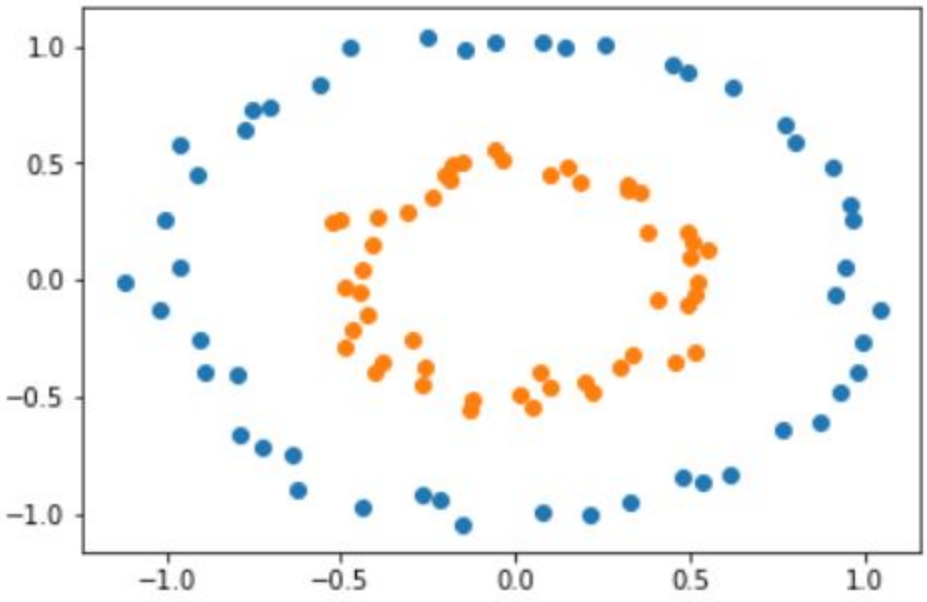
Machine Learning
Assignment 2

Abhishek Agarwal
2016126

Question 1)

Dataset 1:

data_1.h5

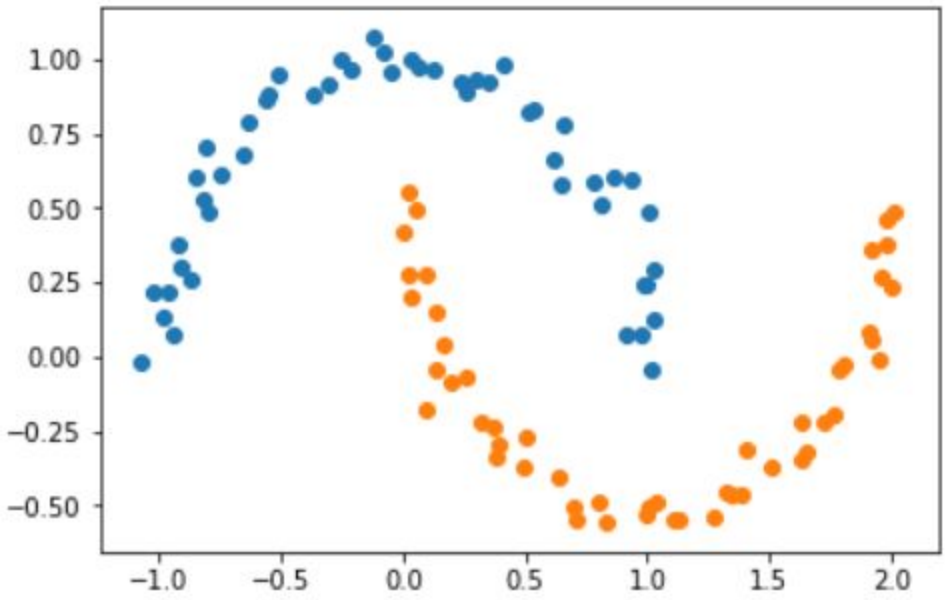


Observations:

- There are total 100 samples
- 2 classes
- 50 samples of each class
- Equally balanced
- Cannot be separated by a linear boundary, a circular boundary will be required to separate the two classes
- Very less noise

Dataset 2:

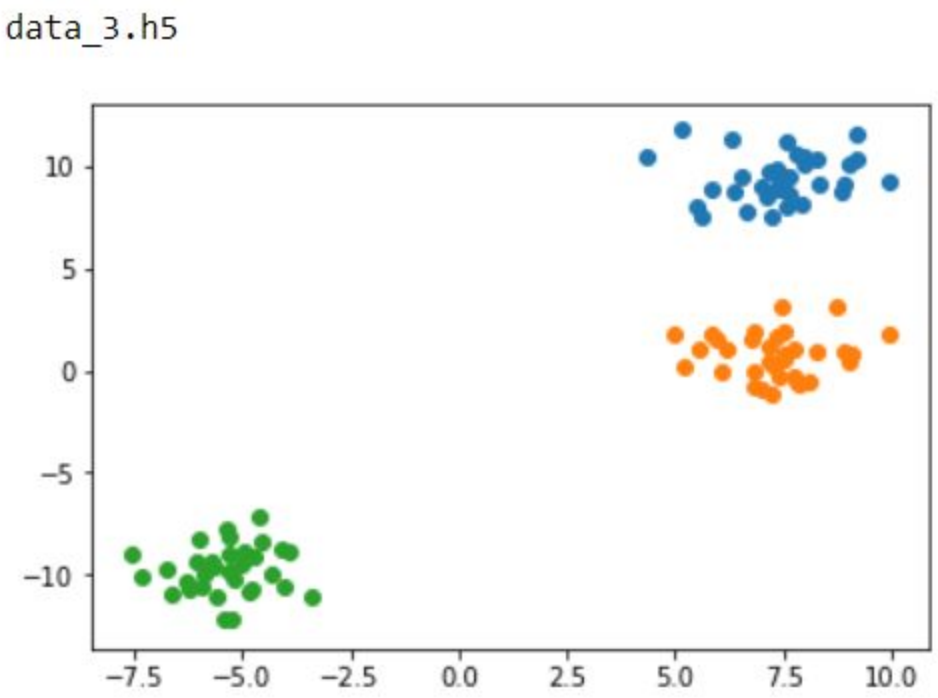
data_2.h5



Observations:

- There are total 100 samples
- 2 classes
- 50 samples of each class
- Equally balanced
- Cannot be separated by a linear boundary, a circular boundary will be required to separate the two classes
- Very less noise

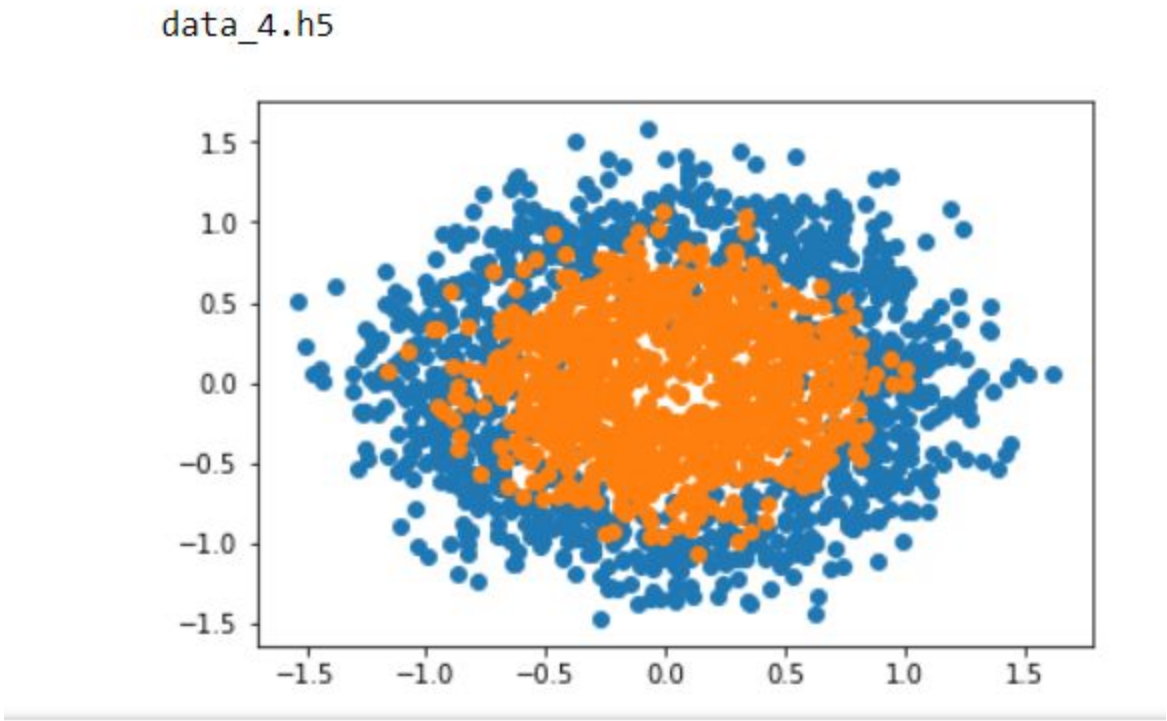
Dataset 3:



Observations:

- There are total 100 samples
- 3 classes
- 33 samples of two classes and 34 of the third class
- Equally balanced
- Linearly separable
- Very less noise

Dataset 4:



Observations:

- There are total 2000 samples
- 2 classes
- 1000 samples of each class
- Equally balanced
- Not separable linearly
- High noise

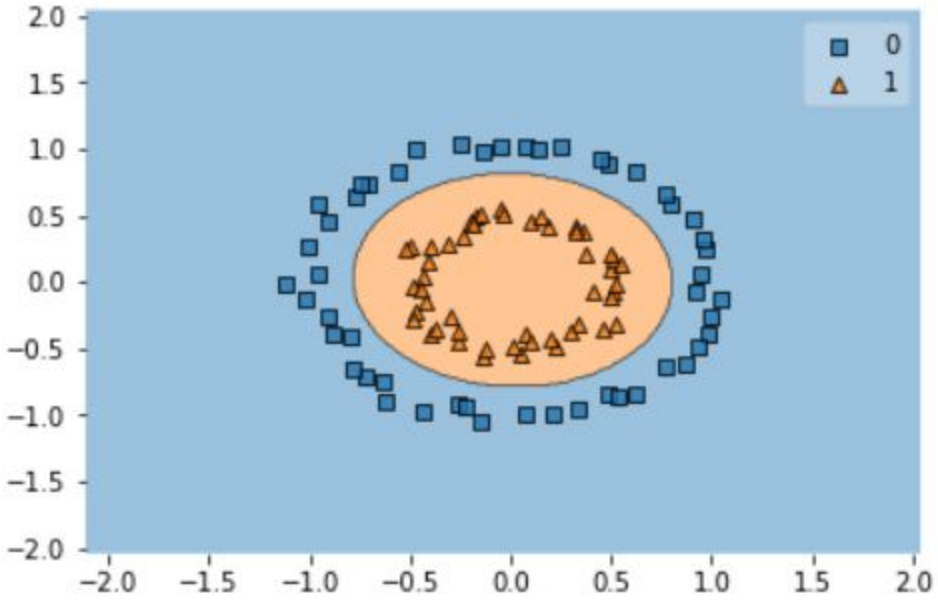
Question 2)

1. **Dataset 1:**
   Kernel used: Polynomial kernel with degree = 2

$$K(x, y) = (1 + x.y)^2$$

Reason: As it is visible from the plot that the two classes can be separated by a circular boundary between them, a polynomial kernel is chosen. The degree is taken to be 2 so that we can obtain a circular boundary.
The kernel equation is the same as the polynomial kernel with degree 2.

Plot of decision boundary:

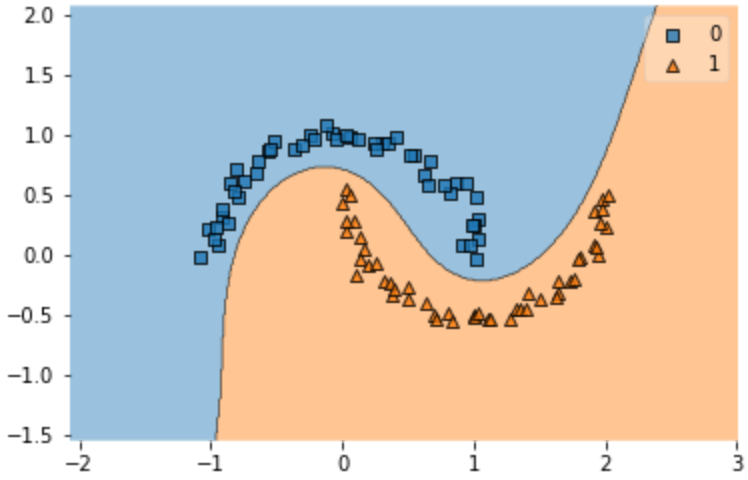


2. **Dataset 2:**
   Kernel used: Polynomial kernel with degree = 3

$$K(x, y) = (1 + x.y)^3$$

   Reason: As it is visible from the plot that the two classes can be separated by the boundary of a cubic function, thus a polynomial kernel with degree 3 is chosen..
   The kernel equation is the same as the polynomial kernel with degree 3.

   Plot of decision boundary:



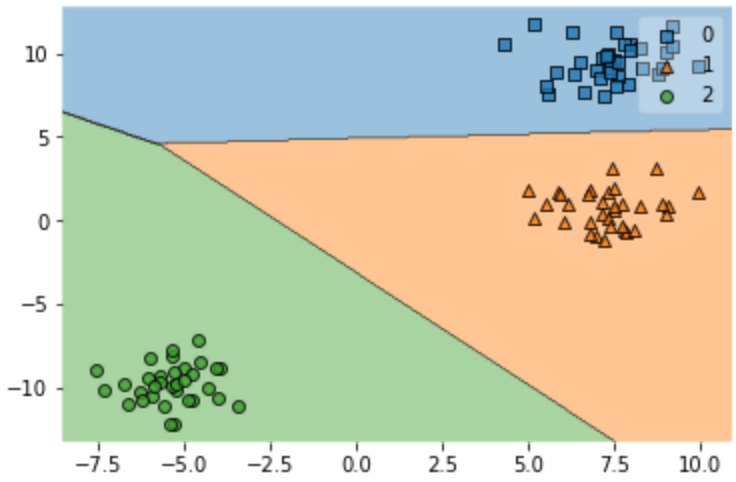
3. **Dataset 3:**

   Kernel used: Linear kernel

   $$K(x,\ y)\ =\ (x.y)$$

   Reason: As it is visible from the plot that the three classes can be separated by making a linear boundary between the classes.
   The kernel equation is the same as the linear kernel.

   Plot of decision boundary:



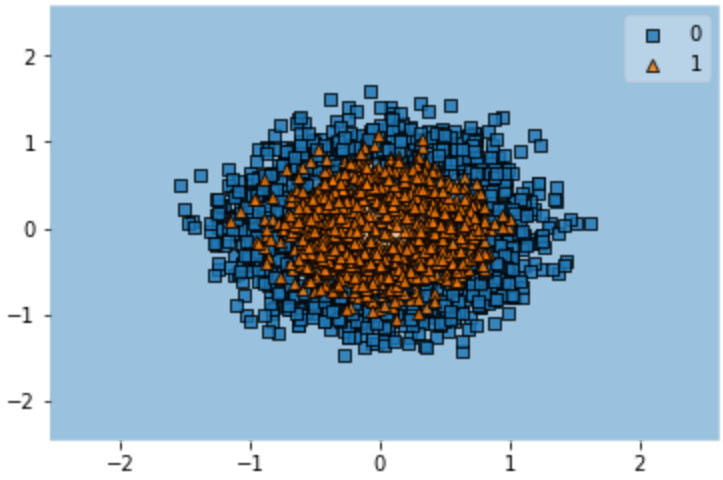
4. **Dataset 4:**

   Kernel used: Polynomial kernel with degree = 2

   $$K(x, y) = (1 + x.y)^2$$

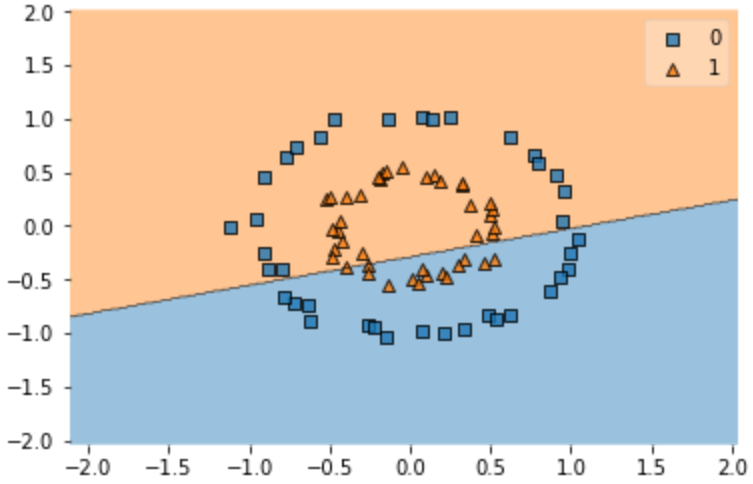
   Reason: As it is visible from the plot that the three classes can be separated by making a circle between them. However, due to huge noise in the data we will not be able to separate them perfectly and a soft margin will be required.
   The kernel equation is the same as the polynomial kernel with degree 2.

   Plot of decision boundary:

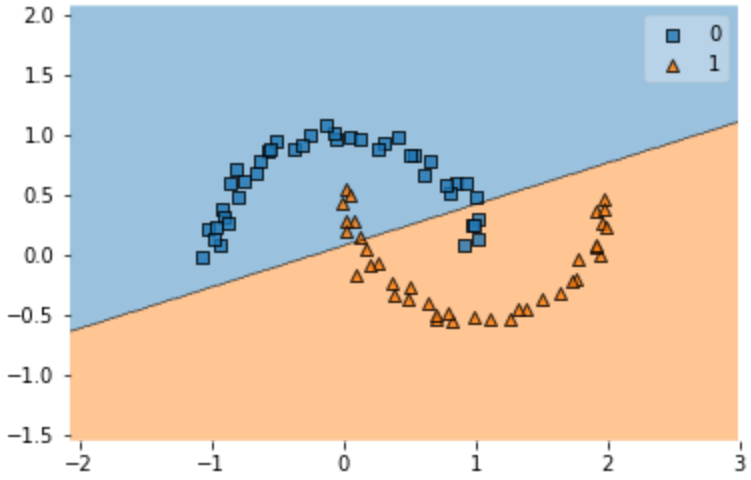


Question 3: SVM with soft margin using Linear Kernel

1. **Dataset 1:**
    a. Train Accuracy: 0.5875
    b. Test Accuracy: 0.45
    c. F1 score for class 0: 0.33
    d. F1 score for class 1: 0.45



2. **Dataset 2:**
    a. Train Accuracy: 0.8875
    b. Test Accuracy: 0.75
    c. F1 score for class 1: 0.89
    d. F1 score for class 2: 0.88

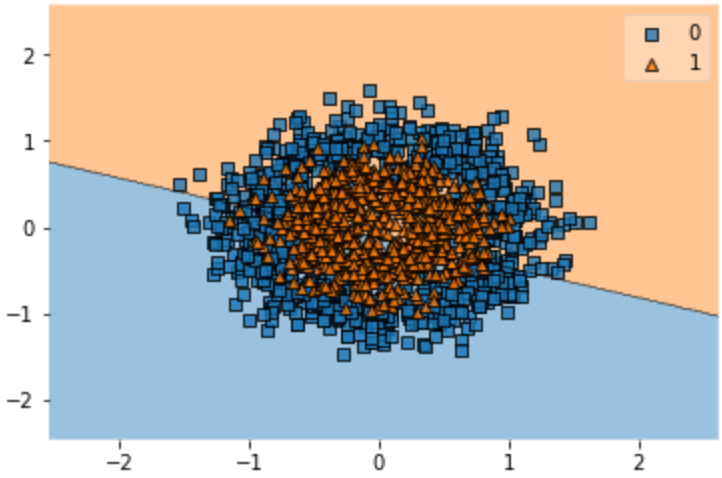


3. **Dataset 3:**

   a. **ONE VS REST**
      i. Train Accuracy: 1
      ii. Test Accuracy: 1
      iii. F1 score for class 1: 1
      iv. F1 score for class 2: 1
      v. F1 score for class 3: 1

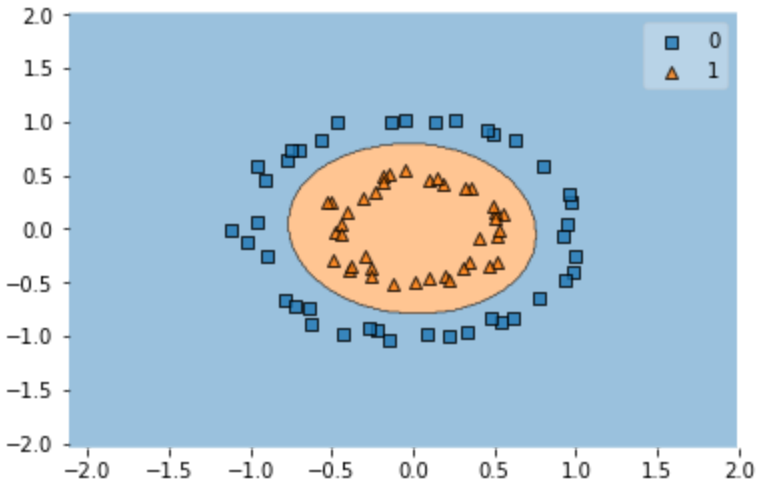
   b. **ONE VS ALL**
      i. Train Accuracy: 1
      ii. Test Accuracy: 1
      iii. F1 score for class 1: 1
      iv. F1 score for class 2: 1
      v. F1 score for class 3: 1

4. **Dataset 4:**
    a. Train Accuracy: 0.54
    b. Test Accuracy: 0.47
    c. F1 score for class 1: 0.51
    d. F1 score for class 2: 0.56

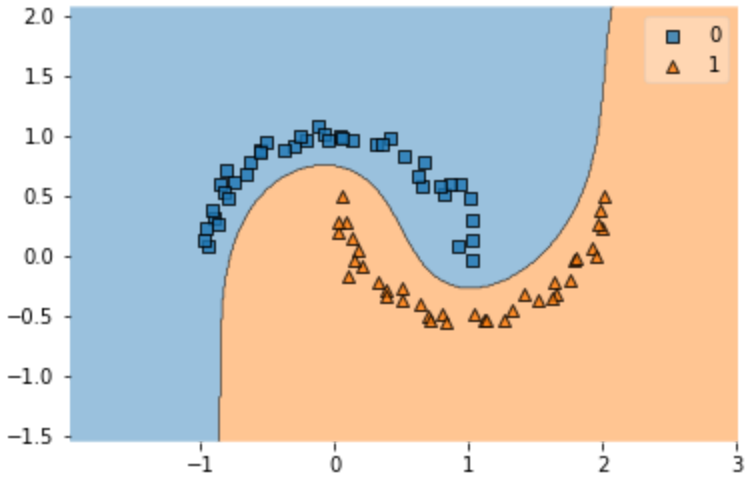


Question 4: SVM with soft margin using RBF Kernel

1. **Dataset 1:**
    a. Train Accuracy: 1
    b. Test Accuracy: 1
    c. F1 score for class 1: 1
    d. F1 score for class 2: 1



2. **Dataset 2:**
    a. Train Accuracy: 1
    b. Test Accuracy: 1
    c. F1 score for class 1: 1
    d. F1 score for class 2: 1

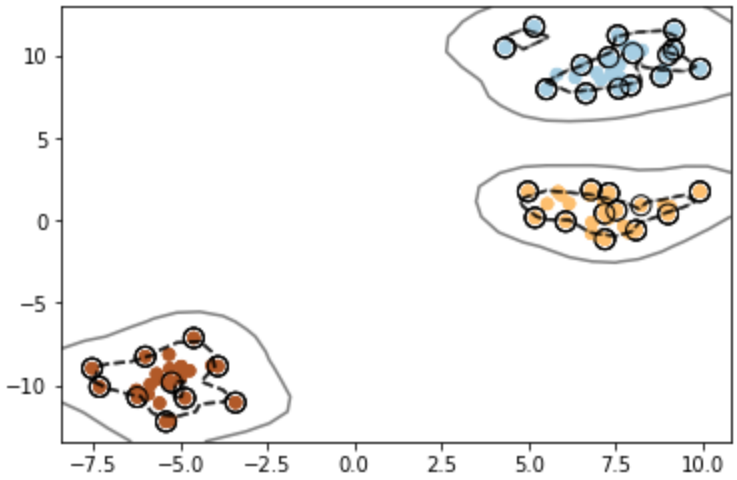


3. **Dataset 3:**
    a. **ONE VS REST**
        i. Train Accuracy: 1

ii. Test Accuracy: 1
iii. F1 score for class 1: 1
iv. F1 score for class 2: 1
v. F1 score for class 3: 1

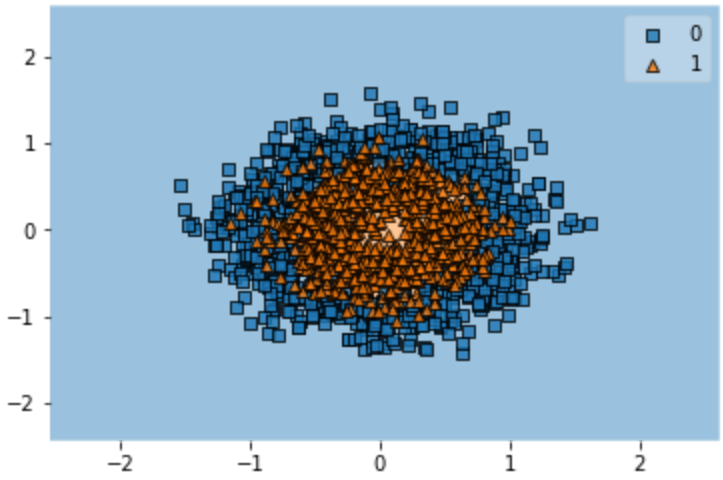
b. ONE VS ALL

i. Train Accuracy: 1
ii. Test Accuracy: 1
iii. F1 score for class 1: 1
iv. F1 score for class 2: 1
v. F1 score for class 3: 1



4. Dataset 4:

a. Train Accuracy: 0.887
b. Test Accuracy: 0.867
c. F1 score for class 1: 0.885
d. F1 score for class 2: 0.887



Question 5) RBF kernel with SVMs to classify the hindi handwritten character dataset.

This data has 5 classes having one hindi alphabet each.
The training set is shuffled and split into training(80%) and validation(20%) sets.
Hyperparameter tuning and shuffling is done with the help of GridSearchCV by passing the entire train set and keeping the value of cross validation(cv) parameter to be 5.

Each image on reading with MatplotLib library gives us a 32*32 matrix, which has been reshaped to an array of 1024 to make it a complete feature vector.
In total there are 1700 images of each character, thus the complete train set comprises of 8500 images reshaped into a feature vector of size 1024.

The following values of C and Gamma were passed for hyperparameter tuning in GridSearchCV.

‘C’ : [1000, 500, 100, 50, 20, 10, 4, 3.5, 3, 2.5, 2, 1, 0.1, 0.005, 0.003],
'gamma' : [10, 4, 3.5, 3, 2.5, 2, 1, 0.1, 0.005, 0.003]

The optimal values chosen were: C = 100, Gamma = 0.005

Accuracy obtained is as follows:

Train set: 1
Test set: 0.994
Validation set: 0.88894118

Question 6)

- **Hyperparameters chosen:**

For each of the 5 datasets, GridSearchCV was run to calculate the optimal value of the hyperparameters. The value obtained was as follows:

- Soft Margin Linear kernel SVM:
  - Dataset 1: C = 3
  - Dataset 2: C = 3
  - Dataset 3: C = 3
  - Dataset 4: C = 0.5
- Soft Margin RBF kernel SVM:
  - Dataset 1: C = 1, gamma = 0.01
  - Dataset 2: C = 1, gamma = 0.01
  - Dataset 3: C = 1, gamma = 0.01
  - Dataset 4: C = 0.5, gamma = 0.01

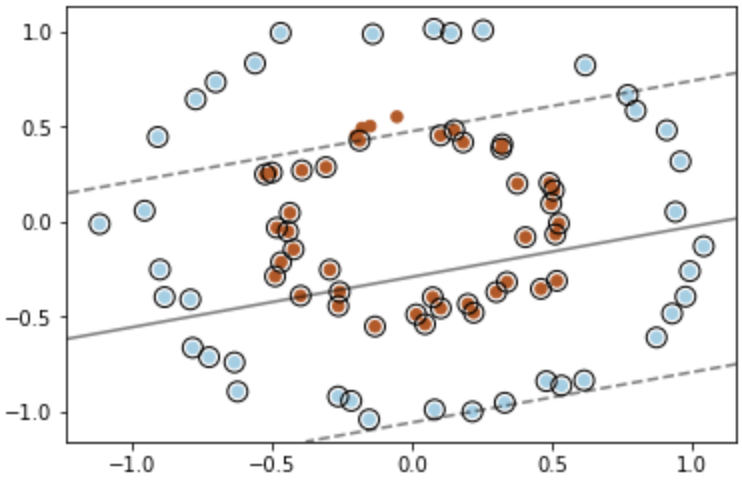
For the Hindi Dataset the hyperparameters chosen have been reported above.

Reason for choice: Since 3 datasets have very less noise we do not need a softer margin and hence high C values were chosen/obtained.

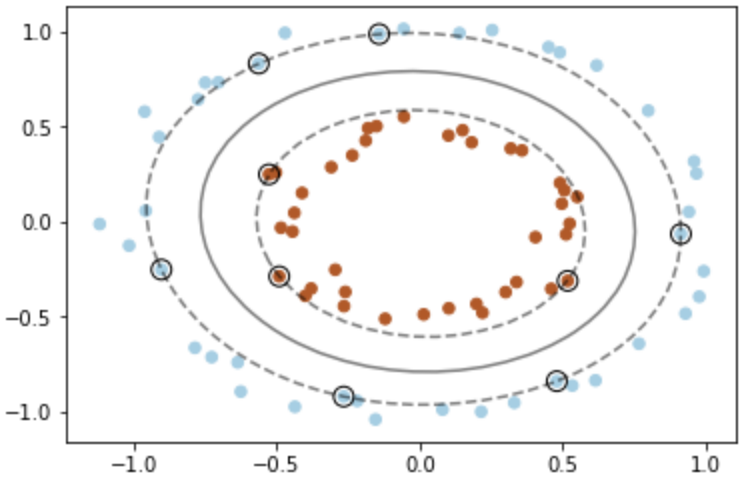
For the 4th dataset which has high noise, we need a softer margin so that we can allow misclassifications and hence a low C value is chosen.

- **(ii) Plot support vectors and margin**

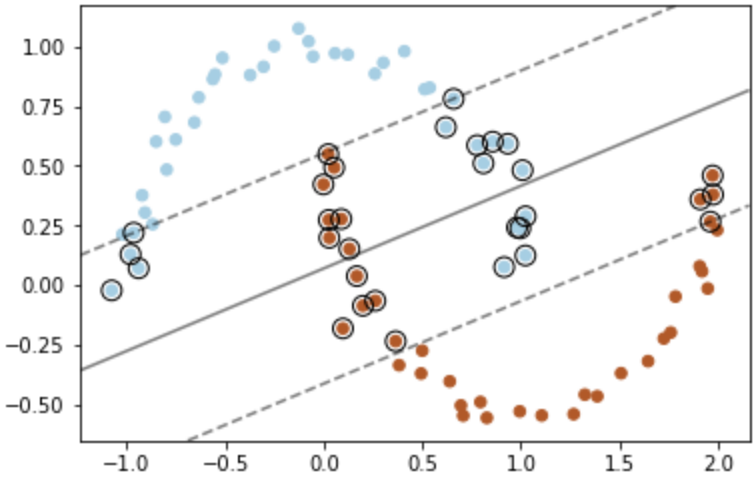
- **Dataset 1:**
  - **Linear:**



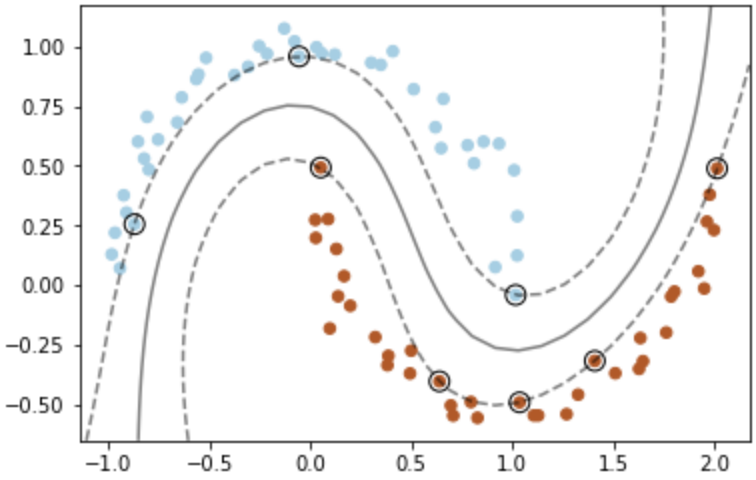
  - **RBF**



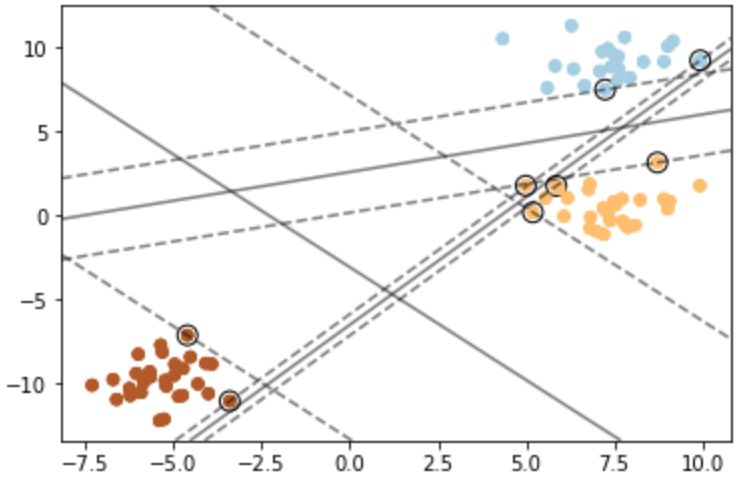
- **Dataset 2:**
  - **Linear:**



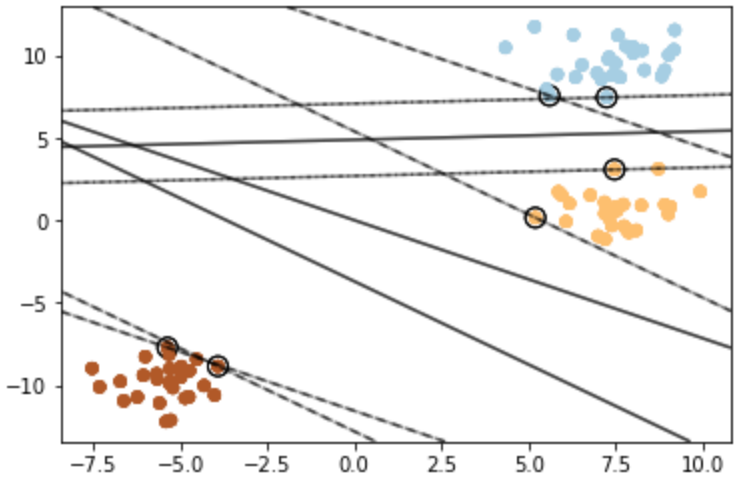
  - **RBF:**



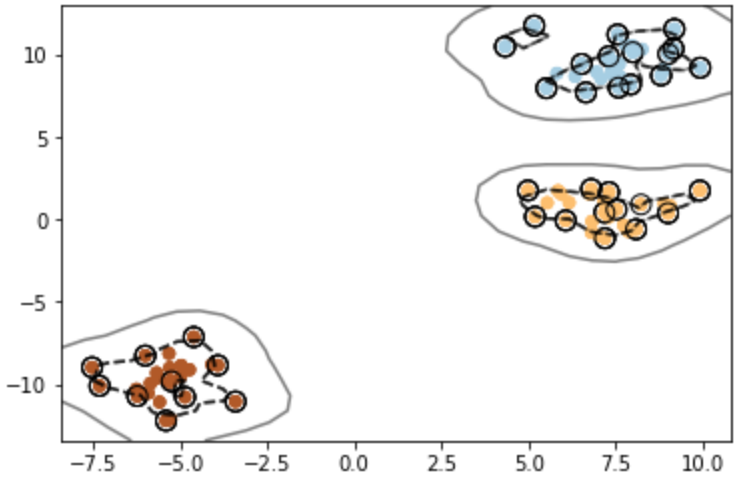
- **Dataset 3:**
  - **Linear:**
    - **One vs Rest**



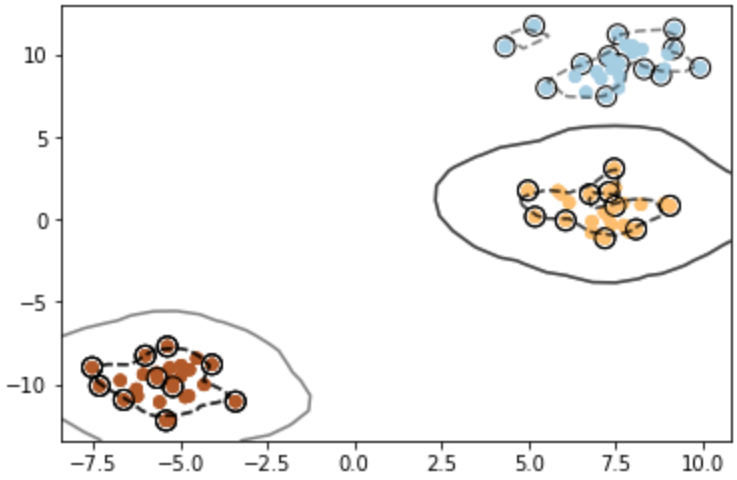
    - **One vs One**



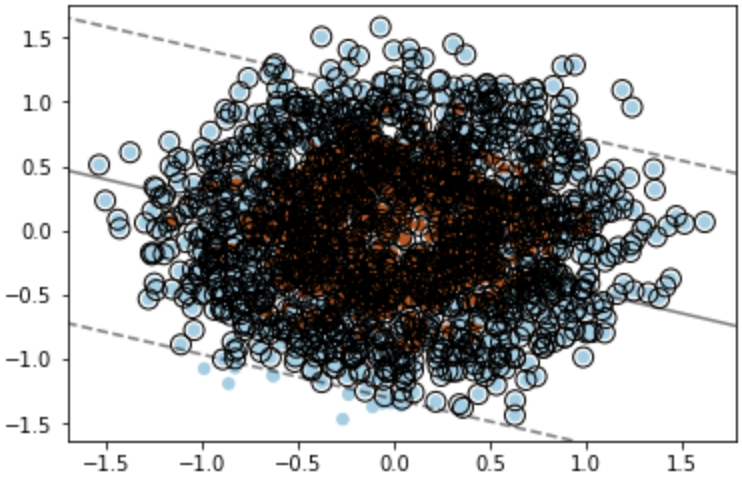
  - **RBF:**
    - **One vs Rest**



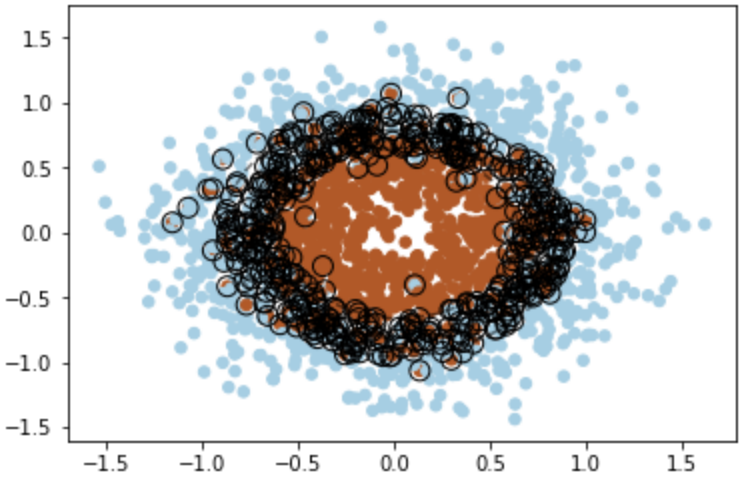
- **One vs One**



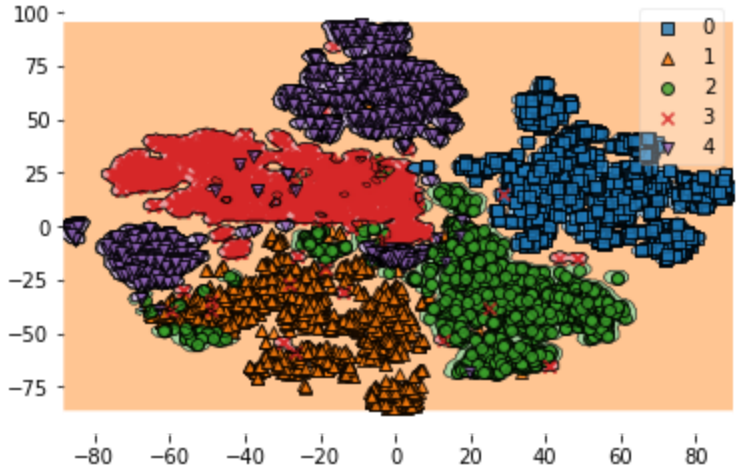
- **Dataset 4:**
  - **Linear:**



- **RBF**



- **Hindi Dataset:**



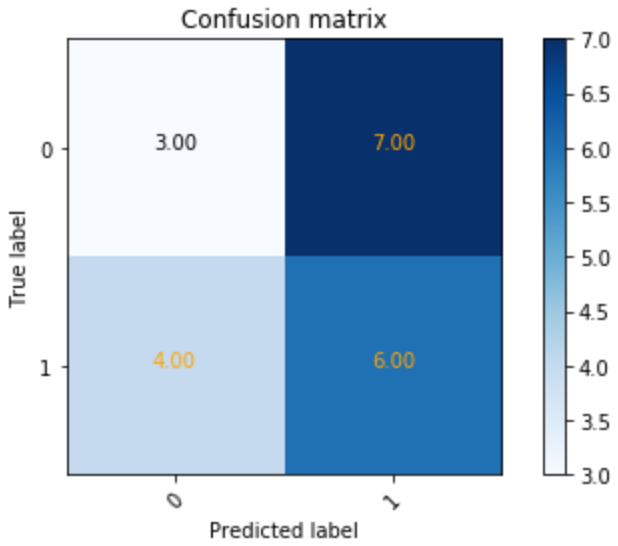
- **(iii) Overfitting Hyperparameters**
    - For Linear kernel models:
      On reducing the value of C the accuracy decreased by a very slight margin which shows underfitting, whereas on increasing the value of C there was no change in the accuracy which is because of the non separability of the data by a linear boundary.

    - For RBF kernel models:
      On varying the values of C and Gamma, the train and test accuracy were not found to differ much for the datasets. Therefore overfitting does not occur on increasing C.

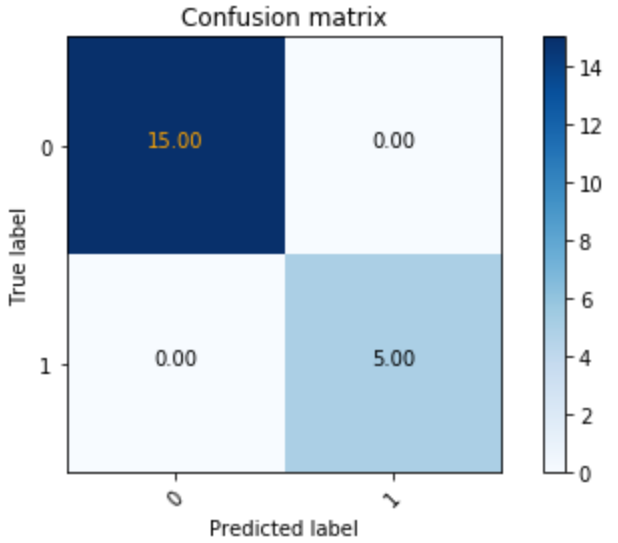
      For both the models we observe that as we keep decreasing C, the model tends to move towards an overfitting one as the penalty becomes less and thus the number of support vectors also keeps increasing.

- **(iv) Linear Kernel vs RBF Kernel:**
    - Linear Kernel works very well for linearly separable data, as we saw in the first question. This type of Kernel is not good for complex data as it cannot construct a hyperplane to separate non linearly separable data. It works well even for multiple classes which can be separated by linear boundaries.
    - RBF kernel has a very good performance on a variety of dataset as we can see the results for all the given datasets.
      However we also see that RBF kernel does not perform as good on the data with high noise which is because of the overlapping nature of the datapoints. Thus soft margin is used to allow some misclassifications and helping us to get a perfect decision boundary.

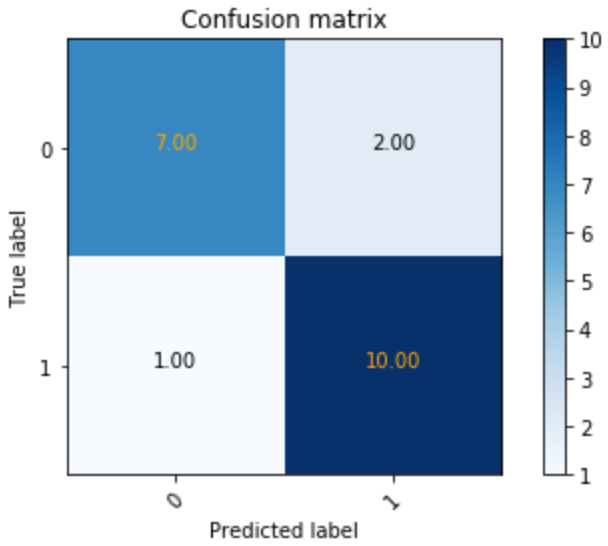
- **(v) Confusion Matrices:**
    - **Dataset 1**
        - **Linear**



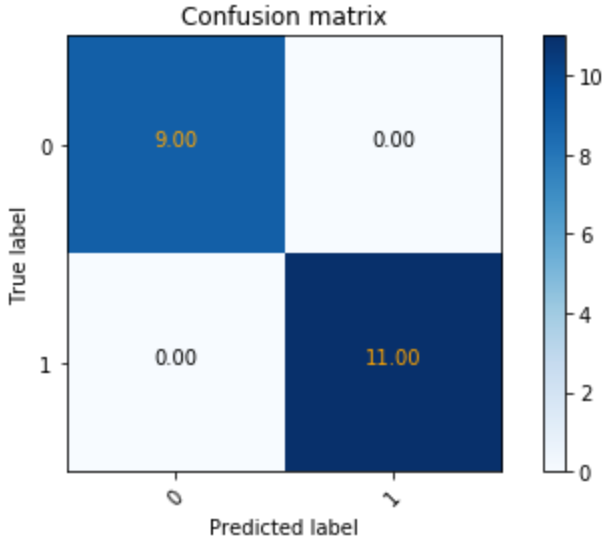
- **RBF**



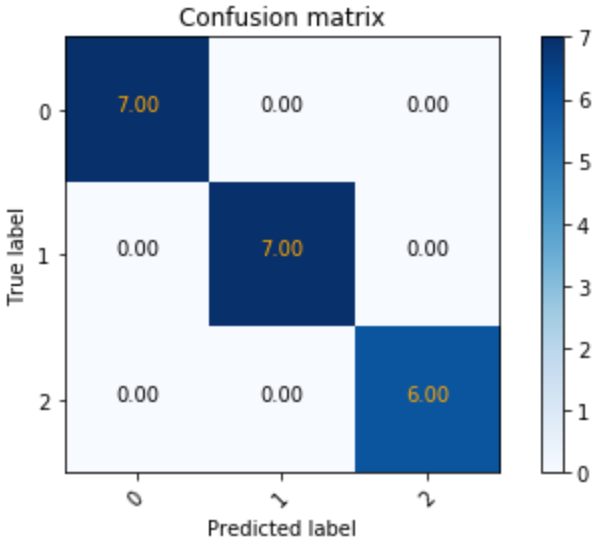
- **Dataset 2**
  - **Linear**



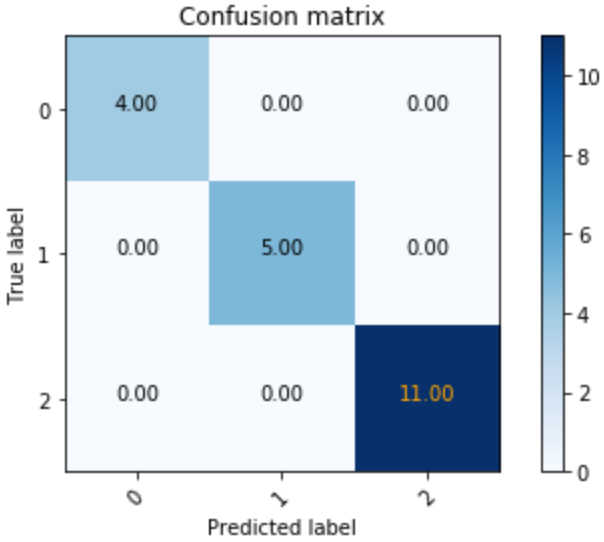
- **RBF**



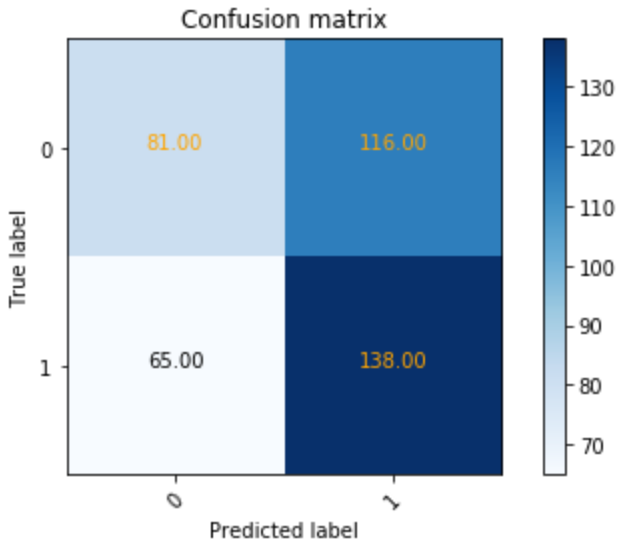
- **Dataset 3**
  - **Linear**



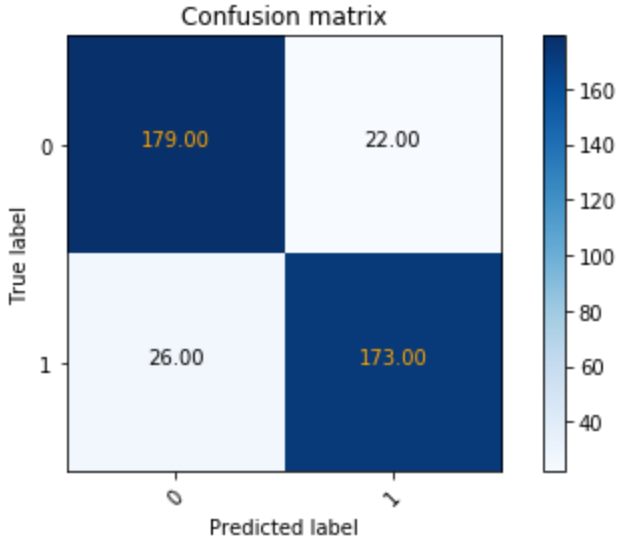
- **RBF**



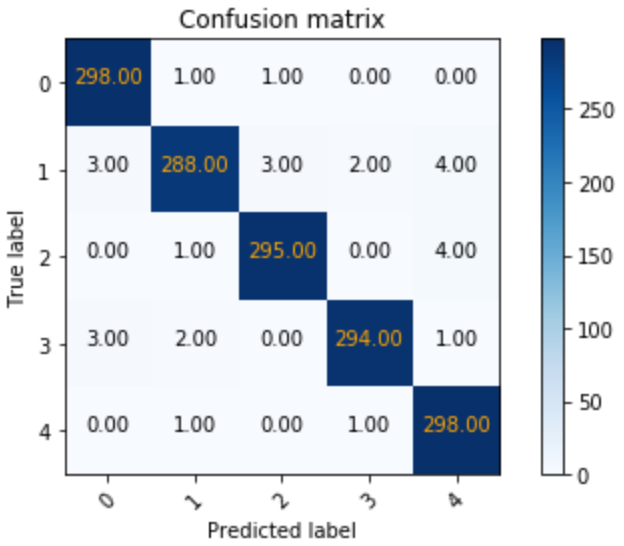
- **Dataset 4**
  - **Linear**



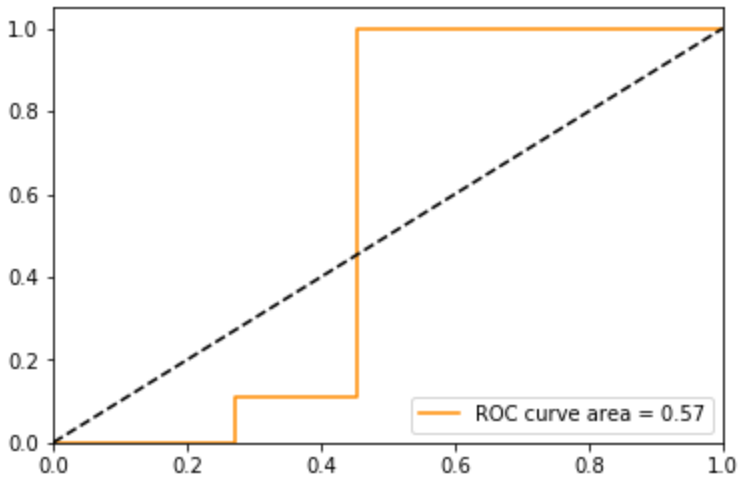
- **RBF**



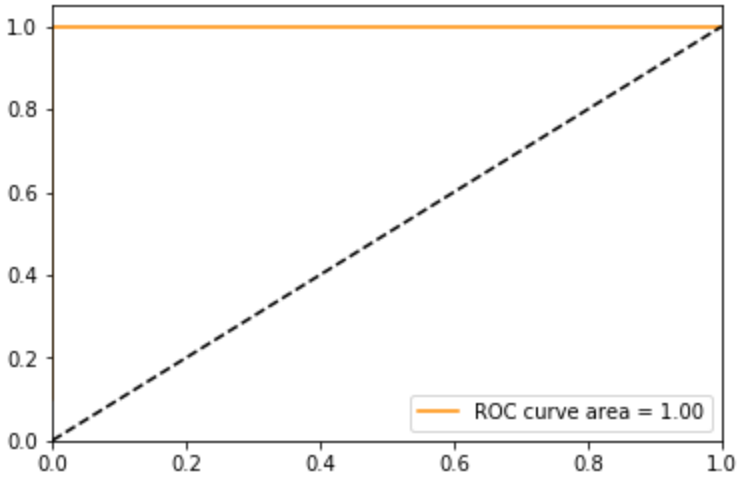
- **Hindi Dataset**



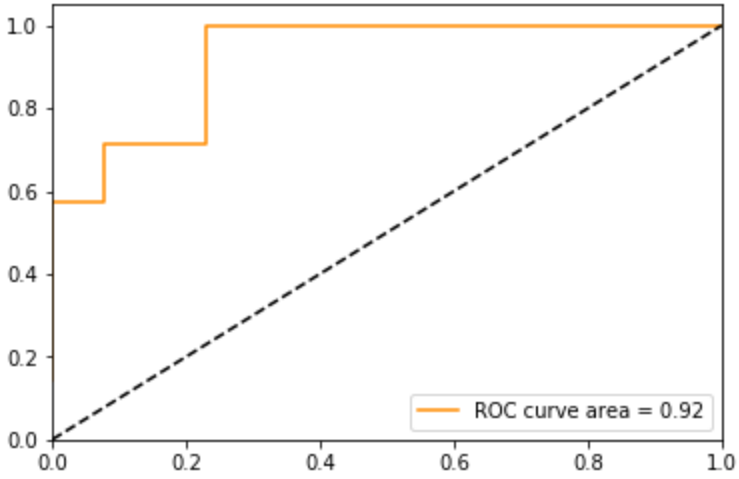
- **(vi) ROC Curves:**
  - Dataset 1
    - Linear



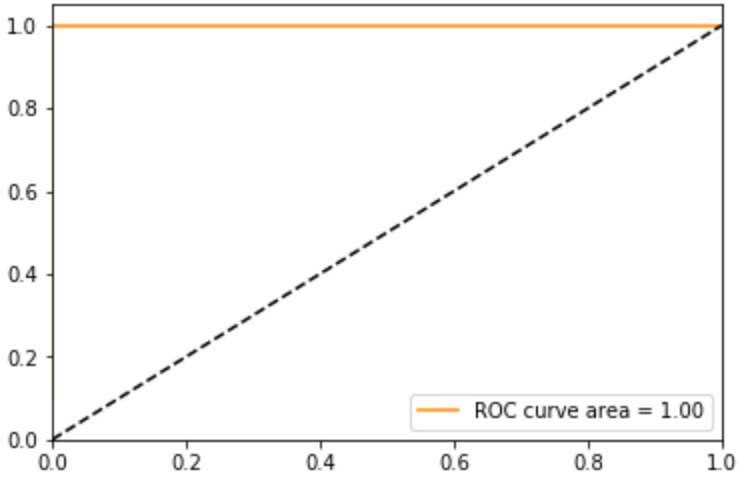
    - RBF



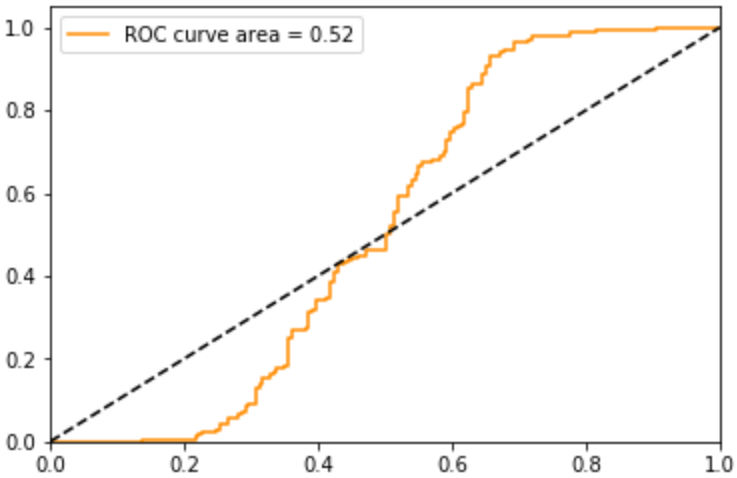
- Dataset 2
    - Linear



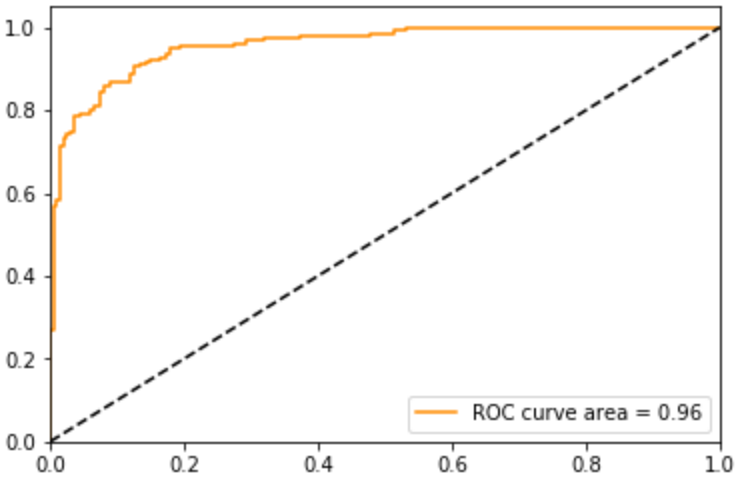
- RBF



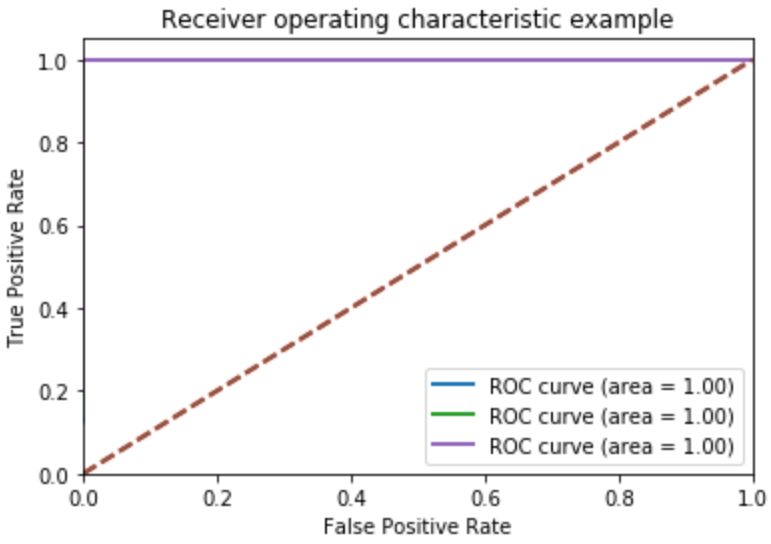
- Dataset 4
    - Linear



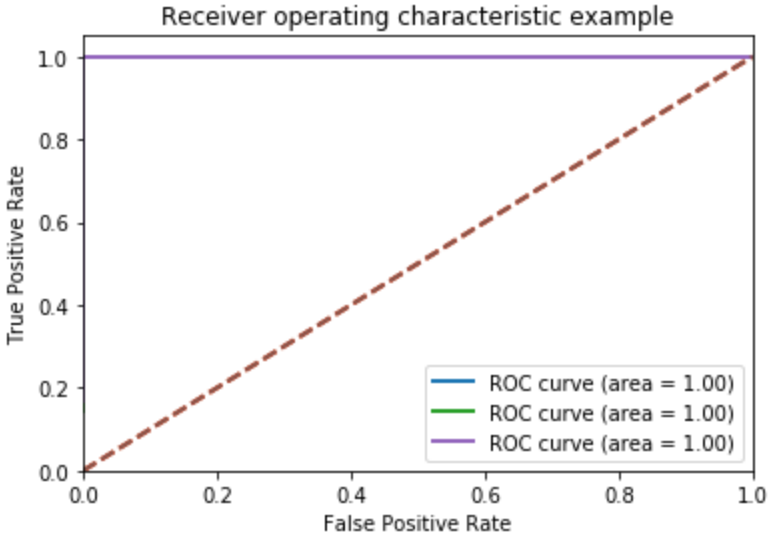
    - RBF

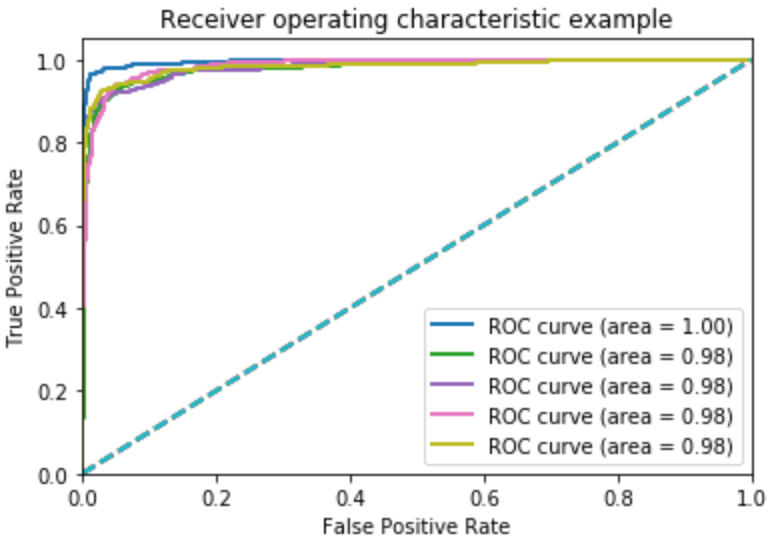


- **(Bonus) Multiclass ROC Curve**
    - **Linear**


Receiver operating characteristic example
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 1.00)
ROC curve (area = 1.00)
ROC curve (area = 1.00)

- **RBF**


Receiver operating characteristic example
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 1.00)
ROC curve (area = 1.00)
ROC curve (area = 1.00)



Functions implemented:

Since the code was written in Jupyter Notebook, all functions are written as different cells.

- kernel_1(X, Y): Takes in two input vectors and returns the value of self defined kernel function for dataset 1.
- kernel_2(X, Y): Takes in two input vectors and returns the value of self defined kernel function for dataset 2.
- kernel_3(X, Y): Takes in two input vectors and returns the value of self defined kernel function for dataset 3.
- kernel_24(X, Y): Takes in two input vectors and returns the value of self defined kernel function for dataset 4.
- plot_support_vectors(X, Y, clf): A function used to plot the support vectors for given input and corresponding labels.
- plot_roc_curve(x_test, y_true, classifier): Plots the ROC curve for a binary classifier.
- plot_roc_curve_multi_class(coordinates, labels, num_classes, kernel_name): Plots ROC curve for multi class classifier, used for dataset 3.
- plot_roc_curve_multi_class_hindi(coordinates, labels, num_classes): Used to plot ROC curve for Hindi dataset

- make_confusion_matrix(y_test, y_predicted, num_classes): This returns the confusion matrix

----------End of Report----------