type == "Elliptical Blobs":
       X, y = make_blobs(n_samples=n_samples, centers=[(-3, -3), (3, 3)], cluster_std=[c
   luster_std_1, cluster_std_2], random_state=42)
   ```

2. **Decision Boundary Visualization in 2D with Contours**:
   - Added **contour plots** to show the decision boundary for Moons, Spiral, and Elliptical Blobs datasets.

   Code Modification:

   ```python
   elif plot_type == "2D":
       x_range = np.linspace(X[:, 0].min() - 0.5, X[:, 0].max() + 0.5, 100)
       y_range = np.linspace(X[:, 1].min() - 0.5, X[:, 1].max() + 0.5, 100)
       z = svm_model.decision_function(xy_mesh).reshape(xx.shape)
       ax.contourf(xx, yy, z, levels=[-1, 0, 1], alpha=0.2, colors=['blue', 'orange'])
       ax.contour(xx, yy, z, levels=[0], linewidths=2, colors='black')
   ```

3. **Error Handling for 1D Dataset**:
   - Addressed **IndexError** in Simple 1D dataset by creating a separate 1D visualization and adjusting the grid creation logic.

   Code Modification:

   ```python
   elif plot_type == "1D":
       fig, ax = plt.subplots(figsize=(10, 6))
       ax.scatter(X[y == 0], y[y == 0], color='blue', label='Class 0', edgecolor='k')
   ```