**ML Assignment 2**

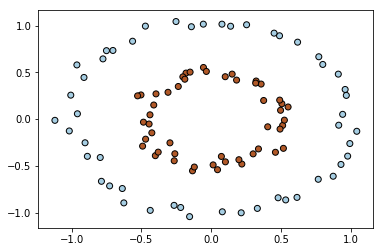
William Scott – MT18026

**1 – Plotting the datasets**

**Dataset 1:**

Observations:

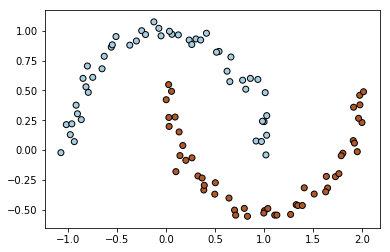
* Dataset size – 100.
* Separability: Separable
* Balance: Balanced
* Noise: Less Noise



**Dataset 2:**

Observations:

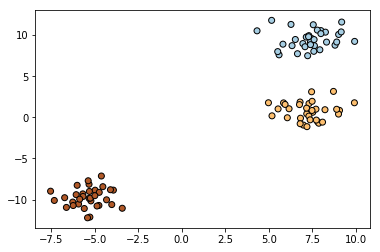
* Dataset size – 100.
* Separability: Separable
* Balance: Balanced
* Noise: Less Noise



**Dataset 3:**

Observations:

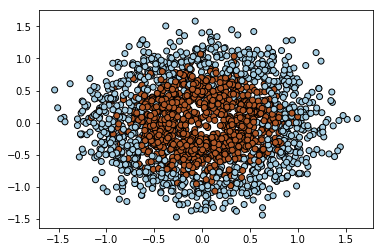
* Dataset size – 100.
* Separability: Separable
* Balance: Balanced
* Noise: Less Noise



**Dataset 4:**

Observations:

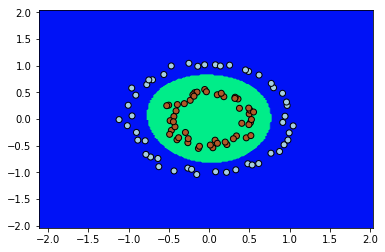
* Dataset size – 2000.
* Separability: Partially Separable
* Balance: Balanced
* Noise: 20% Noise



**2 – Building a kernel to separate the data linearly.**

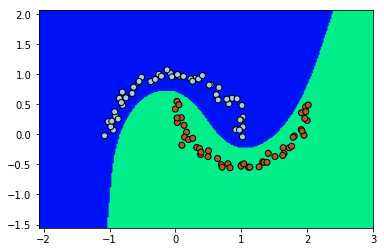
**Dataset 1:**

* Poly Kernel, dimension 2
* As dimension 2 will make it a circle
* Score: 1.0
* F1 Score: 1.0
* Accuracy: 100



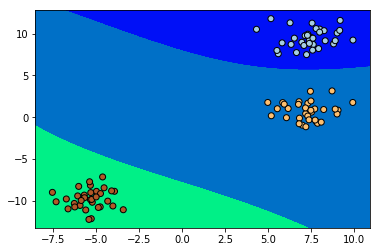
**Dataset 2:**

* Poly Kernel, Dimension 3
* Score: 1.0
* F1 Score: 1.0
* Accuracy: 100



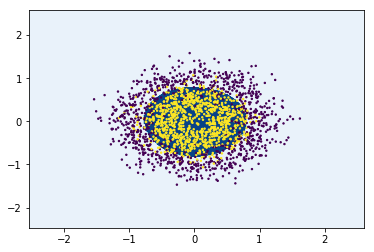
**Dataset 3:**

* Poly Kernel, Dimension 3
* Score: 1.0
* F1 Score: 1.0
* Accuracy: 100



**Dataset 4:**

* Poly Kernel, dimension 2
* Score: 0.887
* F1 Score: 0.8869959318535466
* Accuracy: 1774



**3 – Implementing Soft margin SVM with Linear Kernel**

The below graphs just show the linear margin and soft margins for datasets 1, 2, 4. Plotting the soft margins is not mandatory as per mentioned in the discussion.

**Dataset 1:**

F1 Score: 0.2793103448275862

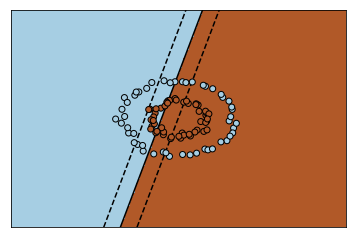
Accuracy: 9

Confusion Matrix

[9, 11, 20]

[0, 0, 0]

[9, 11, 0]



**Dataset 2:**

F1 Score: 1.0

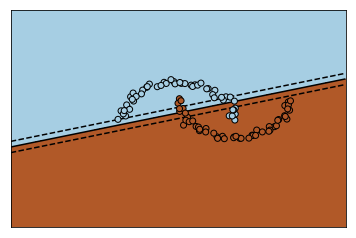
Accuracy: 20

Confusion Matrix

[12, 0, 12]

[0, 8, 8]

[12, 8, 0]



**Dataset 2:**

F1 Score: 0.49268085985944615

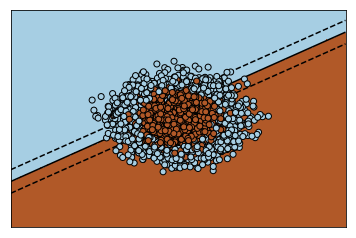
Accuracy: 204

Confusion Matrix

[66, 60, 126]

[136, 138, 274]

[202, 198, 0]



**One vs One – Linear**

Note:

* Data is split to 80 – 20, Train – Test data randomly

**Dataset 1:**

F1 Score: 0.5

Accuracy: 10

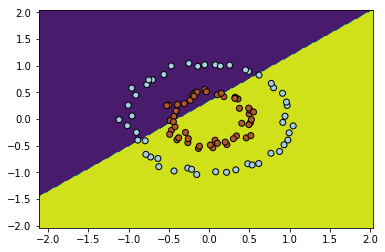
[0 0 0 1 0 0 0 1 0 0] [1, 1, 1, 0, 0, 0, 0, 1, 1, 1]

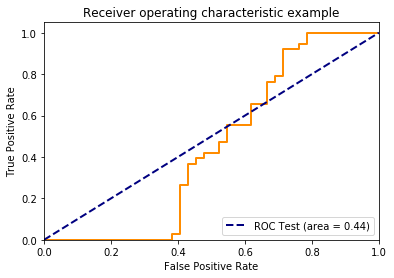
Confusion Matrix

[5, 5, 10]

[5, 5, 10]

[10, 10, 0]





**Dataset 2:**

F1 Score: 0.848849104859335

Accuracy: 17

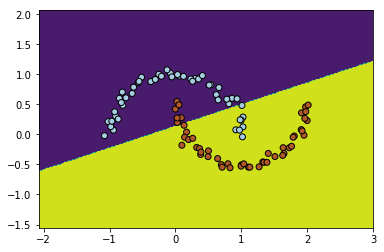
[0 1 0 1 0 0 0 0 1 1] [0, 0, 0, 1, 0, 0, 0, 0, 1, 1]

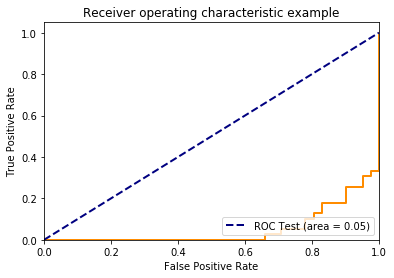
Confusion Matrix

[10, 2, 12]

[1, 7, 8]

[11, 9, 0]



****

**Dataset 3:**

F1 Score: 1.0

Accuracy: 20

[2 1 0 2 0 1 0 1 2 0] [2, 1, 0, 2, 0, 1, 0, 1, 2, 0]

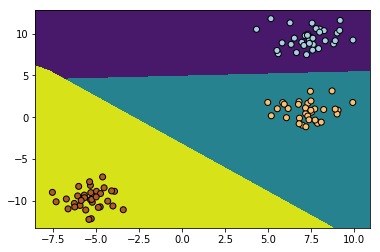
Confusion Matrix

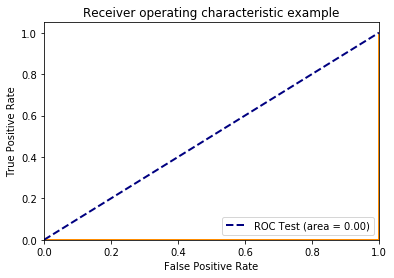
[9, 0, 0, 9]

[0, 5, 0, 5]

[0, 0, 6, 6]

[9, 5, 6, 0]



****

**Dataset 4:**

F1 Score: 0.5025898085140055

Accuracy: 203

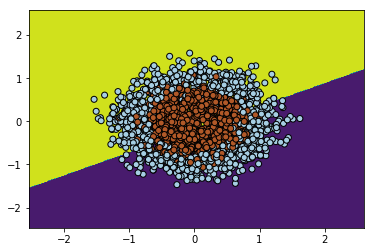
[0 0 1 0 0 1 0 1 0 0] [1, 0, 1, 1, 1, 0, 0, 1, 1, 0]

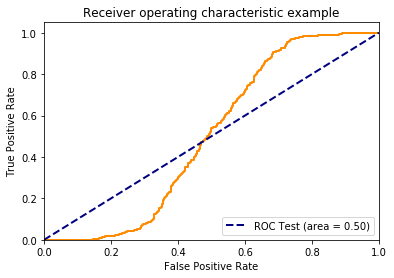
Confusion Matrix

[79, 81, 160]

[116, 124, 240]

[195, 205, 0]





**One vs Rest – Linear**

**Dataset 1:**

F1 Score: 0.2787878787878788

Accuracy: 6

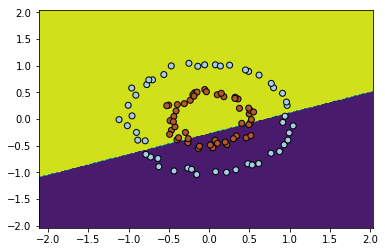
[0 0 0 0 1 1 0 1 0 1] [1, 1, 1, 0, 1, 0, 1, 0, 1, 1]

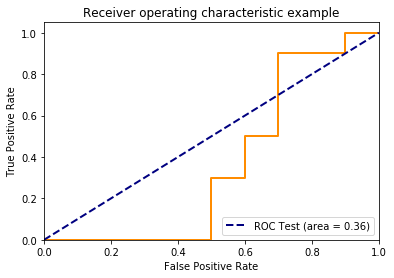
Confusion Matrix

[2, 4, 6]

[10, 4, 14]

[12, 8, 0]



****

**Dataset 2:**

F1 Score: 0.8534526854219949

Accuracy: 17

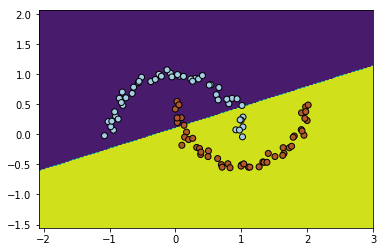
[1 0 1 1 0 0 1 0 0 0] [1, 0, 1, 1, 1, 0, 1, 1, 0, 0]

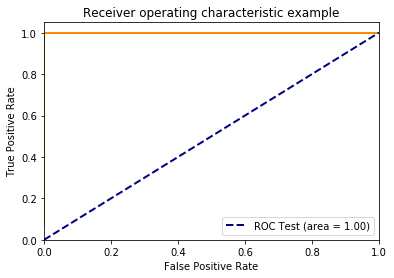
Confusion Matrix

[10, 0, 10]

[3, 7, 10]

[13, 7, 0]



****

**Dataset 3:**

F1 Score: 1.0

Accuracy: 20

[1 0 0 2 2 2 0 0 0 2] [1, 0, 0, 2, 2, 2, 0, 0, 0, 2]

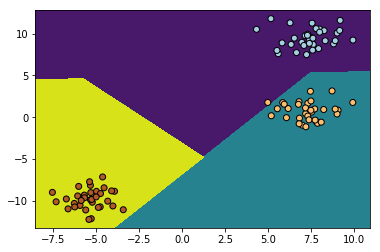
Confusion Matrix

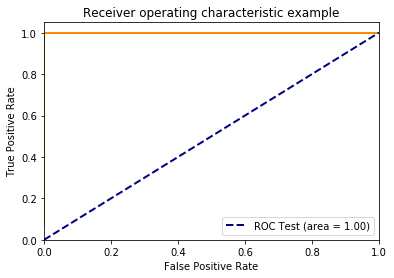
[7, 0, 0, 7]

[0, 3, 0, 3]

[0, 0, 10, 10]

[7, 3, 10, 0]



****

**Dataset 4:**

F1 Score: 0.5250099850855684

Accuracy: 212

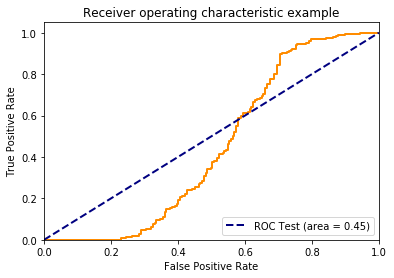
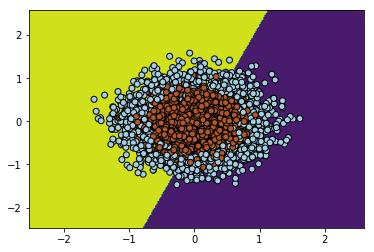
[1 0 0 1 1 1 1 0 0 0] [1, 1, 0, 1, 1, 0, 1, 1, 1, 0]

Confusion Matrix

[85, 74, 159]

[114, 127, 241]

[199, 201, 0]



**Part 4 – Implementing Soft margin SVM with RBF Kernel**

**One vs Rest – RBF**

**Dataset 1:**

F1 Score: 1.0

Accuracy: 20

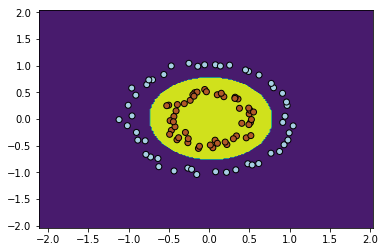
[1 1 0 1 1 0 1 1 1 1] [1, 1, 0, 1, 1, 0, 1, 1, 1, 1]

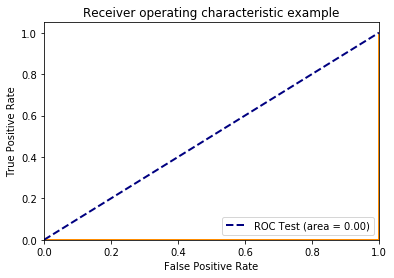
Confusion Matrix

[7, 0, 7]

[0, 13, 13]

[7, 13, 0]



****

**Dataset 2:**

F1 Score: 1.0

Accuracy: 20

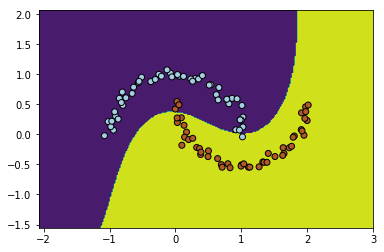
[1 1 0 1 0 1 1 0 0 1] [1, 1, 0, 1, 0, 1, 1, 0, 0, 1]

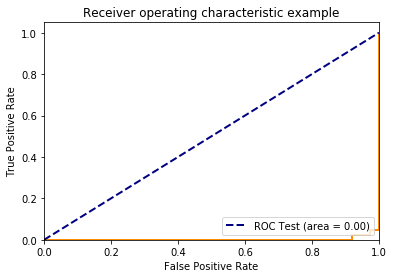
Confusion Matrix

[10, 0, 10]

[0, 10, 10]

[10, 10, 0]



****

**Dataset 3:**

F1 Score: 1.0

Accuracy: 20

[0 1 2 1 0 0 2 2 1 2] [0, 1, 2, 1, 0, 0, 2, 2, 1, 2]

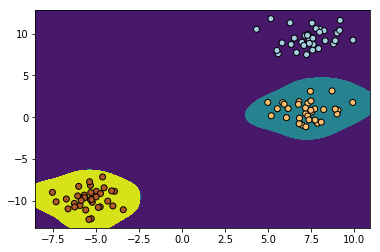
Confusion Matrix

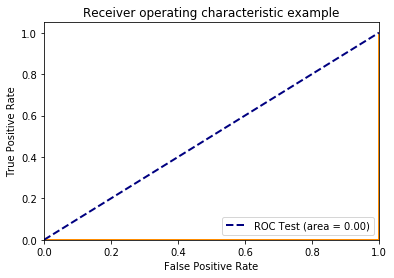
[9, 0, 0, 9]

[0, 6, 0, 6]

[0, 0, 5, 5]

[9, 6, 5, 0]





**Dataset 4:**

F1 Score: 0.87

Accuracy: 348

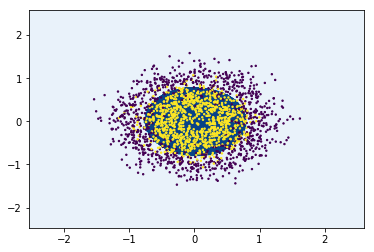
[0 1 1 0 0 0 1 0 1 1] [0, 1, 1, 0, 0, 0, 1, 0, 1, 1]

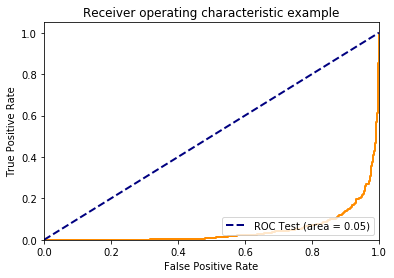
Confusion Matrix

[163, 26, 189]

[26, 185, 211]

[189, 211, 0]





**One vs One – RBF**

**Dataset 1:**

F1 Score: 1.0

Accuracy: 20

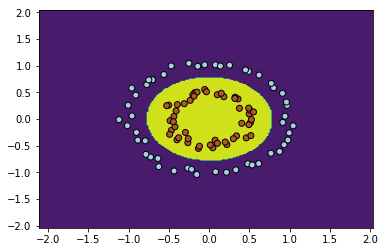
[1 0 0 0 0 1 0 1 1 0] [1, 0, 0, 0, 0, 1, 0, 1, 1, 0]

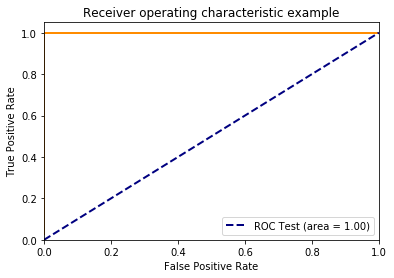
Confusion Matrix

[9, 0, 9]

[0, 11, 11]

[9, 11, 0]



****

**Dataset 2:**

F1 Score: 1.0

Accuracy: 20

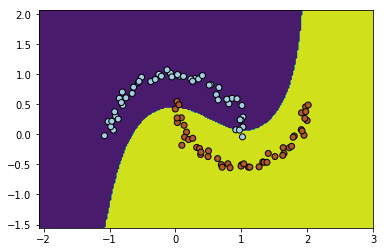
[1 1 0 1 0 1 1 0 1 1] [1, 1, 0, 1, 0, 1, 1, 0, 1, 1]

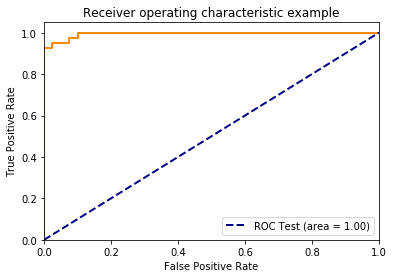
Confusion Matrix

[7, 0, 7]

[0, 13, 13]

[7, 13, 0]



****

**Dataset 3:**

F1 Score: 1.0

Accuracy: 20

[0 2 0 0 1 1 2 2 2 0] [0, 2, 0, 0, 1, 1, 2, 2, 2, 0]

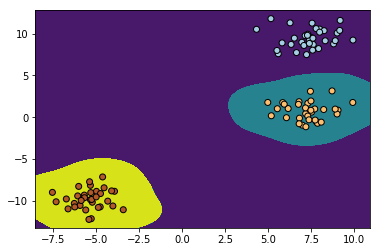
Confusion Matrix

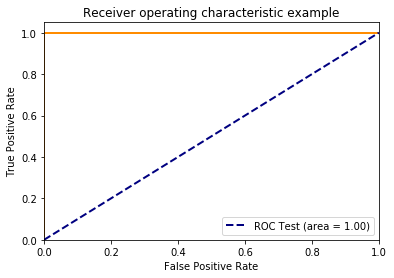
[8, 0, 0, 8]

[0, 5, 0, 5]

[0, 0, 7, 7]

[8, 5, 7, 0]



****

**Dataset 4:**

F1 Score: 0.87

Accuracy: 348

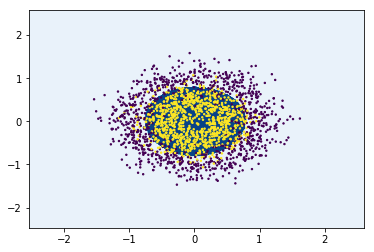
[0 1 1 0 0 0 1 0 1 1] [0, 1, 1, 0, 0, 0, 1, 0, 1, 1]

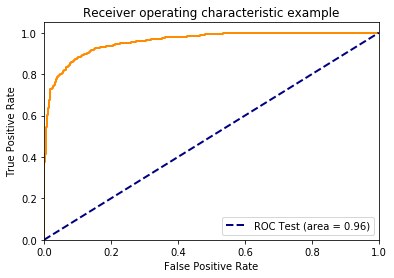
Confusion Matrix

[163, 26, 189]

[26, 185, 211]

[189, 211, 0]





**PART 5**

Split: 6800 1700

Training Score 0.9463235294117647

Validation Score 0.9294117647058824

Split: 6800 1700

Training Score 0.9461764705882353

Validation Score 0.9229411764705883

Split: 6800 1700

Training Score 0.9476470588235294

Validation Score 0.9247058823529412

Split: 6800 1700

Training Score 0.9439705882352941

Validation Score 0.9352941176470588

Split: 6800 1700

Training Score 0.9476470588235294

Validation Score 0.9352941176470588

Best Validation Score: 0.9352941176470588

[[289, 4, 2, 4, 1, 300],

[9, 284, 18, 4, 4, 319],

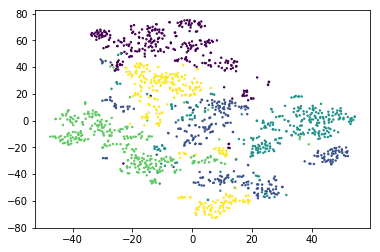
[0, 7, 269, 6, 4, 286],

[1, 3, 8, 284, 5, 301],

[1, 2, 3, 2, 286, 294],

[300, 300, 300, 300, 300, 0]]

Test



Train



**PART 6 Observations:**

* The hyper parameters are given trying different values and then checking the error. The c value is chosen to be large to implement the soft margin
* C value is given incredibly high to find if it is over fitting.
* The performance of RBF and Linear totally depends on the kernel implementation.
* One vs rest seems to be a little computational heavy as it has to compute for more number of chasses taking all the those as one.
* But one vs one has more work to do than the one vs rest
* One vs one has to compute n\*n-1/2 classifiers
* Where as one vs rest has to compute n\*n-1 classifiers