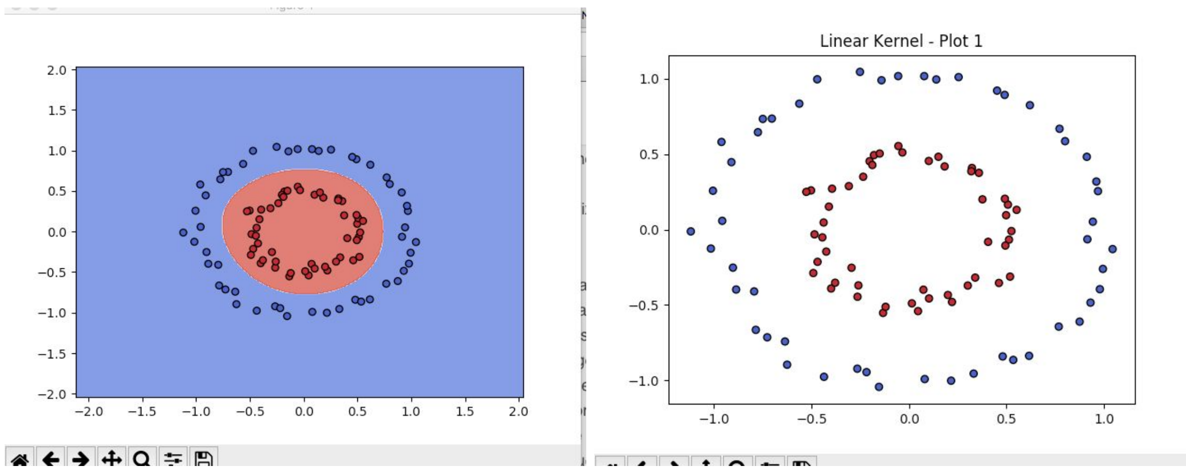
ML REPORT

DATA VISUALIZATION

C=1 was the best choice for all the datasets. Other than that I used all the default parameters since there was no need to change them.
C was chosen on the basis of trial and error.

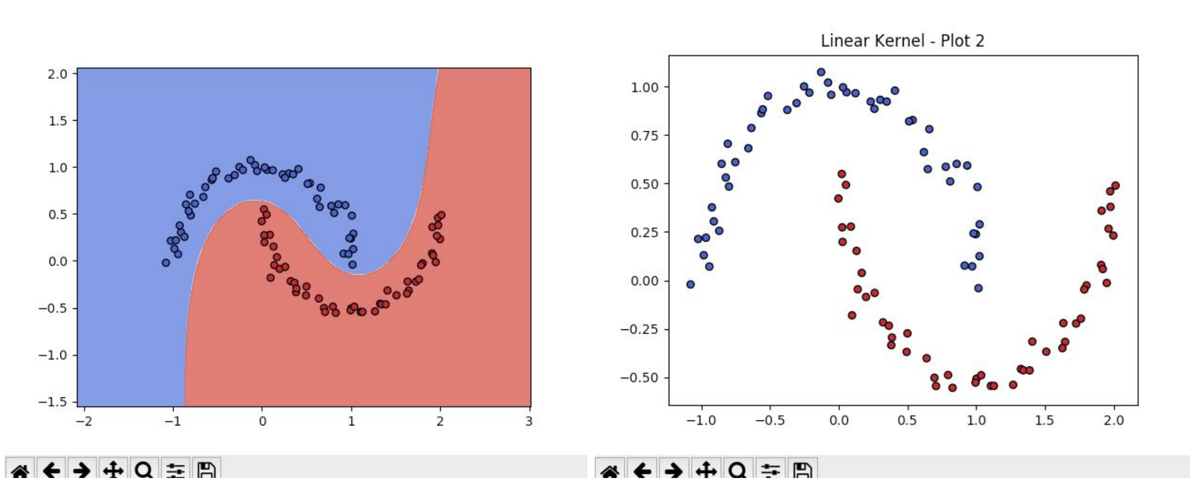
PLOT 1

Concentric circles



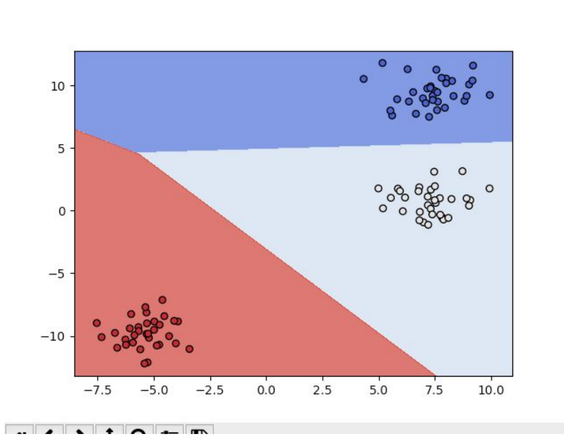
PLOT 2

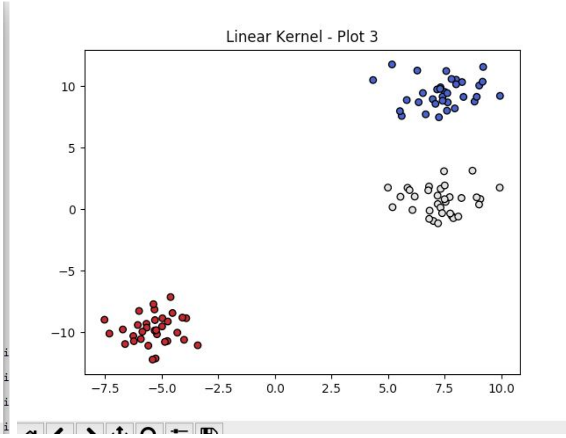
Kernel -> rbf , c = 1



PLOT 3

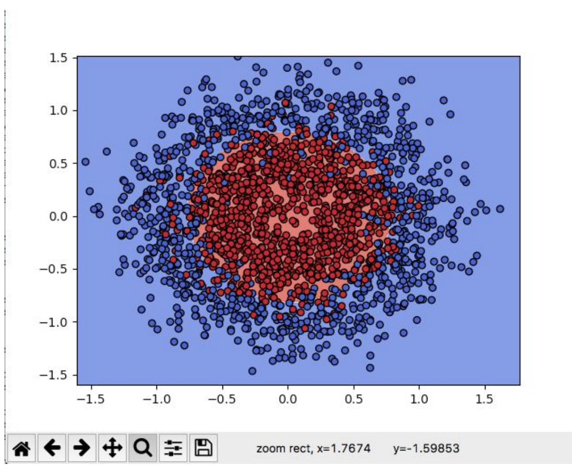
Plotted using linear kernel. It’s already linearly seperable. Kernel - linear

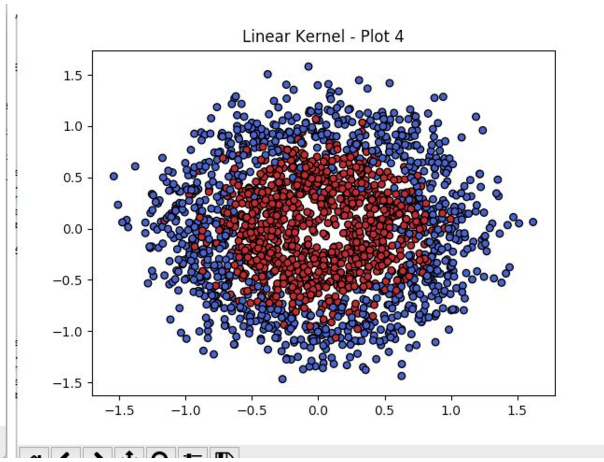




PLOT 4

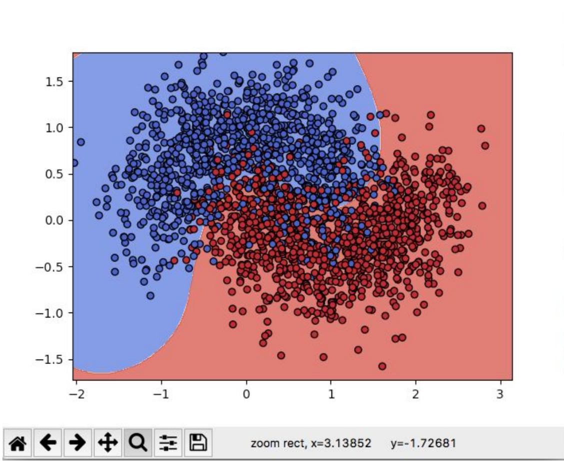
Many outliers. Kernel - rbf
Circles formed with same centre.

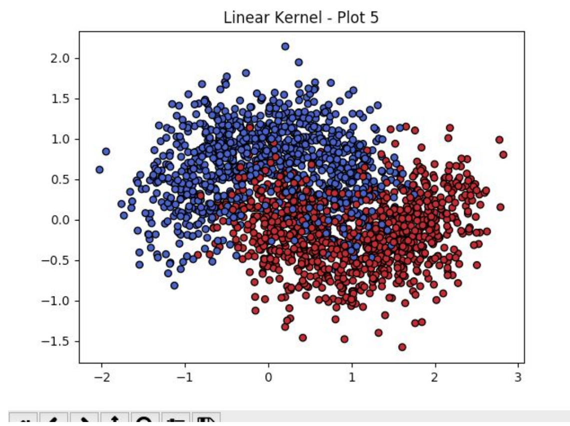




PLOT 5

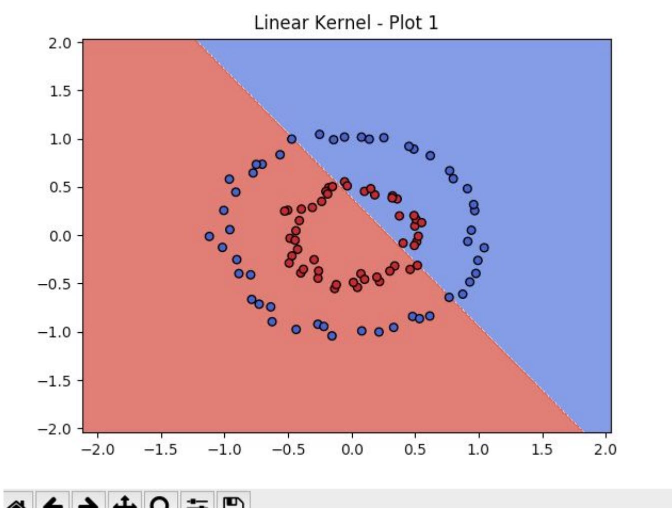
Not linearly seperable. Kernel - rbf





SVM with Linear kernel

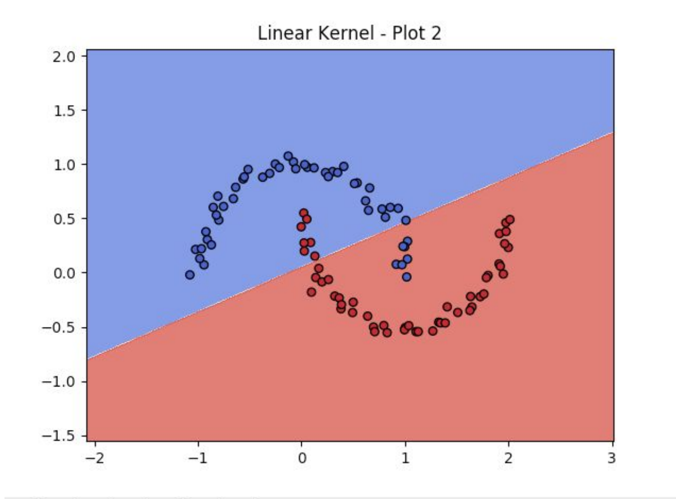
Plot 1



Best Accuracy : 0.5

```
(python2) Akarshas-MacBook-Air:ML  akarsha$ python asgn2.py --data data_1.h5
('accuracy_score', 0.5)
```

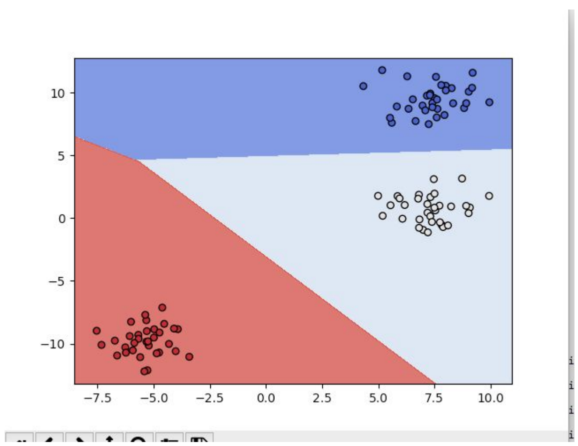
Plot 2



Best Accuracy : 0.8

```
(python2) Akarshas-MacBook-Air:ML  akarsha$ python asgn2.py --data data_2.h5
('accuracy_score', 0.8000000000000004)
```

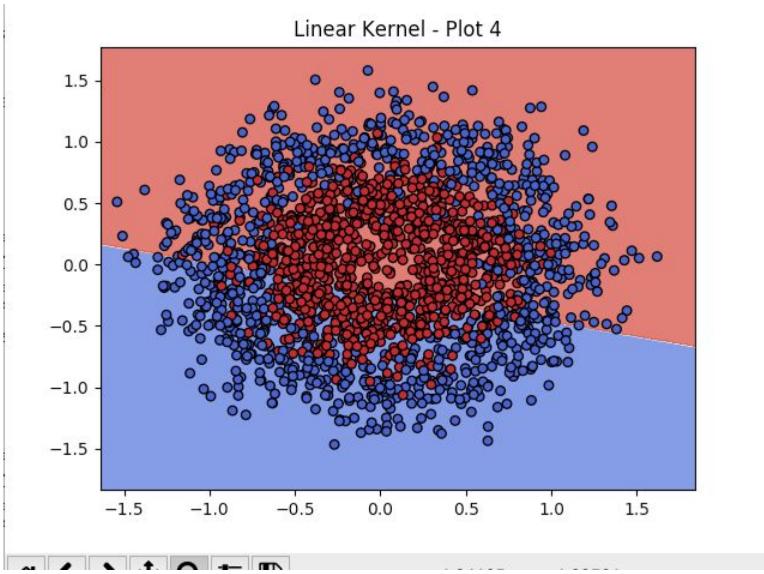
Plot 3



Best Accuracy : 0.6

```
(python2) Akarshas-MacBook-Air:ML  akarsha$ python asgn2.py --data data_3.h5
('accuracy_score', 0.5999999999999998)
```

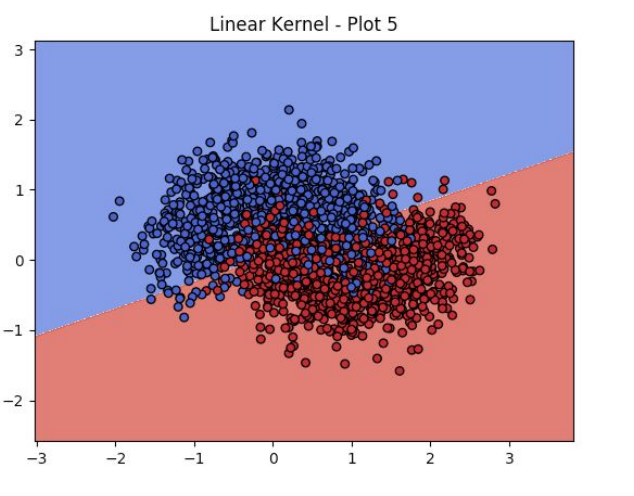
Plot 4



Best Accuracy : 0.54

```
[(python2) Akarshas-MacBook-Air:ML  akarsha$ python asgn2.py --data data_4.h5
 ('accuracy_score', 0.5460000000000004)
```

Plot 5

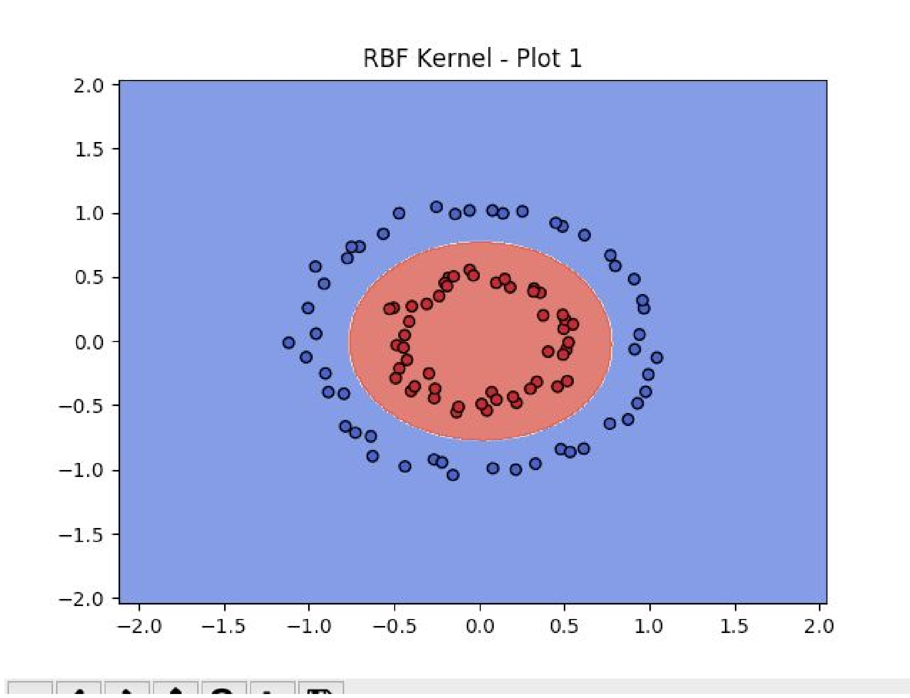


Best Accuracy : 0.83

```
(python2) Akarshas-MacBook-Air:ML  akarsha$ python asgn2.py --data data_5.h5
('accuracy_score', 0.8259999999999996)
```

SVM with RBF kernel

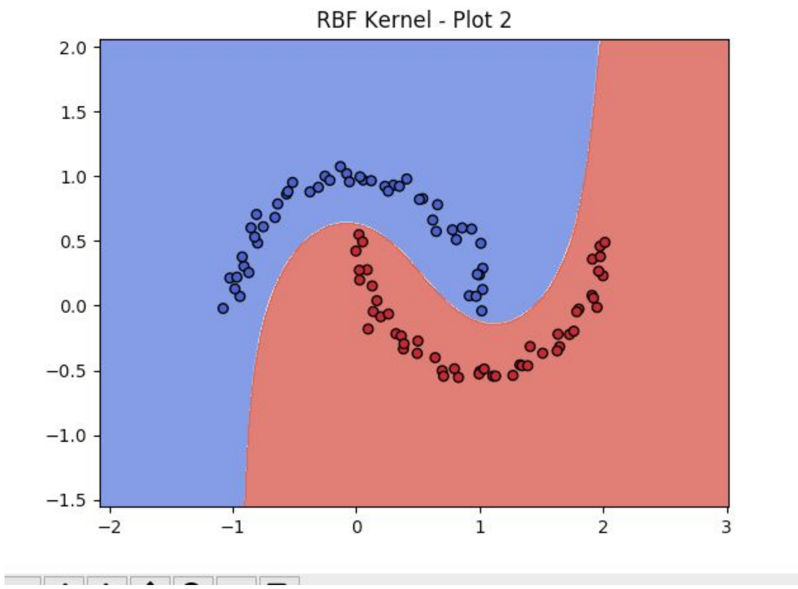
PLOT 1



Accuracy: 1.0

```
[(python2) Akarshas-MacBook-Air:ML  akarsha$ python asgn2.py --data data_1.h5
 accuracy_score of  data_1.h5 is :  1.0
```

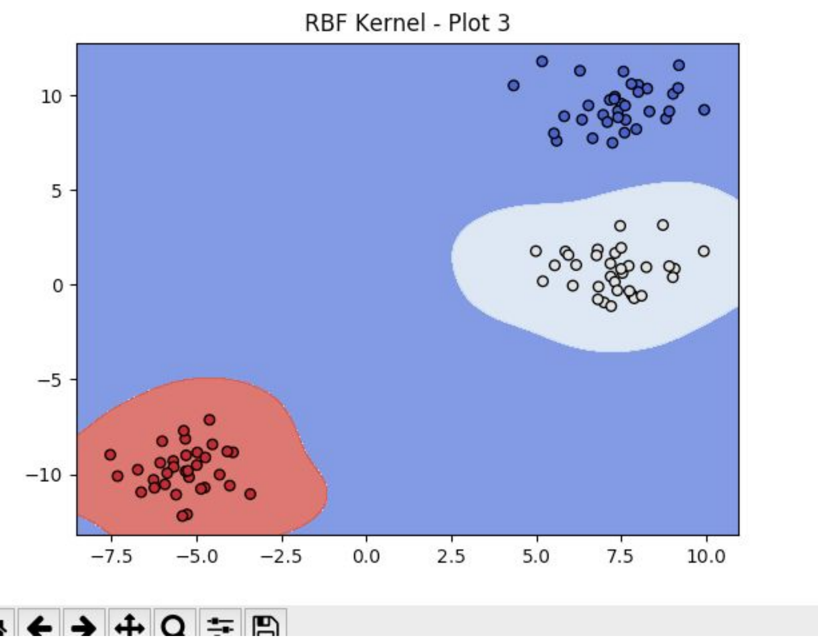
PLOT 2



Accuracy: 1.0

```
(python2) Akarshas-MacBook-Air:ML  akarsha$ python asgn2.py --data data_2.h5
accuracy_score of  data_2.h5 is :  1.0
```

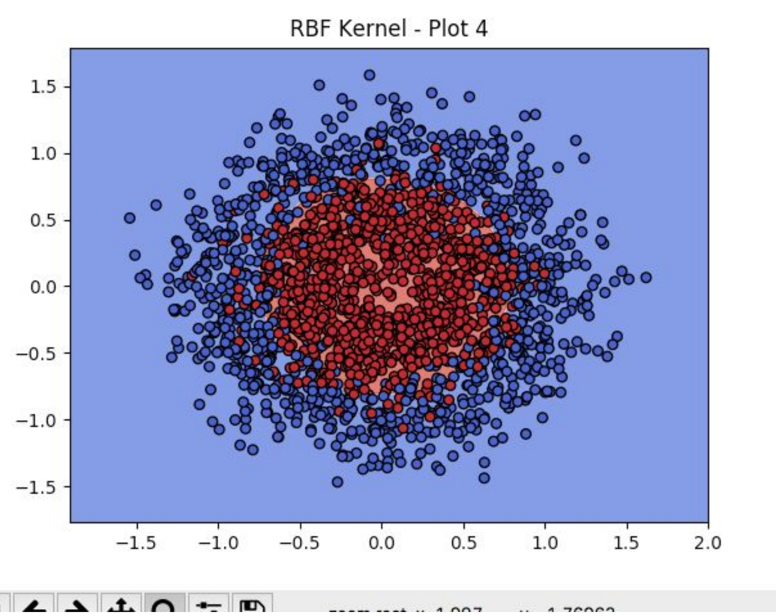
PLOT 3



Accuracy: **1.0**

```
[(python2) Akarshas-MacBook-Air:ML  akarsha$ python asgn2.py --data data_3.h5
 accuracy_score of  data_3.h5 is :  1.0
```

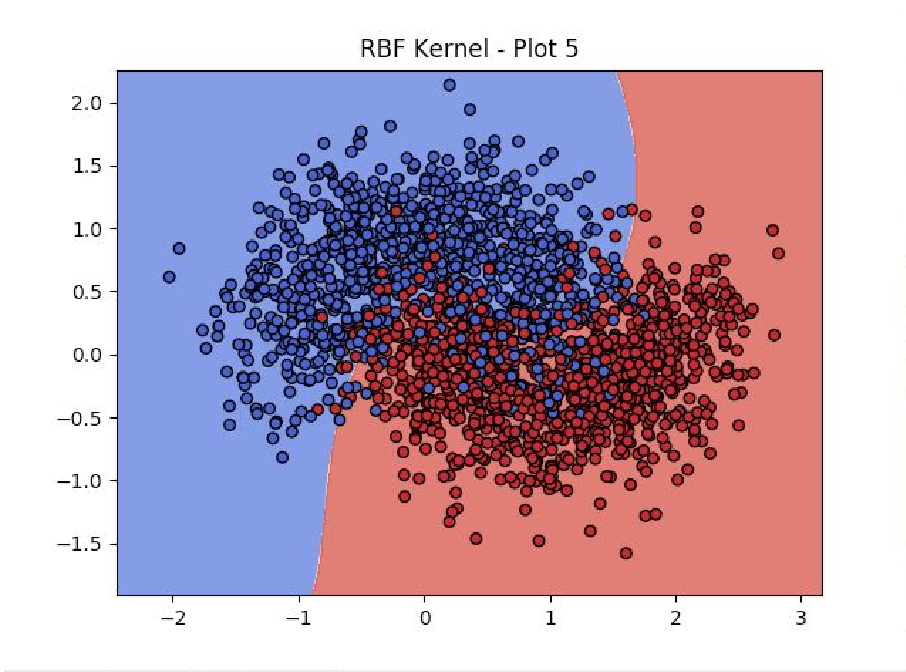
PLOT 4



Accuracy: **0.66**

```
[(python2) Akarshas-MacBook-Air:ML  akarsha$ python asgn2.py --data data_4.h5
accuracy_score of  data_4.h5 is :  0.6625
```

PLOT 5



Accuracy: **0.56**

```
[^[(python2) Akarshas-MacBook-Air:ML  akarsha$ python asgn2.py --data data_5.h5
accuracy_score of  data_5.h5 is :  0.5575
```

Outlier Removal :

Outlier removal can be done by using normal distribution and standard deviation.

One vs One and One vs Rest :

Output came out to be same for all of the above plots.

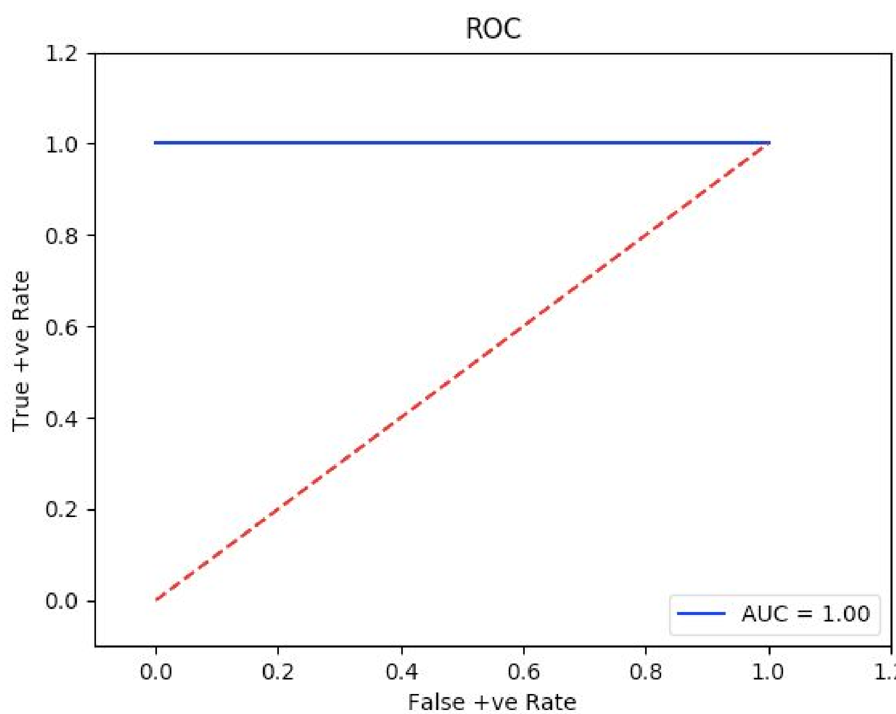
For the third dataset : it came out to be perfectly 1.0.

Confusion matrices:

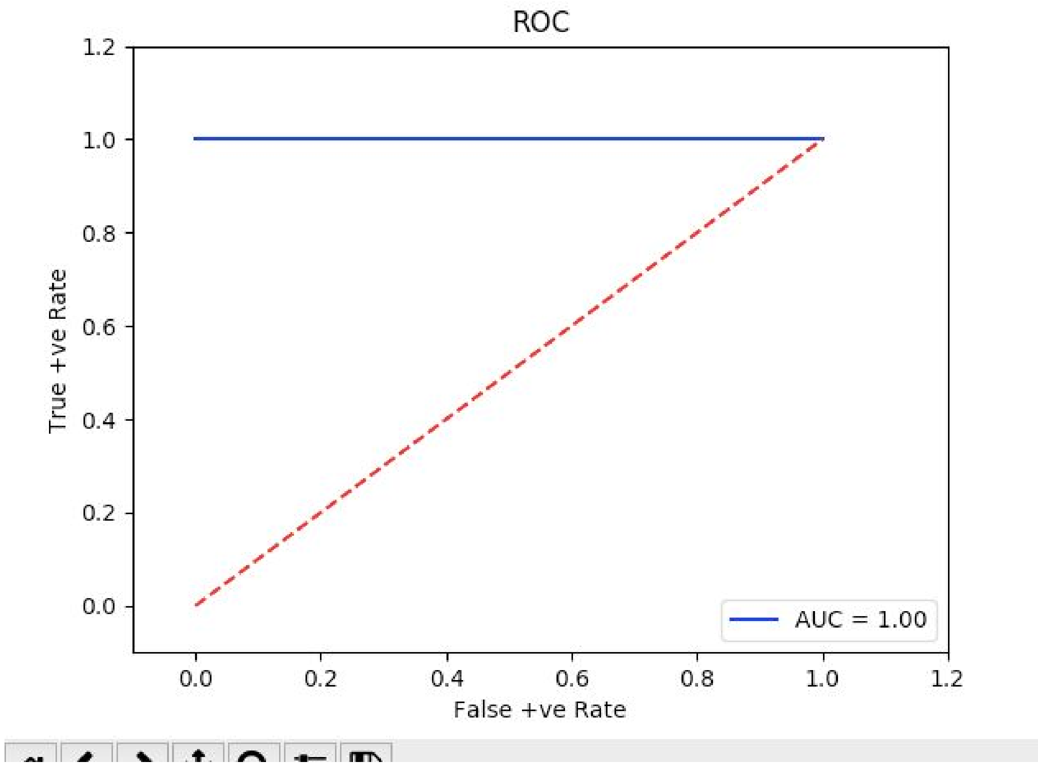
```
[(python2) Akarshas-MacBook-Air:ML  akarsha$ python asgn2.py --data data_1.h5
 Confusion matrix with Normalization
 [[ 1.  0.]
  [ 0.  1.]]
[(python2) Akarshas-MacBook-Air:ML  akarsha$ python asgn2.py --data data_2.h5
 Confusion matrix with Normalization
 [[ 1.  0.]
  [ 0.  1.]]
[(python2) Akarshas-MacBook-Air:ML  akarsha$ python asgn2.py --data data_3.h5
 Confusion matrix with Normalization
 [[ 0.  1.  0.]
  [ 1.  0.  0.]
  [ 1.  0.  0.]]
[(python2) Akarshas-MacBook-Air:ML  akarsha$ python asgn2.py --data data_4.h5
 Confusion matrix with Normalization
 [[ 0.76142132  0.23857868]
  [ 0.12807882  0.87192118]]
[(python2) Akarshas-MacBook-Air:ML  akarsha$ python asgn2.py --data data_5.h5
 Confusion matrix with Normalization
 [[ 0.58571429  0.41428571]
  [ 0.47368421  0.52631579]]
```

ROC Curves:

Data 1



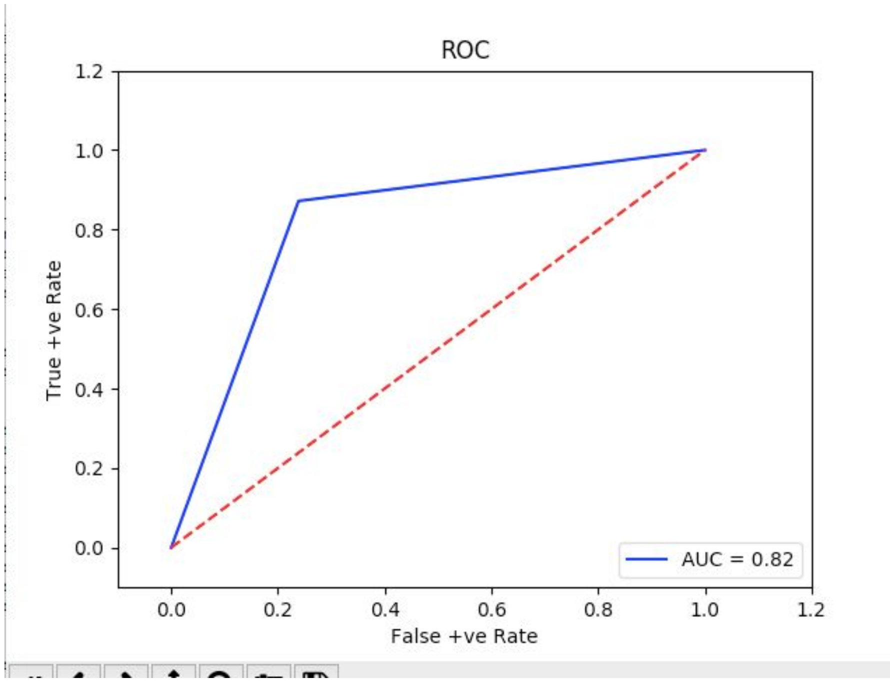
Data 2



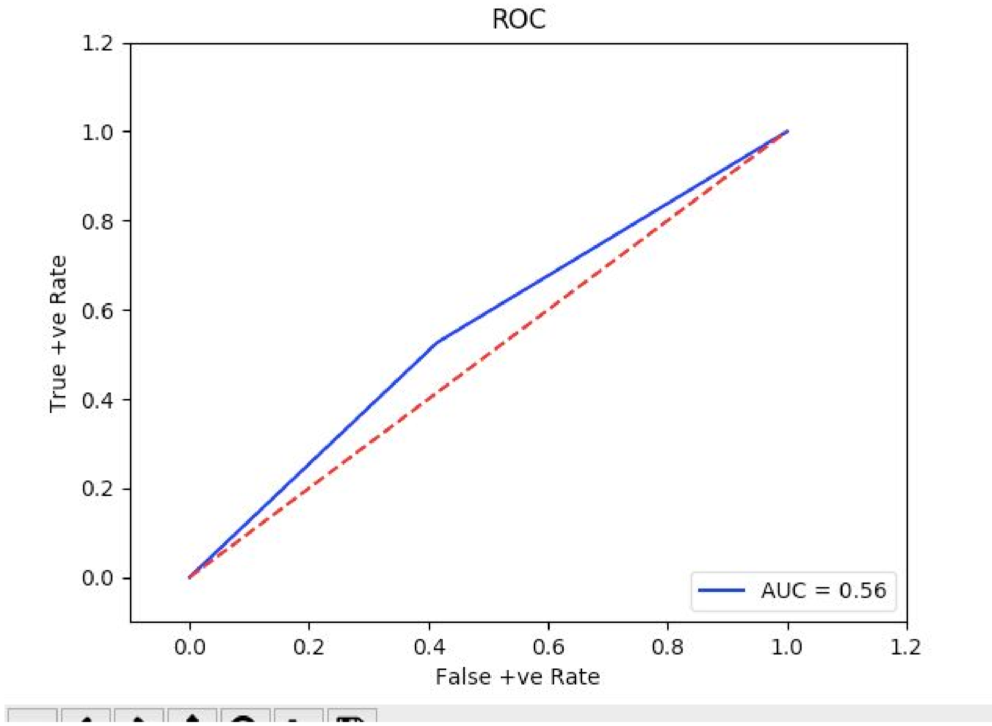
Data 3

ROC couldn't be plotted because the data wasn't binary.

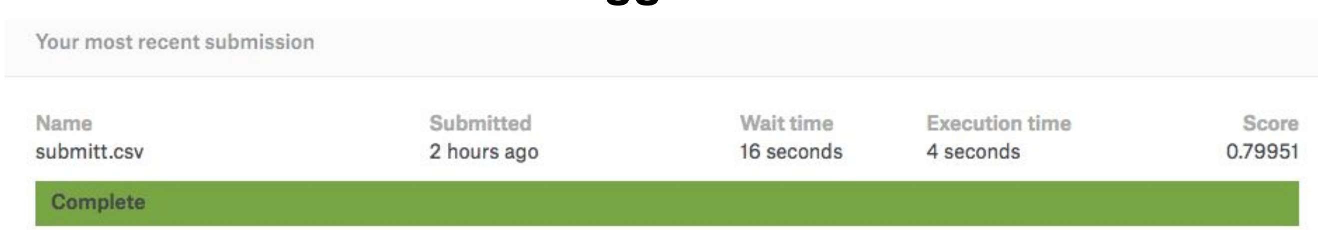
Data 4


ROC
True +ve Rate
False +ve Rate
AUC = 0.82
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2

Data 5



Kaggle



Score: 0.79951

I used ***Tfidf vectorizer*** with the following parameters. I tried different parameters out of which this suited the best.
Parameters: *ngram_range = 1,3*

Reason: the combination of two/three features should also be taken into account for which 1-gram,2-gram,3-grams need to be considered. Increasing n would lead to decrease in efficiency of the code.
Then, after fitting the transform, I used **Linear Support Vector Classifier** to classify the data. Regularizer is taken too low to see the impact on the results.
Parameters: *C= 0.33*