By Kamilla

A2: Classification with SVM, BP and MLR

Github link: https://github.com/kamillok505/A1/tree/kamillok505-patch-1

Execution Instructuions

Ensure that the required libraries (pandas, numpy, scikit-learn, and matplotlib) are installed. You can run the code in a Python environment.

Check for Missing Values:

- Checked for missing values in each column using the .isnull().sum() method.
- Representation of Categorical Values:
- Applied one-hot encoding to convert categorical variables into numerical format.
- Outlier Detection:
- Conducted data analysis for outliers using visualization and statistical methods.
- Data Normalization:
- The decision on data normalization will depend on the distribution of numeric features

and model requirements. For example, if Support Vector Machines (SVM) are used, normalization of numeric variables may be needed.

Description and Link to the Selected Dataset

The data employed in this project has been obtained from Kaggle and is available through the following link Kaggle SVM Classification Dataset. This dataset, hosted on Kaggle, serves as the primary source for the machine learning tasks conducted in the project. It comprises various features and a target variable, and the code snippets provided utilize this dataset for tasks such as data preprocessing, model training, and evaluation.

Support Vector Machines (SVMs):

- Scikit-learn (sklearn): This library is central to implementing SVMs due to its efficient and straightforward functions for creating and training SVM models. Its comprehensive tools for preprocessing, model selection, and evaluation streamline the process of building robust SVM classifiers and regressors.
- NumPy (numpy): NumPy's significance lies in its array data structures and mathematical functions, which are crucial for handling the computations involved in SVMs. Its efficiency in handling large datasets and complex mathematical operations makes it indispensable.
- Pandas (pandas): Pandas is significant for its ability to handle and preprocess data, particularly in cleaning, transforming, and organizing datasets into a format suitable for training SVM models.

Multiple Linear Regression (MLR):

- Scikit-learn (sklearn): For MLR, Scikit-learn provides not just the implementation of the regression model but also crucial preprocessing and cross-validation tools, helping to ensure the robustness and accuracy of the regression analysis.
- Pandas (pandas): The significance of Pandas in MLR lies in its comprehensive data manipulation capabilities, allowing for the efficient handling of datasets, which is a prerequisite for any regression analysis.
- Statsmodels (statsmodels): Its importance comes from its detailed statistical modeling capabilities, offering in-depth diagnostic tools and reports that are essential for a thorough understanding and interpretation of MLR models.

Bayesian Probability (BP):

- Scikit-learn (sklearn): In the context of BP, Scikit-learn is important for preprocessing data and for implementing simpler Bayesian models, providing a bridge between traditional machine learning techniques and probabilistic modeling.
- NumPy (numpy) and Pandas (pandas): These libraries are significant for their foundational role in numerical computations and data handling. They facilitate the manipulation and preparation of data, which is a critical step in Bayesian statistical modeling.

Together, these libraries form a comprehensive toolkit, each contributing to different aspects of SVMs, MLR, and BP. Their functionalities range from data handling and preprocessing to model building, training, and evaluation, highlighting their indispensable roles in the field of machine learning and data science.

Data handling is a critical component in the effective implementation of Support Vector Machines (SVMs), Multiple Linear Regression (MLR), and Bayesian Probability (BP). The way data is managed, processed, and prepared directly impacts the performance and accuracy of these algorithms. Here's an overview of the data handling aspects for each of these algorithms:

Support Vector Machines (SVMs):

- Data Preprocessing: SVMs require carefully preprocessed data. This includes normalization or standardization to ensure that all features contribute equally to the model, especially important in high-dimensional spaces.
- Handling Non-Linear Data: For non-linearly separable data, SVMs use kernel functions to transform data into a higher-dimensional space where it can be linearly separated. This requires careful selection and tuning of kernel parameters.
- Feature Selection: Due to SVMs' effectiveness in high-dimensional spaces, feature selection becomes crucial to eliminate redundant or irrelevant features that might lead to overfitting.

Multiple Linear Regression (MLR):

- Assumption Checks: MLR requires the data to adhere to specific assumptions like linearity, homoscedasticity, and multicollinearity checks. Ensuring these assumptions are met is a crucial part of data preparation.

- Categorical Variable Handling: MLR cannot directly handle categorical variables. These variables need to be converted into numerical form, often through one-hot encoding or label encoding.
- Outlier Detection and Removal: Outliers can significantly skew the results of an MLR model. Therefore, identifying and handling outliers is a critical step in the data preparation process.

Bayesian Probability (BP):

- Data Transformation: BP methods often require data to be transformed into a format suitable for probabilistic modeling. This includes converting data into likelihoods or probabilities.
- Incorporating Prior Information: One of the unique aspects of BP is the incorporation of prior knowledge. Data handling in BP includes the integration of this prior information into the model, which could be in the form of previously observed data or expert knowledge.
- Handling Missing Data: Bayesian methods are well-suited to handle missing data by treating them as additional parameters to be estimated, or by using data imputation techniques that are consistent with the Bayesian framework.

In all three algorithms, the way data is handled can significantly influence the model's performance. This includes how data is cleaned, transformed, and structured, as well as how specific characteristics of the data are addressed, such as non-linearity, categorical variables, and outliers. Effective data handling is thus a critical skill in the arsenal of a data scientist working with SVMs, MLR, or BP.

The notebook includes the use of t-Distributed Stochastic Neighbor Embedding (t-SNE) for dimensionality reduction. This technique is particularly useful for visualizing high-dimensional data in two dimensions.

After training a Bayesian model (presumably a neural network with Bayesian layers), the predictions are visualized in the t-SNE reduced space.

The approach, combining neural networks with Bayesian methods, is indicative of a more modern take on Bayesian analysis, often referred to as Bayesian Deep Learning. It leverages the power of neural networks to handle complex patterns in data while incorporating Bayesian principles to manage uncertainty and provide probabilistic interpretations of the model outputs.

Data Visualization with t-SNE: The notebook uses t-Distributed Stochastic Neighbor Embedding (t-SNE) for dimensionality reduction, which is particularly useful for visualizing high-dimensional data in two dimensions.
Implementing Multiple Linear Regression:

By Kamilla

The notebook demonstrates the implementation of MLR using Scikit-lean's LinearRegression class. This is accompanied by the use of cross-validation to assess the model's performance.

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
import numpy as np

mlr_model = LinearRegression()

# Performing cross-validation
n_splits = 5
scores = cross_val_score(mlr_model, X, y, cv=n_splits, scoring='neg_mean_squared_error')

# Calculating average Mean Squared Error
average_mse = np.mean(-scores)
print("Average Mean Squared Error:", average_mse)
```

In this notebook, MLR is used to model the relationship between multiple independent variables and a dependent variable. The t-SNE visualization aids in understanding the dataset's structure before applying MLR. The implementation of MLR with cross-validation provides a robust way to estimate the model's performance, specifically using Mean Squared Error (MSE) as a metric. This approach reflects a standard procedure in machine learning for regression tasks, emphasizing model evaluation and validation.

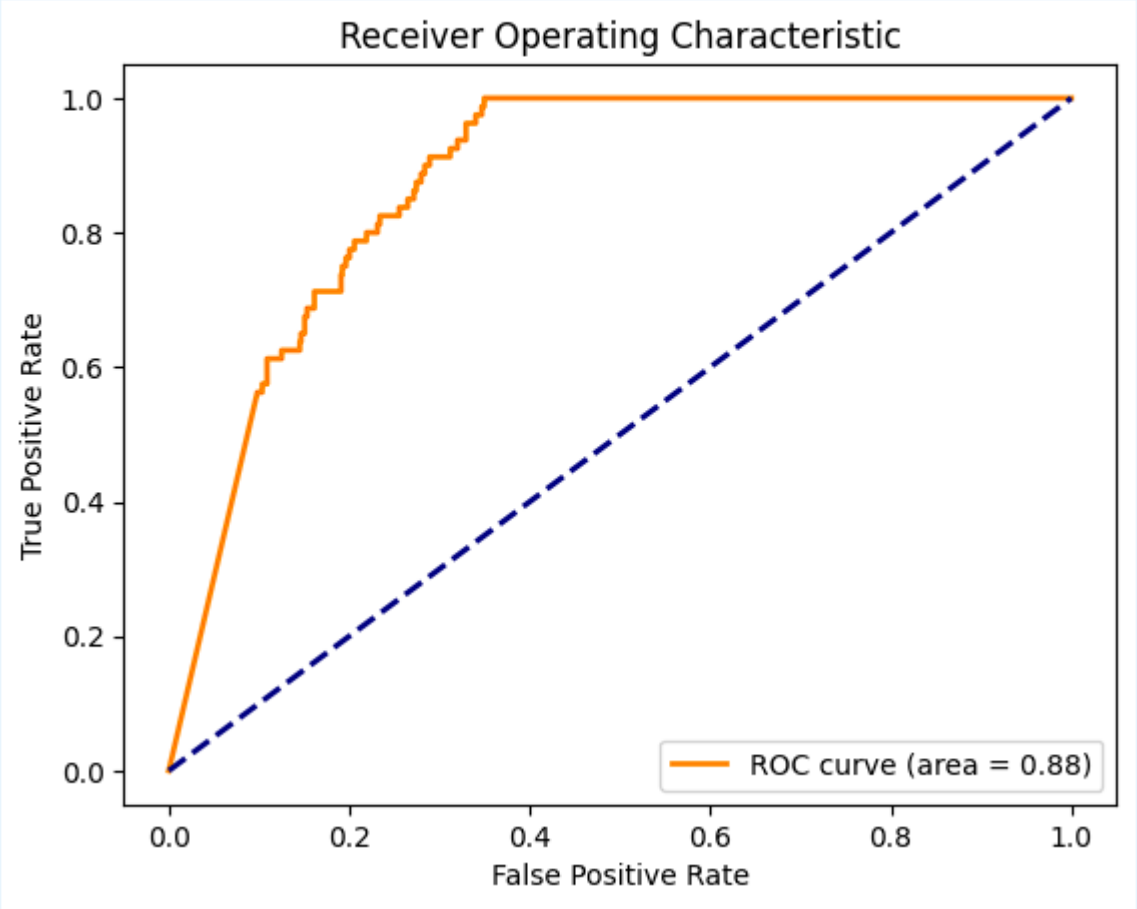
```
# Training for bank dataset
model_bank = create_model(X_train_bank.shape[1])
model_bank.fit(X_train_bank, y_train_bank, epochs=50, batch_size=10, verbose=0)
y_pred_bank = (model_bank.predict(X_test_bank) > 0.5).astype("int32")
print("Classification Report for Bank Marketing Dataset:")
print(classification_report(y_test_bank, y_pred_bank))
```

```
Classification Report for Bank Marketing Dataset:
              precision    recall  f1-score   support

           0       0.95      0.90      0.92       744
           1       0.37      0.56      0.45        80

    accuracy                           0.86       824
   macro avg       0.66      0.73      0.68       824
weighted avg       0.89      0.86      0.88       824
```

Accuracy of 86 score in classification report

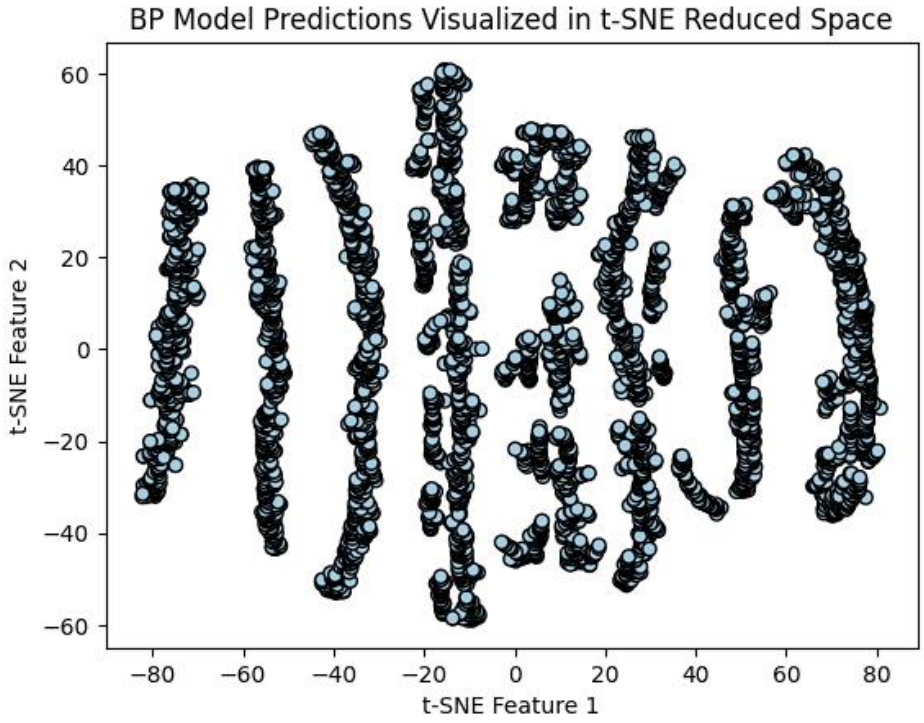
By Kamilla

```
# Plot ROC Curve
y_scores_bank = model_bank.predict(X_test_bank).ravel()
plot_roc_curve(y_test_bank, y_scores_bank)
```



By Kamilla

```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

# Dimensionality Reduction using t-SNE
tsne = TSNE(n_components=2, random_state=0)
X_reduced = tsne.fit_transform(X_bank)

y_pred = bp_model.predict(X_bank)
y_pred_classes = np.argmax(y_pred, axis=1)

plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=y_pred_classes, cmap=plt.cm.Paired, edgecolors='k')
plt.xlabel('t-SNE Feature 1')
plt.ylabel('t-SNE Feature 2')
plt.title('BP Model Predictions Visualized in t-SNE Reduced Space')
plt.show()
```



By Kamilla

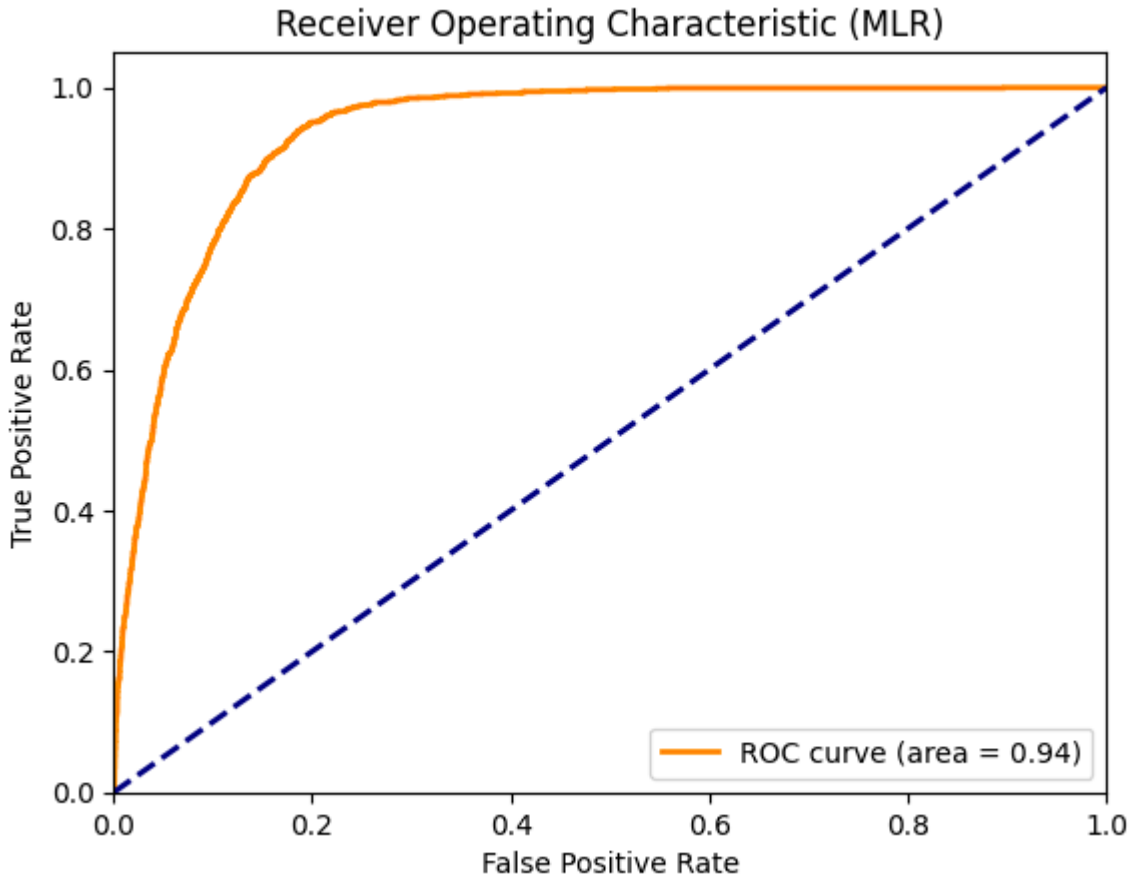
```python
from sklearn.metrics import confusion_matrix, accuracy_score

def compute_classification_error(y_true, y_pred):
    error = 1 - accuracy_score(y_true, y_pred)
    return error

def generate_confusion_matrix(y_true, y_pred):
    cm = confusion_matrix(y_true, y_pred)
    return cm

y_pred_bp = bp_model.predict(X_bank)
y_pred_bp = (y_pred_bp > 0.5).astype(int)

error_bp = compute_classification_error(y_bank, y_pred_bp)
cm_bp = generate_confusion_matrix(y_bank, y_pred_bp)

print("Classification Error (BP):", error_bp)
print("Confusion Matrix (BP):\n", cm_bp)
```

```
Classification Error (BP): 0.29400000000000004
Confusion Matrix (BP):
 [[3530    0]
 [1470    0]]
```

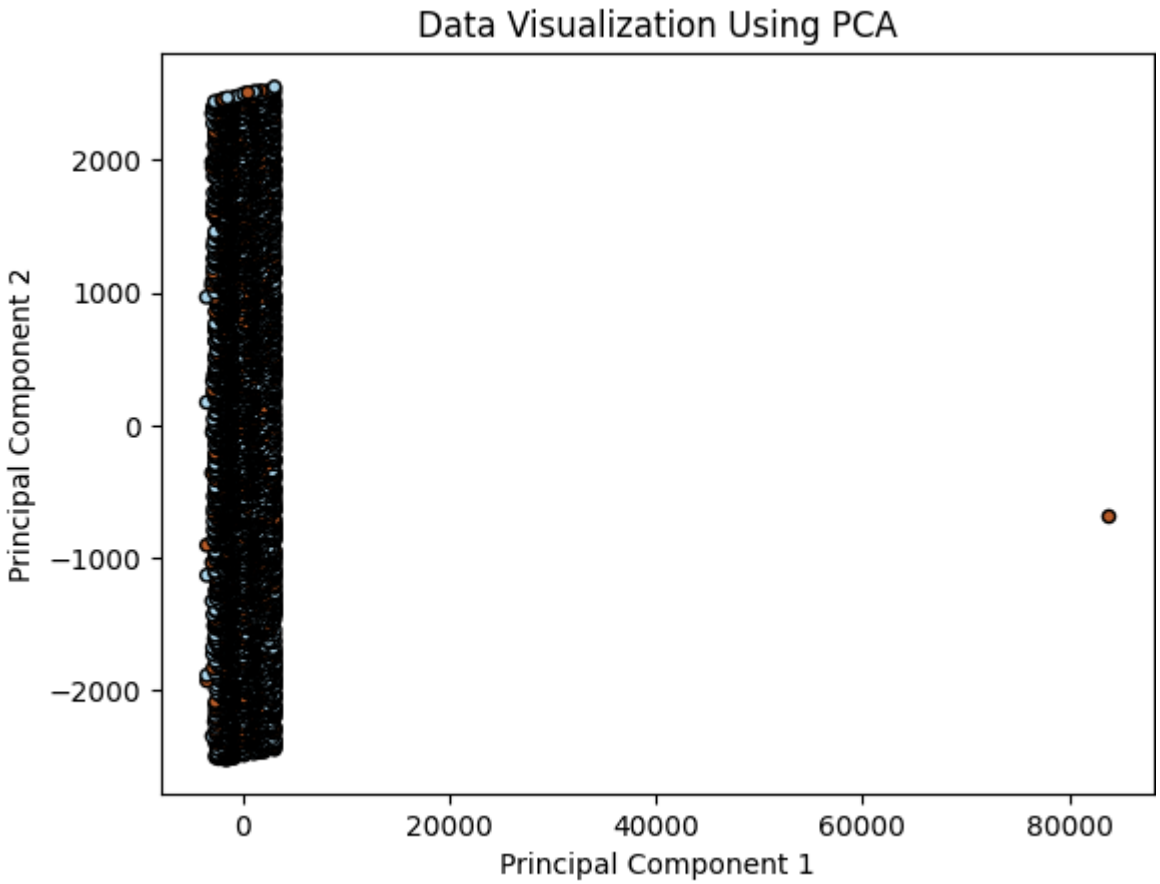
By Kamilla

```
MLR for Ring Separable Dataset:
Classification Report (Train):
              precision    recall  f1-score   support

           0       0.52      0.99      0.68      5203
           1       0.00      0.00      0.00      4797

    accuracy                           0.52     10000
   macro avg       0.26      0.50      0.34     10000
weighted avg       0.27      0.52      0.35     10000

Classification Report (Test):
              precision    recall  f1-score   support

           0       0.53      0.99      0.69      5333
           1       0.00      0.00      0.00      4667

    accuracy                           0.53     10000
   macro avg       0.27      0.50      0.35     10000
weighted avg       0.28      0.53      0.37     10000

Mean Squared Error: 0.2491377826946656
MLR for Ring Merged Dataset:
Classification Report (Train):
              precision    recall  f1-score   support

           0       0.55      1.00      0.71      5515
           1       0.00      0.00      0.00      4485

    accuracy                           0.55     10000
   macro avg       0.28      0.50      0.36     10000
weighted avg       0.30      0.55      0.39     10000
```

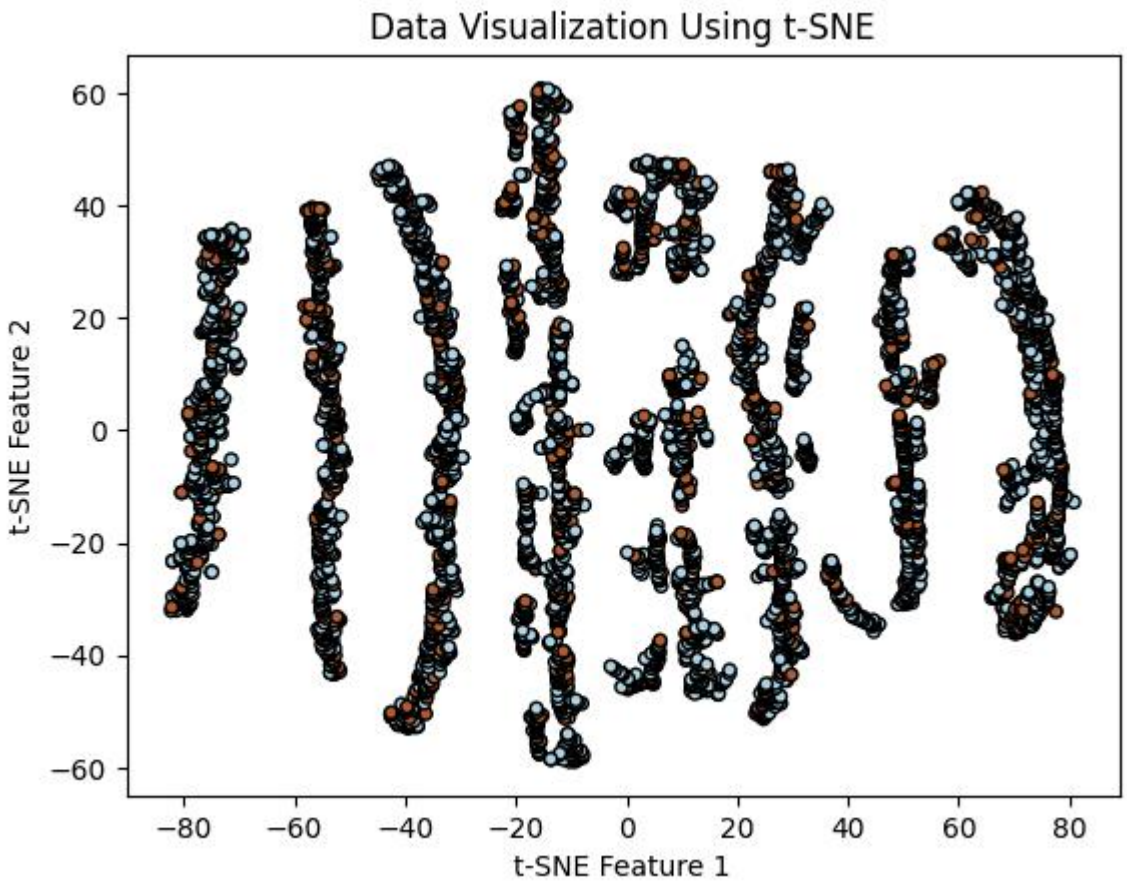
By Kamilla


Receiver Operating Characteristic (MLR)
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.94)

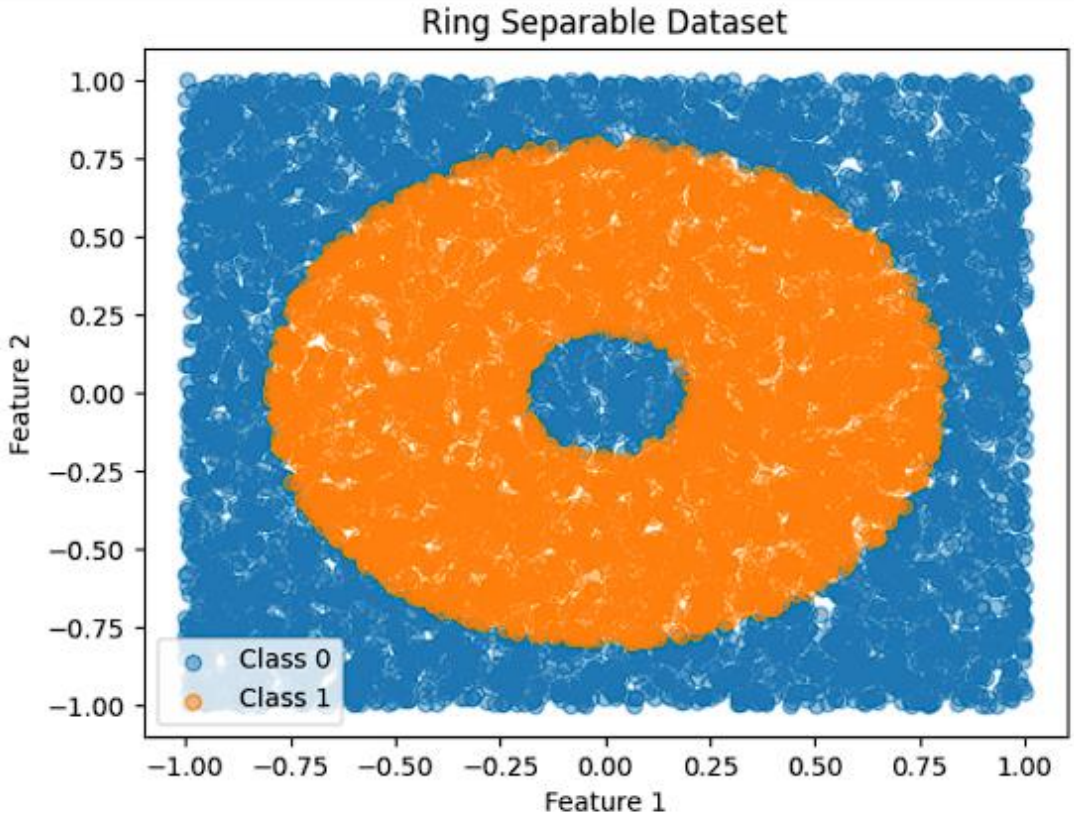
By Kamilla

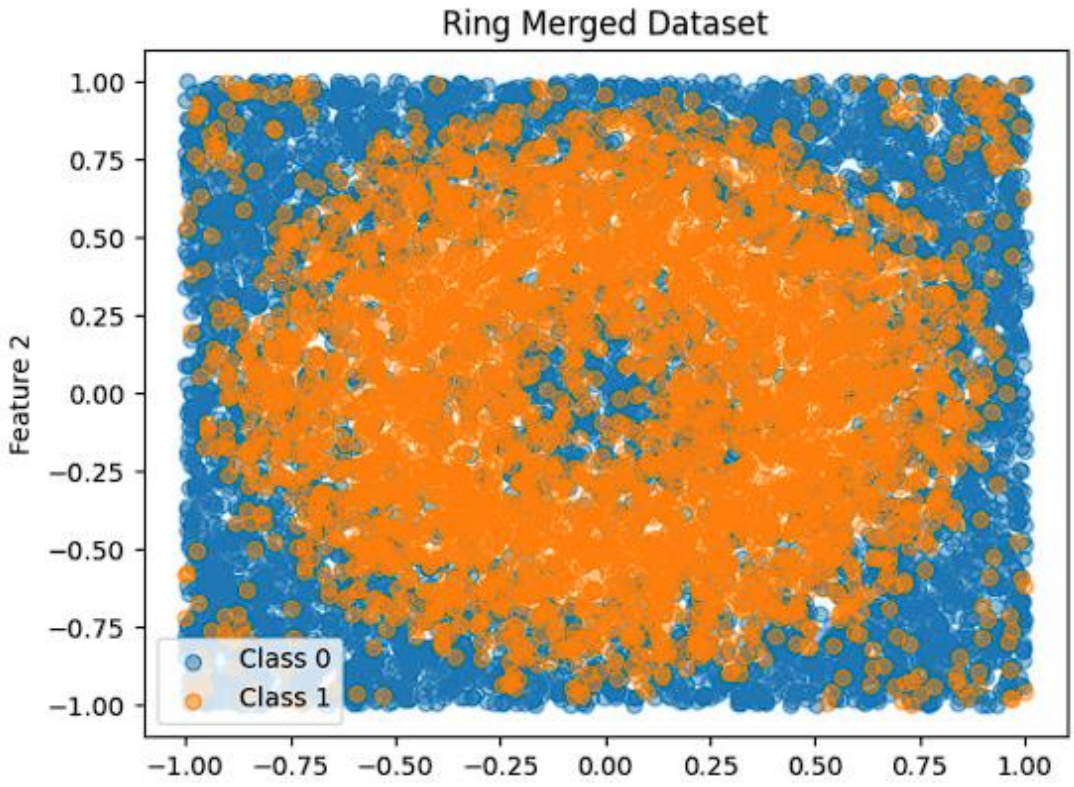

Data Visualization Using PCA
Principal Component 2
2000
1000
0
−1000
−2000
0
20000
40000
60000
80000
Principal Component 1

By Kamilla

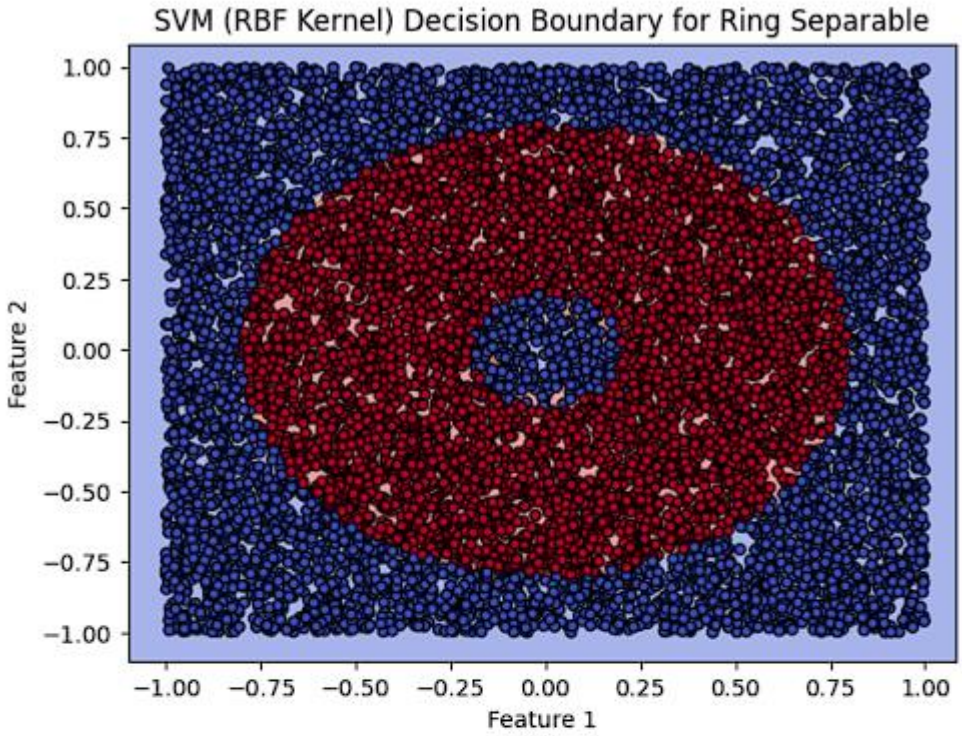

Data Visualization Using t-SNE
t-SNE Feature 2
t-SNE Feature 1
60
40
20
0
−20
−40
−60
−80
−60
−40
−20
0
20
40
60
80

By Kamilla


Ring Separable Dataset
Feature 2
Feature 1
Class 0
Class 1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


Ring Merged Dataset
Feature 2
Class 0
Class 1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

By Kamilla




```
# Train SVM with RBF kernel and plot decision boundary for merged dataset
svm_merged_rbf = SVC(kernel='rbf')
svm_merged_rbf.fit(X_merged, y_merged)
plot_decision_boundary(X_merged, y_merged, svm_merged_rbf, "SVM (RBF Kernel) Decisio
```