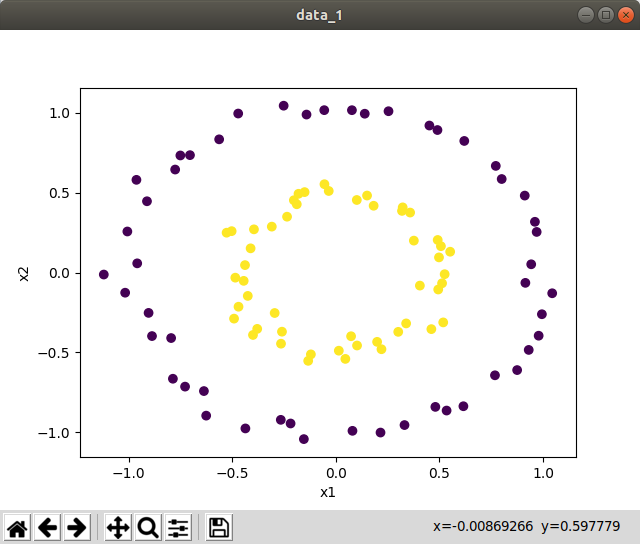
**Report**

**1.**

**i. Dataset Analysis and plotting:**

**data\_1**:

****

Balance: Yes

Number of samples: 100

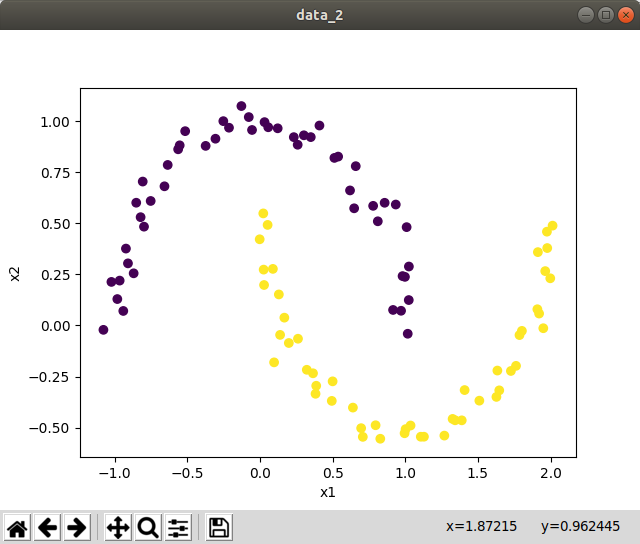
Dimension: 2 dimensions

Separability: Non-Linear

Noise: No

Classes: {0,1}

**data\_2**:

****

Balance:Yes

Number of samples:100

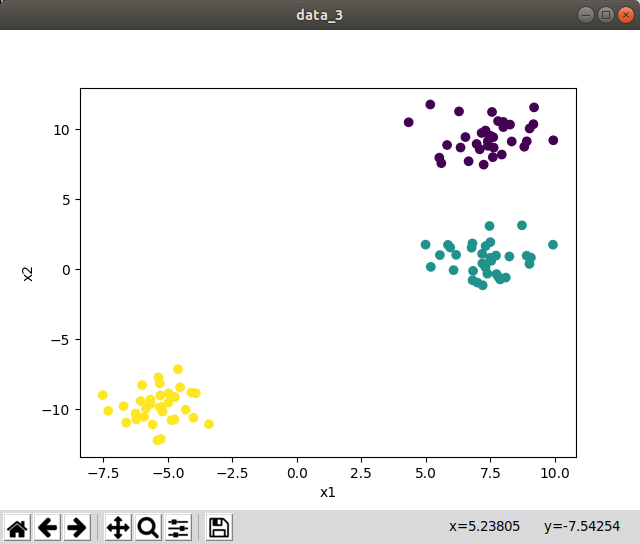
Dimension: 2 dimensions

Separability: Non Linear

Noise: No

Classes: {0,1}

**data\_3**:

****

Balance:No

Number of samples:100

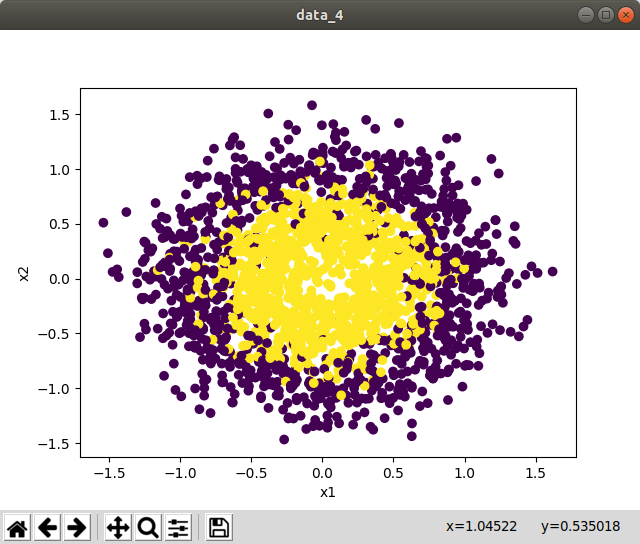
Dimension: 2 dimensions

Separability: Linear

Noise: No

Classes: {0,1,2}

**data\_4**:

****

Balance:Yes

Number of samples:2000

Dimension: 2 dimensions

Separability: Non Linear

Noise: Yes

Classes: {0,1}

**ii. SVM Decision boundaries for Datasets:**

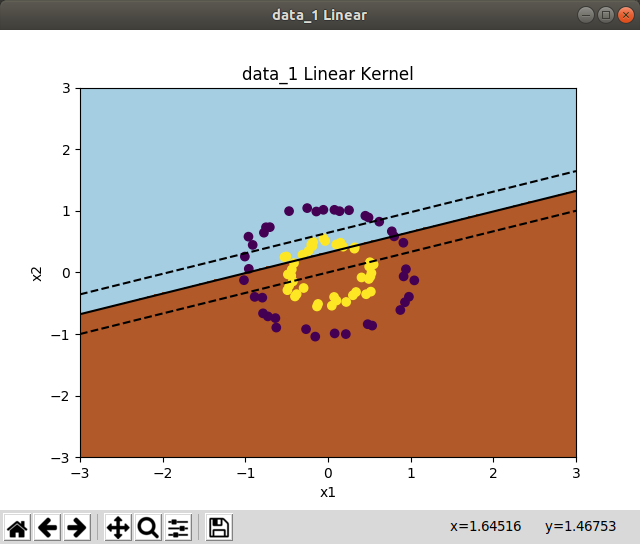
By observing the dataset, I chose following kernel and implemented by taking appropriate parameters(constant , degree)

|  |  |  |
| --- | --- | --- |
| Dataset | Custom Kernel | Decision Boundary |
| Data\_1 | Polynomial  (c=1,degree=2) |  |
| data\_2 | Polynomial  (c=1,degree=3) |  |
| data\_3 | Linear |  |
| data\_4 | Polynomial  (c=1,  degree=2) |  |

Used Custom Polynomial kernel: np.power((c+np.dot(x,y.T)),d)

**iii. SVM with Linear Kernel:**

**data\_1:**

****

F1-Score: 0.56

Accuracy:0.45

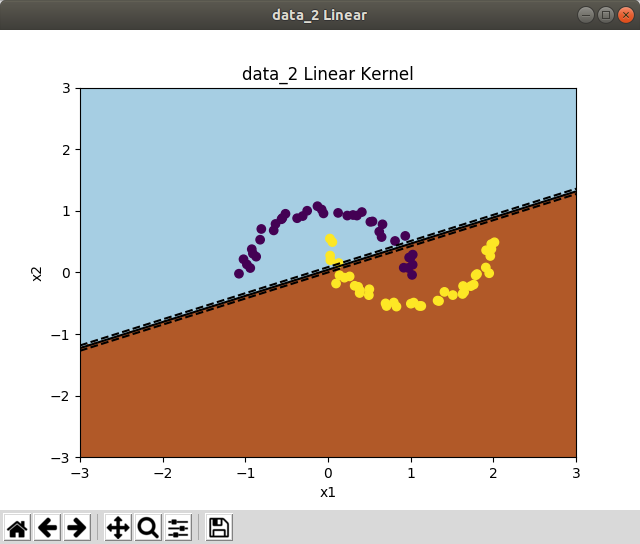
bias(b) : 0.38793275

Confusion matrix:

TP: 2 FP: 9

TN:2 TN: 7

**data\_2:**

****

F1-Score: 0.8750000000000001

Accuracy:0.9

bias(b) : 0.08580343

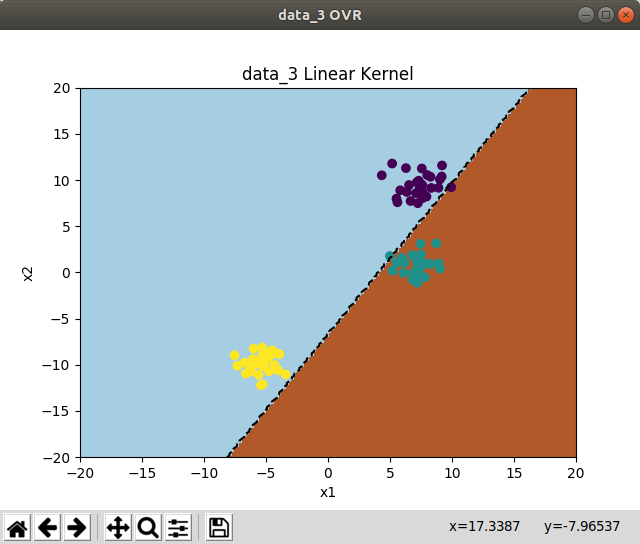
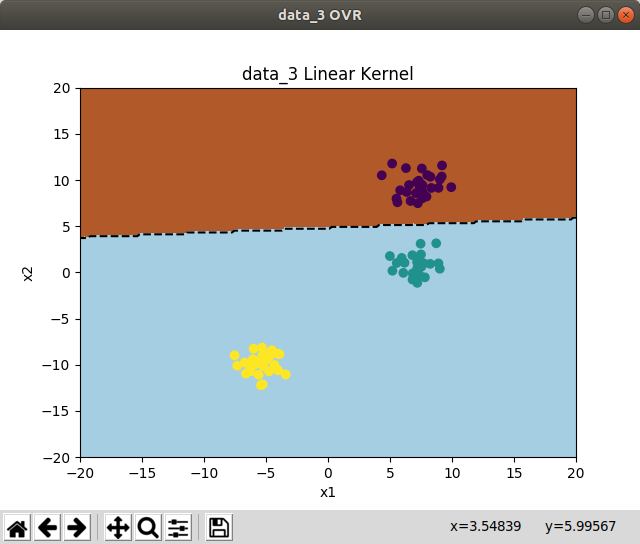
Confusion matrix:

TP: 11 FP: 0

TN:2 TN: 7

**data\_3 OVR:**

Accuracy for all individual plotting are 1.0

****

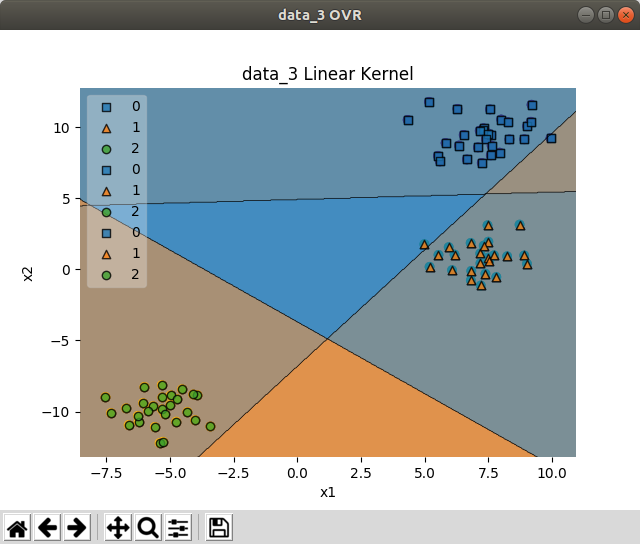
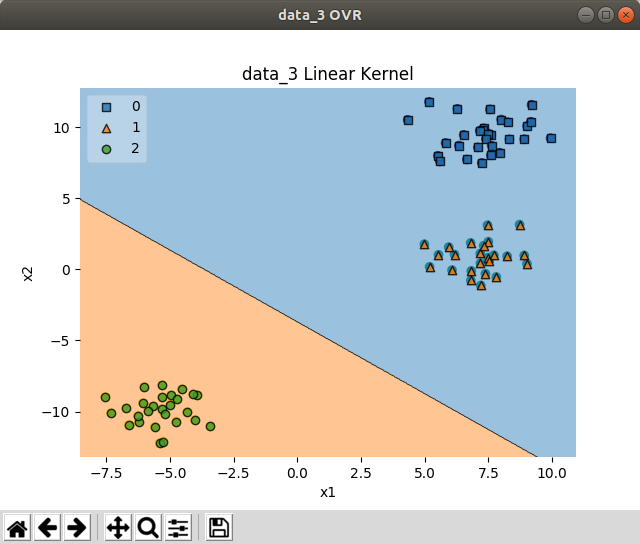
class 0 vs rest: class 1 vs Rest:

bias: -2.23721209 bias: -5.01318078

Confusion3: confusion3:

[[15 0] [[11 0]

[ 0 5]] [ 0 9]]

****

bias : -0.40477513

confusion3:

[[14 0]

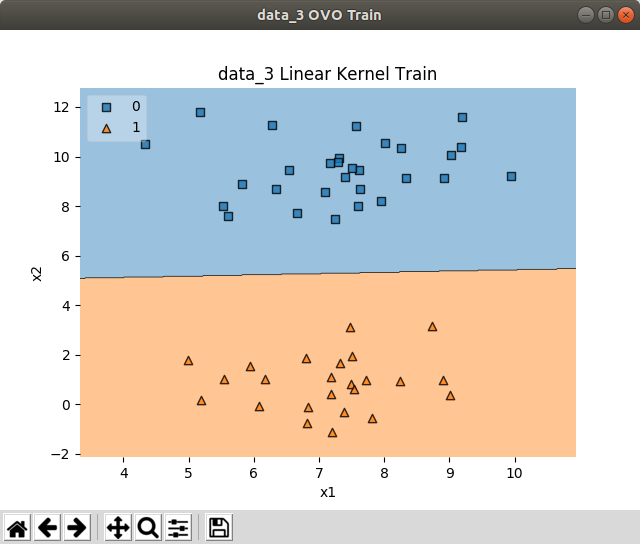
[ 0 6]]

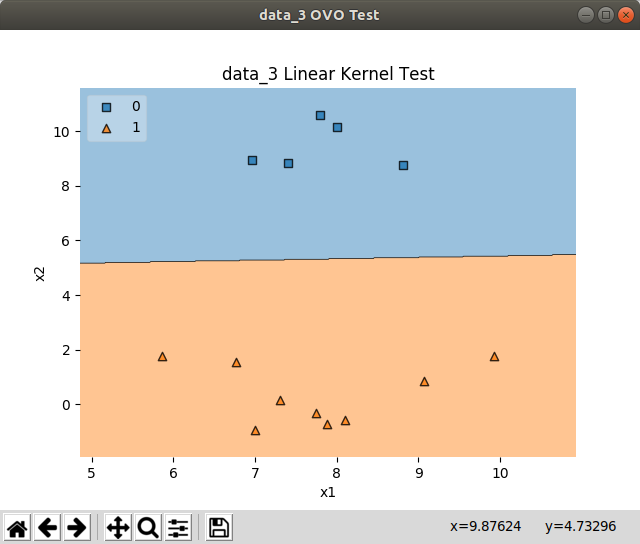
**data\_3 OVO:**

Accuracy for all individual plotting are 1.0

**class 0 vs 1 Train and Test**

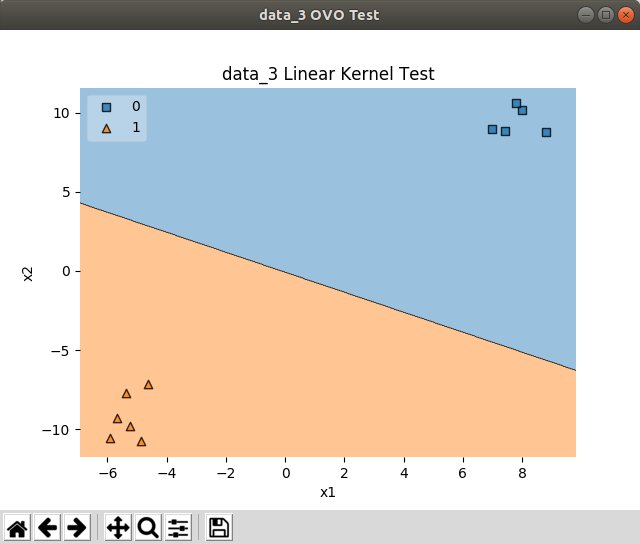
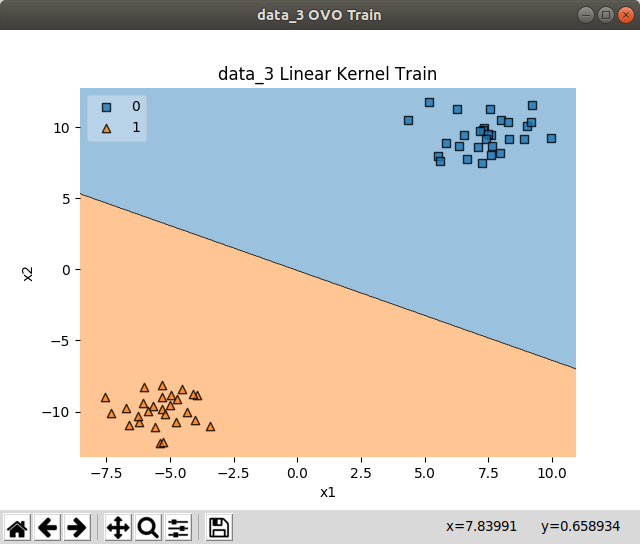
bias : 2.23721216

****

****

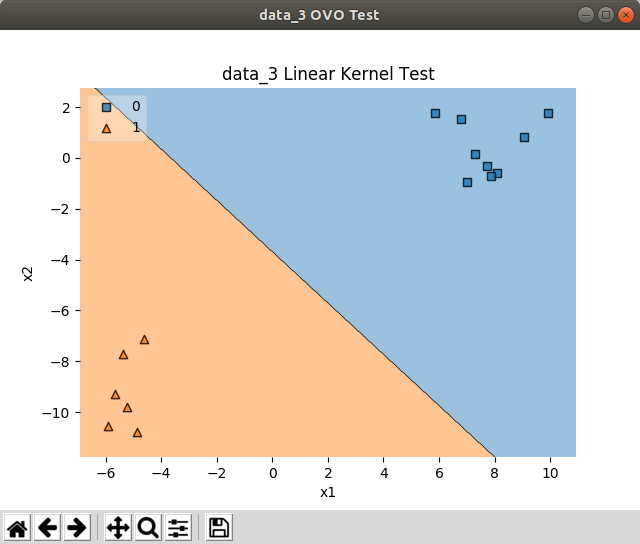
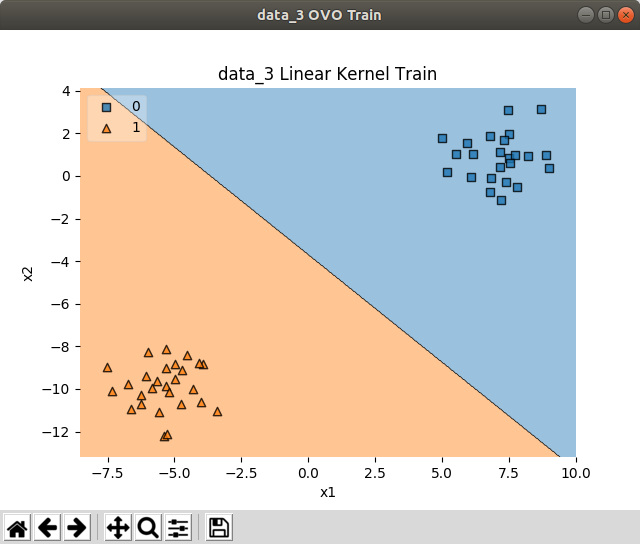
**class 0 vs 2 Train and Test:**

bias : -0.00693931

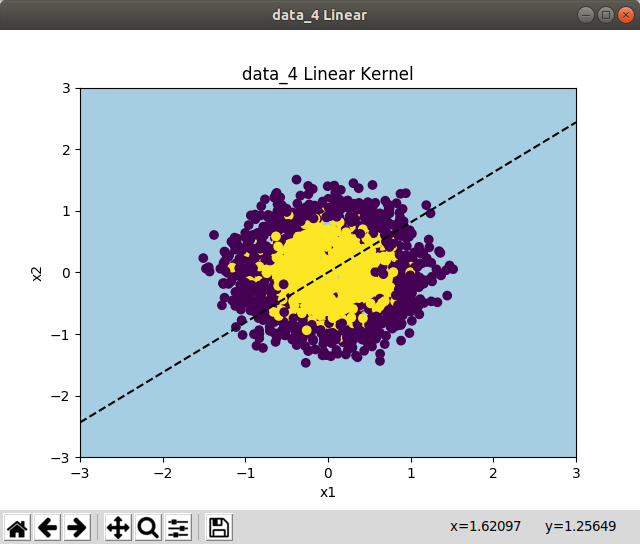


**class 1 vs 2 Train and Test:**

bias : -0.40477513



**data\_4:**



Accuracy for test: 0.4625

confusion4:

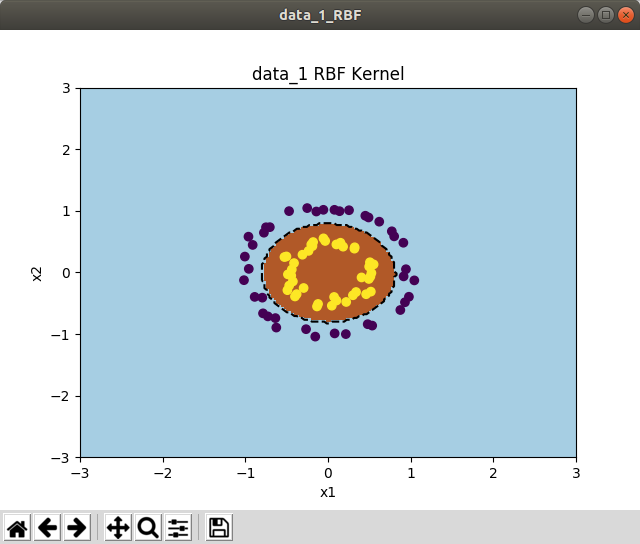
[[ 0 11]

[ 0 9]]

bias4: [0.99993001]

**iv. SVM with RBF Kernel:**

**data\_1:**

****

Accuracy for test: 1.0

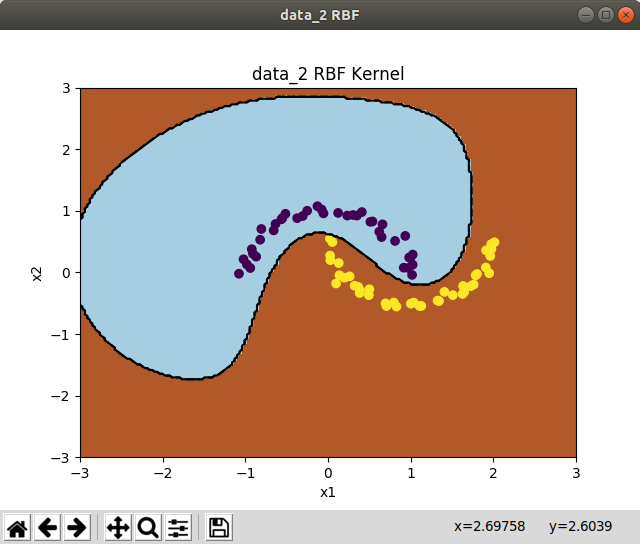
confusion1:

[[11 0]

[ 0 9]]

bias1: [-2.6212776]

**data\_2:**

****

f1 Score: 0.9411764705882353

Accuracy for test: 1.0

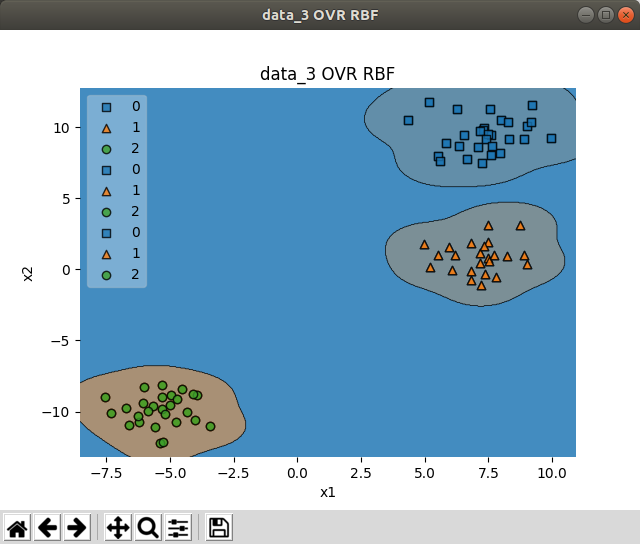
confusion2:

[[11 0]

[ 0 9]]

bias2: [0.11650227]

**data\_3 OVR:**

****

class 0 vs Rest:

F1 : 1.0

bias3: [-0.22586886]

class 1 vs Rest:

F1: 1.0

intercept 3: [-0.38168894]

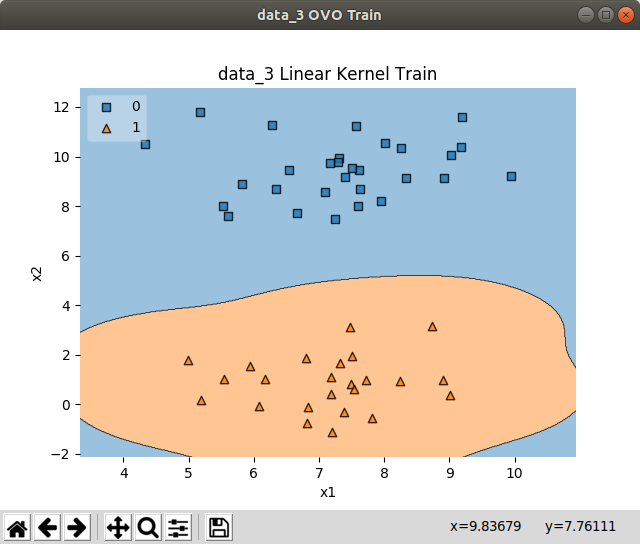
class 2 vs Rest:

F1: 1.0

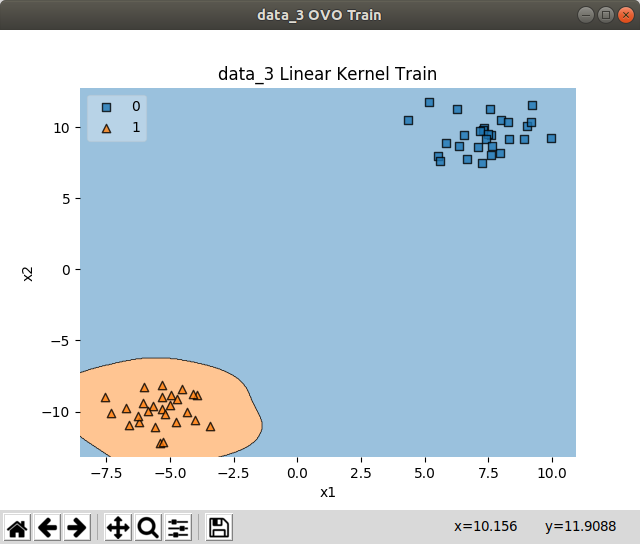
bias3: [-0.40387025]

**data\_3 OVO:**

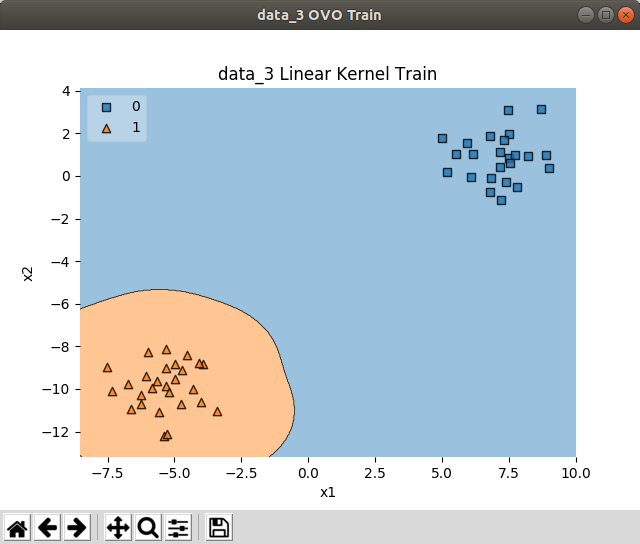
**class 0 vs 1:**

****

**class 0 vs 2:**

****

**class 1 vs 2:**

****

class 0 vs 1:

F1 : 1.0

intercept 3: [-0.11057571]

class 0 vs 2:

F1 1.0

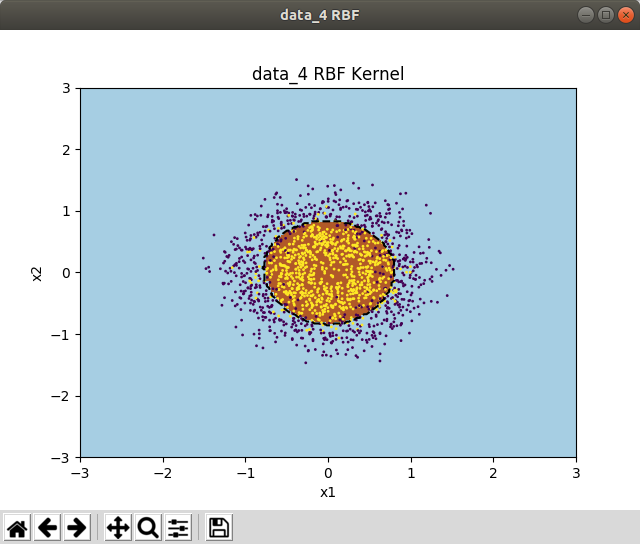
intercept 3: [-0.12318428]

class 1 vs 2:

F1: 1.0

intercept 3: [-0.012874]

**data\_4:**

****

F1 score: 0.8475452196382429

Accuracy for test: 0.8525

confusion matrix:

[[9 2]

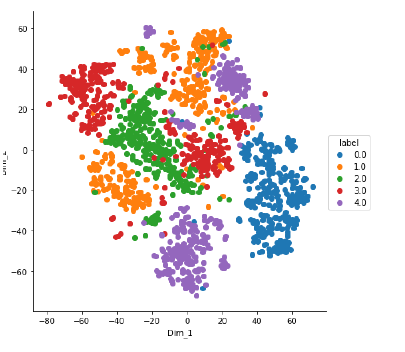
[0 9]]

bias: [-4.18883444]

**v. Hindi Handwritten Characters using RBF:**

In this, there are 5 hindi characters. So, I labelled them as classes from 0 to 4.

Initially loaded the data images and vectorized them and then trained on SVM using RBF Kernel.

**T-SNE:**

here I used cross validation=2

Validation error in fold:1: 0.85

Validation error in fold2:0.86