ASSIGNMENT 3 REPORT

Answer 1.

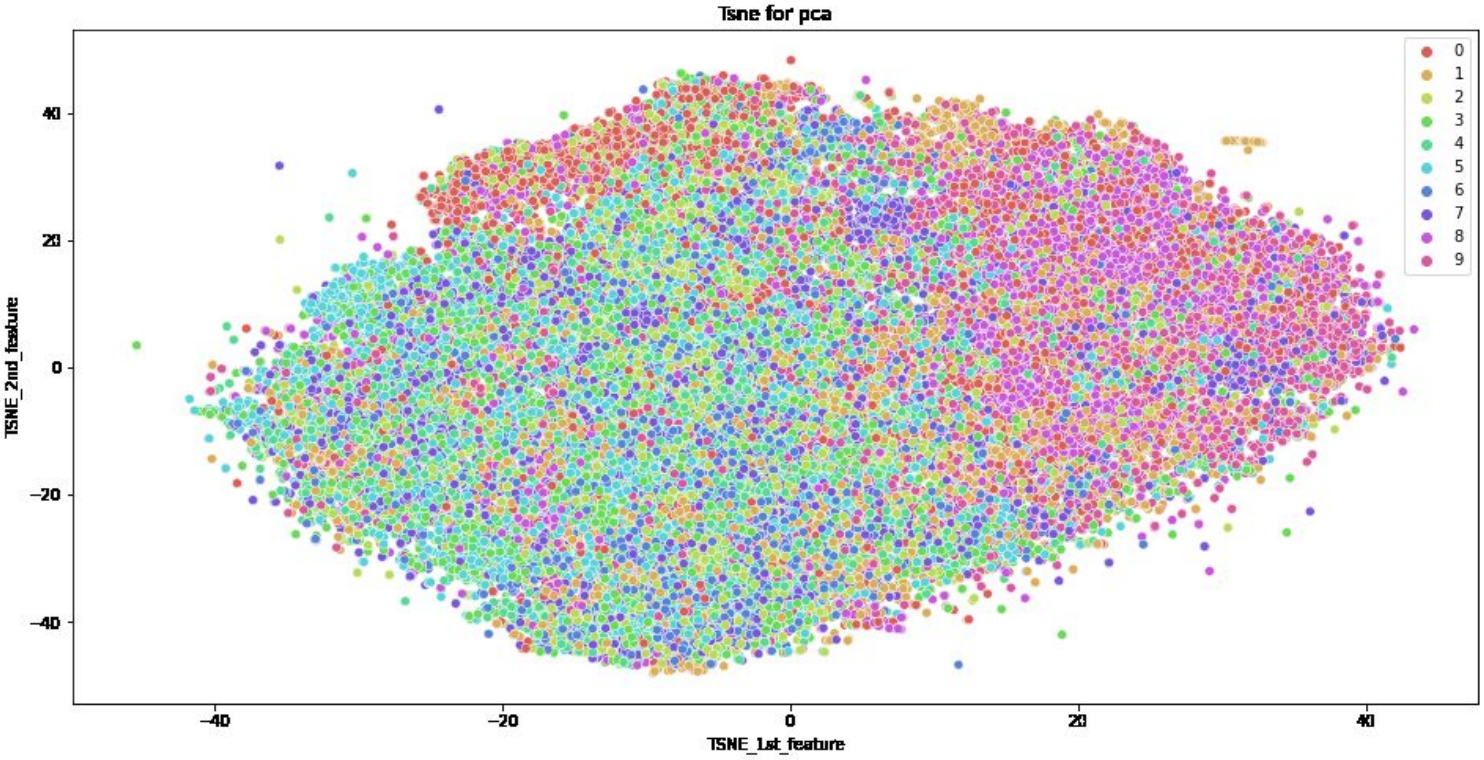
1a) Performed PCA using sklearn on the dataset such that 90% of the total variance is retained - feature descriptor(1)

The number of components that retain 90% of the total variance after performing PCA turned out to be 99. The implementation of the above observation is done in the code file attached for question 1.

1b) Combined HOG and color histogram, implemented from scratch on a whole
ie., (hog + color hist) - feature descriptor(2). The total components turned out to be 324. The implementation of this too is done in the code file attached for question 1.

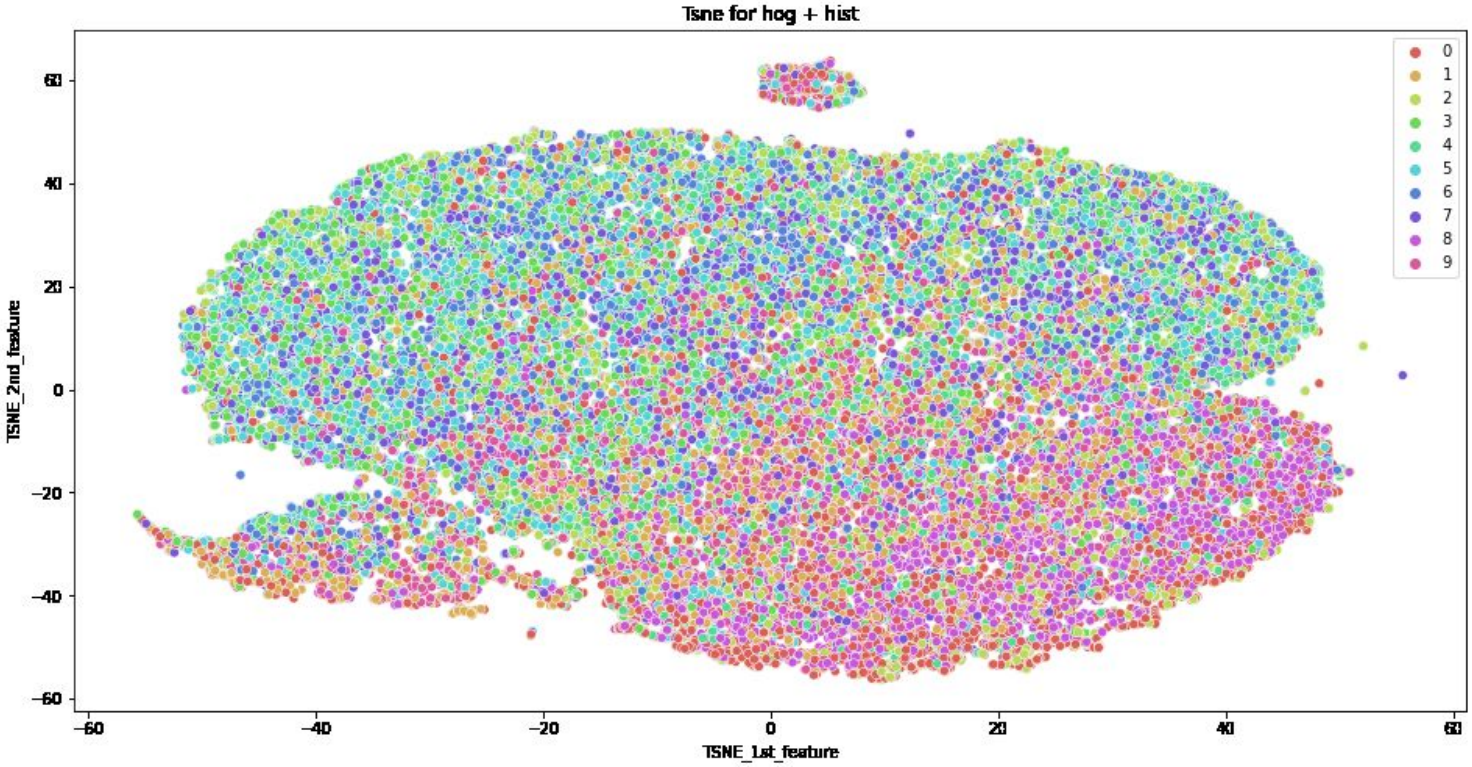
2. This is the 2-D TSNE plot for pca - feature descriptor 1 that is computed in part 1.

As is visible, classes 4 and 5 form a cluster where they occur together, same with classes 8 and 9 and classes 0 and 2. Classes 6 and 7 are all over the plot. Same with class 1 and 3.



This is the 2-D TSNE plot for hog + colour histogram- feature descriptor 2 that is computed in part 1.

Here too, classes 4 and 5 form a cluster where they occur together, same with classes 8 and 9 and classes 0 and 2. Classes 6 and 7 are all over the plot. Same with class 1 and 3.



3

Grid search for descriptor 1 - pca

The best performing kernel acc to the above distribution would be radial basis function kernel with C = 10. So to save time only these parameters were passed to the GridsearchCV function.

Test accuracy - 55%

Train accuracy - 92%

Run time - 80 minutes

Gridsearch for descriptor 2 - hog + colorhist
Same is true for this descriptor too. The gridsearch is run over rbf kernel with C = 10.

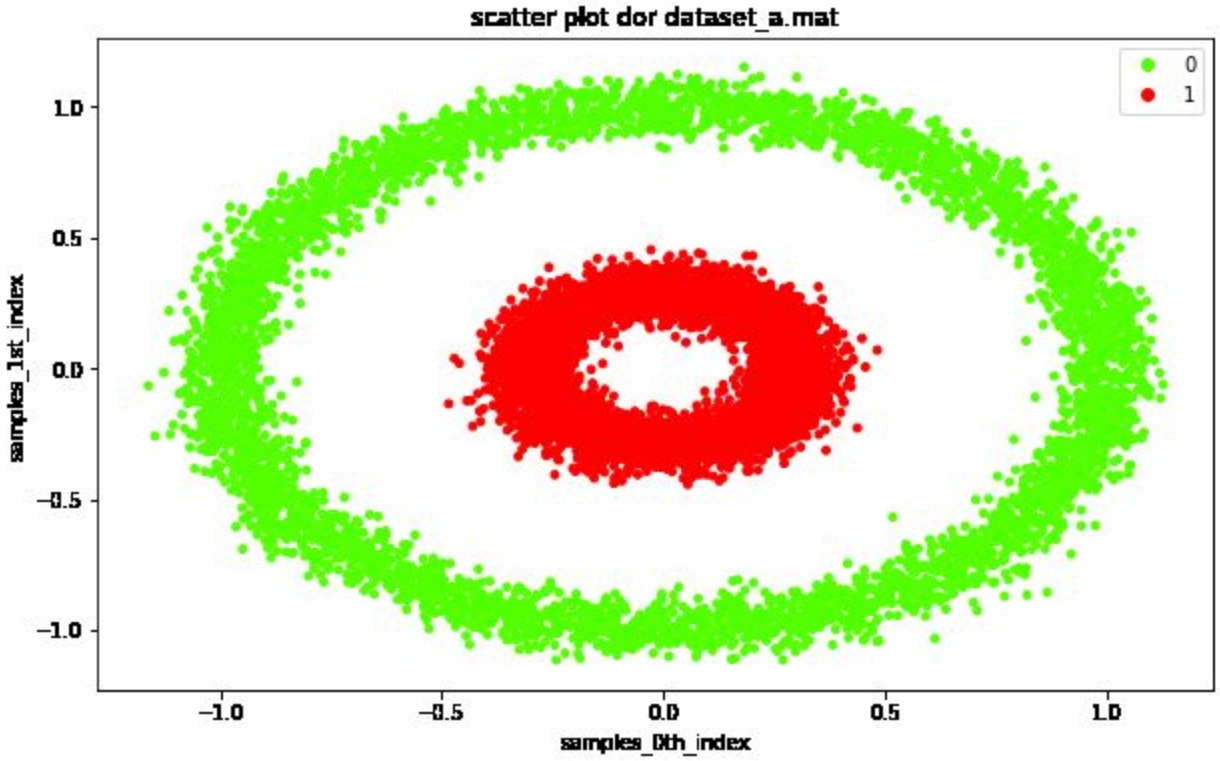
Train accuracy - 88.8%
Test accuracy - 62.31%
Run time - 130 minutes

4
A new training set by extracting the support vectors from the SVM fitted in (3) is formed.
Now another SVM model is prepared with the new training set and fit and predict are done.

Accuracies for pca
Train accuracy - 92.3%
Test accuracy - 56%
Accuracies for hog
Train accuracy - 88.9%
Test accuracy - 62%

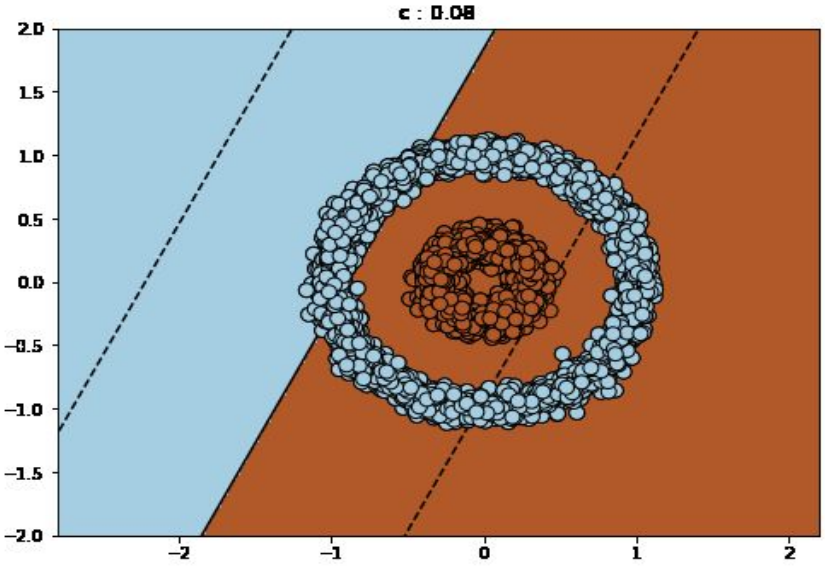
Answer 2.

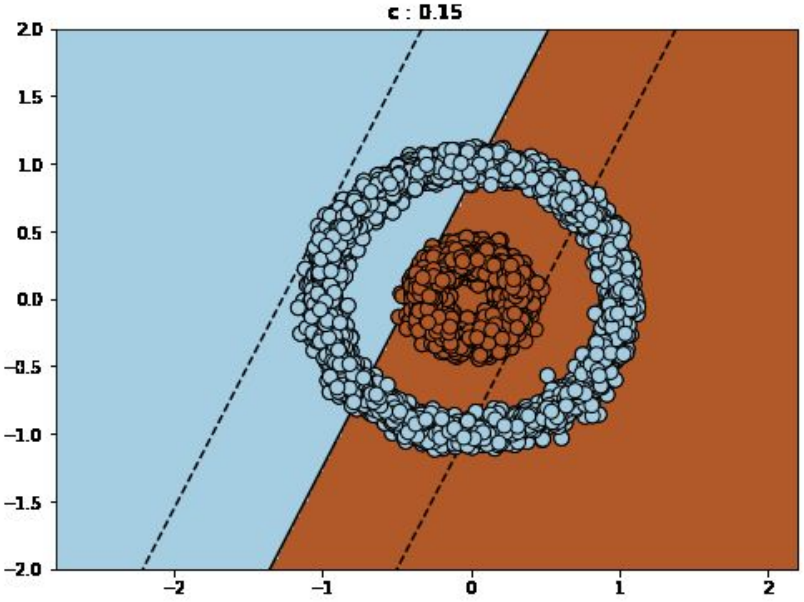
1. This is the visualization of the dataset_a in the form of a scatter plot. The two classes do not have an overlap and are separable, though not by a linear classifier.

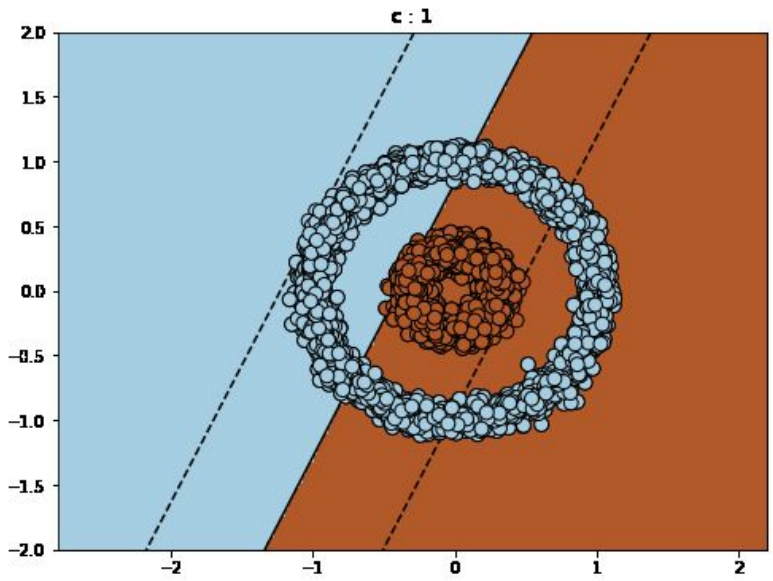


2. Divided the dataset into 80-20 ratio. The 20% is the test set. Used the fit() function from the sklearn for the SVM, predict implemented from the scratch using the attributes of the model created.

   For linear model three different values for c were taken out of which c = 0.15 performs best. Though the accuracy is only 68% but that is the best we can achieve using linear kernel. As the dataset is not linearly separable, we need to apply a different kernel here.







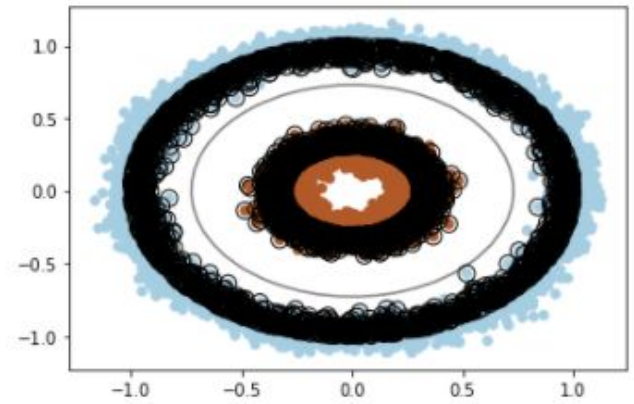

```
Accuracy for c =  0.08  is :  0.6015
Accuracy for c =  0.15  is :  0.683
Accuracy for c =  1  is :  0.681
```

2(c). In this part, 6 different C and Gamma values are taken. These are:

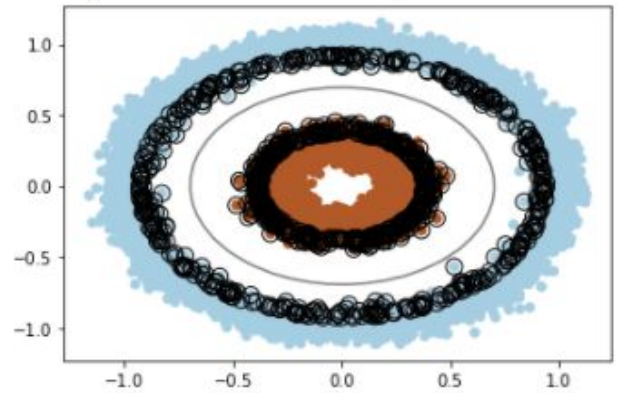
```
[[0.08,0.1],[0.1,0.1],[0.15,0.2],[0.3,0.4],[0.5,0.6],[1,0.8]]
```

Therefore, the plots generated from these values are as shown below:

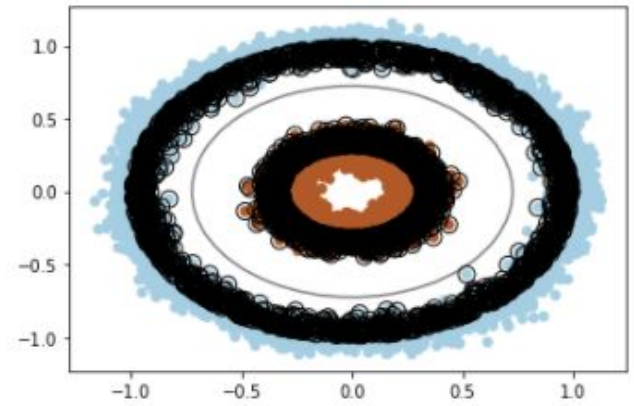
Accuracy for c = 0.08 and for gamma = 0.1 is : 1.0



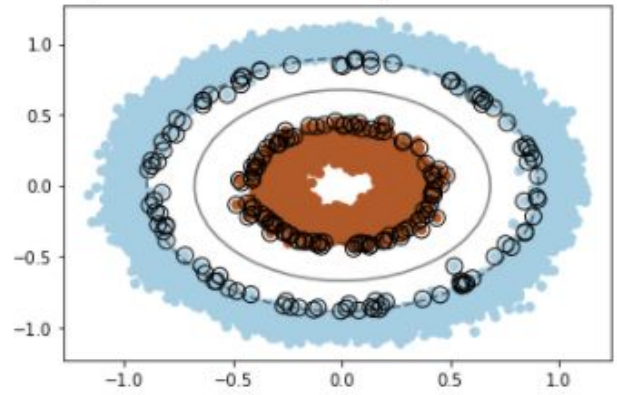
Accuracy for c = 0.15 and for gamma = 0.2 is : 1.0



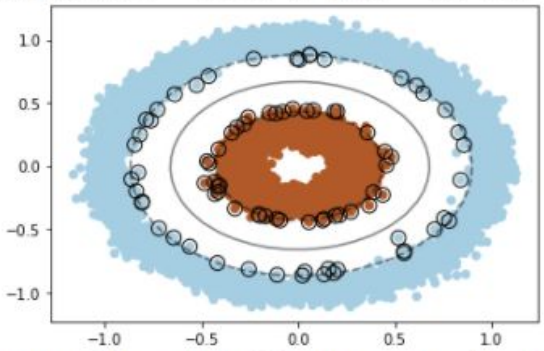
Accuracy for c = 0.1 and for gamma = 0.1 is : 1.0



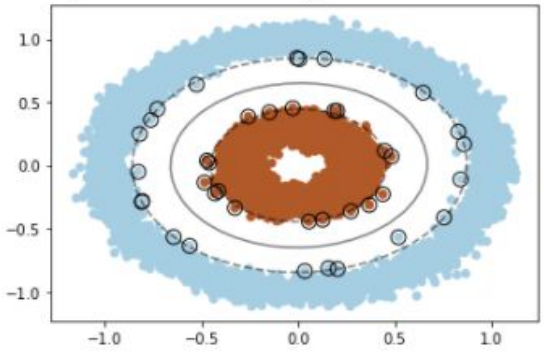
Accuracy for c = 0.3 and for gamma = 0.4 is : 1.0



Accuracy for c = 0.5 and for gamma = 0.6 is : 1.0



Accuracy for c = 1 and for gamma = 0.8 is : 1.0



Therefore, looking at the diagrams above, we see that we get the same accuracy for all values of C and Gamma. Hence, based on the diagram, we have taken C = 0.15 and Gamma = 0.1 for the next part with SVM (RBF kernel) in sklearn.

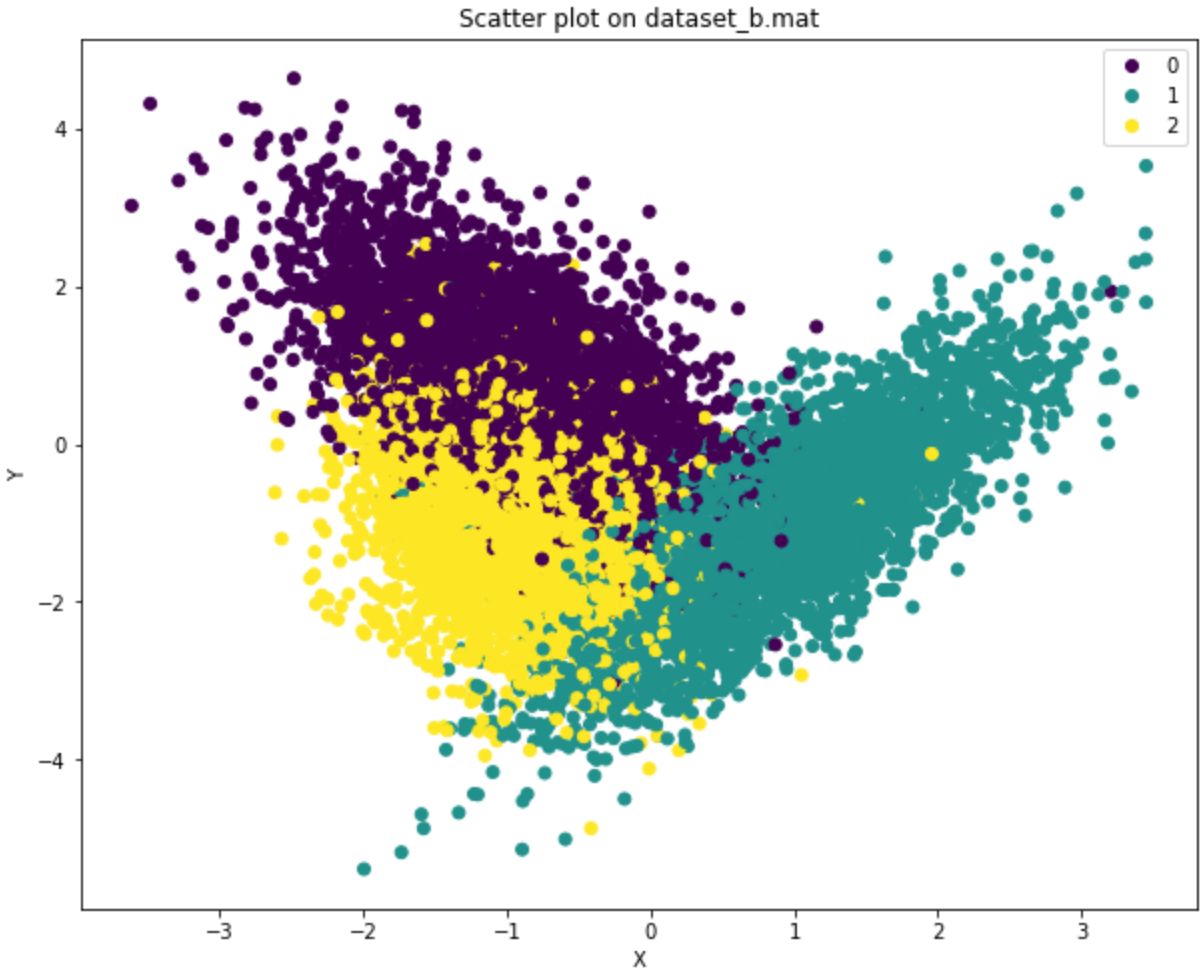
2(d). Using SVM from sklearn in this part, we used the optimum value of 0.15 for C in linear SVM and C = 0.15, gamma = 0.1 for SVM with RBF kernel.

The accuracies achieved are similar to the ones achieved in Question 2(b) and 2(c) respectively. The accuracies are as shown below:

```
Accuracy in Linear SVM for optimum C value 0.15 is  0.683
Accuracy in SVM with RBF kernel for optimum C value 0.15 and optimal Gamma value 0.1 is  1.0
```

Answer 3

3(a). The dataset “dataset_b” given is visualized using matplotlib and the plot obtained is as shown below:



3(b). The class wise accuracies on different values of C and Gamma using OVR are as shown below:
(`cval = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0],gval = [0.1,0.2,0.4,0.6,0.8,1.0,2.0,3.0]`)

	Fold No.	C	Gamma	Class	Accuracy
0	0	0.1	0.1	0	89.05
1	0	0.1	0.1	1	95.20
2	0	0.1	0.1	2	88.55
3	0	0.1	0.2	0	89.45
4	0	0.1	0.2	1	95.30
...	...	...	...	...	...
1195	4	1.0	2.0	1	95.85
1196	4	1.0	2.0	2	90.15
1197	4	1.0	3.0	0	91.50
1198	4	1.0	3.0	1	95.85
1199	4	1.0	3.0	2	90.15

C_mean

```
[[0.1, 91.60875],
 [0.2, 91.63833333333334],
 [0.3, 91.645],
 [0.4, 91.66083333333333],
 [0.5, 91.66999999999999],
 [0.6, 91.67083333333333],
 [0.7, 91.66749999999999],
 [0.8, 91.67291666666667],
 [0.9, 91.67458333333335],
 [1.0, 91.67166666666667]]
```

G_mean

```
[[0.1, 91.72533333333332],
 [0.2, 91.72533333333332],
 [0.4, 91.72533333333332],
 [0.6, 91.72533333333332],
 [0.8, 91.72533333333332],
 [1.0, 91.72533333333332],
 [2.0, 91.72533333333332],
 [3.0, 91.72533333333332]]
```

From the values of the table and the two outputs, we see that the optimum value of C is 0.9 and that of Gamma shows the same accuracy in every value, hence taking Gamma value as 0.1 for comparing it with the values obtained from sklearn in part d.

3(c). The class wise accuracies on different values of C and Gamma using OVO are as shown below:
(`cval = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0],gval = [0.1,0.2,0.4,0.6,0.8,1.0,2.0,3.0]`)

	Fold No.	C	Gamma	Class i	Class j	Accuracy
0	0	0.1	0.1	0	1	89.35
1	0	0.1	0.1	1	2	68.40
2	0	0.1	0.1	2	0	69.45
3	0	0.1	0.2	0	1	88.70
4	0	0.1	0.2	1	2	68.20
...	...	...	...	...	...	...
1195	4	1.0	2.0	1	2	64.40
1196	4	1.0	2.0	2	0	70.45
1197	4	1.0	3.0	0	1	90.10
1198	4	1.0	3.0	1	2	64.95
1199	4	1.0	3.0	2	0	71.20

C_mean2

```
[[0.1, 87.22333333333331],
 [0.2, 87.03833333333333],
 [0.3, 86.97208333333333],
 [0.4, 86.97000000000001],
 [0.5, 86.97791666666664],
 [0.6, 87.00499999999998],
 [0.7, 87.02374999999998],
 [0.8, 87.04833333333332],
 [0.9, 87.08916666666666],
 [1.0, 87.11874999999999]]
```

G_mean2

```
[[0.1, 86.98033333333332],
 [0.2, 86.98033333333332],
 [0.4, 86.98033333333332],
 [0.6, 86.98033333333332],
 [0.8, 86.98033333333332],
 [1.0, 86.98033333333332],
 [2.0, 86.98033333333332],
 [3.0, 86.98033333333332]]
```

Thus, in the case, we see from the above tables that the optimum C and Gamma values are both 0.1. Hence, we use these values to compare with that obtained using sklearn in part d.

3(d). Using sklearn for SVM with RBF kernel, we first apply OVR by putting decision_function_shape = 'ovr' and then obtaining the accuracies over folds as follows:

	Fold No.	Accuracy
0	0	91.15
1	1	91.50
2	2	91.40
3	3	91.80
4	4	92.85

From the above table, we see that the accuracies over folds are similar to the one obtained in Q3(b), thus, the implemented OVR works the same as the SVM OVR decision function shape.

Now for OVO, we put decision_function_shape = 'ovo' and obtain the accuracies as follows:

	Fold No.	Accuracy
0	0	86.90
1	1	87.00
2	2	87.00
3	3	87.40
4	4	88.15

From the above table for OVO, we see that the accuracies obtained from sklearn SVM with decision function shape as OVO is similar to the ones obtained class wise in Q3(c), thus we can infer that the OVO manual implementation was correct in terms of the sklearn SVM decision function shape used.