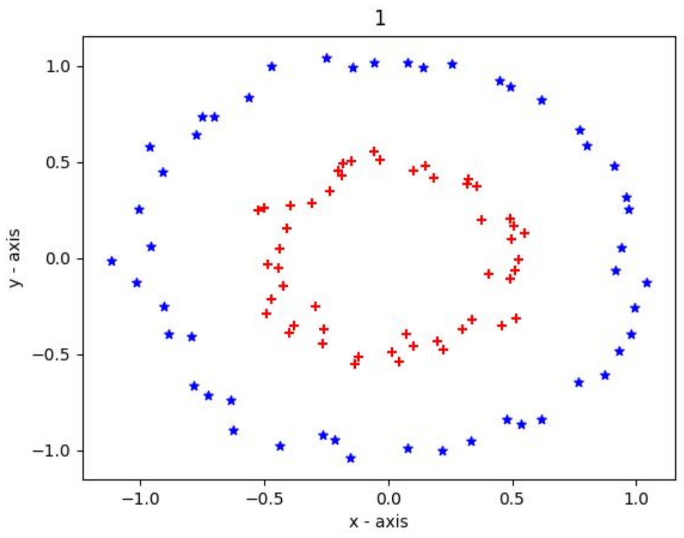
Machine Learning
Assignment 2

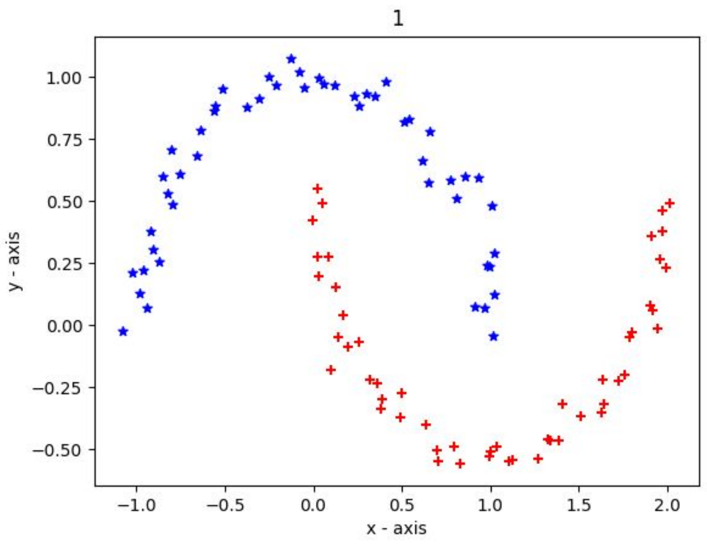
If there is memory error while running .py file please run the file using a jupyter notebook which is provided.

Question1

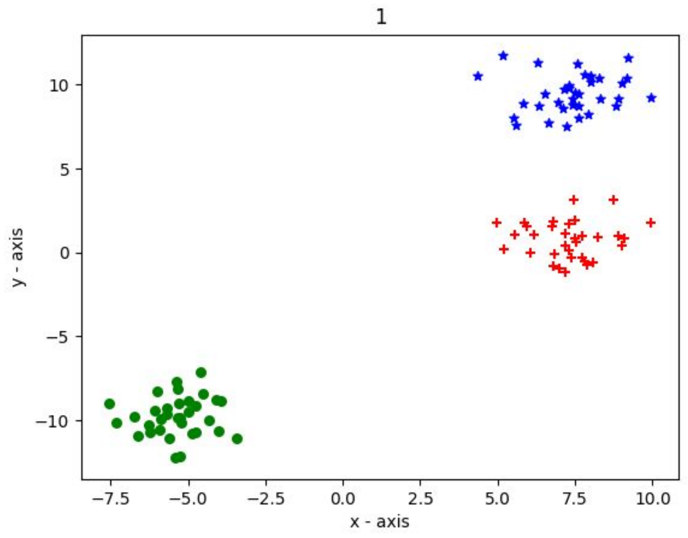
Question1 Part A



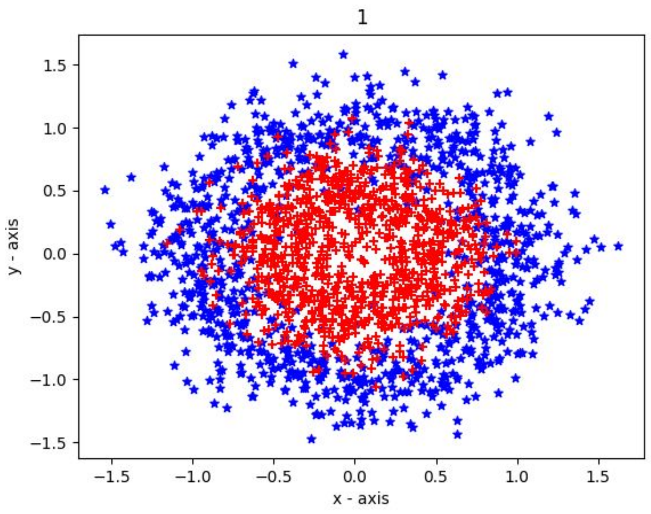
There is no noise or outliers. It is not linearly separable directly. So we use the kernel to make it linearly separable. There are two classes of data.



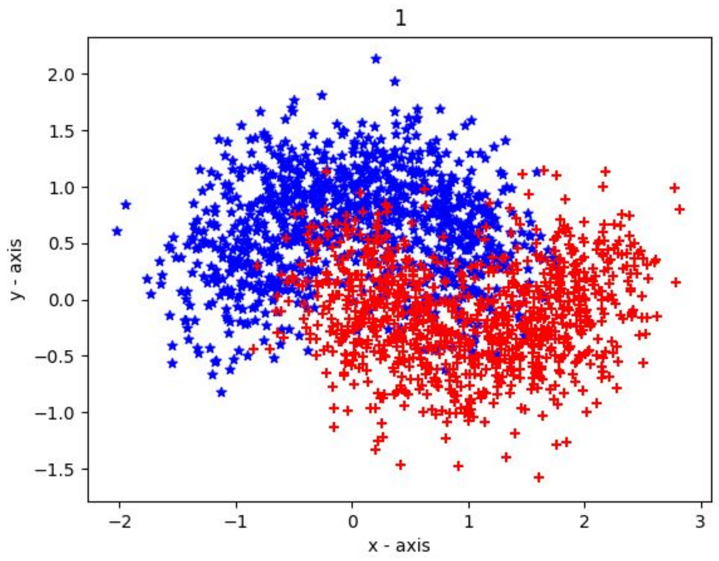
There is no noise or outliers. It is not linearly separable directly. So we use the kernel to make it linearly separable. There are two classes of data.



There is no noise or outliers. It is linearly separable directly. So we don't need any kernel. There are 3 classes of data.

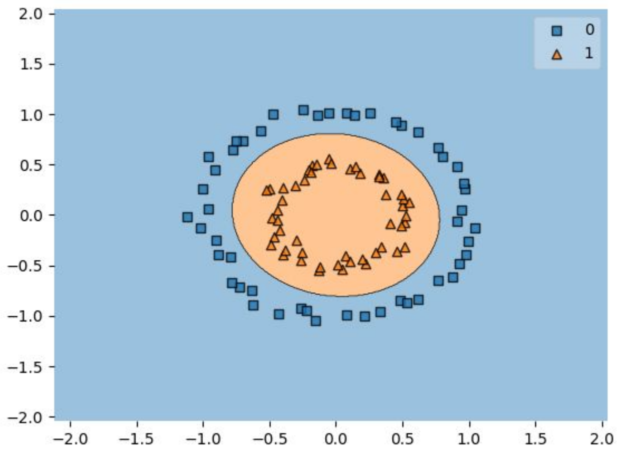


This data set has noise as well as outliers. It is not linearly separable directly. So we use the kernel to make it linearly separable. There are two classes of data.

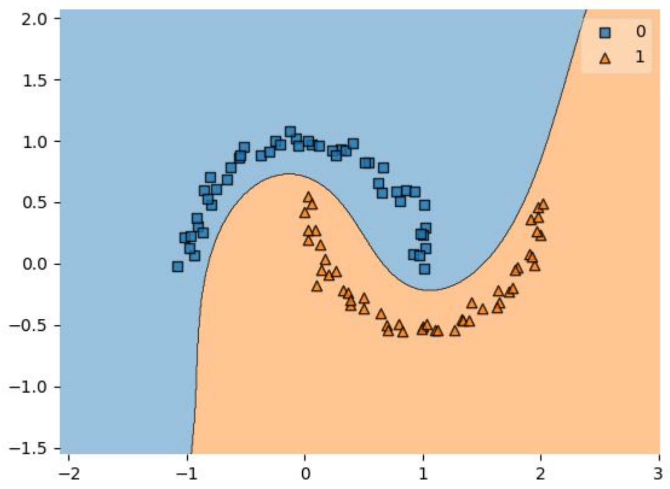


This data set has noise as well as outliers. It is not linearly separable directly. So we use the kernel to make it linearly separable. There are two classes of data.

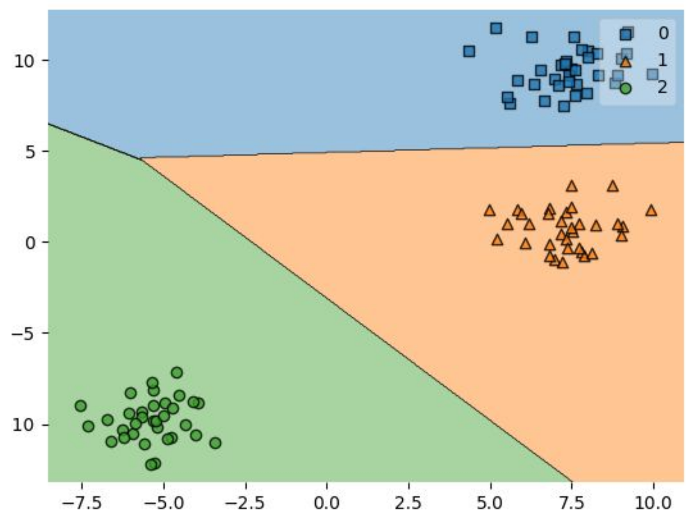
Question1 Part B



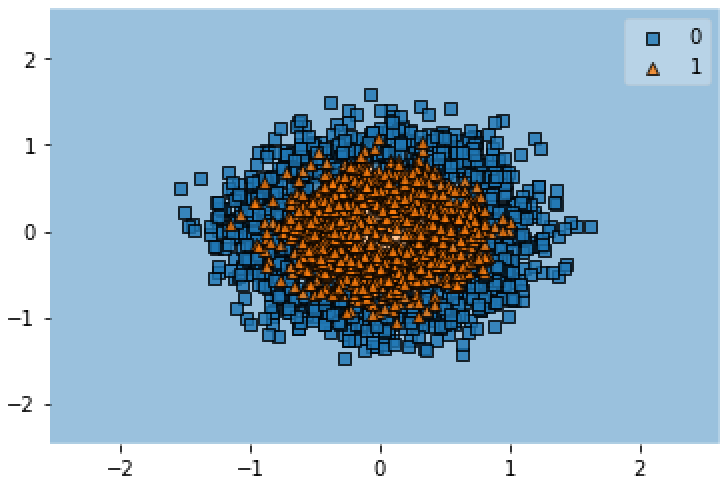
Using the inbuilt poly kernel of degree 2 this is made linearly separable. Plotted using plot_decision_regions. We can also use higher degree polynomial kernel but it will increase complexity without increasing the accuracy.



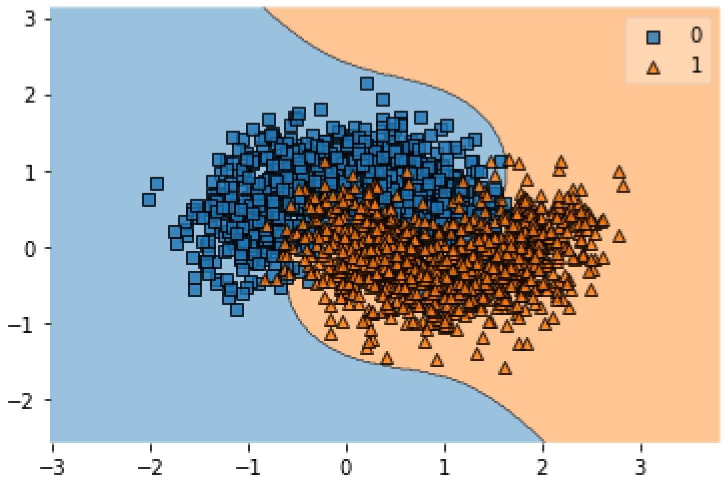
Using kernel ((x@y.T)+1)**3 this is made linearly separable. Plotted using plot_decision_regions. The polynomial of degree 2 was not sufficient for separating the data.



This is linearly separable. The inbuilt linear kernel has been used to plot the same. Plotted using plot_decision_regions.



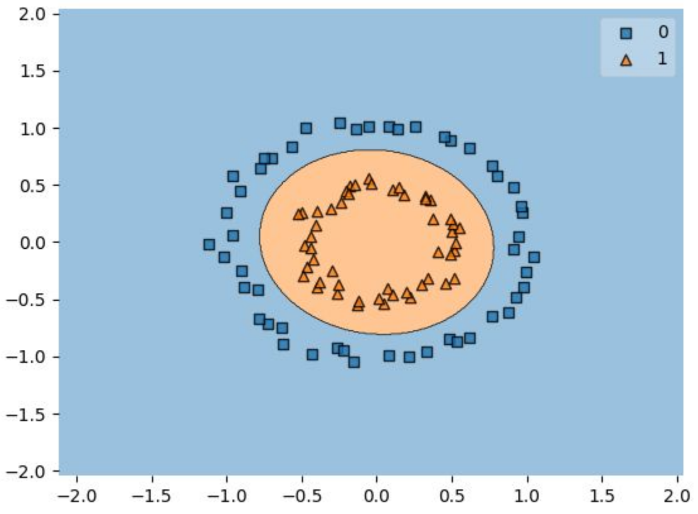
Using kernel ((x@y.T)+1)**2 this is made linearly separable. Plotted using plot_decision_regions. Due to the high density of class 2 data we are unable to see the decision boundary. Due to noise some of the data is classified incorrectly.



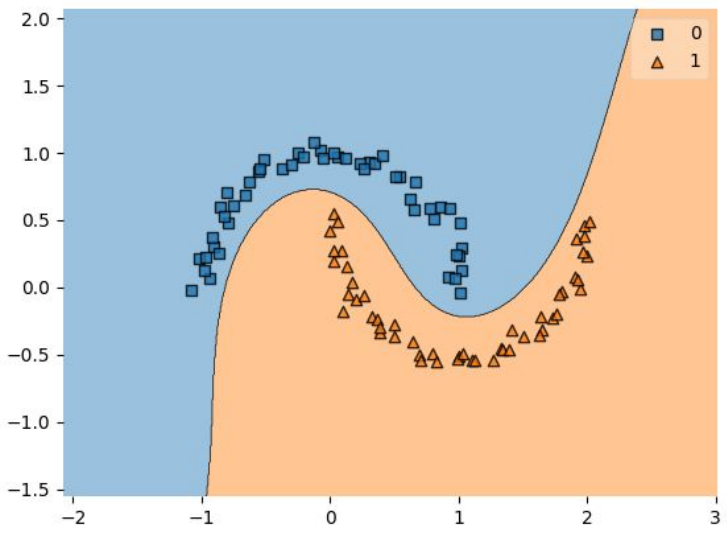
Using kernel ((x@y.T)+1)**3 this is made linearly separable. Plotted using plot_decision_regions. Due to noise some of the data is classified incorrectly. We could have used higher-order too but it would only increase complexity without changing the accuracy much.

Question1 Part C

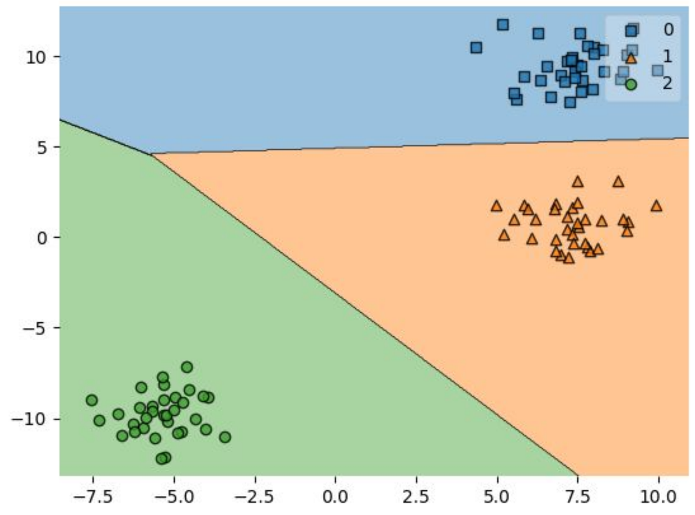
I have used the zscore method which utilizes the mean and variance to predict the vectors which are too far from the original data.



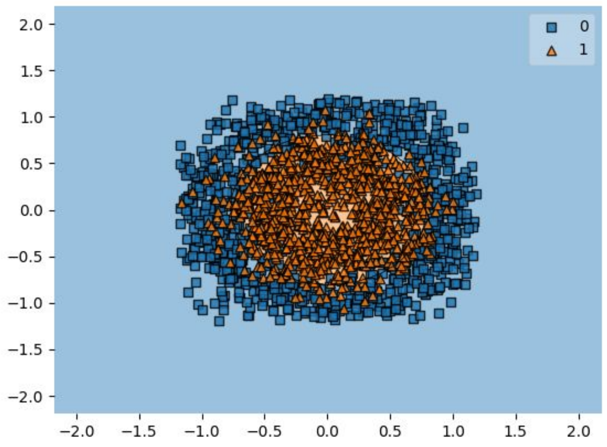
There were no outliers in this case. Hence, the same plot.



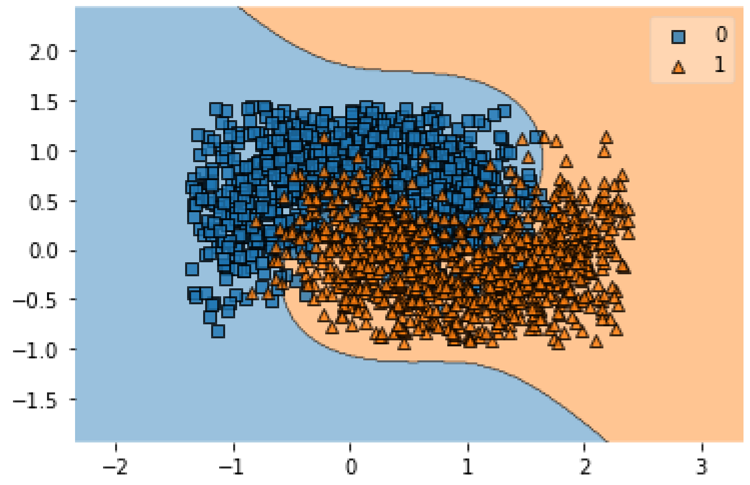
There were no outliers in this case. Hence, the same plot.



There were no outliers in this case. Hence, the same plot.



Setting the zscore limit as 2 we get many outliers. Those are printed while running the file. The outliers were removed and then this plot was plotted.



Setting the zscore limit as 2 we get many outliers. Those are printed while running the file. The outliers were removed and then this plot was plotted.

Question1 Part D

Linear_predict is for predicting the output of the linear kernel. rbfKernel is the self-implemented RBF kernel. rbfPredict is for predicting the output of the RBF kernel.

For Dataset 4

---------Linear-----------

My accuracy for test = 56.05263157894737

My accuracy for train = 51.81040157998683

SVC accuracy for test = 56.05263157894736

SVC accuracy for train = 51.81040157998683

-----------RBF-----------

SVC accuracy for test = 86.8421052631579

My accuracy for test = 86.84210526315789

SVC accuracy for train = 88.21593153390388
My accuracy for train = 88.0184331797235

For dataset 5

---------Linear-----------

My accuracy for test = 78.87700534759358

My accuracy for train = 82.62032085561498

SVC accuracy for test = 81.81818181818183

SVC accuracy for train = 83.62299465240642

-----------RBF-----------

SVC accuracy for test = 87.16577540106952

My accuracy for test = 86.89839572192513

SVC accuracy for train = 87.5
My accuracy for train = 87.23262032085562

Question2

Question2 Part A

Using Data_batch3

I have picked 500 images for each class. Then used the 5 fold method which means my every model runs of 4000 training data and 1000 testing data. For the sake of clarity, I have included only the best plots of each model. Other plots can be seen while running the file.

The accuracy along the 1-v-1 and 1-v-all are the same corresponding to the same datasets. The 1-vs-1 method has 45 classifiers, here I have shown the 10 classifiers(Due to high running time) among them on the ROC curve.

RBF is the best kernel for the data set as it provides us with an infinite dimensional vector. It is evident from accuracy as well as the ROC plots and confusion matrices.

Mean accuracy

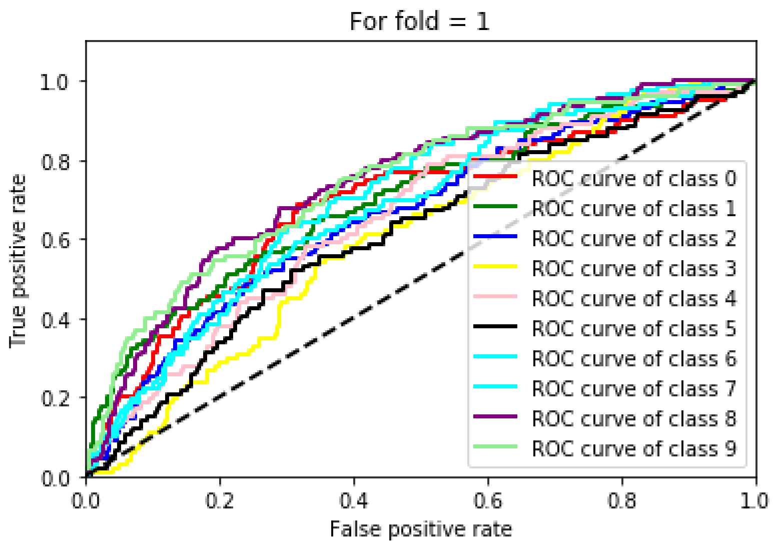
1. RBF kernel: 43.76%
2. Polynomial kernel: 41.3%
3. No kernel: 32.88%

SVM with no kernel(linear) One-vs-rest

Weights of linear models are printed while running the .py file.

```
Confusion Martix= [[42  5  6  3  5  4  5  5 20  4]

 [ 7 44  5  6  3  4  2  5  9  5]

 [10  5 40 12 16  5  5 10  6  2]

 [ 6  4 11 13 17 17 14  6  4  6]

 [ 1  1 23 12 26 10  9 13  3  0]

 [ 8  5 17 14 11 22  3 14  5  7]

 [ 2  9 21 16  9  9 32  2  3  0]

 [ 8  2 16  7 11  5  3 22  2  5]

 [13 14  3  7  4  1  1  1 60  4]
 [11 18  5  2  1  3  8  5  9 44]]
```

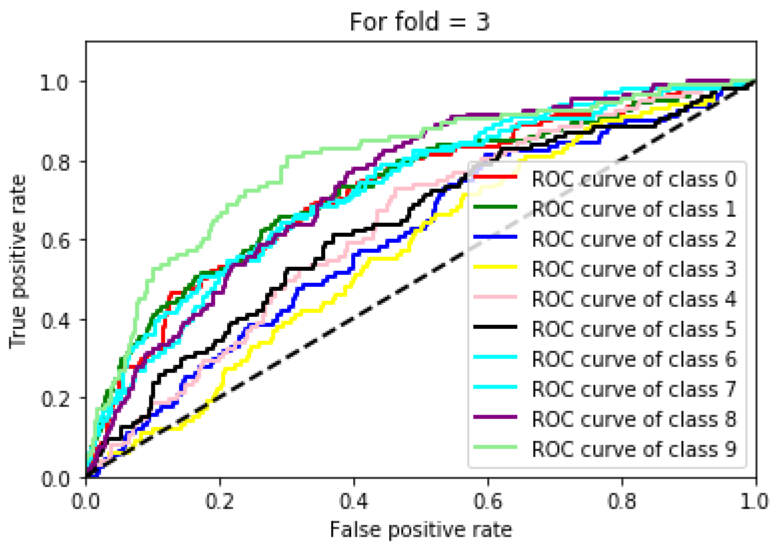
Accuracy = .345



SVM with no kernel(linear) One-vs-one

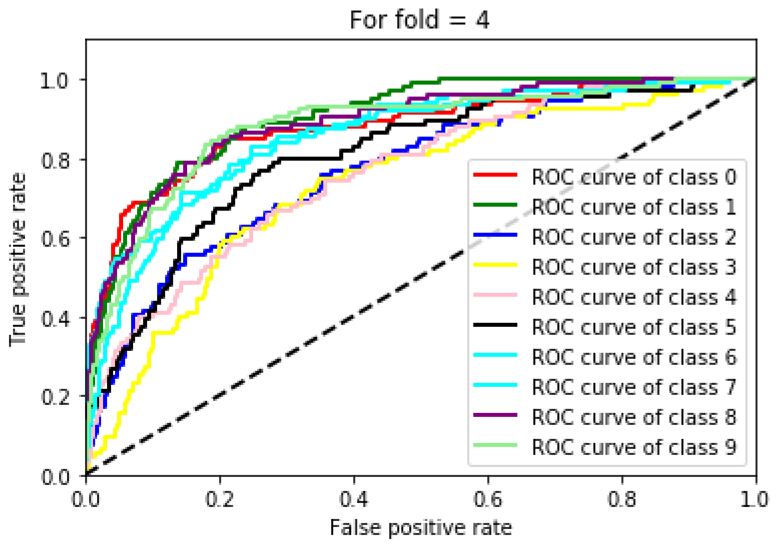
```
Confusion Martix= [[52 10  6  7  4  2  3  3 15  6]

 [12 40 10  3  5  3  3  7  9 13]

 [16  3 29  9  9  7  8  3  3  4]

 [ 7  9 25 12 10 11 10  4  6  6]

 [ 7  6 14 17 26  9 10 10  2  2]

 [12  8 10 21 12 20 10  4  4  4]

 [ 5  3 11 20 11  5 21  4  2  4]

 [ 7  8 14  7 16  8  4 30  4  3]

 [24  7  7  5  1  1  2  3 53  5]

 [14 21  6  2  5  3  4  5  3 30]]
```

Accuracy = 0.313



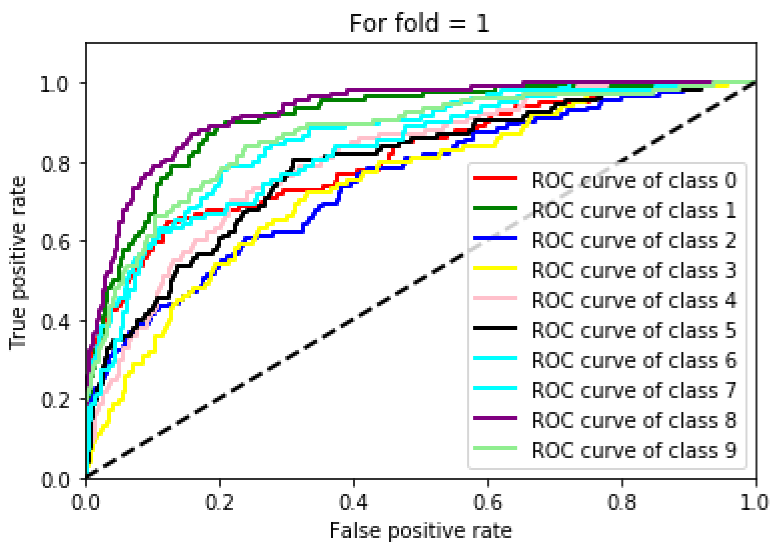
SVM with RBF One-vs-rest

```
Confusion Martix= [[49  3  5  3  2  1  4  2 21  3]

 [ 5 55  2  4  2  1  1  2 10 16]

 [ 8  3 33  5 17  5 12  4  4  3]

 [ 5  3  7 30 10 13 16  3  2  6]

 [ 5  3 16  8 29  1 14 14  2  1]

 [ 3  3 10 24  6 25 10  8  2  3]

 [ 2  4 10  7  9 10 54  2  1  5]

 [ 2  3 10  9 14  6  8 52  2  4]

 [12 11  2  3  0  3  2  1 59 10]

 [ 8 19  2  2  4  3  2  5 15 56]]
Accuracy = 0.442
```



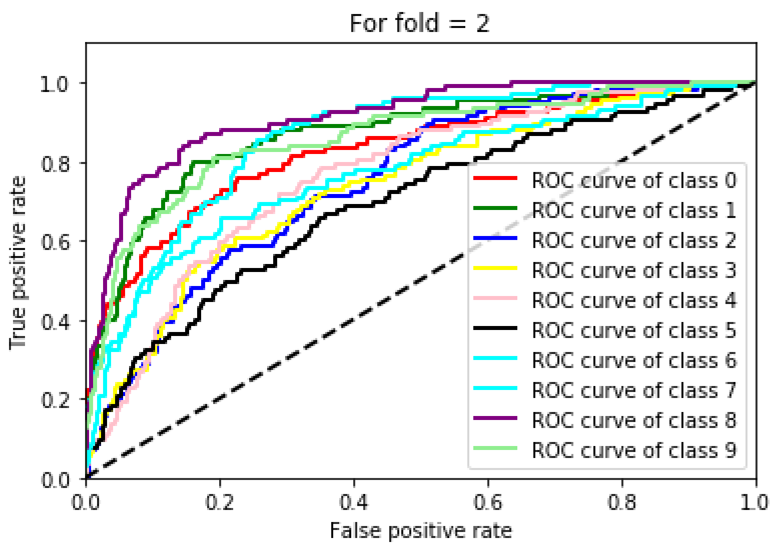
SVM with RBF One-vs-one

```
Confusion Martix= [[48  7  3  3  4  3  7  3 15  6]

 [ 2 50  1  8  1  1  4  4  5 14]

 [10  4 38  7 13  7 21  7  3  1]

 [ 6  9  8 33  6 10 15  2  4  5]

 [ 3  1 15  3 39  2 25  5  2  3]

 [ 5  4 10 27 12 27  9  6  5  1]

 [ 3  3 11  9 13  2 57  3  1  1]

 [ 3  0  4 13 12  3  6 30  5  5]

 [ 8 12  2  4  4  3  1  1 62 11]

 [ 5 22  3  2  2  1  3  3 13 52]]
Accuracy = 0.436
```



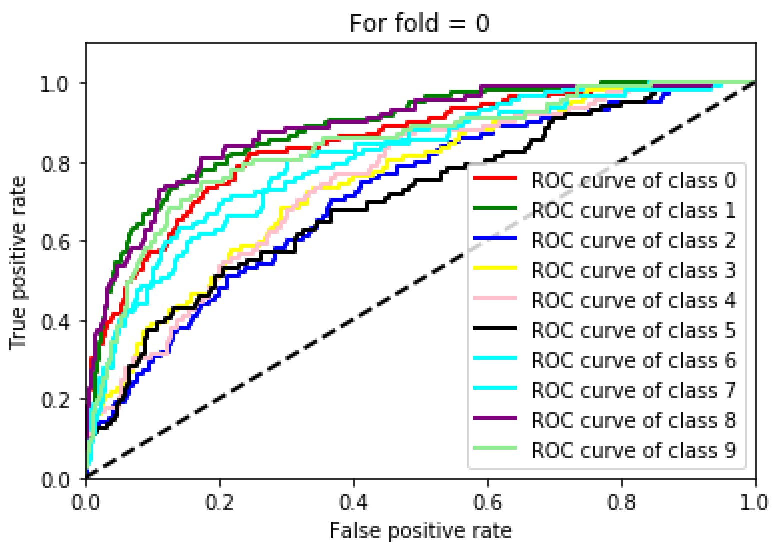
SVM with Polynomial One-vs-rest

```
Confusion Martix= [[51  6 10  3  0  3  4  2  5  7]

 [ 4 38  4  5  2  2  8  4 10 13]

 [ 7  0 45  6 13  4 18  6  3  2]

 [ 5  5  9 37  7 21 16  2  2  2]

 [10  1 21  4 56  2 14  2  0  6]

 [ 3  3 12 14 14 24 13  4  5  1]

 [ 1  5 11  6 13  4 54  2  0  0]

 [ 9  3  9  9 27  8  7 38  2  5]

 [13  6  4  5  2  2  0  1 54  6]

 [ 3 18  4  4  1  4  5  3 12 40]]
Accuracy = 0.437
```



SVM with Polynomial One-vs-one

```
Confusion Martix= [[50  7 10  3  8  7  2  5 12  5]

 [ 2 49  9  2  2  2  5  6 12 28]

 [ 9  2 32  7 14  9 13 10  3  1]

 [ 5  3 15 36  4 16 13  3  4  2]

 [ 5  2 15  5 34  7  9  8  0  5]

 [ 3  1 19 18 14 27 13  4  3  0]

 [ 1  3 20 15 11  4 52  2  1  2]

 [ 5  2 13  6 11 10  3 33  1  7]

 [11  3  3  5  7  3  3  4 46  3]

 [ 5 20  4  3  4  4  5  3  4 39]]
Accuracy = 0.398
```

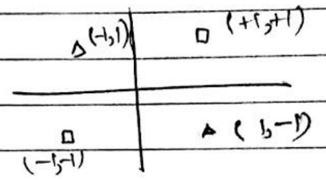

For fold = 0
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve of class 0
ROC curve of class 1
ROC curve of class 2
ROC curve of class 3
ROC curve of class 4
ROC curve of class 5
ROC curve of class 6
ROC curve of class 7
ROC curve of class 8
ROC curve of class 9

We cannot seperate the XOR linearly with a kernel.

(3) Yes, to model the XOR problem we use polynomial Kernel with degree 2.,

Two class
△ → Class +1
□ → Class −1

$$k(x, x_i) = (1 + x' x_i)^2$$



Input (x)	Output (y)
(−1,−1)	−1
(−1,+1)	+1
(+1,−1)	+1
(+1,+1)	−1

Calculating the Kernel gram matrix, we get

$$K = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}$$

From Dual form ; $Q(\alpha) = \sum_{i=1}^{4} \alpha_i - \frac{1}{2} \sum_{i=1}^{4} \sum_{j=1}^{4} \alpha_i \alpha_j y_i y_j \, K(x_i, x_j)$

We get,

$$Q(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}\big(9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha_1\alpha_4 + 9\alpha_2^2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2\big)$$

Optimizing $Q(\alpha)$ w.r.t. Lagranges multiplier ⇒

$\frac{\partial Q}{\partial \alpha_1} = 1 - \frac{1}{2}(18\alpha_1 - 2\alpha_2 - 2\alpha_3 + 2\alpha_4) = 0$

$9\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 = 1$ — (1)

$\frac{\partial Q}{\partial \alpha_2} = 1 - \frac{1}{2}(18\alpha_2 - 2\alpha_1 + 2\alpha_3 - 2\alpha_4) = 0$

$-\alpha_1 + 9\alpha_2 + \alpha_3 - \alpha_4 = 1$ — (2)

$\frac{\partial Q}{\partial \alpha_3} = 1 - \frac{1}{2}(18\alpha_3 + 2\alpha_2 - 2\alpha_1 - 2\alpha_4) = 0$

$-\alpha_1 + \alpha_2 + 9\alpha_3 - \alpha_4 = 1$ — (3)

$\frac{\partial Q}{\partial \alpha_4} = 1 - \frac{1}{2}(2\alpha_1 - 2\alpha_2 - 2\alpha_3 + 18\alpha_4)$

$\alpha_1 - \alpha_2 - \alpha_3 + 9\alpha_4 = 1$ — (4)

Solving (1), (2), (3) & (4)

We get $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 1/8$

$Q(\alpha)$ has optimum value $= 1/4$

For decision boundary

$$\sum_{i=1}^{4} \alpha_i y_i K(x_2, x_i) = 0$$

x is of the for $[x_1, x_2]' = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

Soln

$$\Rightarrow \frac{1}{8}(-1)(1-x_1-x_2)^2 + \frac{1}{8}(1)(1-x_1+x_2)^2$$
$$+\frac{1}{8}(1)(1+x_1-x_2)^2 + \frac{1}{8}(-1)(1+x_1+x_2)^2$$
$$= 0$$

$$= -(\cancel{1}+\cancel{x_1^2}+\cancel{x_2^2}-\cancel{2x_1}-\cancel{2x_2}+2x_1x_2)$$
$$+(\cancel{1}+\cancel{x_1^2}+\cancel{x_2^2}-\cancel{2x_1}+\cancel{2x_2}-2x_1x_2)$$
$$+(\cancel{1}+\cancel{x_1^2}+\cancel{x_2^2}-\cancel{2x_2}+\cancel{2x_1}-2x_1x_2)$$
$$-(\cancel{1}+\cancel{x_1^2}+\cancel{x_2^2}+\cancel{2x_1}+\cancel{2x_2}+2x_1x_2)$$

$$-8\,x_1x_2 = 0$$
$$\boxed{-x_1x_2 = 0}$$

Hence $-x_1x_2 = y$ is our decision boundary.

(4) Given, $K(x, x') = (1+x^T x')^2$.
Also $x = [x_1, x_2]^T$

$$K(x, x') = \left(1 + [x_1\ x_2]\begin{bmatrix} x_1' \\ x_2' \end{bmatrix}\right)^2$$
$$= (1 + x_1x_1' + x_2x_2')^2$$

$$= 1 + x_1^2x_1'^2 + x_2^2x_2'^2 + 2x_1x_1' + 2x_2x_2'$$
$$+ 2x_1'x_1x_2x_2'$$

Also $K(x, x') = \phi(x)\,\phi(x')$
~~1 is bias, then $\phi(x) =$~~
$$\phi(x) = [\underbrace{1}_{\text{bias}}, x_1^2, x_2^2, \sqrt{2}\,x_1, \sqrt{2}\,x_2, \sqrt{2}\,x_1x_2]$$

classmate
Date
Page

$\boxed{5}$ (i) Equation of hyperplane optimum = $\underline{w^T x + b}$

We need to find b.

Equation of primal form is

$$w = \sum_{i=1}^{N} \alpha_i x_i y_i$$

~~wz~~ $\underline{N=10}$

$$w = \alpha_1 x_1 y_1 + \alpha_4 x_4 y_4 + \alpha_7 x_7 y_7 + \alpha_9 y_9 x_9$$

Because $\alpha_i = 0$ of other i's

Putting values from table.

$$w = 0.414 \begin{bmatrix} 4 \\ 2.9 \end{bmatrix}(1) + 1.18 \begin{bmatrix} 2.5 \\ 1 \end{bmatrix}(-1)$$

$$+ 1.18 \begin{bmatrix} 3.5 \\ 4 \end{bmatrix}(1) + 0.414 \begin{bmatrix} 2 \\ 2.1 \end{bmatrix}(-1)$$

$$= \begin{bmatrix} 1.656 \\ 1.2006 \end{bmatrix} - \begin{bmatrix} 2.95 \\ 1.18 \end{bmatrix} + \begin{bmatrix} 4.13 \\ 4.72 \end{bmatrix} - \begin{bmatrix} 0.828 \\ 0.8694 \end{bmatrix}$$

$$w = \begin{bmatrix} 2.008 \\ 3.8712 \end{bmatrix}$$

For SVM to belong to class 1
$w^T x + b = 1 \Rightarrow b = 1 - w^T x$
For SVM to belong to class -1
$w^T x + b = -1 \Rightarrow b = -1 - w^T x$

$b_1 = 1 - \begin{bmatrix} 2.008 & 3.8712 \end{bmatrix} \begin{bmatrix} 4 \\ 2.9 \end{bmatrix}$
$= -18.63$

$b_2 = -1 - \begin{bmatrix} 2.008 & 3.8712 \end{bmatrix} \begin{bmatrix} 2.5 \\ 1 \end{bmatrix}$
$= -9.90$

$b_3 = 1 - \begin{bmatrix} 2.008 & 3.8712 \end{bmatrix} \begin{bmatrix} 3.5 \\ 4 \end{bmatrix} = -21.56$

$b_4 = -1 - \begin{bmatrix} 2.008 & 3.8712 \end{bmatrix} \begin{bmatrix} 2 \\ 2.1 \end{bmatrix} = -13.15$

Avg value of $b = -15.81$

So eq^n of plane becomes
$w^T x + b \Rightarrow \begin{bmatrix} 2.008 & 3.8712 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - 15.81 = 0$

$$\boxed{2.008 x_1 + 3.8712 x_2 - 15.81 = 0}$$

(ii) Support vector is when $\alpha_i \neq 0$ i.e.
x_1, x_4, x_7 & x_9.

(iii) As calculated above $w = \begin{bmatrix} 2.008 \\ 3.8712 \end{bmatrix}$

for point (3, 3)

$w^T x + b = \begin{bmatrix} 2.008 & 3.8712 \end{bmatrix} \begin{bmatrix} 3 \\ 3 \end{bmatrix} - 15.81$
$= 1.8276$

classmate
Date
Page

which is greater than 1. Hence it belongs to class +1.

(6) $K(u,v) = e^{(-\gamma \|u-v\|^2)}$ (given)

$u, v \in \mathbb{R}^1$ (given)

$K(u,v) = \exp(-\gamma(u^2+v^2-2uv))$

$= \exp(-\gamma u^2)\exp(-\gamma v^2)\exp(2\gamma uv)$

By taylor expansion we know $e^x = 1+x+\frac{x^2}{2!}---$

so $e^{2\gamma uv} = 1+2\gamma uv+\frac{(2\gamma uv)^2}{2!}---$

$\underline{\gamma = 1}$

$e^{2uv} = 1+2uv+\frac{(2uv)^2}{2!}+---$

$\phi(u) = e^{-\gamma u^2}\left[1, \sqrt{2\gamma}\,u, \sqrt{\frac{2^2\gamma^2}{2!}}u^2 --- , \sqrt{\frac{(2\gamma)^n}{n!}}u^n --\right]$

When u is very high i.e. $\underline{\gamma=1}$
$u^{1000000}$, coff been $=\sqrt{\frac{(2)^{1000000}}{1000000!}}$

we know $n! > 2^n$ for $n > 3$

Hence $\sqrt{\frac{(2)^{100000}}{1000000!}} \xrightarrow{\text{tends}} 0$

So coff of $u^{1000000}$ or any higher order are tending to 0.