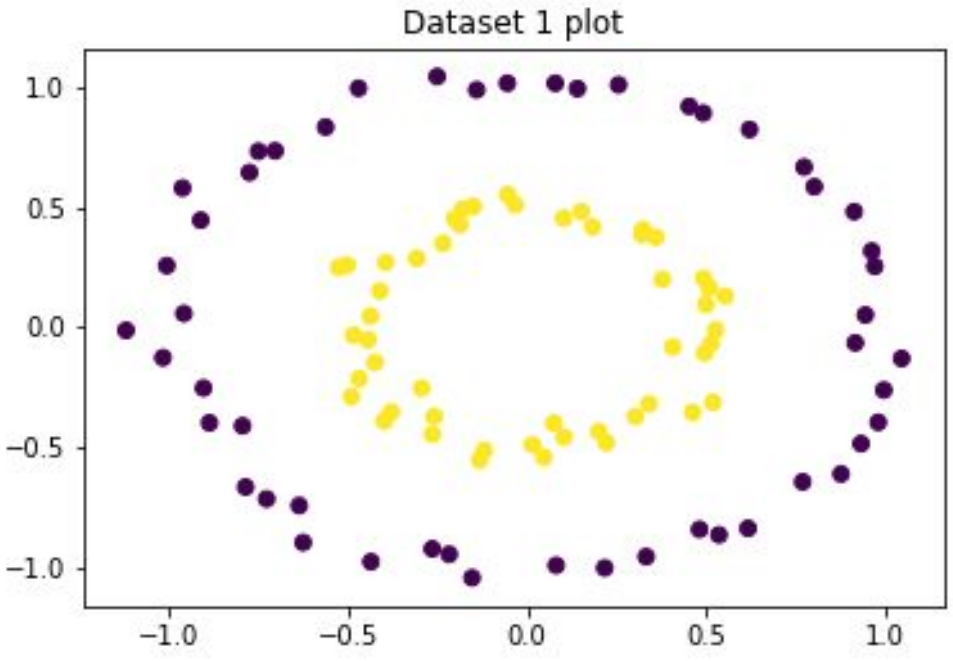
Class and functions defined by me:

- Class MySVM: uses sklearn's .fit but .predict is defined by me which takes input features as input and returns the output class corresponding to those features.
- def confusion_matrix(y_true, y_pred): returns confusion matrix plot
- def roc_plot(y_test, preds): returns roc plot
- def quad(X, Y): custom polynomial kernel with degree =2
- def accuracy(y_pred, y_true): returns accuracy
- def rbf_kernel(X, gamma): custom rbf kernel
- def plot_decision_boudary(clf, X, y, kernel=None, gamma=1): plots decision boundary
- def f1_score(y_true, y_pred): return f1 score
- def one_vs_all(X, y): trains the classifier according to one vs all approach
- def one_vs_one(X, y, kernel=None, gamma=1): trains the classifier according to one vs one approach

i) Explore the datasets, plot them and write your observations and findings of the datasets. Particularly, comment on the number of samples, balance among classes, separability and noise in the dataset.

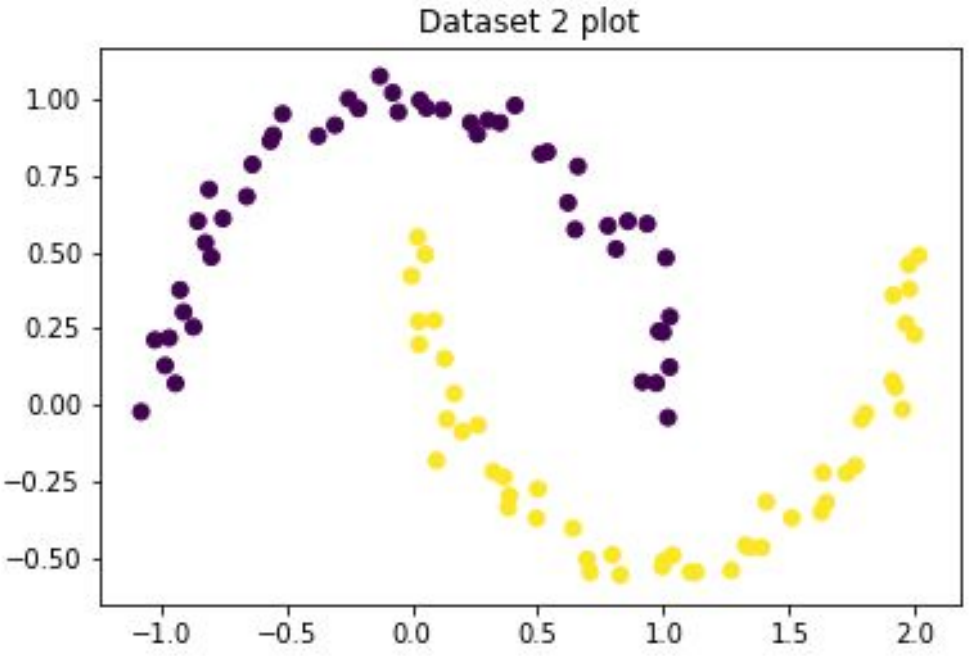
Dataset 1:



Number of samples: 100
Number of classes: 2
Distribution of samples among classes: [50 50]

As it can be seen that how the data is distributed among classes, the number of samples are enough to classify the dataset, and can be seperated using polynomial kernel with degree 2. Both classes has same number of samples which is good because the model can learn the distribution more effectively for each class. Hence, there is a very good balance among classes. From the plot, it can be seen that there is no noise (or outlier) which could cause the problem here, One can easily classify the dataset using non-linear boundary as mentioned earlier.

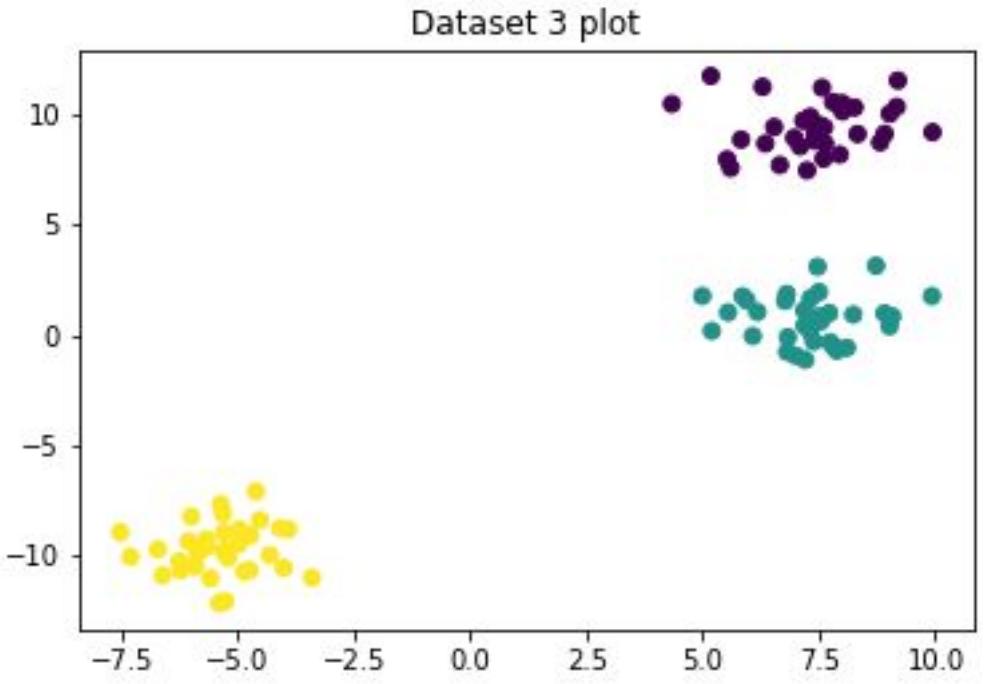
Dataset 2:



Number of samples: 100
Number of classes: 2
Distribution of samples among classes: [50 50]

As it can be seen that how the data is distributed among classes, the number of samples are enough to classify the dataset, and can be seperated using non-linear decision boundary. Both classes has same number of samples which is good because the model can learn the distribution more effectively for each class. Hence, there is a very good balance among classes. From the plot, it can be seen that there is no noise (or outlier) which could cause the problem here, One can easily classify the dataset using rbf kernel.

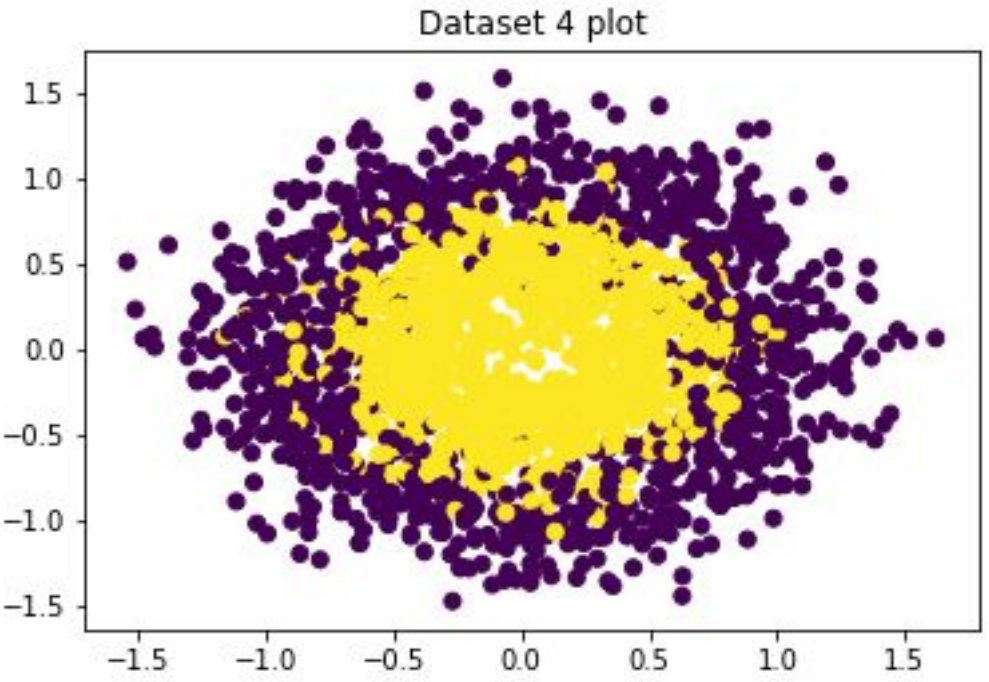
Dataset 2:



Number of samples: 100
Number of classes: 3
Distribution of samples among classes: [34 33 33]

As it can be seen that how the data is distributed among classes, the number of samples are enough to classify the dataset, and can be classified using one-vs-all or one-vs-one technique as it is multi-class problem. Each classes has almost same number of samples which is good because the model can learn the distribution more effectively for each class. Hence, there is a very good balance among classes. From the plot, it can be seen that there is no noise (or outlier) which could cause the problem here, One can easily seperate the dataset using linear decision boundary.

Dataset 4:

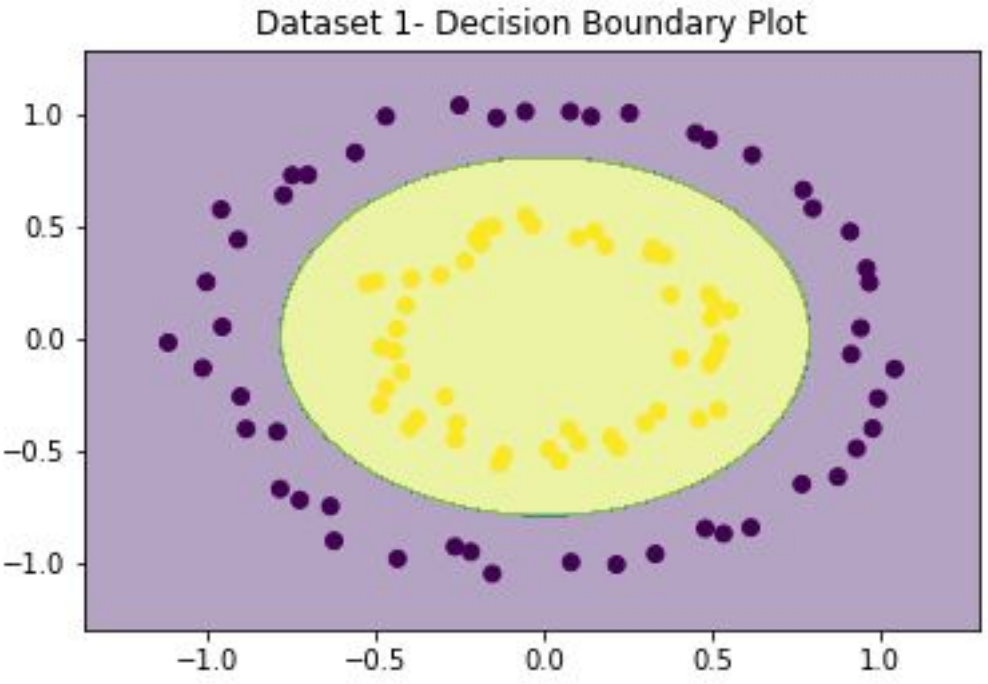


Number of samples: 2000
Number of classes: 2
Distribution of samples among classes: [1000 1000]

As it can be seen that how the data is distributed among classes, the number of samples are enough to classify the dataset. Both classes has same number of samples which is good because the model can learn the distribution more effectively for each class. Hence, there is a very good balance among classes. From the plot, it can be seen that **there is a lot of noise** (or outliers) in the dataset which would cause the problem here. Some Yellow data-points lie in the region of Purple Class, This dataset can be seen as the more complicated version of Dataset 1. One could seperate it using non-linear decision boundary but there would be high probability of misclassification in the training set as well as in the test set.

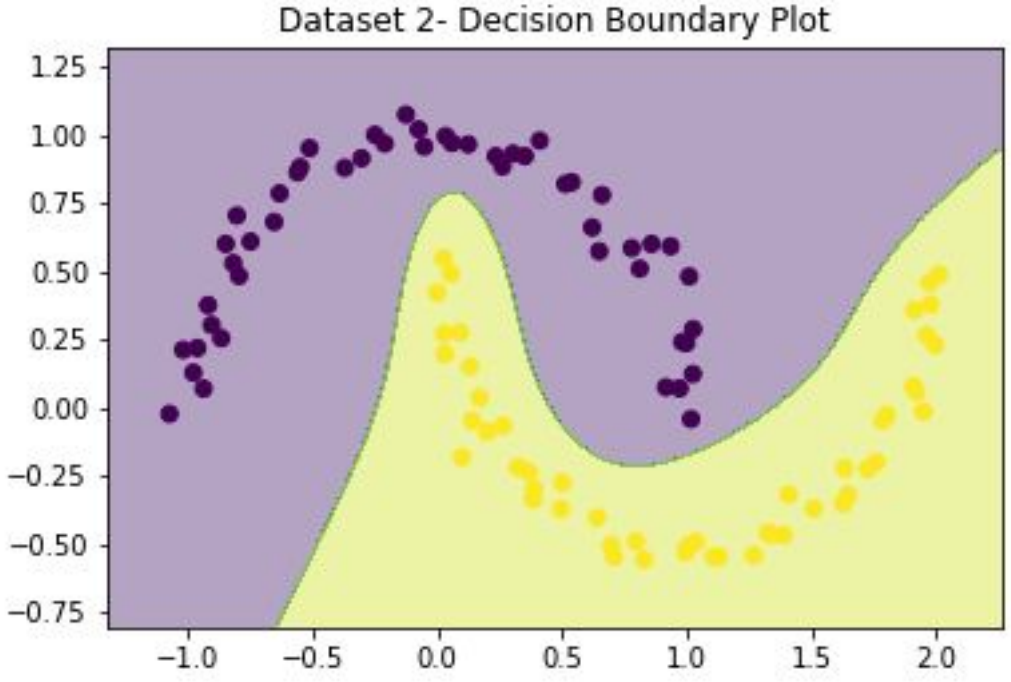
ii) **(20 points) For each dataset, write a kernel (if required) to make them linearly separable. Explain the choice of each kernel. Plot the datasets with decision boundaries corresponding to those kernels.**

Dataset 1:



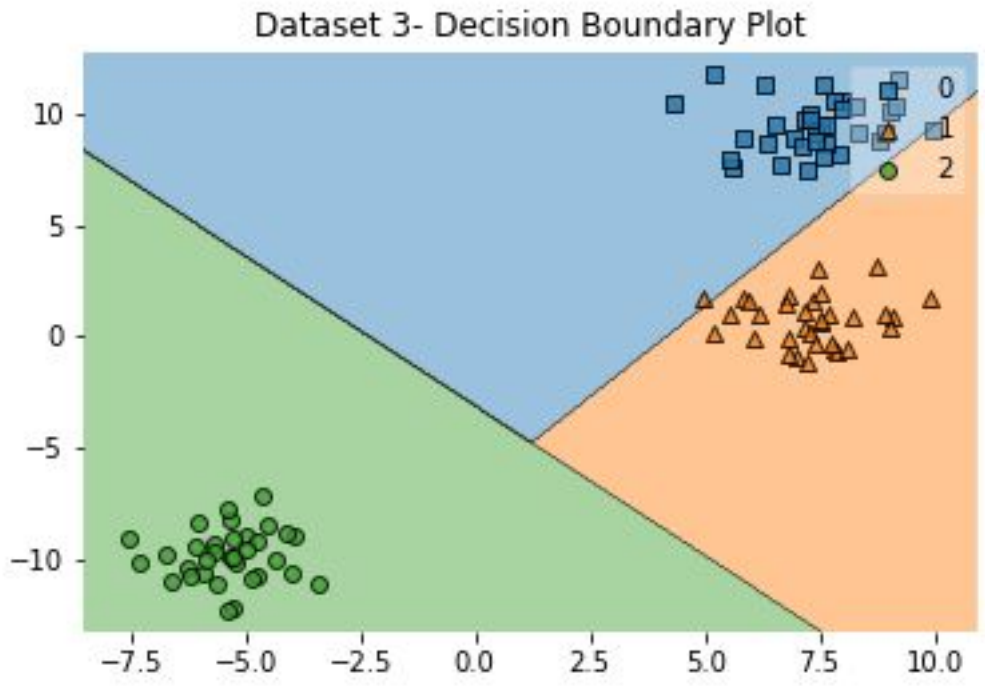
Choice of Kernel: I used polynomial kernel with degree = 2 to seperate the data because from the plot, it can be seen that one can seperate the data using the equation of an ellipse (which is a polynomial equation with degree = 2).

Dataset 2:



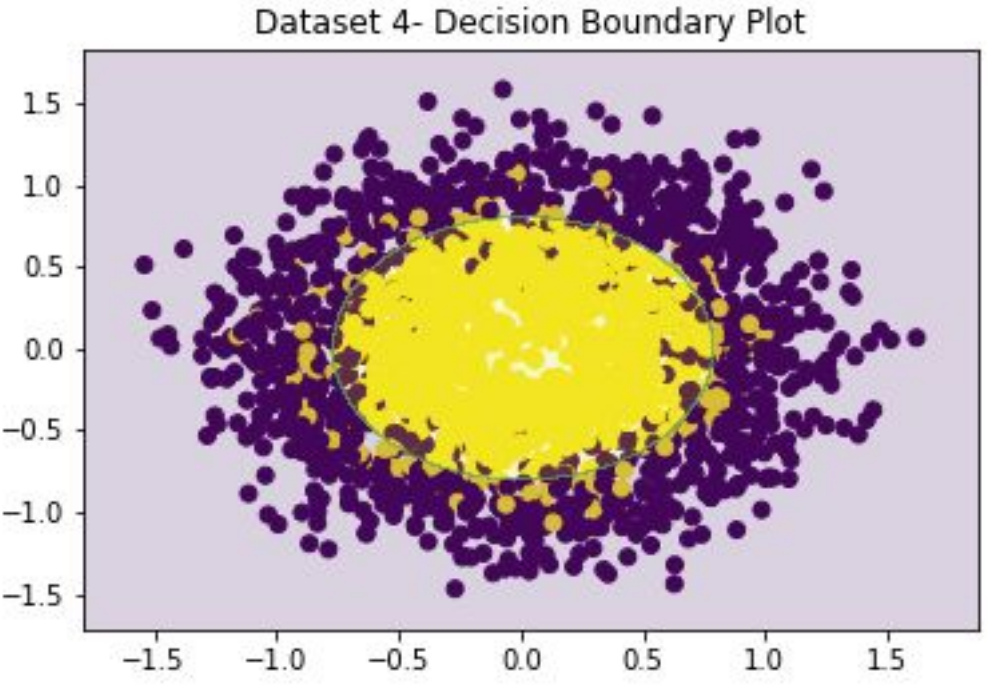
Choice of Kernel: I used rbf kernel with gamma = -4 to seperate the data because from the plot, it can be seen that one can seperate the data using the equation of an ellipse (which is a polynomial equation with degree = 2).

Dataset 3:



Choice of Kernel: I used linear kernel to seperate the data because from the plot, it can be seen that the data is linearly seperable. I used both All-vs-One and One-vs-One approach.

Dataset 4:

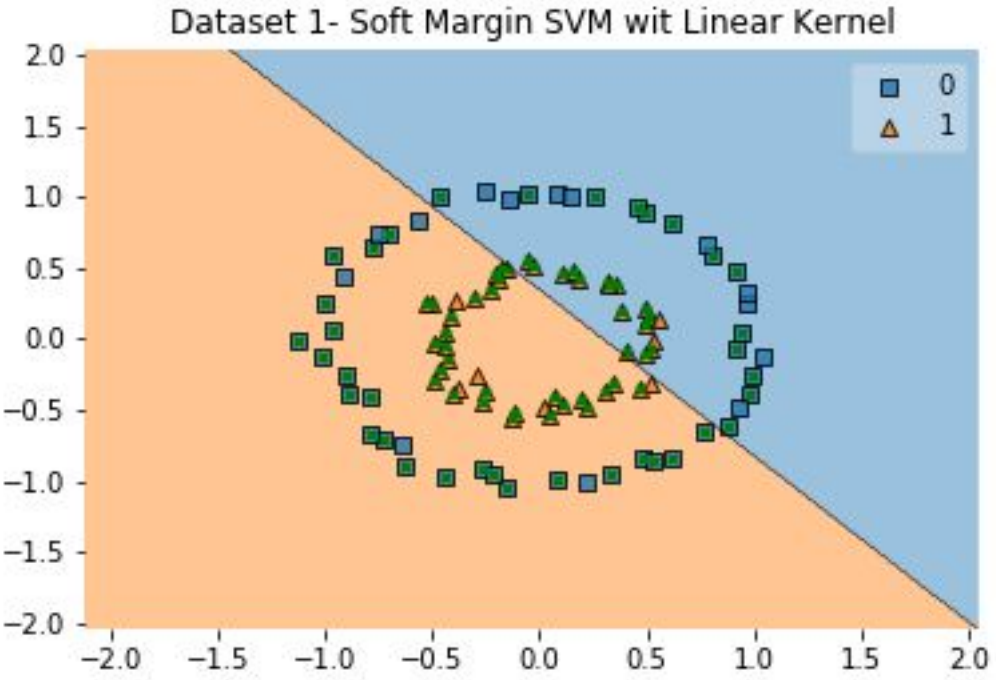


Choice of Kernel: As mentioned in the solution of question 1, there is a lot of noise (or outliers) in the dataset and this dataset can be seen as the more complicated version of Dataset 1. Hence, I used polynomial kernel with degree = 2.

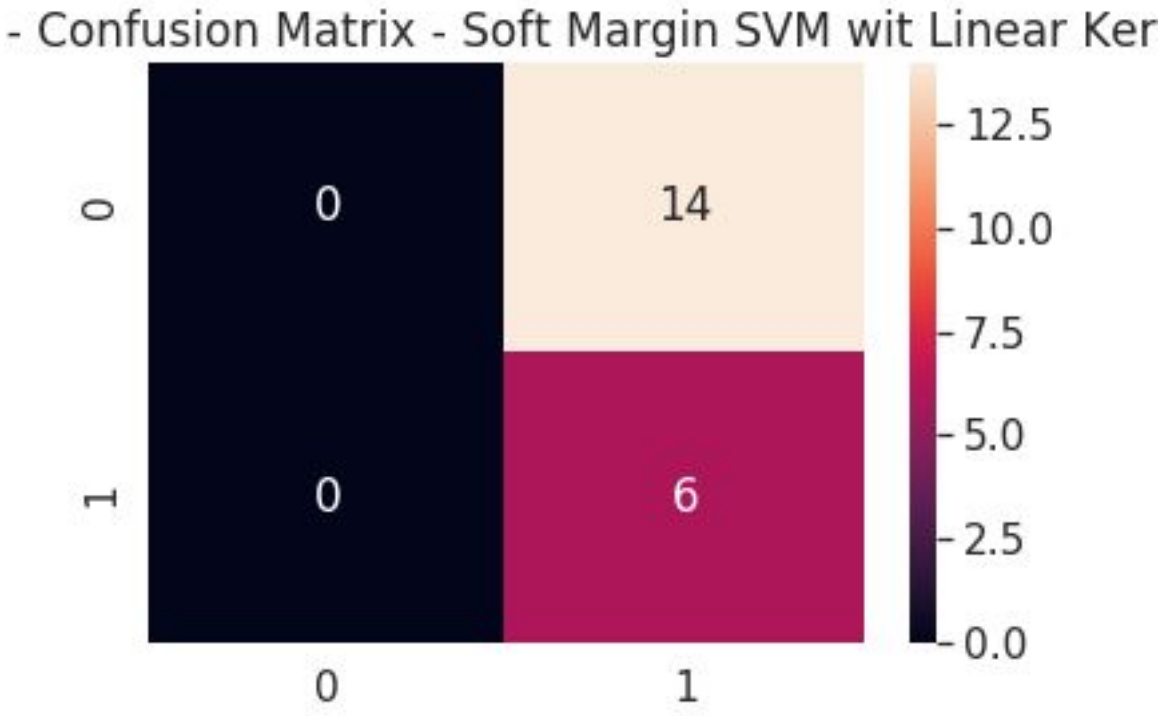
(iii) (27 points) Implement Soft margin SVM with linear kernel. Report the accuracy and F1 score on each dataset.

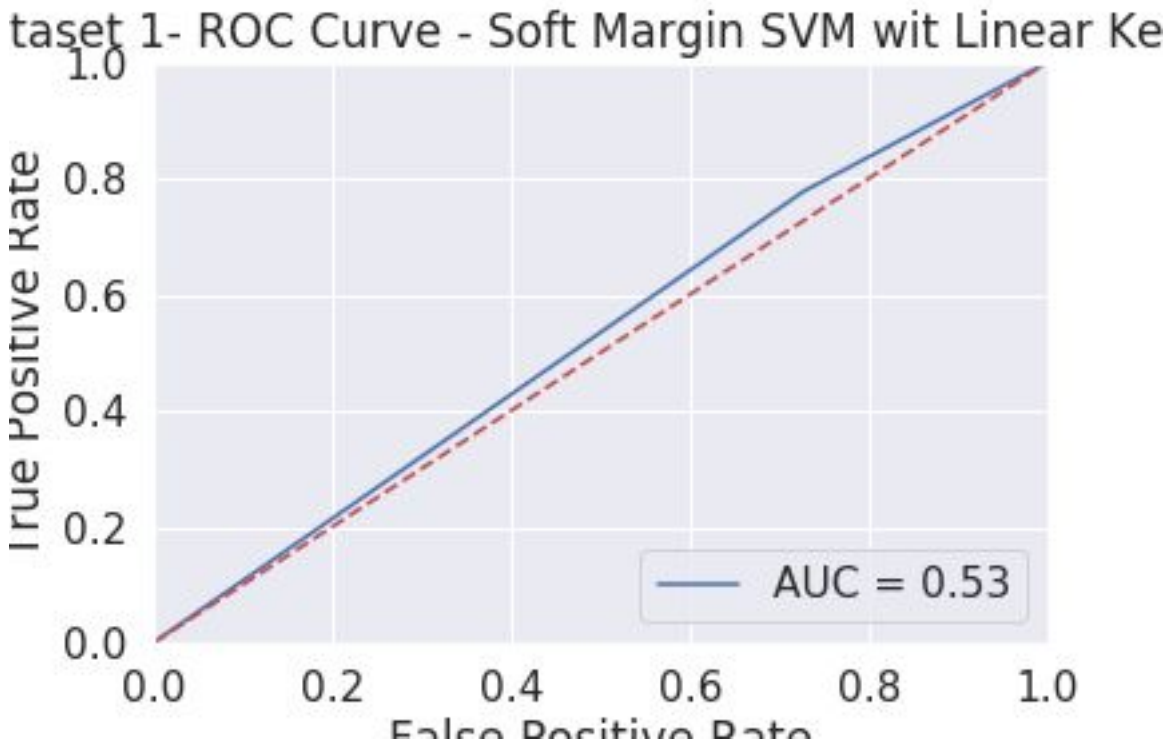
Note: Support Vectors are marked with green color.

Dataset 1:

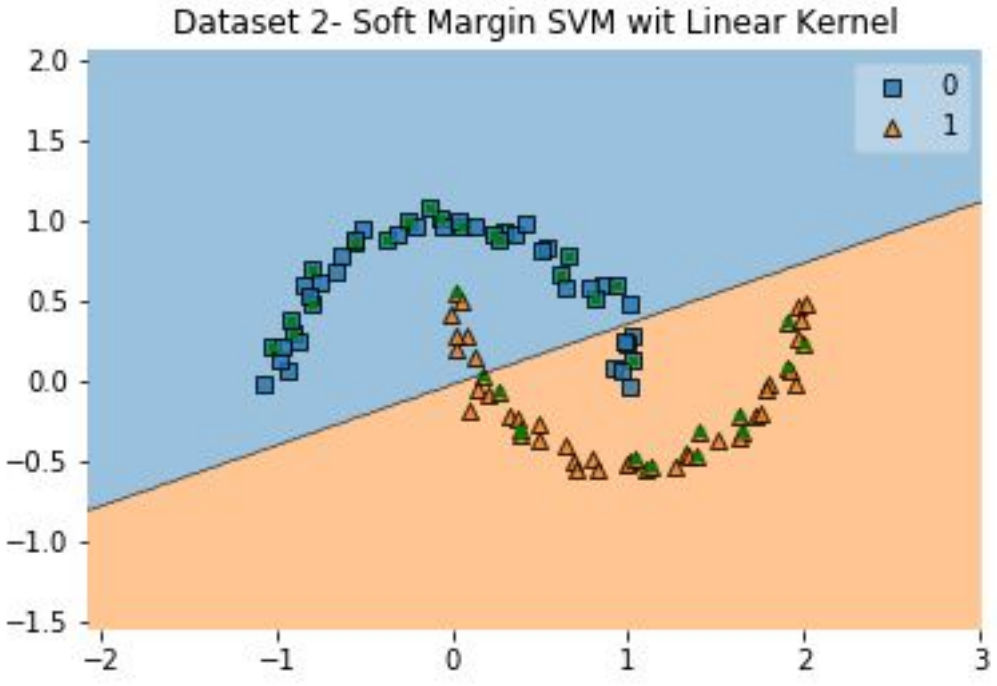


Accuracy: 0.4
F1 Score: 1.0

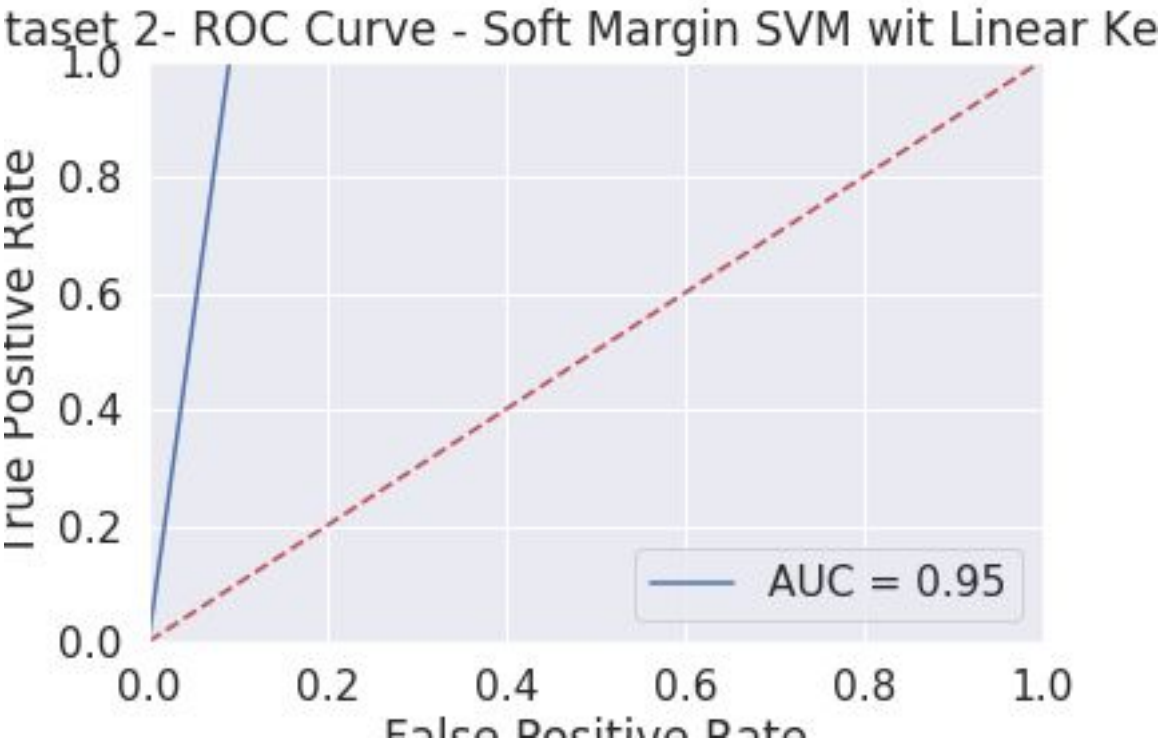




Dataset 2:

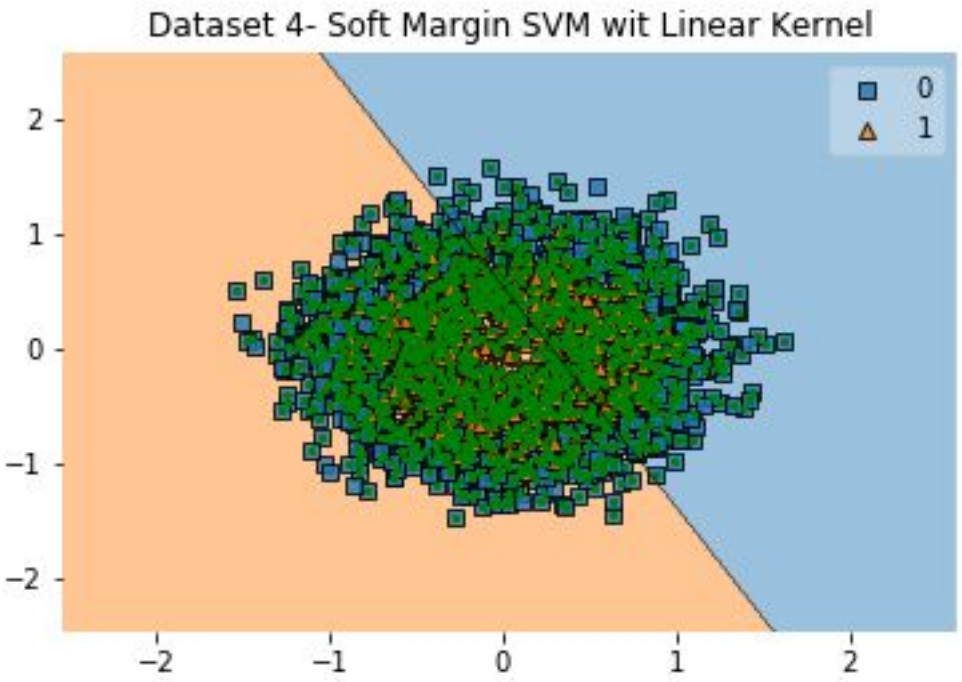


Accuracy: 0.9
F1 Score: 1.0

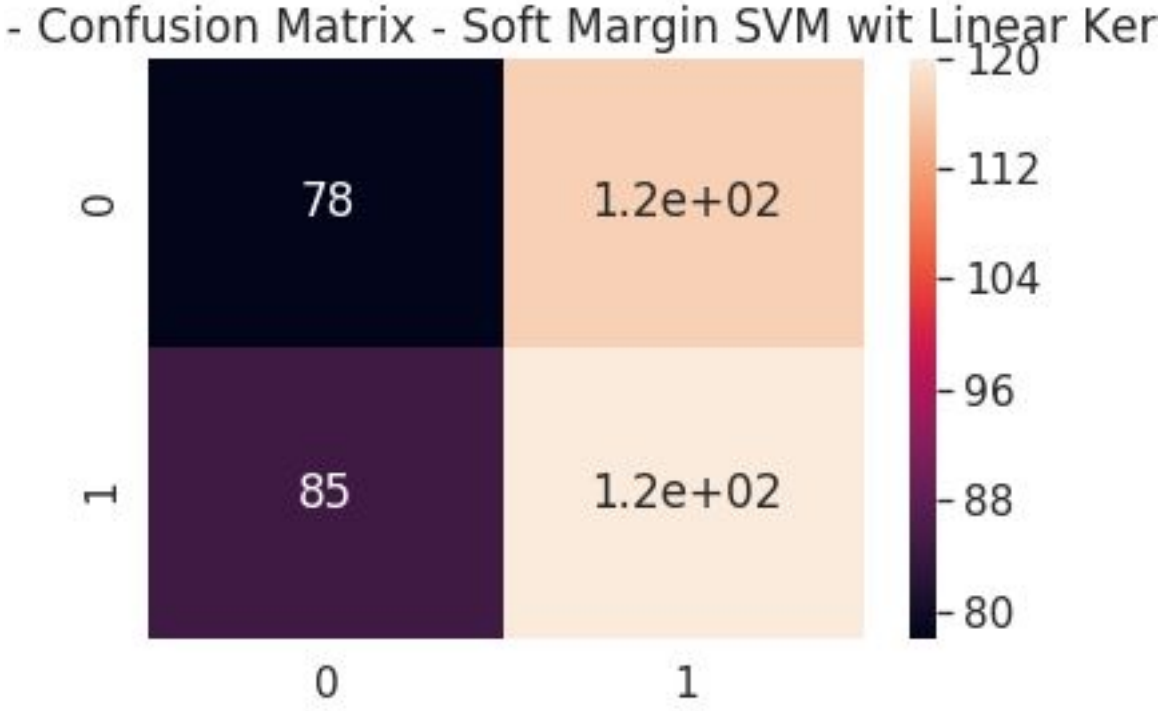


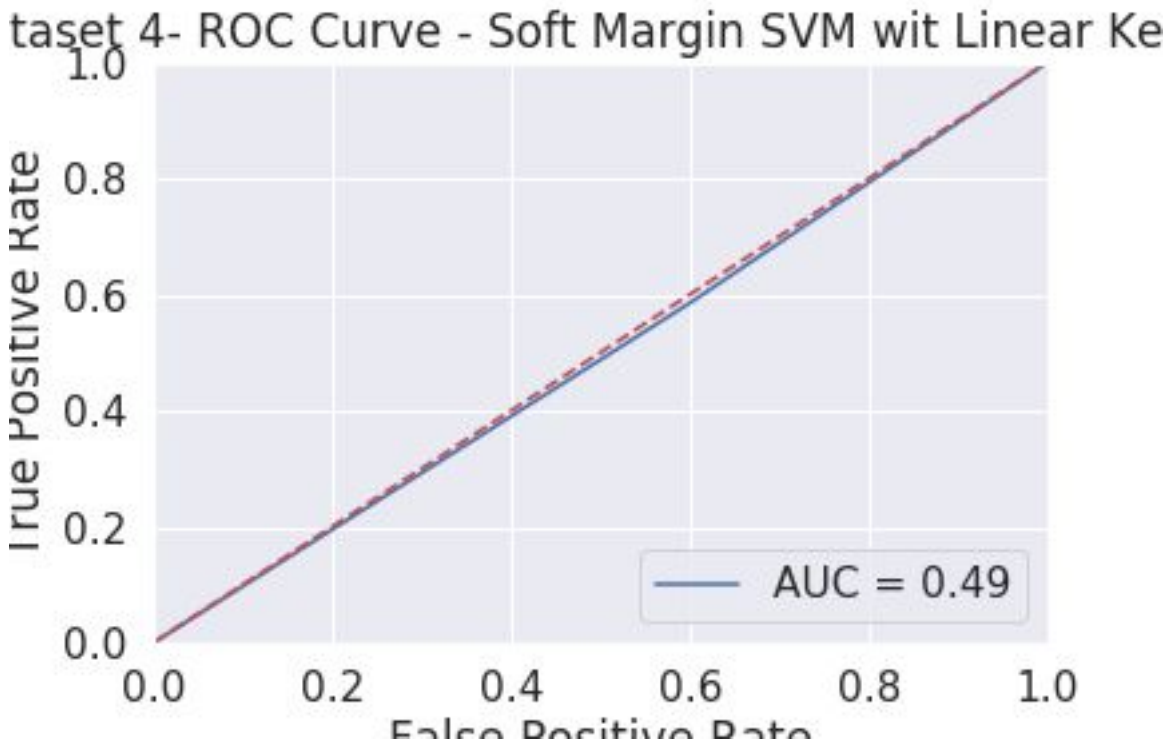
Dataset 3:
Accuracy: 1.0
F1 Score: 1.0

Dataset 4:



Accuracy: 0.5375
F1 Score: 1.0

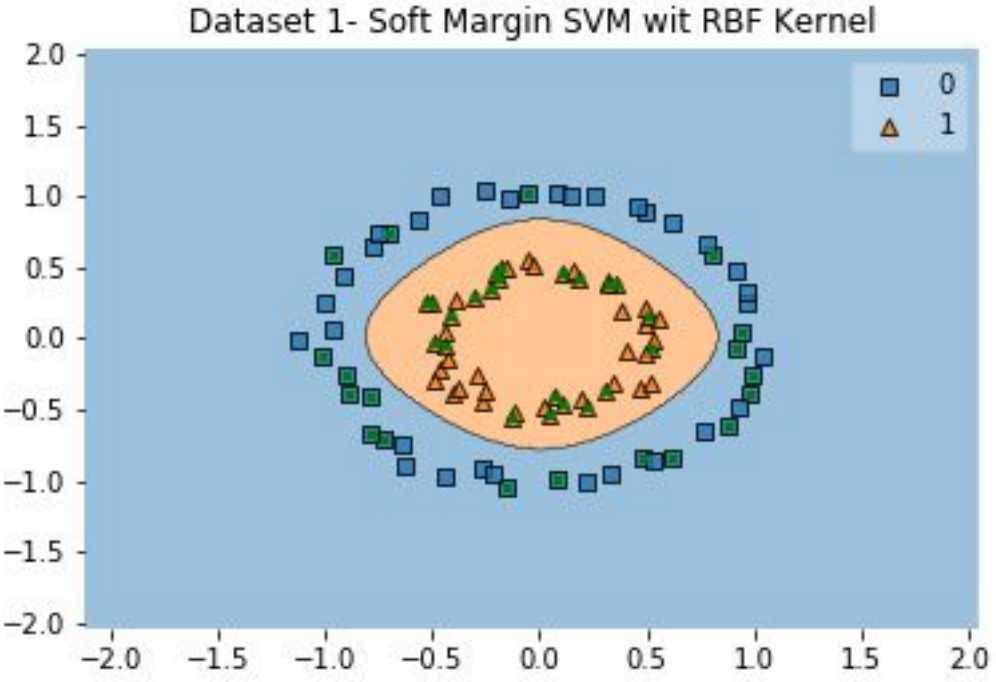




(iv) (10 points) Implement Soft margin SVM with RBF kernel. Report the accuracy and F1 score on each dataset.

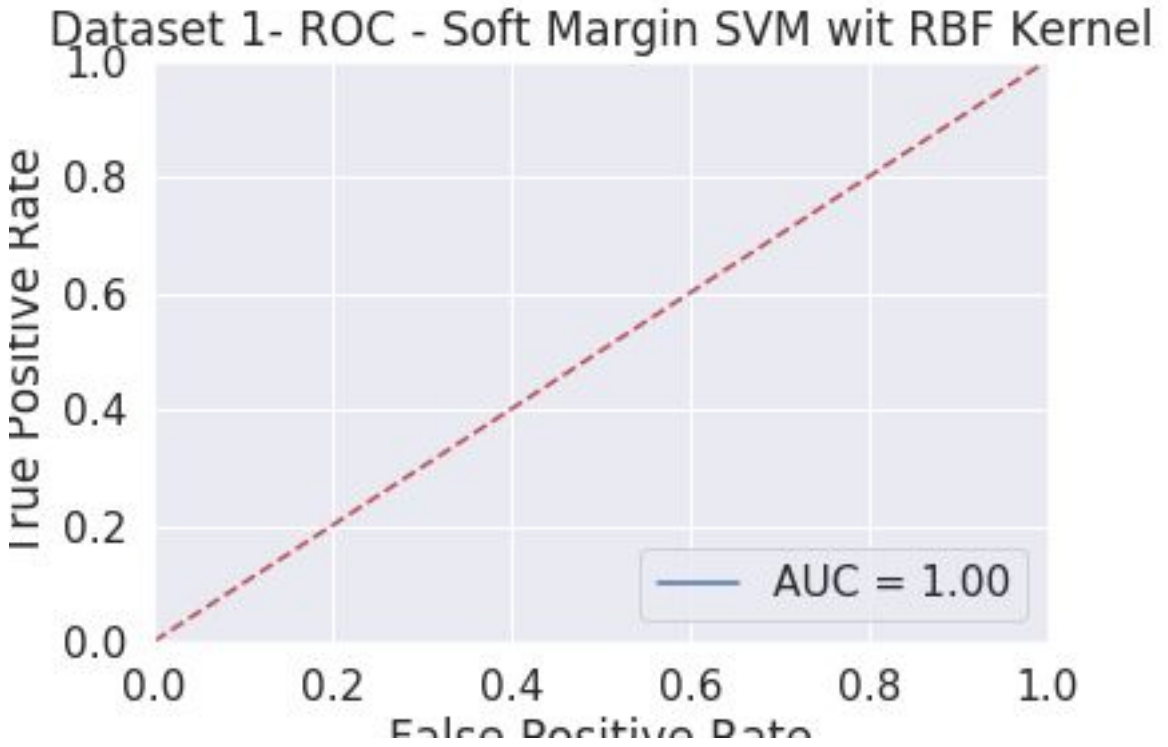
Note: Support Vectors are marked with green color.

Dataset 1:

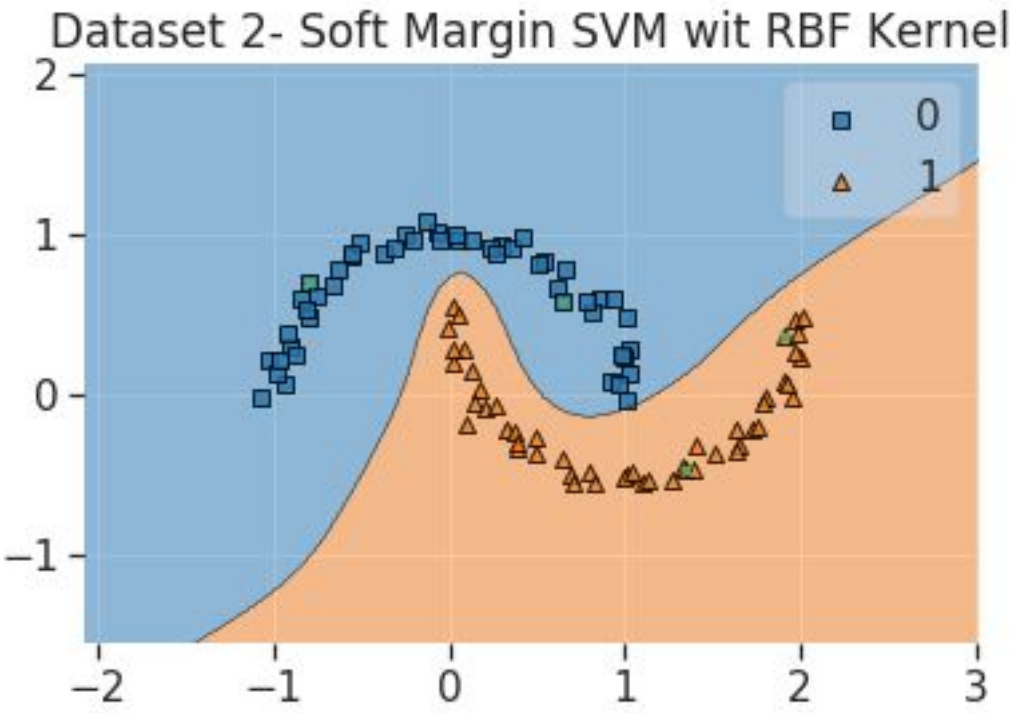


Accuracy: 1.0
F1 Score: 1.0

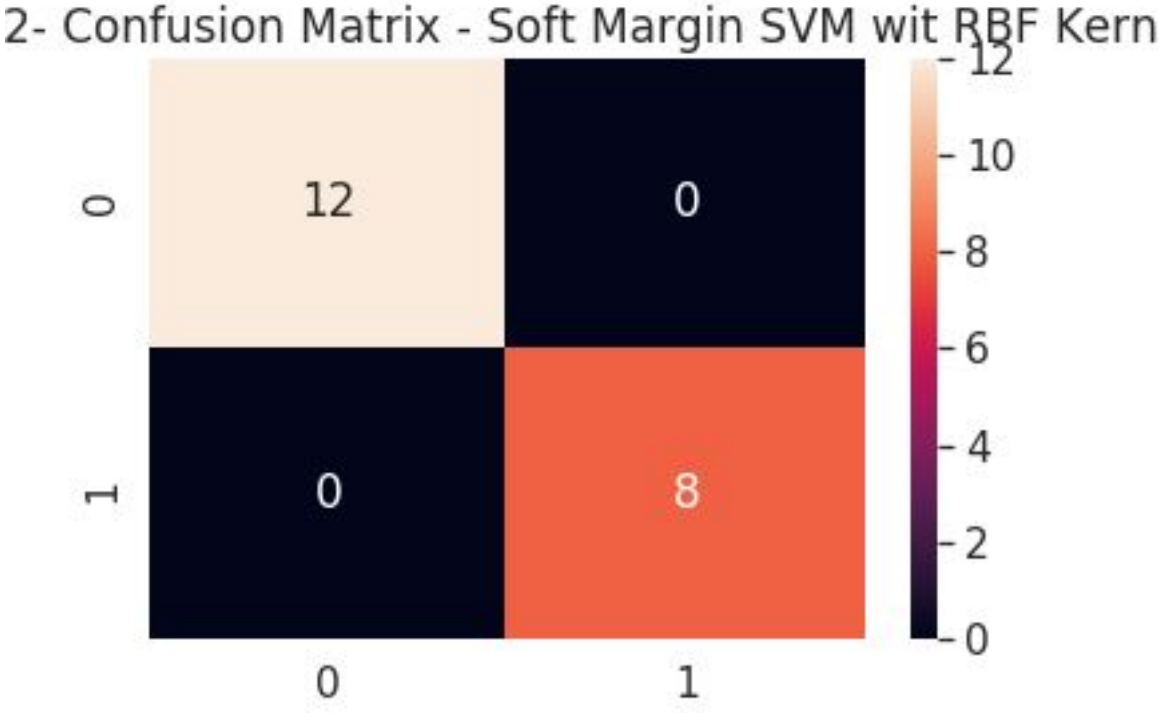

1- Confusion Matrix - Soft Margin SVM wit RBF Kern
7
0
0
13
0
1
0
1
12.5
10.0
7.5
5.0
2.5
0.0

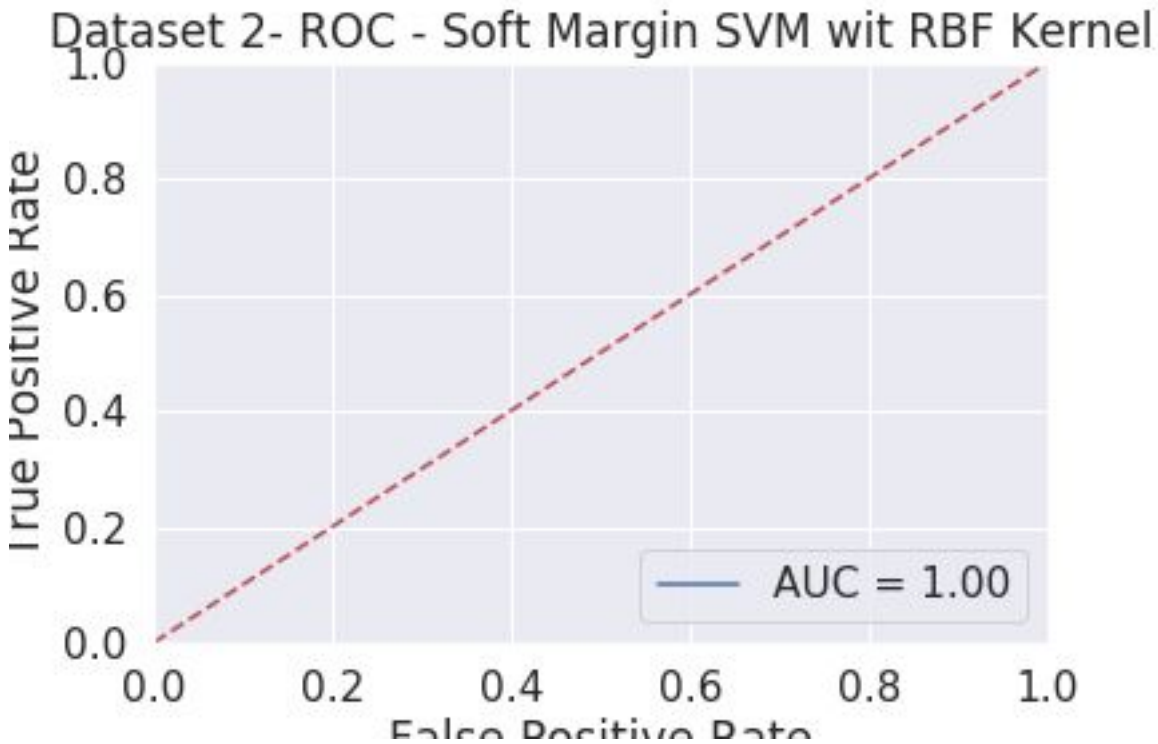

Dataset 1- ROC - Soft Margin SVM wit RBF Kernel
True Positive Rate
False Positive Rate
AUC = 1.00
0.0
0.2
0.4
0.6
0.8
1.0

Dataset 2:

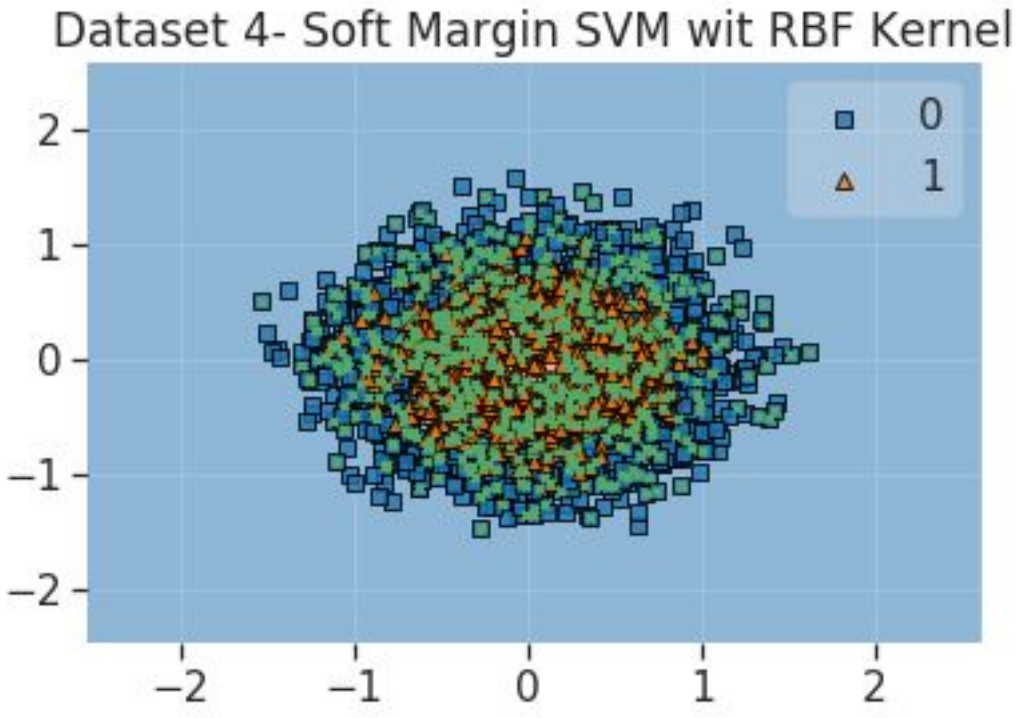


Accuracy: 1.0
F1 Score: 1.0

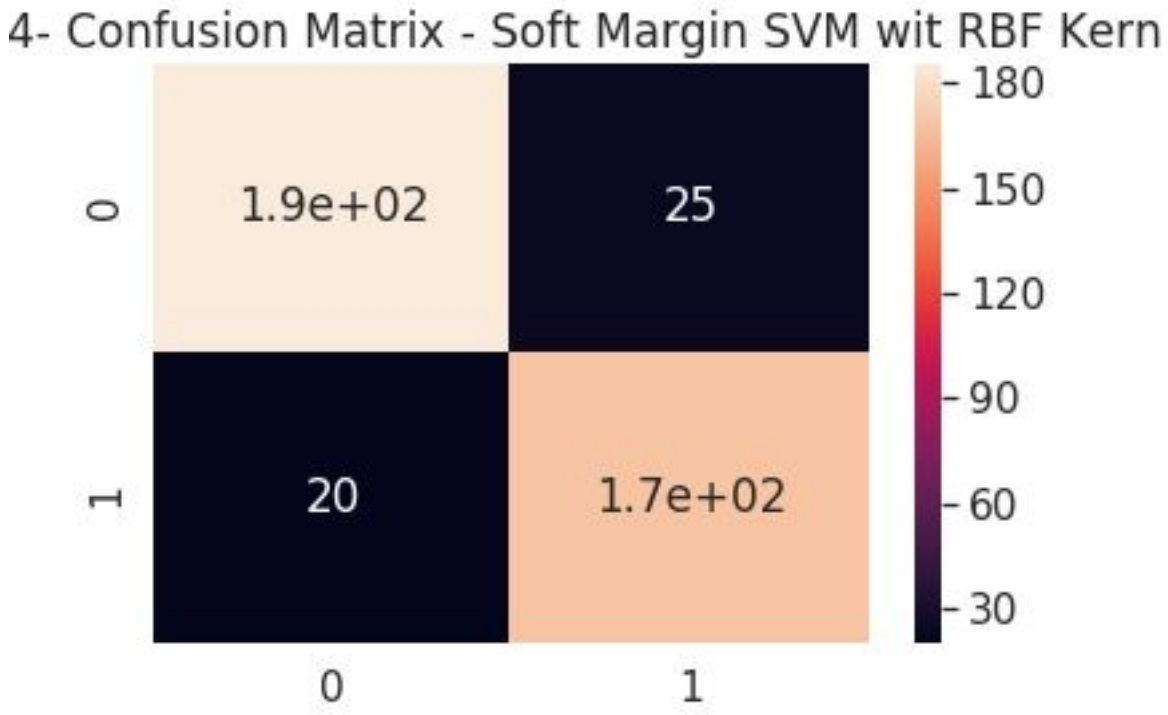


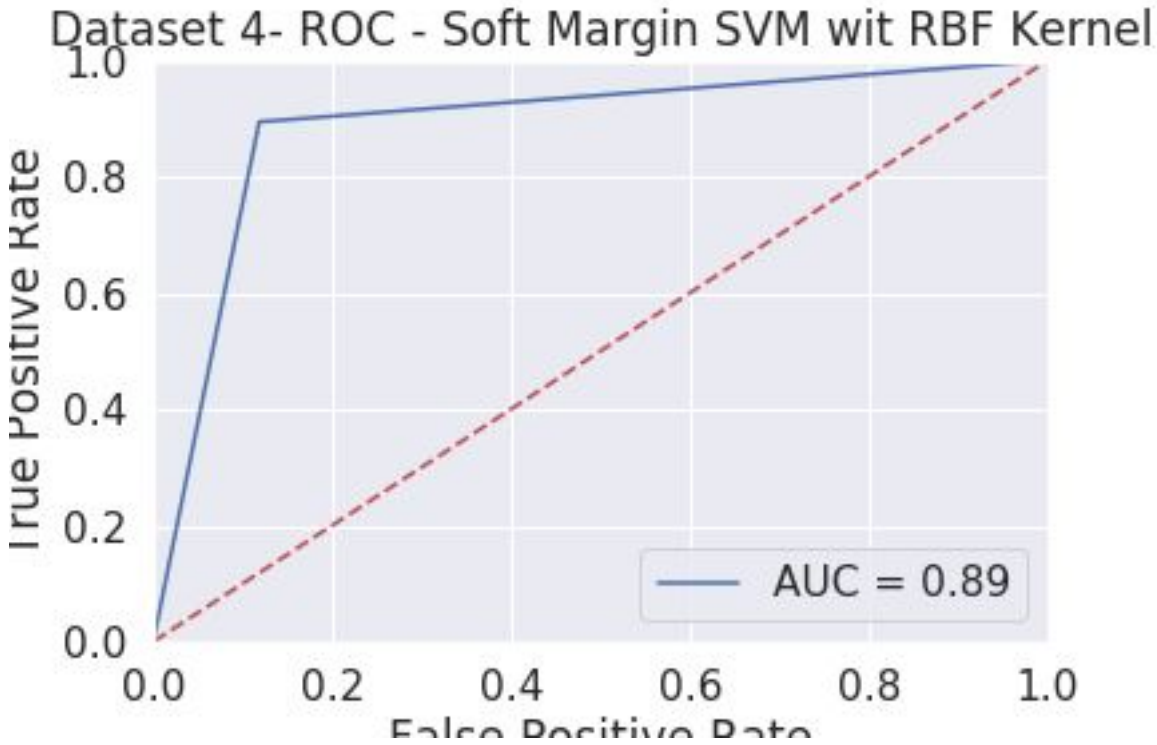


Dataset 3:
Accuracy: 1.0
F1 Score: 1.0
Dataset 4:



Accuracy: 0.8975
F1 Score: 1.0





v) (25 points) Use the RBF kernel with SVMs to classify the hindi handwritten character dataset. You are only given the training (attached as 'Train val Handwritten Hindi dataset.zip' file) and the test dataset (attached as Test Handwritten Hindi dataset.zip file). Split the data as you see fit and train your kernelized SVM. Report the training, validation and test error. Make sure that you do not use the test set for any training/cross-validation.

I used K-Fold cross validation with K = 5. Following are the result:
Train Accuracy for fold 1, 2, 3, 4, 5 respectively:
0.8402941176470589, 0.8366176470588236, 0.8391176470588235, 0.8348529411764706, 0.8354411764705882
Validation Accuracy for fold 1, 2, 3, 4, 5 respectively:
0.8288235294117647, 0.8423529411764706, 0.831176470588235, 0.85, 0.8482352941176471
Test Accuracy for fold 1, 2, 3, 4, 5 respectively:
0.625, 0.661, 0.655, 0.671, 0.665

vi) (25+10 points) For both of the above parts, report the following analysis:

• What hyperparameters did you choose? Why? How did you choose the optimal values for the hyperparameters?

• Plot the support vectors and the margin separating hyperplane.
Done. See (iii), (iv)

• Choose a value of hyperparameters such that the model overfits the training data. Discuss how overfitting affects the number of support vectors.
Number of support vectors increased drastically when I tried to overfit the training data.

• Discuss your observations on performance of the linear kernel vs the RBF kernel.
RBF kernel gave better accuracy and f1 score in all the datasets.

• Plot confusion matrices for each dataset. You have to calculate it yourself, and cannot use any library for this.
Done. See (iii), (iv)

• Plot ROC curves for data 1, data 2 and data 4 using the sklearn library.
Done. See (iii), (iv)

• (Bonus) Plot multi-class ROC curve for data 3 and the hindi handwritten character data.