DS203 : Exercise 4

Nirav Bhattad

Last updated September 7, 2024

Pre-requisites

0.1. Importing Libraries

I imported the necessary libraries for this assignment, like `pandas` for data manipulation, `numpy` for numerical operations, `matplotlib` for plotting, and `sklearn` for machine learning models and metrics. I also imported `os` to create directories for saving the plots and metrics.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, accuracy_score, roc_auc_score, roc_curve
import os
```

Listing 1: Importing Libraries

0.2. Creating Directories

Here, I created two directories, `Images` and `Metrics`, to save the plots and metrics generated in the subsequent steps.

```
if not os.path.exists('Images'):
    os.makedirs('Images')

if not os.path.exists('Metrics'):
    os.makedirs('Metrics')
```

Listing 2: Creating Directories

0.3. Loading the Data

I loaded the data from the 3 CSV files into `pandas DataFrame`. Each file contains 1440 rows and 3 columns. First 2 columns give the location of the point, and the last column gives the class of the point.

```
data_v0 = pd.read_csv('clusters-4-v0.csv')
data_v1 = pd.read_csv('clusters-4-v1.csv')
data_v2 = pd.read_csv('clusters-4-v2.csv')
```

Listing 3: Loading the Data

Step 1

Step 1

Divide each data set into 'train' and 'test' datasets, once, and use them for all subsequent steps.

To divide the data into 'train' and 'test' datasets, I used the `train_test_split` method from the `sklearn.model_selection` module. I divided the data into 80% training and 20% testing datasets. I used the same split for all the subsequent steps.

```
x_train_v0, x_test_v0, y_train_v0, y_test_v0 = train_test_split(data_v0[['x1','x2']], data_v0['y'],
    test_size=0.2, random_state=42)
x_train_v1, x_test_v1, y_train_v1, y_test_v1 = train_test_split(data_v1[['x1','x2']], data_v1['y'],
    test_size=0.2, random_state=42)
x_train_v2, x_test_v2, y_train_v2, y_test_v2 = train_test_split(data_v2[['x1','x2']], data_v2['y'],
    test_size=0.2, random_state=42)
```

Listing 4: Dividing the Data into Train and Test Datasets

Step 2

Step 2

Review the data using appropriate plots and understand the overall structure of the data. Comment on the data and anticipate how well `LogisticRegression` will perform on the data.

This is how I plotted the 3 datasets

```
datasets = [(data_v0, "Dataset 1"),
            (data_v1, "Dataset 2"),
            (data_v2, "Dataset 3")]

for i, (data, label) in enumerate(datasets):
    plt.figure(figsize=(16, 10))

    plt.scatter(data['x1'], data['x2'], c=data['y'], cmap='viridis', s=10)
    plt.title(f'Scatter Plot of {label}')
    plt.xlabel('x1')
    plt.ylabel('x2')

    plt.tight_layout()
    plt.savefig(f'Images/dataset-{i+1}-overview.png', dpi=400)
    plt.show()

    class_counts = data['y'].value_counts()
    print(f"Class balance in {label}:\n{class_counts}\n")
```

Listing 5: Plotting the Datasets

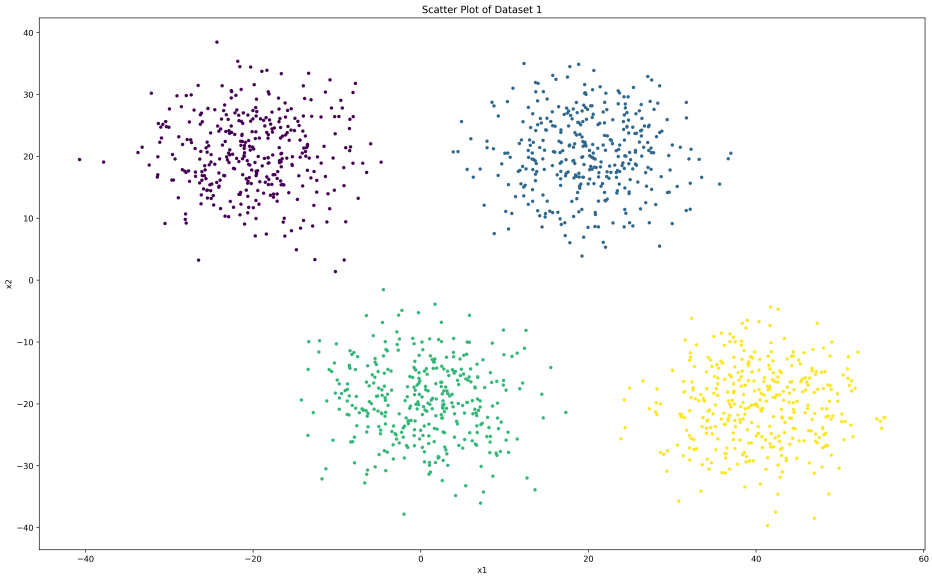

Figure 1: Overview of Data Set 1

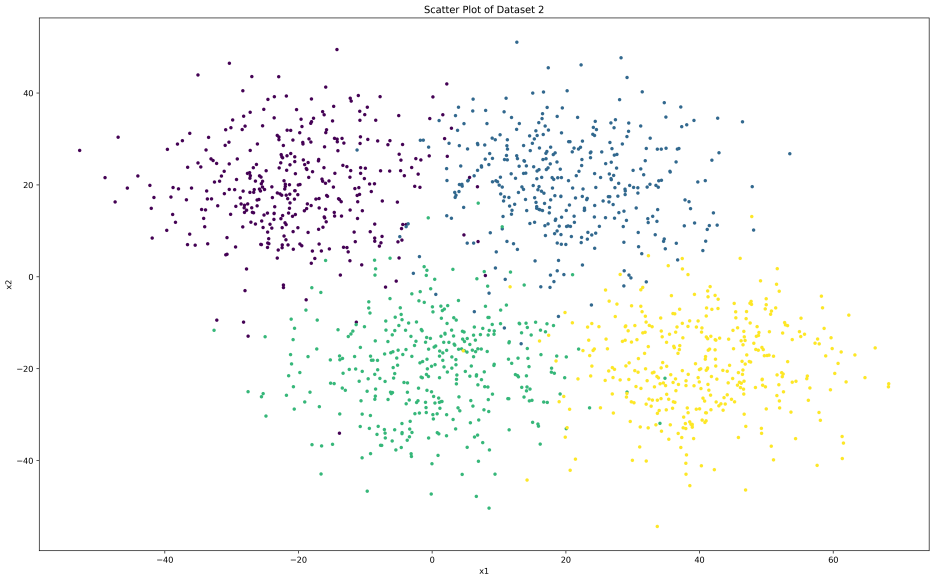

Figure 2: Overview of Data Set 2

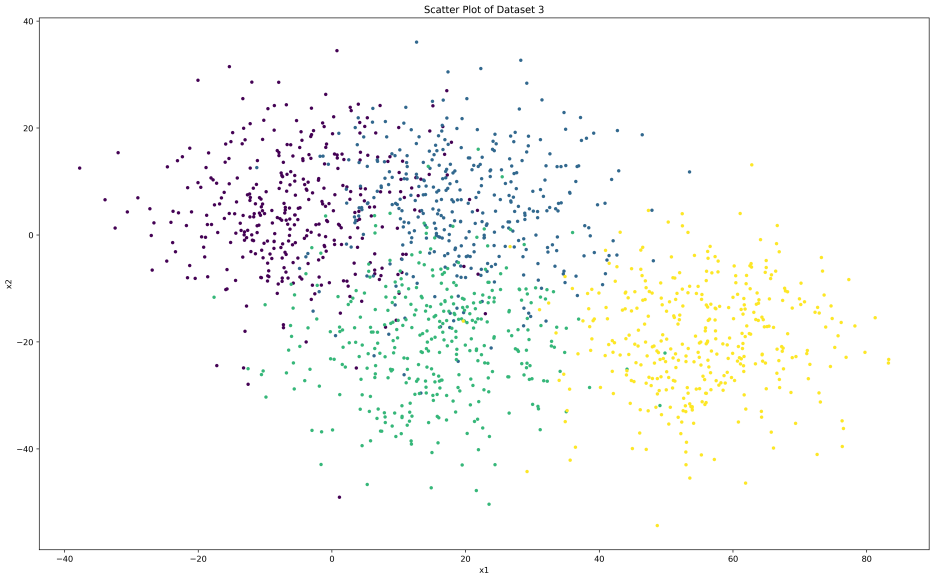

Figure 3: Overview of Data Set 3

We see that in each dataset, there are 360 points. The data is balanced in all the datasets. The data in Dataset 1 is linearly separable, the data in Dataset 2 is not linearly separable, and the data in Dataset 3 is more complex than the data in Dataset 2.

This is how I plotted the training and testing datasets.

```
plt.figure(figsize=(16, 6))
plt.subplot(1, 2, 1)
plt.scatter(x_train_v0['x1'], x_train_v0['x2'], c=y_train_v0, cmap='viridis', s=10)
plt.title('Training data for Dataset 1')

plt.subplot(1, 2, 2)
plt.scatter(x_test_v0['x1'], x_test_v0['x2'], c=y_test_v0, cmap='viridis', s=10)
plt.title('Test data for Dataset 1')
plt.savefig('Images/dataset-1-train-test.png', dpi=400)
plt.show()

plt.figure(figsize=(16, 6))
plt.subplot(1, 2, 1)
plt.scatter(x_train_v1['x1'], x_train_v1['x2'], c=y_train_v1, cmap='viridis', s=10)
plt.title('Training data for Dataset 2')

plt.subplot(1, 2, 2)
plt.scatter(x_test_v1['x1'], x_test_v1['x2'], c=y_test_v1, cmap='viridis', s=10)
plt.title('Test data for Dataset 2')
plt.savefig('Images/dataset-2-train-test.png', dpi=400)
plt.show()

plt.figure(figsize=(16, 6))
plt.subplot(1, 2, 1)
plt.scatter(x_train_v2['x1'], x_train_v2['x2'], c=y_train_v2, cmap='viridis', s=10)
plt.title('Training data for Dataset 3')

plt.subplot(1, 2, 2)
plt.scatter(x_test_v2['x1'], x_test_v2['x2'], c=y_test_v2, cmap='viridis', s=10)
plt.title('Test data for Dataset 3')
plt.savefig('Images/dataset-3-train-test.png', dpi=400)
plt.show()
```

Listing 6: Plotting the Training and Testing Datasets

This is how the test and train datasets look for each data set.

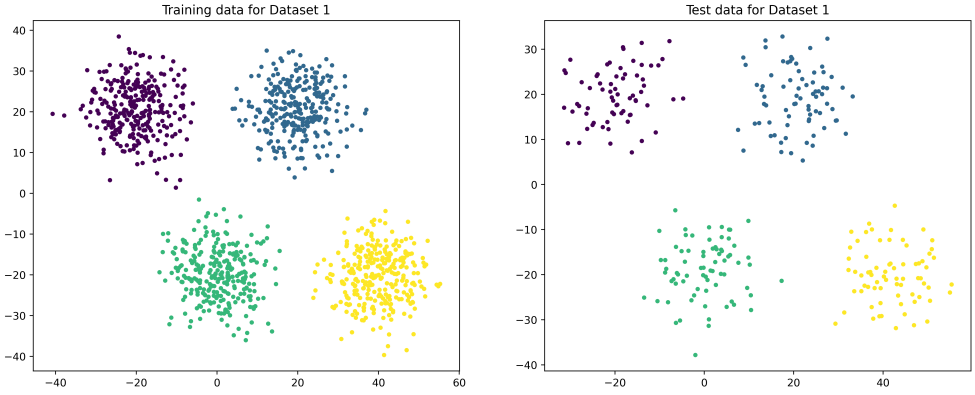

Figure 4: Training and Testing Datasets for Data Set 1

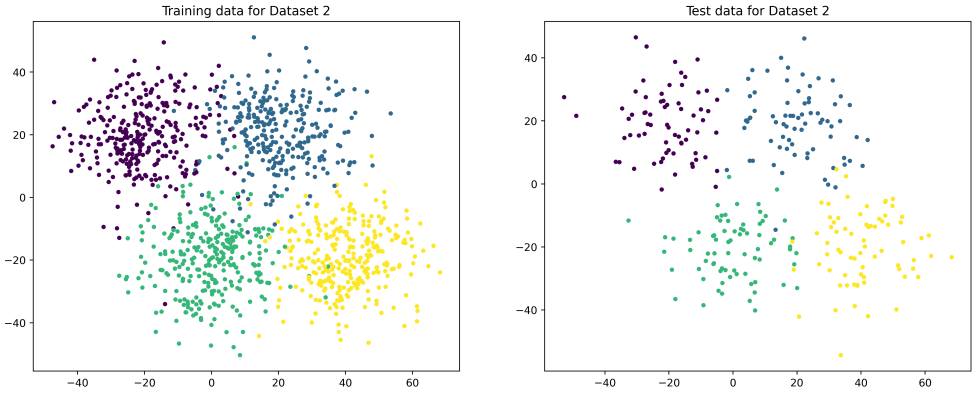

Figure 5: Training and Testing Datasets for Data Set 2

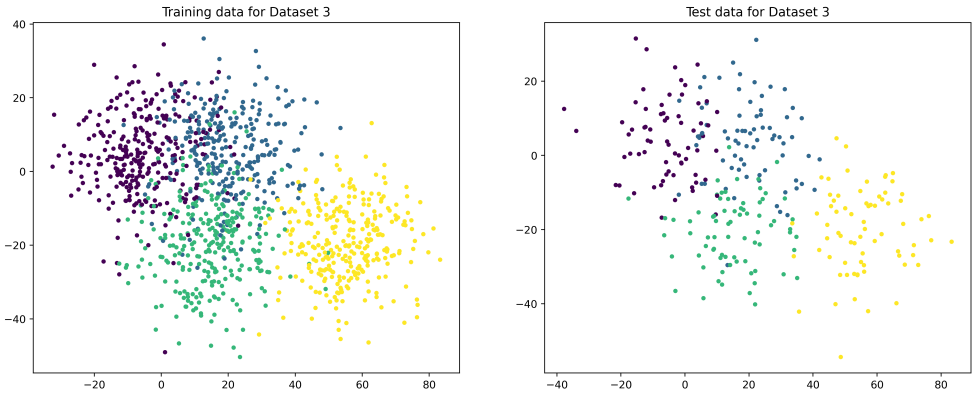

Figure 6: Training and Testing Datasets for Data Set 3

Reviewing the plots above, we can see that the training data and the test data are well balanced in all the 3 situations. The data is somewhat linearly separable in Dataset 1, but not so much in Dataset 2 and Dataset 3. The data in Dataset 3 is more complex than in Dataset 2.

Logistic Regression will perform well on Dataset 1 as it is linearly separable. It will perform poorly on Dataset 2 and Dataset 3 as they are not linearly separable. We can expect that the model will have a high accuracy on Dataset 1 and a low accuracy on Dataset 2 and Dataset 3.

Step 3

Step 3

Use the following algorithms / variants to process the datasets:

a. Logistic Regression
b. SVC – with '`linear`' kernel (what is '`linear`'?)
c. SVC – with '`rbf`' kernel (what is '`rbf`'?)
d. Random Forest Classifier – with `min_samples_leaf=1`
e. Random Forest Classifier – with `min_samples_leaf=3`
f. Random Forest Classifier – with `min_samples_leaf=5`
g. Neural Network Classifier – with `hidden_layer_sizes=(5)`
h. Neural Network Classifier – with `hidden_layer_sizes=(5,5)`
i. Neural Network Classifier – with `hidden_layer_sizes=(5,5,5)`
j. Neural Network Classifier – with `hidden_layer_sizes=(10)`