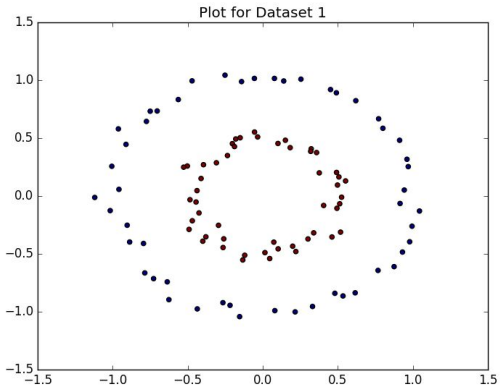
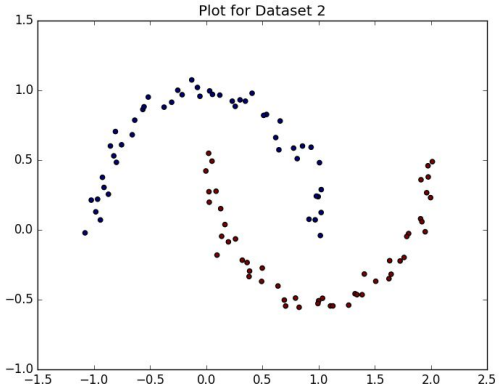
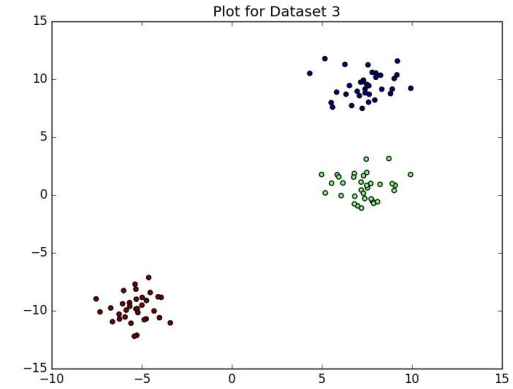
Assignment 2 Report

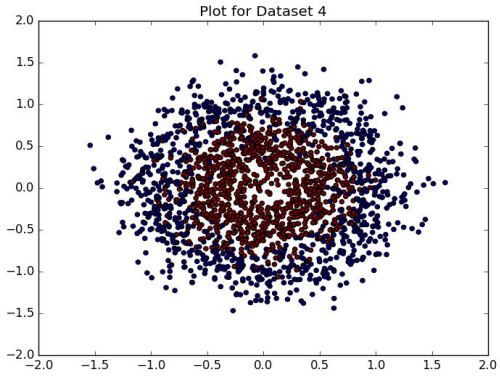
Shubham Khanna [2015179]

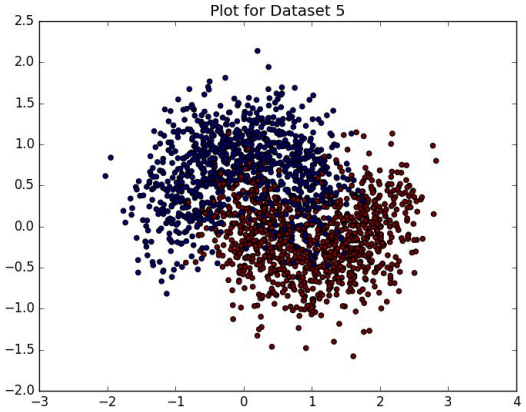
Question 1







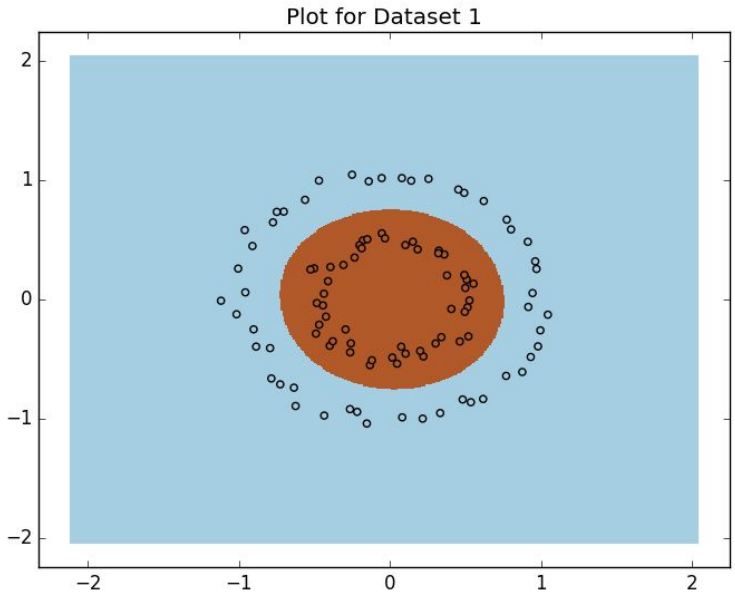




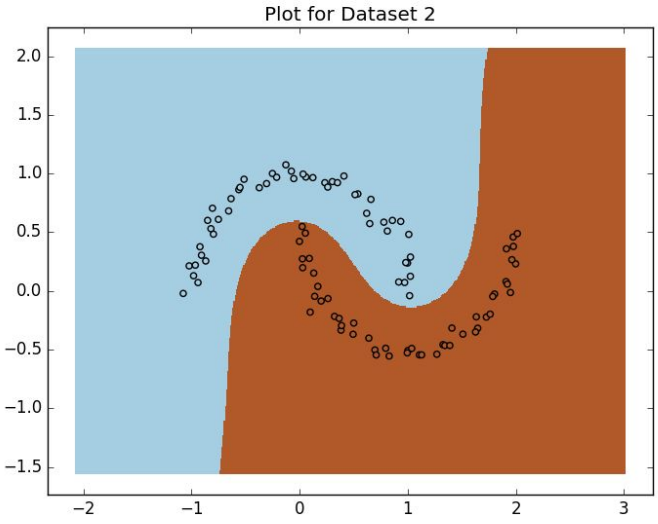
1.1 Dataset - 1: The dataset isn't linearly separable. It can be classified by using a kernel and the decision boundary would be circular. It contains 2 classes without any outliers.
Dataset - 2: They aren't linearly separable. 2 classes
Dataset - 3: They are linearly separable. Contains 3 Classes
Dataset - 4: This data set isn't linearly separable. Contains 2 classes with plenty of outliers

Dataset - 5: This isn't linearly separable. 2 classes with outliers.

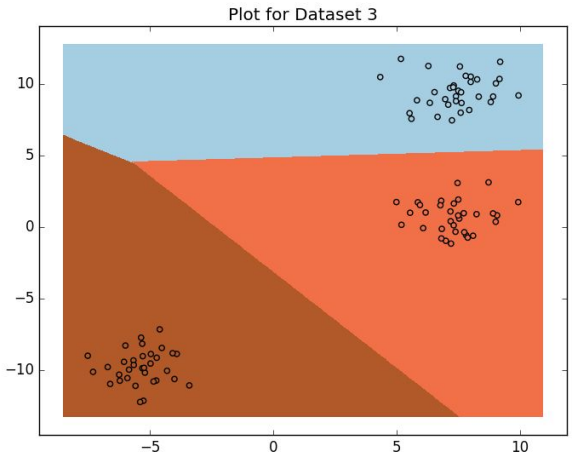
1.2 Dataset - 1: Since it isn't linearly separable, I used a custom kernel which implemented a Gaussian Kernel.

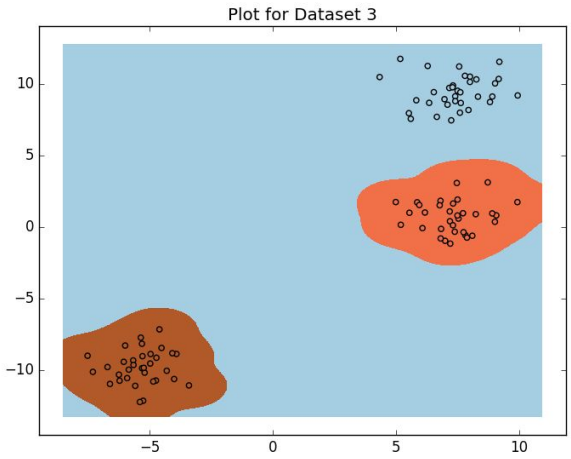


Dataset - 2: Since dataset isn't linearly separable, I used rbf kernel like above and plotted the decision boundaries.

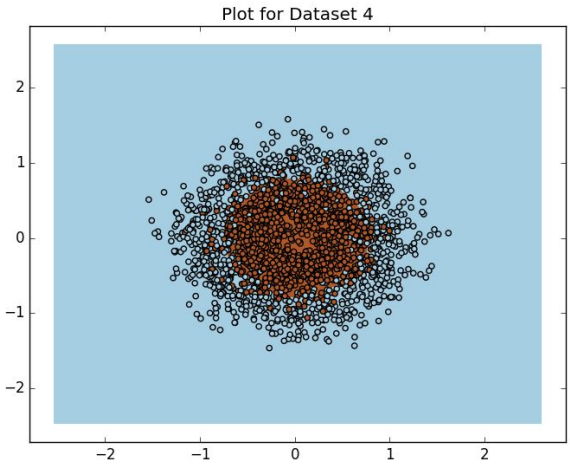


Dataset - 3: Data is linearly separable, Hence I used Linear and Rbf kernels to get different curves for the Multiclass data.

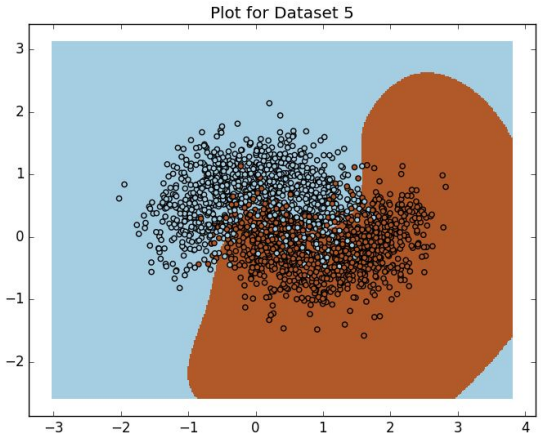




Dataset - 4: This dataset isn't linearly separable. Hence, I used custom Gaussian kernel for the classification.



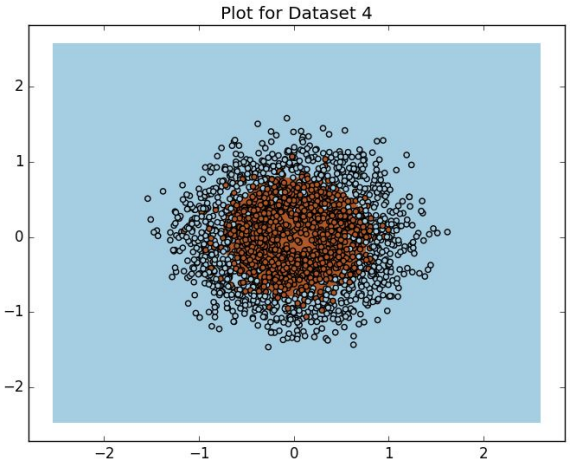
Dataset - 5: This dataset isn't linearly separable. Therefore, the obvious choice for kernel is an Gaussian kernel.

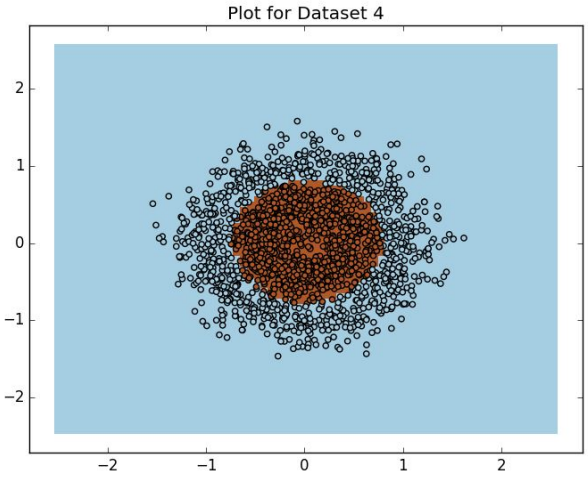


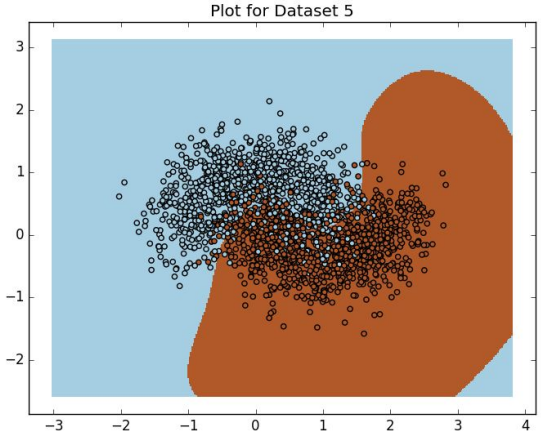
1.3

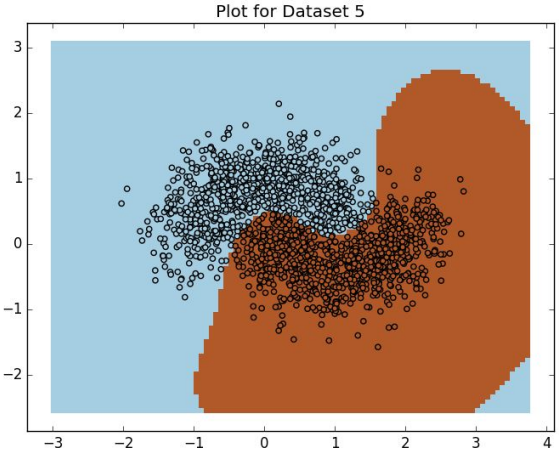
To remove outliers, I tested my model using the training data -> Removed all the incorrect classifications -> re-trained the model with the new training data(which contains all single)

Only dataset 4 and 5 had considerable outliers.





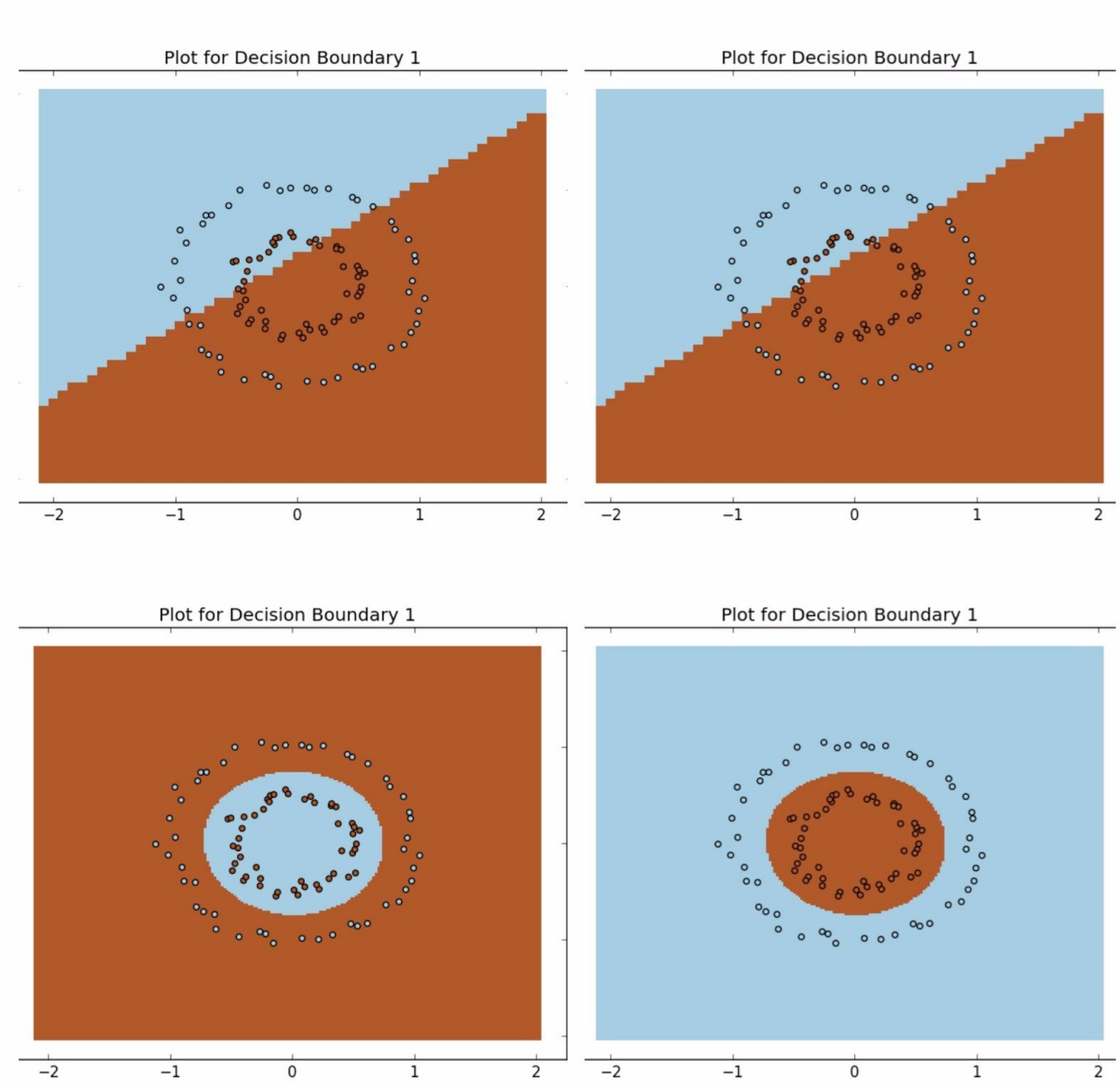




Question 2

Each dataset is trained after randomizing the training data at the start.

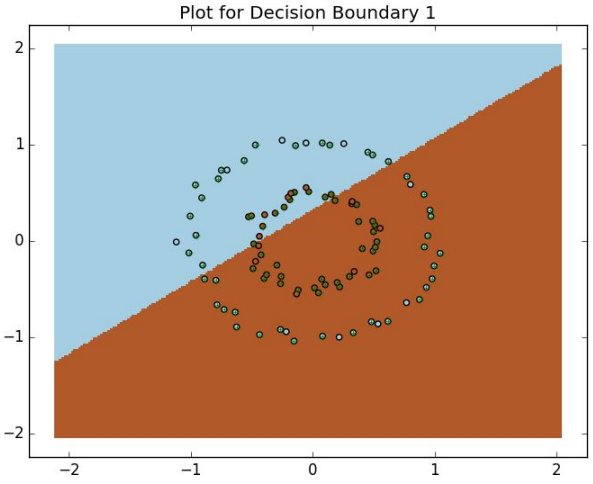
Dataset 1



Plot No	**Model**	**Accuracy**	**Observations**	**Best Parameter(Grid Search)**
1(0,0)	Linear One vs One	**0.38**	Bad accuracy, Decent performance	*C=1*
2(0,1)	Linear One vs Rest	**0.43**	Slightly better than above, but still bad performance.	*C=1*
3(1,1)	RBF One vs One	**1.0**	Excellent Accuracy, Decent speed	*gamma=0.7*
4(1,0)	RBF One vs Rest	**1.0**	Excellent Accuracy, Decent speed	*gamma=0.7*

Linear OvO

[[29. 41.]
 [21.   9.]]



Linear OvR

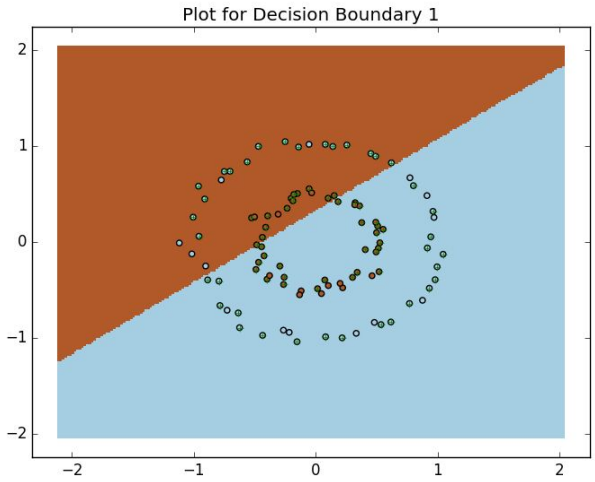
[[12. 18.
 [38. 32.]]

RBF OvO

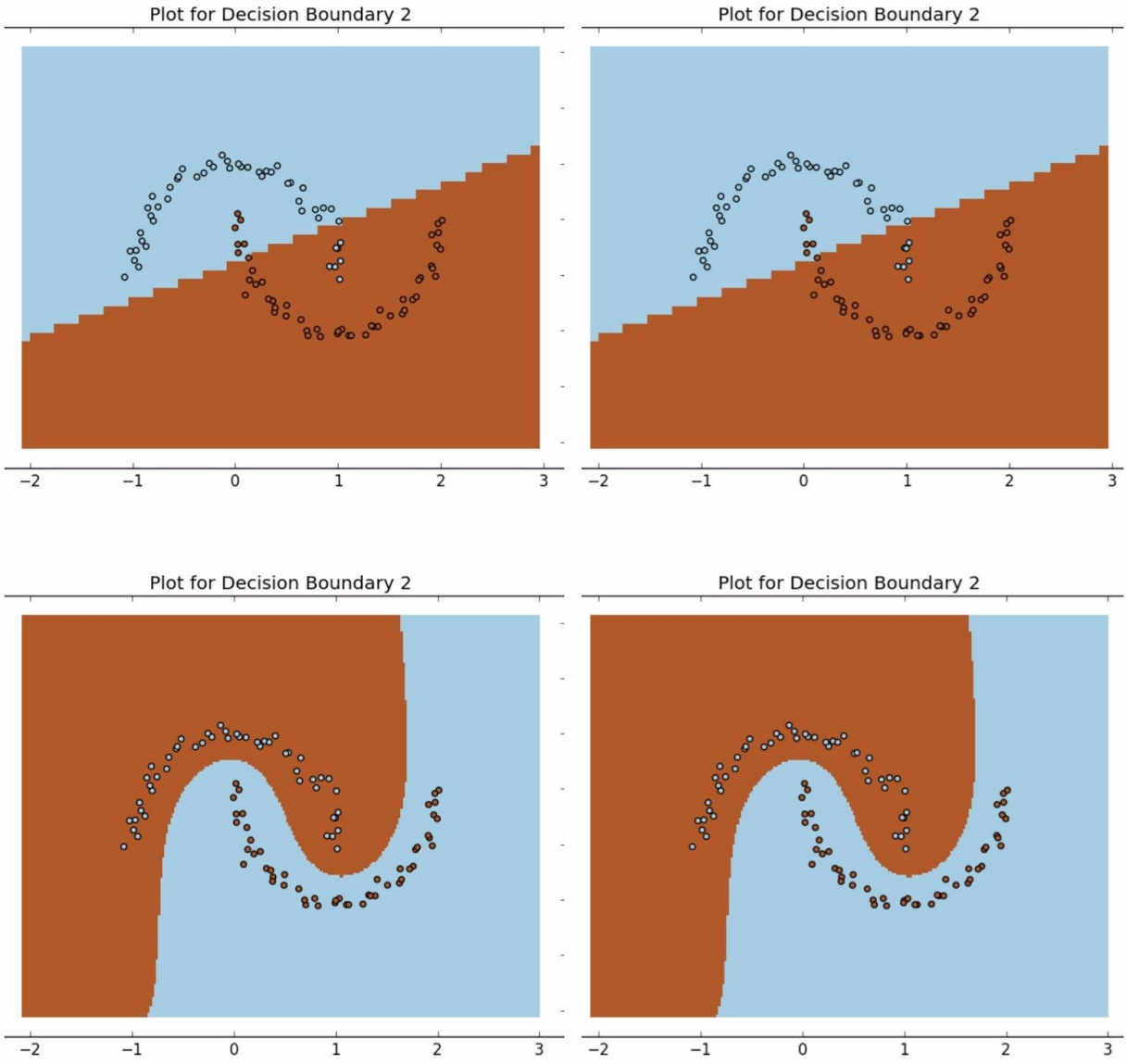
[[50. 0.
 [0. 50.]]

RBF OvR

[[50. 0.
 [0. 50.]]



Dataset 2



Plot No	**Model**	**Accuracy**	**Observations**	**Best Parameter(Grid Search)**
1	Linear One vs One	**0.86**	Decent accuracy, Decent performance	*C=1*
2	Linear One vs Rest	**0.86**	Decent accuracy, decent performance.	*C=1*
3	RBF One vs One	**1.0**	Excellent Accuracy,	*gamma=0.7*

			Decent speed	
4	RBF One vs Rest	**1.0**	Excellent Accuracy, Decent speed	*gamma=0.7*

Confusion Matrices

Linear OvO

[[43. 7.]
 [7. 43.]]

Linear OvR

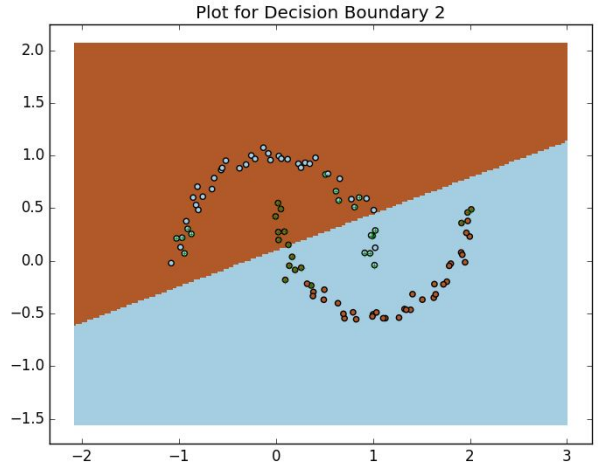
[[43. 7.
 [7. 43.]]

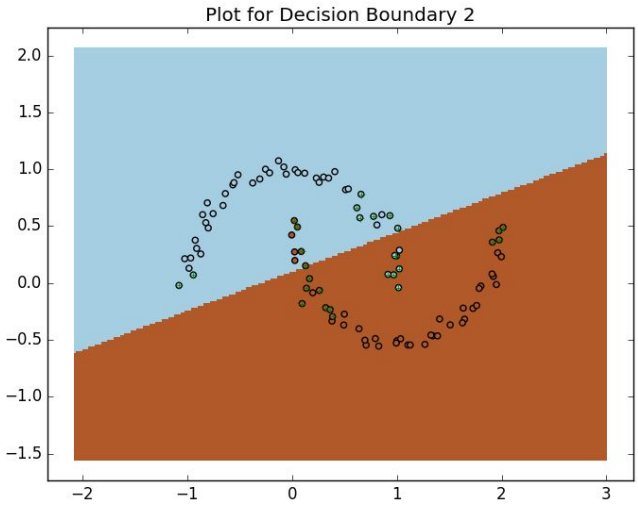
RBF OvO

[[50. 0.
 [0. 50.]]

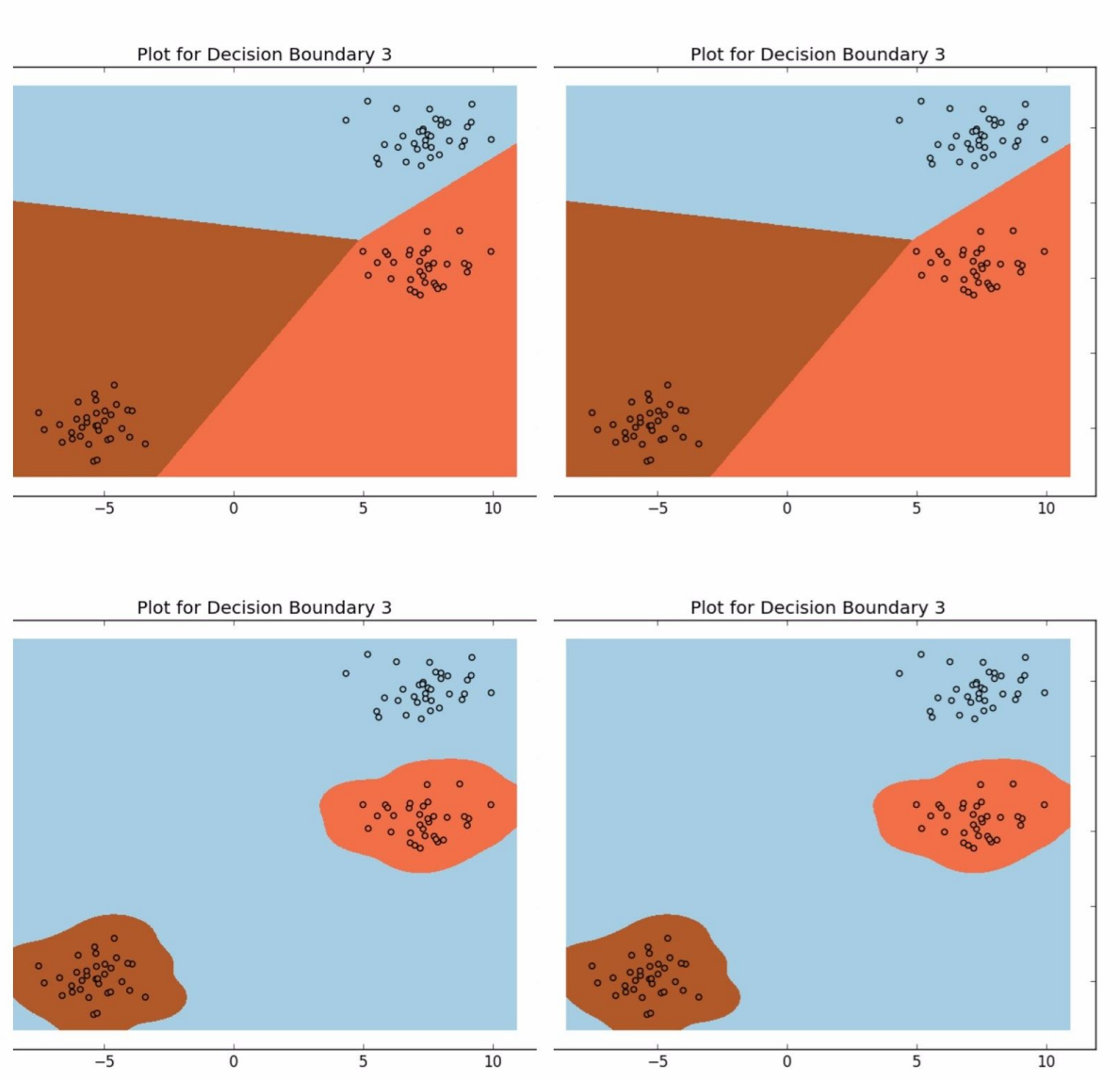
RBF OvR

[[50. 0.
 [0. 50.]]





Dataset 3



Plot No	**Model**	**Accuracy**	**Observations**	**Best Parameter(Grid Search)**
1(0,0)	Linear One vs One	**1.0**	Excellent Accuracy, Good speed	*C=1*
2(0,1)	Linear One vs Rest	**1.0**	Decent accuracy,	*C=1*

			Good speed.	
3(1,0)	RBF One vs One	**1.0**	Excellent Accuracy, Decent speed	*gamma=0.7*
4(1,1)	RBF One vs Rest	**1.0**	Excellent Accuracy, Decent speed	*gamma=0.7*

Confusion Matrices

Linear OvR

[[34. 0. 0.]
 [0. 33. 0.]
 [0. 0. 33.]]

Linear OvO

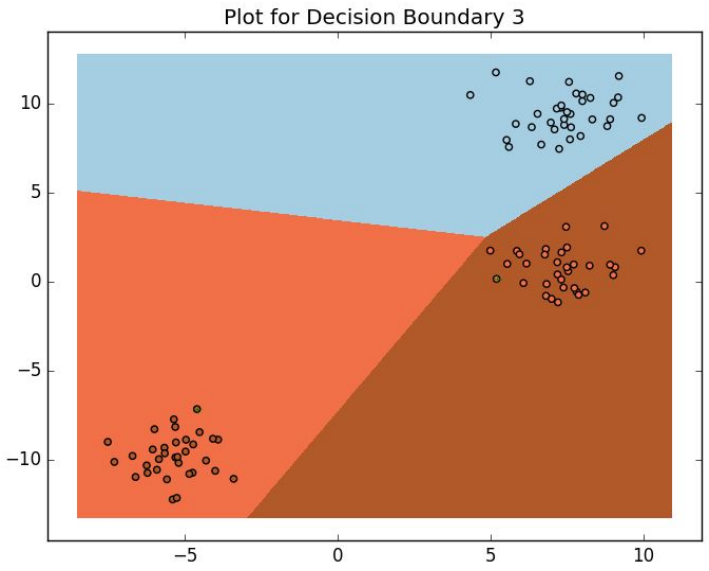
[[34. 0. 0.]
 [0. 33. 0.]
 [0. 0. 33.]]

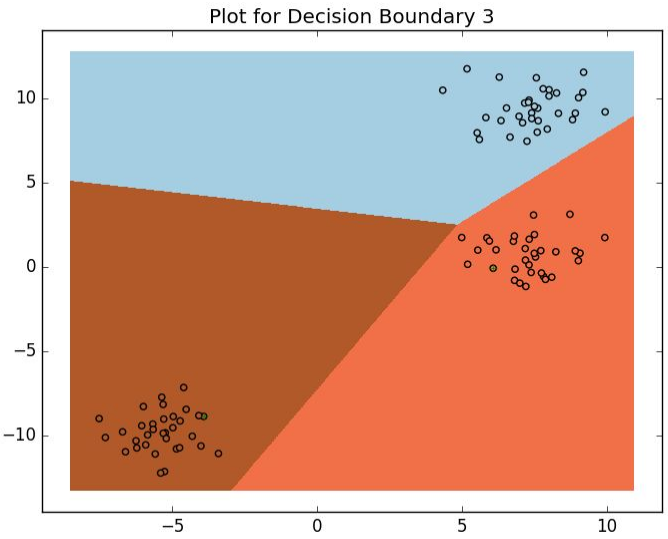
RBF OvO

[[33. 0. 0.]
 [0. 33. 0.]
 [0. 0. 34.]]

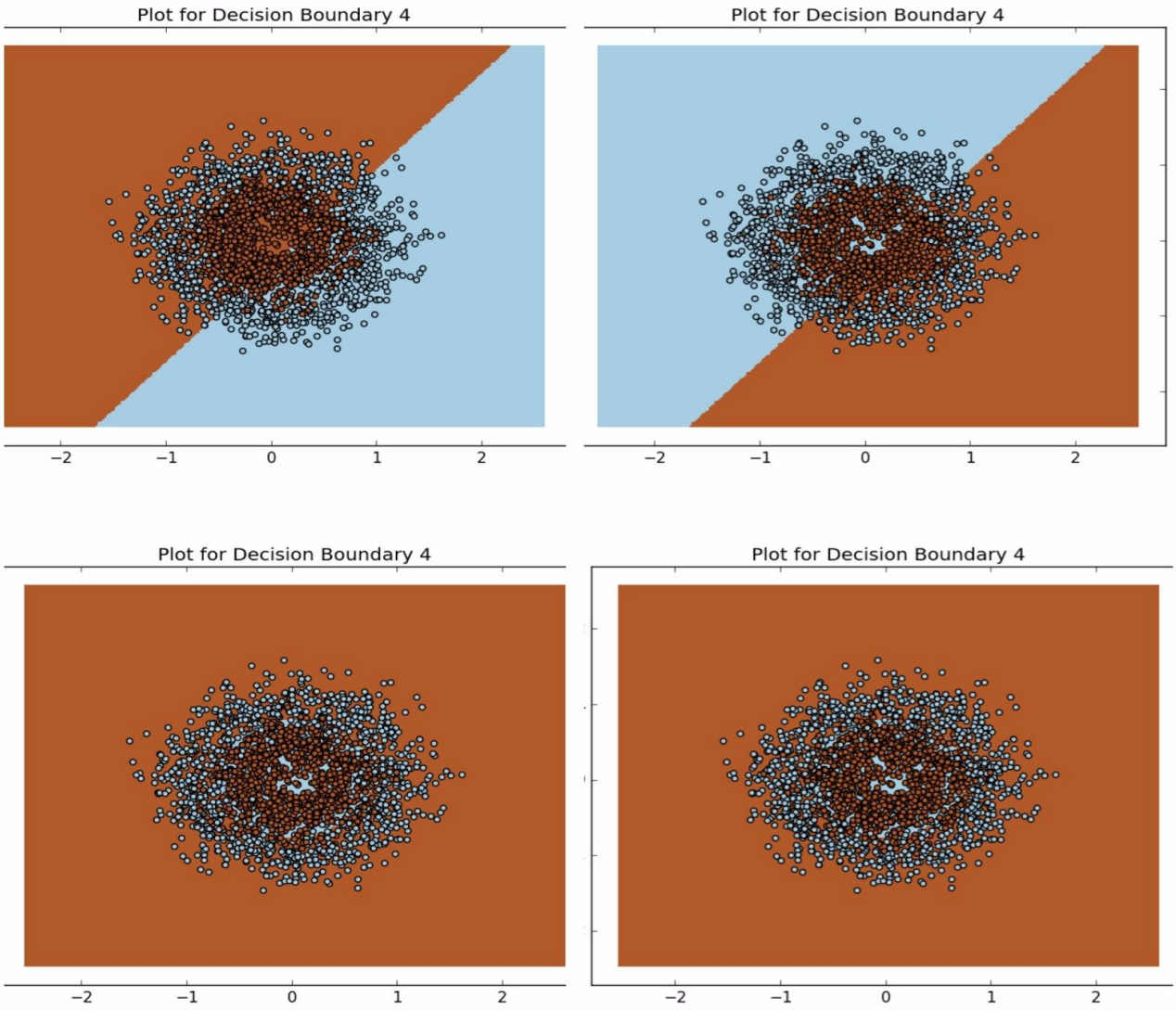
RBF OvR

[[34. 0. 0.]
 [0. 33. 0.]
 [0. 0. 33.]]





Dataset 4



Plot No	**Model**	**Accuracy**	**Observations**	**Best Parameter(Grid Search)**
1(0,0)	Linear One vs One	**0.5315**	Bad Accuracy(linearly inseparable)	*C=1*
2(0,1)	Linear One vs Rest	**0.469**	Bad accuracy	*C=1*

3(1,0)	RBF One vs One	**0.8805**	Good Accuracy, Some outliers, decent speed	*gamma=0.7*
4(1,1)	RBF One vs Rest	**0.8765**	Good Accuracy, Some outliers, decent speed	*gamma=0.7*

Confusion Matrices

Linear OvO

[[523. 478.]
 [477. 522.]]

Linear OvR

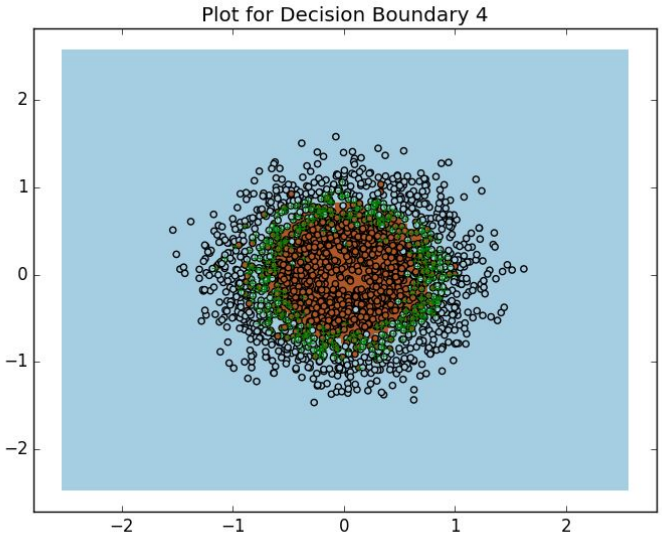
[[535. 557.]
 [465. 443.]]

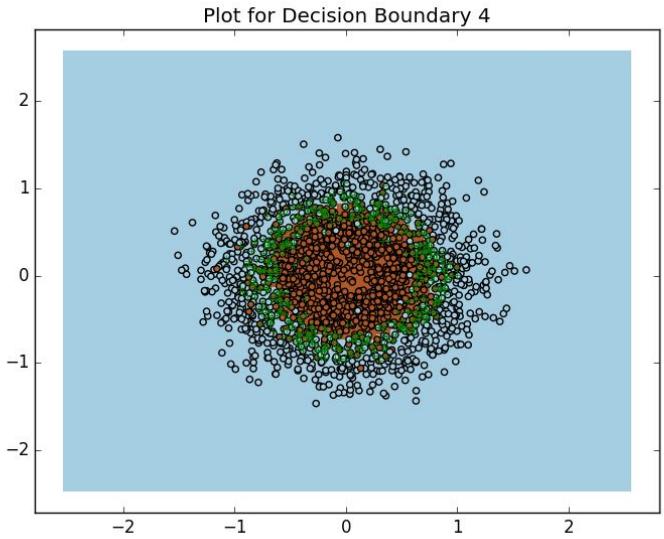
RBF OvO

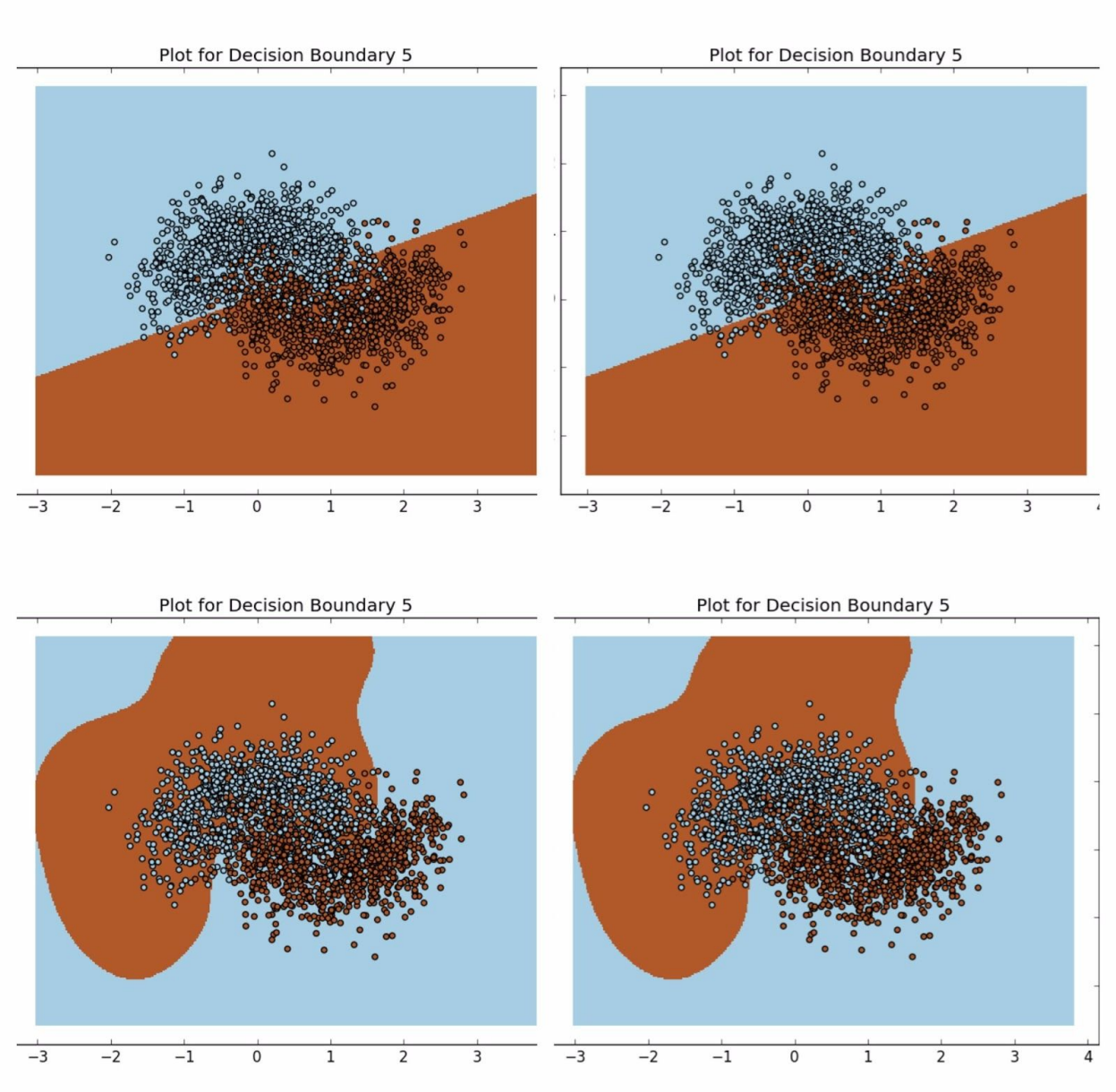
[[915. 150.]
 [85. 850.]]

RBF OvR

[[916. 157.]
 [84. 843.]]







Dataset - 5:

Plot No	**Model**	**Accuracy**	**Observations**	**Best Parameter(Grid Search)**
1(0,0)	Linear One vs One	**0.8445**	Decent Accuracy(linearly	*C=1*

			separable with some outliers)	
2(0,1)	Linear One vs Rest	**0.845**	Decent accuracy	*C=1*
3(1,0)	RBF One vs One	**0.877**	Better Accuracy, Some outliers, decent speed	*gamma=0.7*
4(1,1)	RBF One vs Rest	**0.8805**	Better Accuracy than linear kernel, Some outliers, decent speed	*gamma=0.7*

Linear OvO

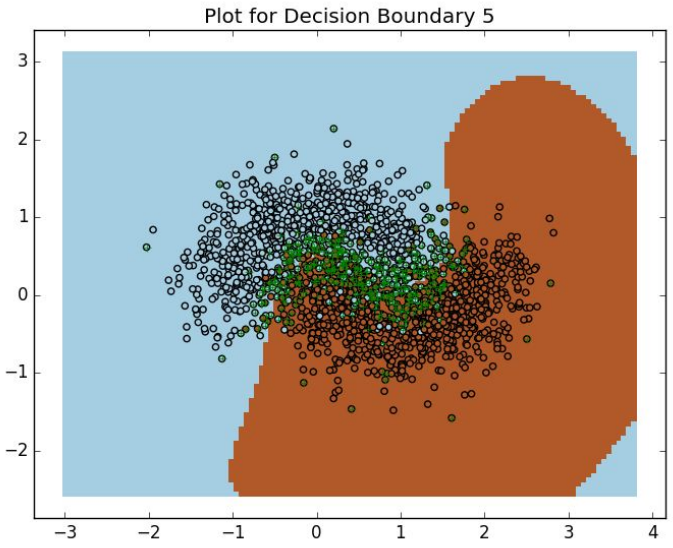
[[840. 151.]
 [160. 849.]]

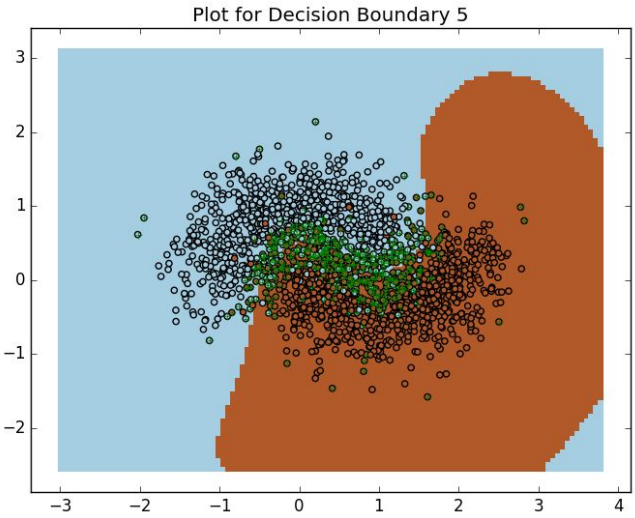
Linear OvR

[[839. 151.]
 [161. 849.]]

RBF OvO
[[855. 101.]
 [145. 899.]]

RBF OvR
[[861. 100.]
 [139. 900.]]





Previous Datasets

Dataset - C
Confusion Matrices
Linear OvO
[[376. 14.]
 [17. 393.]]

Linear OvR
[[376. 11.]
 [17. 396.]]

RBF OvO

[[371. 4.]
 [22. 403.]]

RBF OvR

[[367. 6.]
 [26. 401.]]

Plot No	Model	Accuracy	Observations	Best Parameter(Grid Search)
1(0,0)	Linear One vs One	**0.96750**	Great accuracy	*C=1*
2(0,1)	Linear One vs Rest	**0.9650**	Good accuracy	*C=1*
3(1,0)	RBF One vs One	**0.96750**	Great Accuracy	*gamma=0.9*
4(1,1)	RBF One vs Rest	**0.95750**	Good Accuracy	*gamma=0.9*

Dataset - A
Linear OvO
confusion_matrix for k: 0
[[104. 0. 1. 0. 1. 0. 0. 0. 2. 1.]
 [0. 94. 2. 0. 2. 0. 0. 1. 2. 0.]
 [0. 0. 77. 0. 0. 2. 0. 0. 2. 0.]
 [0. 0. 0. 56. 0. 21. 0. 0. 3. 0.]
 [0. 0. 2. 0. 40. 2. 0. 3. 0. 49.]
 [0. 0. 0. 13. 0. 51. 0. 0. 5. 2.]

```
 [  1.   0.   0.   0.   1.   1.  81.   0.   1.   0.]
 [  0.   0.   0.   0.  32.   0.   0.  63.   3.  34.]
 [  0.   0.   2.   7.   0.   1.   0.   0.  57.   2.]
 [  2.   0.   1.   1.   1.   3.   0.   4.   1.   3.]]
confusion_matrix for k: 1
[[ 61.   0.   1.   0.   0.   0.   0.   0.   1.   0.]
 [  0.  95.   1.   0.   4.   0.   2.   1.   2.   1.]
 [  2.   1.  77.   3.   0.   9.   0.   0.   1.   0.]
 [  1.   0.   1.  80.   1.  22.   1.   0.   0.   0.]
 [  0.   0.   0.   1.  42.   1.   0.   3.   0.  32.]
 [  0.   0.   0.   5.   0.  45.   1.   0.   5.   1.]
 [  0.   0.   0.   0.   1.   2.  82.   0.   2.   0.]
 [  0.   0.   2.   0.  40.   0.   0.  64.   0.  32.]
 [  1.   0.   2.   7.   0.   2.   0.   2.  56.   0.]
 [  0.   0.   2.   3.   4.   0.   0.  18.   2.  12.]]
confusion_matrix for k: 2
[[ 75.   0.   2.   0.   0.   0.   1.   0.   1.   0.]
 [  0.  113.   1.   1.   3.   0.   1.   5.   4.   0.]
 [  0.   0.  59.   2.   0.   6.   0.   0.   2.   0.]
 [  0.   0.   2.  66.   0.  13.   0.   0.   0.   2.]
 [  0.   0.   0.   0.  40.   1.   0.   4.   0.  23.]
 [  0.   0.   0.  12.   0.  60.   0.   0.   7.   1.]
 [  1.   1.   0.   0.   0.   2.  79.   0.   1.   0.]
 [  0.   0.   2.   1.  36.   1.   0.  71.   0.  35.]
 [  0.   0.   3.   4.   0.   3.   0.   0.  70.   0.]
 [  0.   0.   4.   1.   3.   0.   0.   9.   3.   3.]]
confusion_matrix for k: 3
[[ 72.   0.   0.   0.   0.   1.   2.   0.   0.   1.]
 [  0.  92.   1.   1.   1.   0.   1.   4.   1.   0.]
 [  0.   0.  52.   0.   0.  13.   0.   0.   1.   0.]
 [  1.   0.   1.  63.   0.   7.   0.   0.   1.   0.]
 [  1.   0.   1.   0.  39.   0.   0.   3.   0.  41.]
 [  0.   0.   0.  11.   0.  48.   0.   0.   6.   2.]
 [  1.   0.   0.   0.   1.   3.  92.   0.   0.   0.]
 [  0.   0.   4.   1.  34.   0.   1.  70.   2.  39.]
 [  0.   1.   3.  14.   0.   3.   0.   0.  79.   1.]
 [  0.   1.   2.   2.   3.   0.   0.  13.   0.   2.]]
confusion_matrix for k: 4
[[ 77.   0.   2.   0.   0.   0.   1.   0.   0.   2.]
 [  0.  95.   1.   1.   2.   0.   0.   2.   2.   0.]
 [  0.   1.  76.   1.   0.   5.   0.   0.   0.   1.]
 [  0.   0.   2.  57.   0.  25.   0.   0.   3.   0.]
 [  0.   0.   0.   0.  47.   0.   0.  11.   1.  20.]
```

```
 [ 0.  0.  0.  5.  0. 43.  0.  0.  9.  0.]
 [ 0.  0.  0.  0.  2.  0. 96.  0.  1.  0.]
 [ 0.  0.  3.  0. 29.  0.  0. 71.  1. 32.]
 [ 0.  0.  0.  2.  0.  0.  0.  0. 69.  0.]
 [ 0.  0.  1.  3.  7.  2.  0.  9.  1. 19.]]
```

RBF OvO

```
confusion_matrix for k: 0
[[ 87.   0.   0.   0.   0.   0.   0.   0.   0.   1.]
 [  0. 100.   1.   0.   0.   0.   0.   0.   2.   1.]
 [  1.   1.  67.   0.   0.   0.   0.   1.   1.   2.]
 [  6.   2.   7.  84.   3.   1.   2.   3.   3.   2.]
 [  0.   0.   0.   0.  62.   0.   2.   0.   0.   3.]
 [  1.   0.   0.   2.   1.  69.   2.   0.   1.   0.]
 [  1.   0.   0.   0.   0.   0.  83.   0.   1.   0.]
 [  1.   0.   1.   0.   0.   0.   0.  72.   0.   5.]
 [  0.   0.   0.   1.   0.   2.   0.   1.  72.   0.]
 [  0.   0.   0.   0.   2.   0.   0.   4.   1.  72.]]
confusion_matrix for k: 1
[[ 58.   0.   0.   0.   0.   0.   0.   1.   0.   1.]
 [  0. 105.   2.   1.   1.   0.   1.   2.   1.   0.]
 [  0.   0.  82.   0.   0.   0.   0.   0.   0.   0.]
 [  3.   3.   3.  75.   0.   8.   4.   1.   2.   3.]
 [  0.   2.   0.   0.  65.   0.   0.   0.   0.   3.]
 [  1.   0.   0.   4.   1.  83.   0.   0.   4.   0.]
 [  0.   0.   0.   0.   0.   1.  77.   0.   1.   1.]
 [  0.   2.   1.   0.   0.   0.   0.  74.   1.   2.]
 [  0.   0.   0.   3.   0.   1.   1.   1.  75.   0.]
 [  1.   0.   0.   1.   2.   1.   0.   1.   1.  72.]]
confusion_matrix for k: 2
[[ 76.   0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  0.  85.   0.   0.   2.   0.   1.   2.   1.   0.]
 [  0.   0.  65.   2.   0.   0.   0.   0.   0.   0.]
 [  0.   1.   0.  63.   0.   2.   0.   3.   3.   2.]
 [  0.   0.   0.   0.  85.   2.   0.   0.   0.   4.]
 [  0.   0.   0.   1.   0.  71.   0.   0.   0.   0.]
 [  1.   1.   0.   0.   0.   0.  96.   0.   0.   0.]
 [  0.   0.   0.   0.   0.   0.   0.  68.   0.   3.]
 [  6.   5.   6.   7.   0.   6.   6.   3.  71.   6.]
 [  1.   0.   0.   0.   4.   1.   0.   1.   0.  77.]]
confusion_matrix for k: 3
[[ 65.   0.   1.   0.   0.   0.   1.   0.   1.   0.]
 [  0.  89.   0.   1.   2.   0.   2.   1.   1.   0.]
```

```
 [ 0.  1. 74.  2.  0.  0.  0.  0.  0.  0.]
 [ 4.  2.  3. 80.  2.  9.  2.  4.  8.  2.]
 [ 0.  1.  1.  0. 78.  2.  0.  0.  1.  3.]
 [ 0.  1.  0.  2.  0. 62.  1.  0.  3.  2.]
 [ 0.  0.  0.  0.  1.  3. 75.  0.  2.  0.]
 [ 0.  2.  1.  1.  1.  0.  0. 95.  1.  1.]
 [ 0.  0.  0.  1.  0.  2.  0.  0. 76.  0.]
 [ 0.  0.  0.  1.  5.  1.  0.  0.  0. 57.]]
confusion_matrix for k: 4
[[ 80.  0.  1.  0.  0.  0.  1.  0.  0.  0.]
 [ 0. 87.  1.  0.  3.  0.  0.  2.  1.  0.]
 [ 1.  1. 70.  0.  1.  0.  1.  2.  0.  0.]
 [ 6.  3.  5. 86.  7.  7.  2.  3.  2.  2.]
 [ 0.  0.  0.  0. 84.  0.  1.  1.  0.  4.]
 [ 0.  0.  0.  3.  0. 62.  0.  0.  2.  0.]
 [ 0.  0.  0.  0.  2.  0. 78.  0.  1.  0.]
 [ 0.  0.  1.  0.  0.  0.  0. 83.  0.  3.]
 [ 0.  0.  0.  0.  0.  0.  1.  0. 68.  0.]
 [ 0.  0.  0.  3.  2.  2.  1.  2.  2. 59.]]
```

Plot No	**Model**	**Accuracy**	**Observations**	**Best Parameter(Grid Search)**
1(0,0)	Linear One vs One	**0.74642**	Decent accuracy	*C=1*
2(0,1)	Linear One vs Rest		Good accuracy	*C=1*
3(1,0)	RBF One vs One	**0.90452**	Great Accuracy	*gamma=0.7*
4(1,1)	RBF One vs Rest		Good Accuracy	*gamma=0.7*

Dataset - B
Confusion Matrices
Linear OvO

```
[[ 1063.   754.]
 [ 1034.  1349.]]
```

Linear OvR

```
[[ 1067.   758.]
 [ 1030.  1345.]]
```

RBF OvO

[[1238. 836.]
 [859. 1267.]]

RBF OvR
[[1236. 810.]
 [861. 1293.]]

Plot No	Model	Accuracy	Observations	Best Parameter(Grid Search)
1(0,0)	Linear One vs One	**0.57428**	Poor accuracy	*C=1*
2(0,1)	Linear One vs Rest	**0.57428**	Good accuracy	*C=1*
3(1,0)	RBF One vs One	**0.59642**	Poor Accuracy	*gamma=0.7*
4(1,1)	RBF One vs Rest	**0.60214**	Better than previous Accuracy, Still poor	*gamma=0.7*

Question 3

The data contains list of dictionaries.
I considered the array of “X” as a list of words for each data point and use TfidfVectorizer tokenizer by converting the list of words of “X” into sentences(space separated string) and tokenizing it by using TfidfVectorizer() with suitable parameters like *token_pattern, ngram_range* and fitting the data into LinearSVC() model classifier. I optimize the parameters to consider single letter words as well.

To optimize, I removed all the incorrect classifications from training data after first round of predictions and fit the model with new training data. This achieved better accuracy for given test data.
The parameter C, for the svm model, was optimised to around 0.543
TfidfVectorizer