***Name**: Souham Biswas* ***CWID**: A20365242* ***Email**: sbiswas7@hawk.iit.edu*

CS 584 – Machine Learning, HW 4

1. Problem Statement-

Our objective in this assignment is to implement various Support Vector Machine algorithms.

2. Proposed Solution-

The various algorithms utilized in this assignment include mainly, SVMs implemented with hard and soft margins. The algorithms also include kernel based SVMs utilizing Polynomial and Gaussian kernels.

3. Implementation Details-

We utilize the scikit-learn machine learning python library for the purposes of calculating various metrics like precision, accuracy etc in this assignment along with the matplotlib library for plotting and visualization of various observations and metrics we encounter throughout the experiments. The urllib library has been used to fetch the datasets from the UCI machine learning repository.

Multiple challenges were faced in performing the experiments. A few of them are highlighted below-

- Tackling the exceptions pertaining to incompetent matrix ranks thrown by the solvers.qp function was quite challenging as it required step-by-step analysis of the numeric values.
- Vectorizing the computations for performance gain.
- Minimizing the use of slow pythonic for-loops and replacing them with numpy statements accordingly required morphing of the data structures so as to be compatible with numpy.

Instructions for using the code-

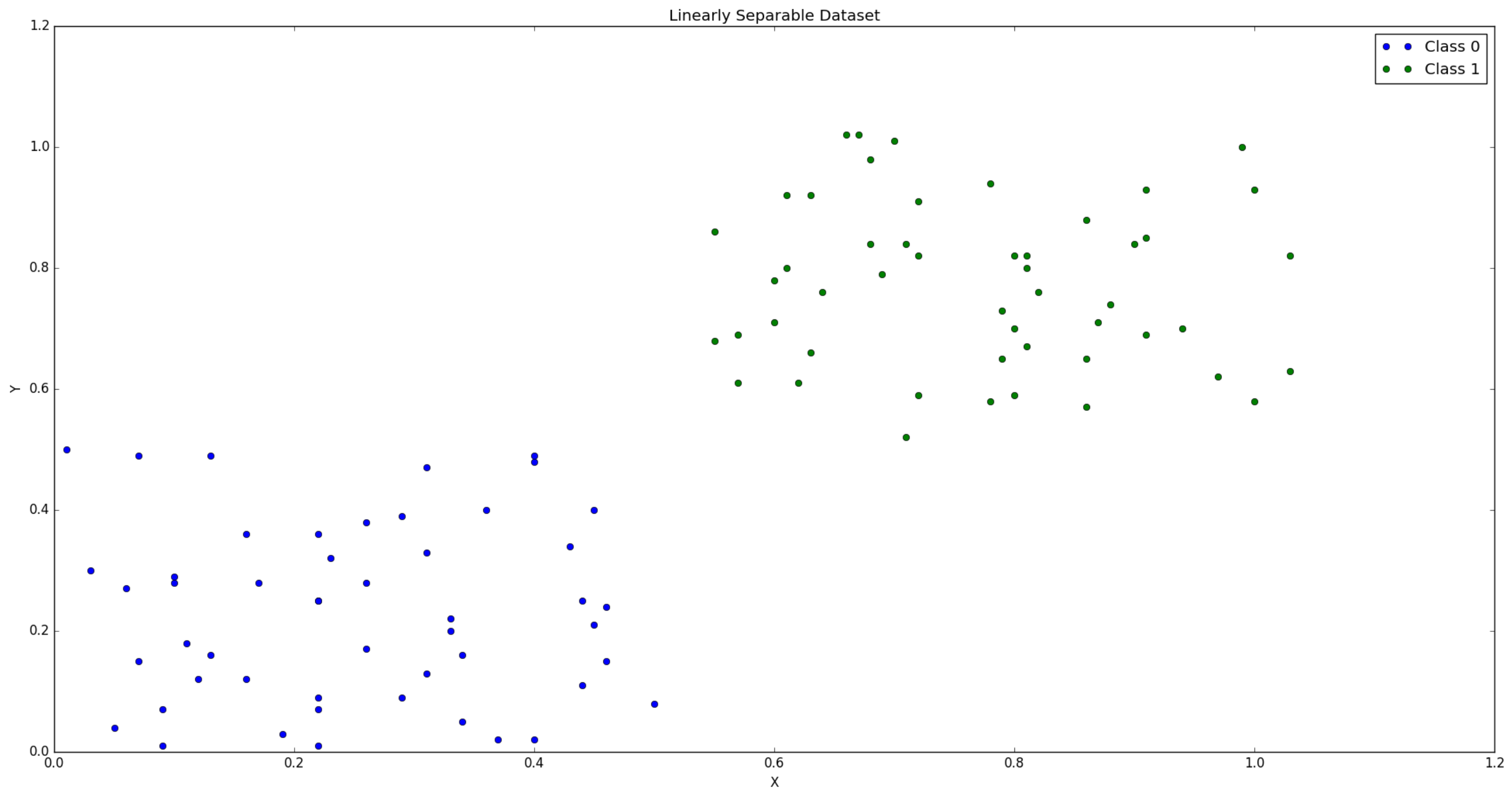
Simply double click the code file and everything should run smoothly. Do note that the code needs an active internet connection to fetch the datasets from the UCI repository.

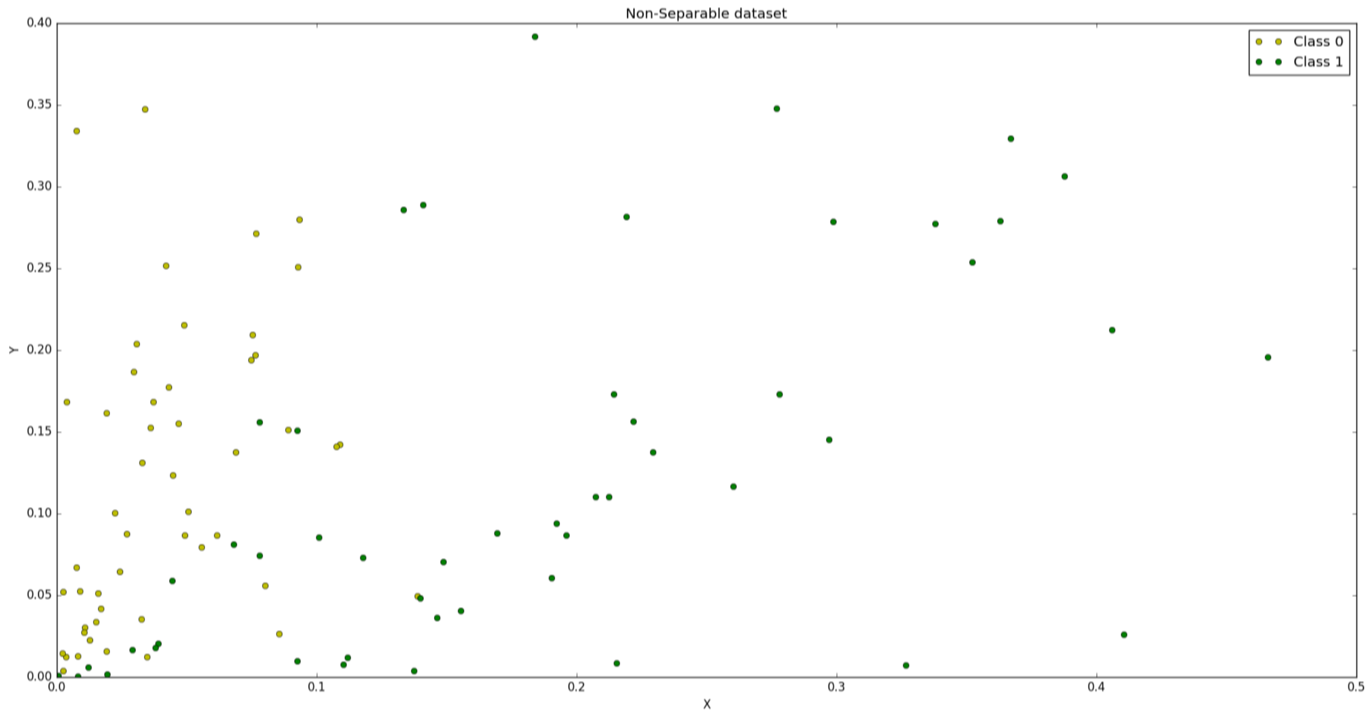
4. **Results and Discussion-**

All the experiments have been performed with 4 fold cross validation. The results shown below are the means of the metrics obtained from each of the cross validation iterations.

Answer 1-

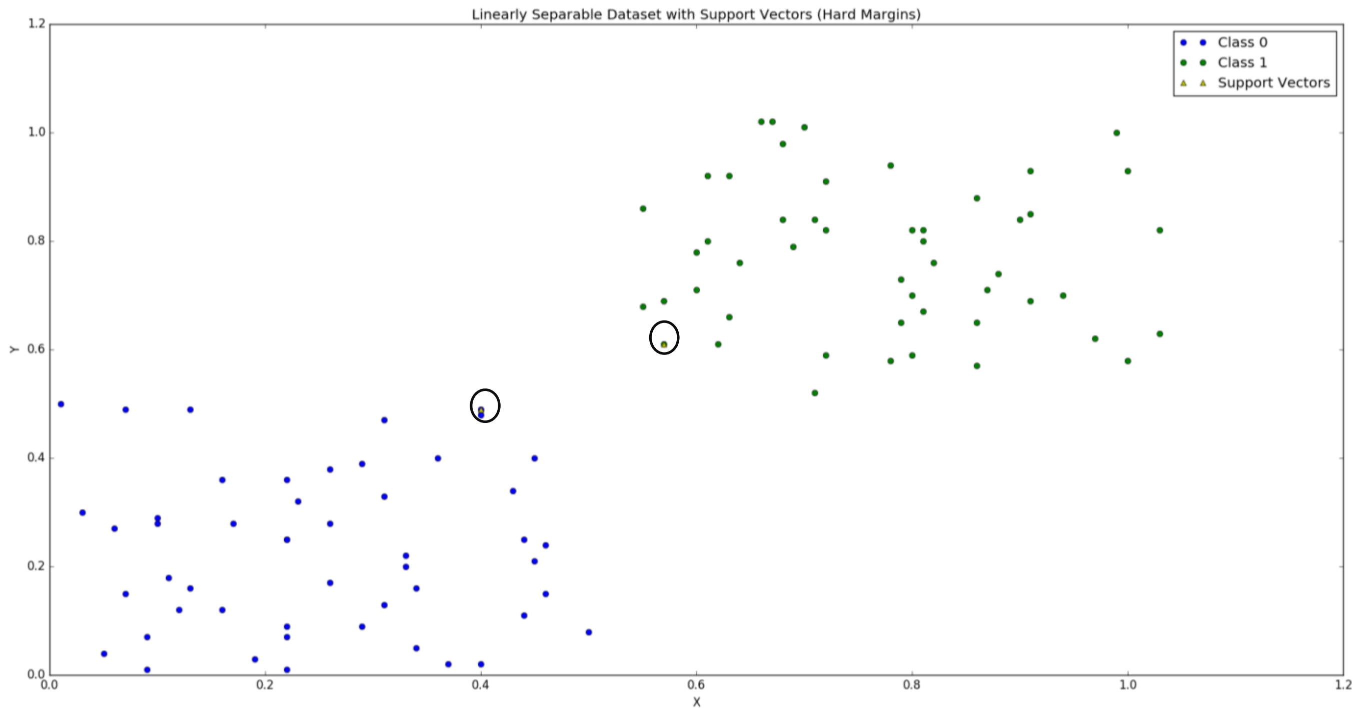
The datasets so generated are illustrated as follows-

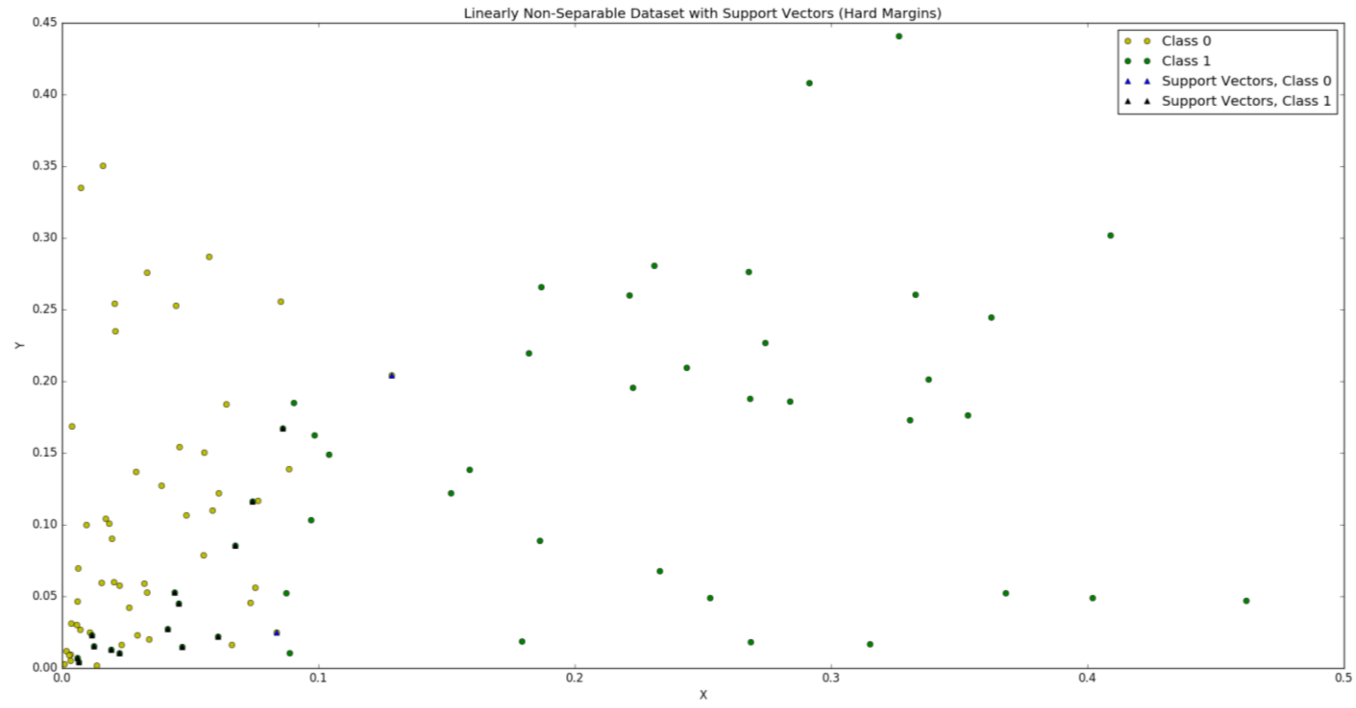



Non-Separable dataset
Class 0
Class 1
Y
X
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.0
0.1
0.2
0.3
0.4
0.5

Answer 2-

A Linear SVM with hard margins was implemented. Presented below are the visual illustrations of the support vectors identified –





As can be seen above, for case of the hard margins, the support vectors identified in the non-separable case tend to belong to mostly one of the classes.

The following results were obtained –

```
Separable Dataset Metrics (Hard Margins)-
Accuracy = 0.99
Precision = [ 0.97727273  1.        ]
Recall = [ 1.          0.98333333]
F-Score = [ 0.98809524  0.99137931]
Confusion Matrix -
[[ 12.5    0.  ]
 [  0.25  12.25]]

Non-Separable Dataset Metrics (Hard Margins)-
Accuracy = 0.5
Precision = [ 0.5  0. ]
Recall = [ 1.  0.]
F-Score = [ 0.66444959  0.        ]
Confusion Matrix -
[[ 12.5   0. ]
 [ 12.5   0. ]]
```

The performance of the hard margin SVM is not at all impressive on the non-separable dataset. However, it’s performance is near perfect for the separable dataset. This is so because the hard margin tries to approximate a linear line/plane/hyper-plane which contradicts with the very nature of the non-separable dataset in which the boundary between both the class data points is irregular.

Answer 3-

Primal Objective -

$$L_P = \frac{1}{2}\|W\|^2 + C\sum_{i=1}^{m}\varepsilon_i - \sum_{i=1}^{m}\alpha_i\left(y^{(i)}(W^T x_i + W_0) - 1 + \varepsilon_i\right) - \sum_{i=1}^{m}\beta\varepsilon_i$$

Condn^s -

- $\varepsilon_i > 0$
- $y^{(i)}(W^T x^{(i)} + W_0) > 1 - \varepsilon_i$

Setting $\frac{\partial L_P}{\partial W} = 0$,

$$\frac{\partial L_P}{\partial W} = \frac{1}{2}2W - \sum_{i=1}^{m}\alpha_i y^{(i)} x^{(i)}$$

$$\Rightarrow W = \sum_{i=1}^{m}\alpha_i y^{(i)} x^{(i)}$$

Setting $\frac{\partial L_P}{\partial W_0} = 0$,

$$\Rightarrow \sum_{i=1}^{m}\alpha_i y^{(i)} = 0$$

Setting $\frac{\partial L_P}{\partial \varepsilon_i} = 0$,

$$\Rightarrow C - \alpha_i - \beta_i = 0$$

Putting in Primal eq. -

$$L_P = \frac{1}{2}\left(\sum_{i=1}^{m}\alpha_i y^{(i)} x^{(i)}\right)^T \sum_{i=1}^{m}\alpha_i y^{(i)} x^{(i)} + C\sum_{i=1}^{m}\xi_i$$

$$-\sum_{i=1}^{m}\alpha_i y^{(i)}\left(\sum_{j=1}^{m}\alpha_j y^{(j)} x^{(j)}\right)^T x^{(i)}$$

$$-\sum_{i=1}^{m}\alpha_i y^{(i)} w_0 + \sum_{i=1}^{m}\alpha_i - \sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\beta_i\xi_i$$

$\because \alpha_i y^{(i)} = 0$ & $c - \alpha_i = \beta_i$.

$$\therefore \Rightarrow L_P = \frac{-1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^{(i)} y^{(j)} x^{(i)} x^{(j)} + \sum_{i=1}^{m}\alpha_i$$

$\hookrightarrow$ DUAL $\rightarrow$ Needs to be maximized

constraints -

$\rightarrow \alpha_i \geq 0$

$\rightarrow \sum_{i=1}^{m}\alpha_i y^{(i)} = 0$

$\rightarrow c - \alpha_i - \beta_i = 0 \;\forall i,\; \beta_i \geq 0$

$\Rightarrow c - \alpha_i = \beta_i$

$\because \beta_i \geq 0 \Rightarrow c - \alpha_i \geq 0$

$\Rightarrow \alpha_i \leq c$

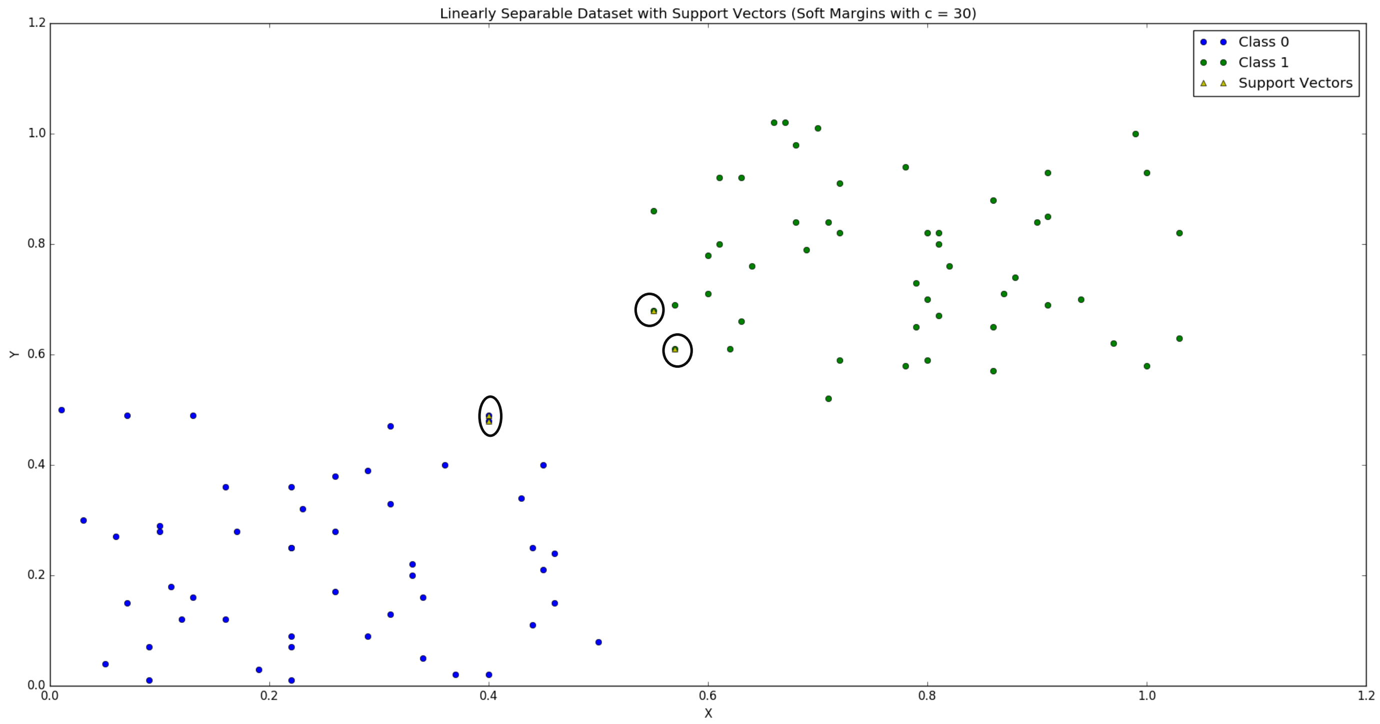
$\because \alpha_i \geq 0$

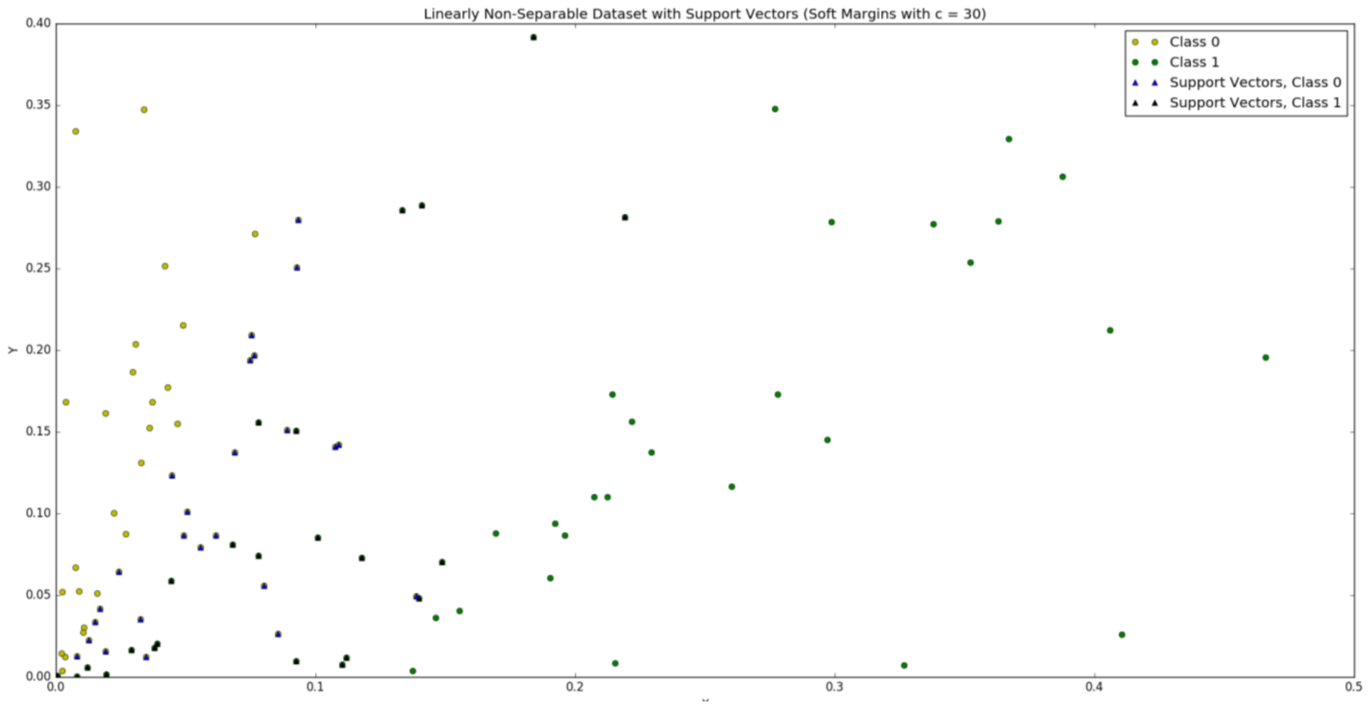
$\Rightarrow 0 \leq \alpha_i \leq c$

Using these constraints L_p can be maximized using the QP solver and the final result may be obtained.

Answer 4-

SVMs with soft margins were trained and implemented on the generated datasets. The support vectors identified are as follows –





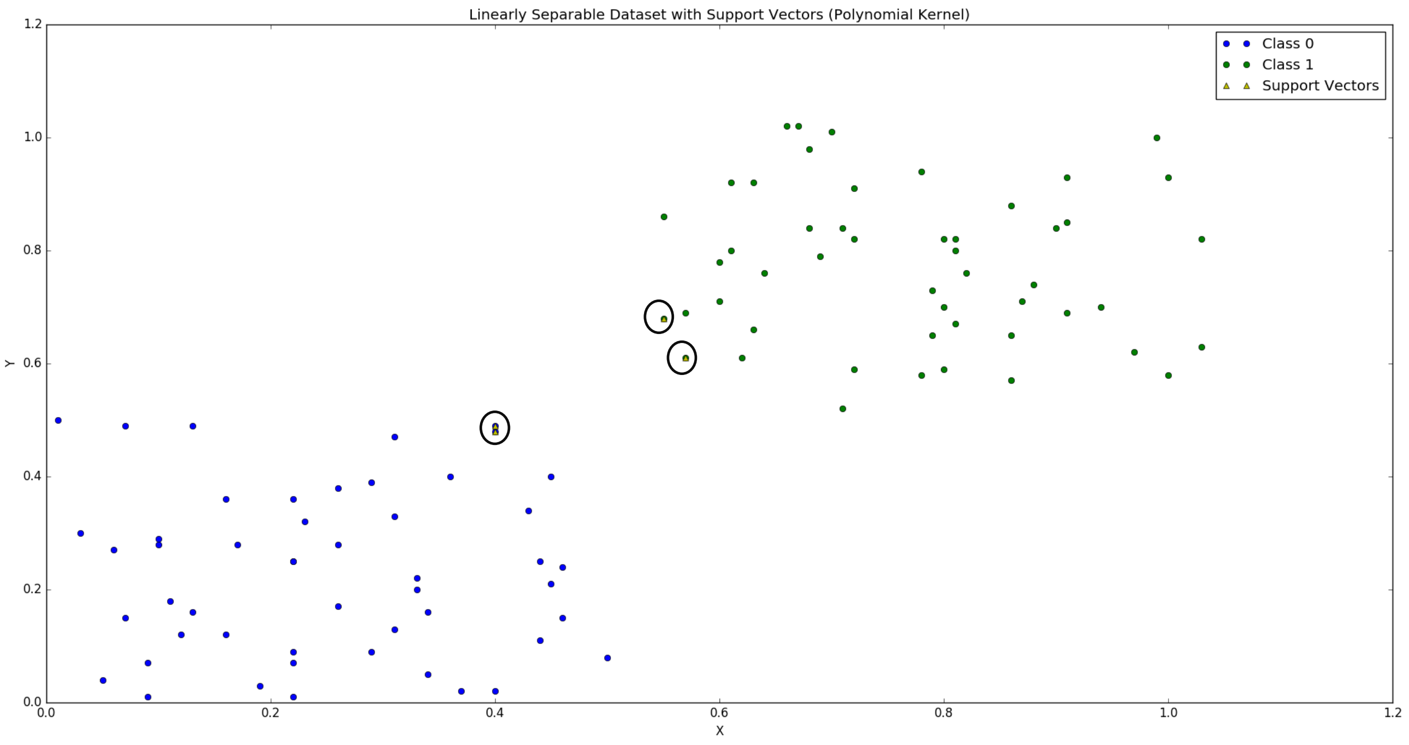
As can be seen this case, the support vectors identified in the non-separable case are more legitimate. The results in this case are also better; they're presented below-

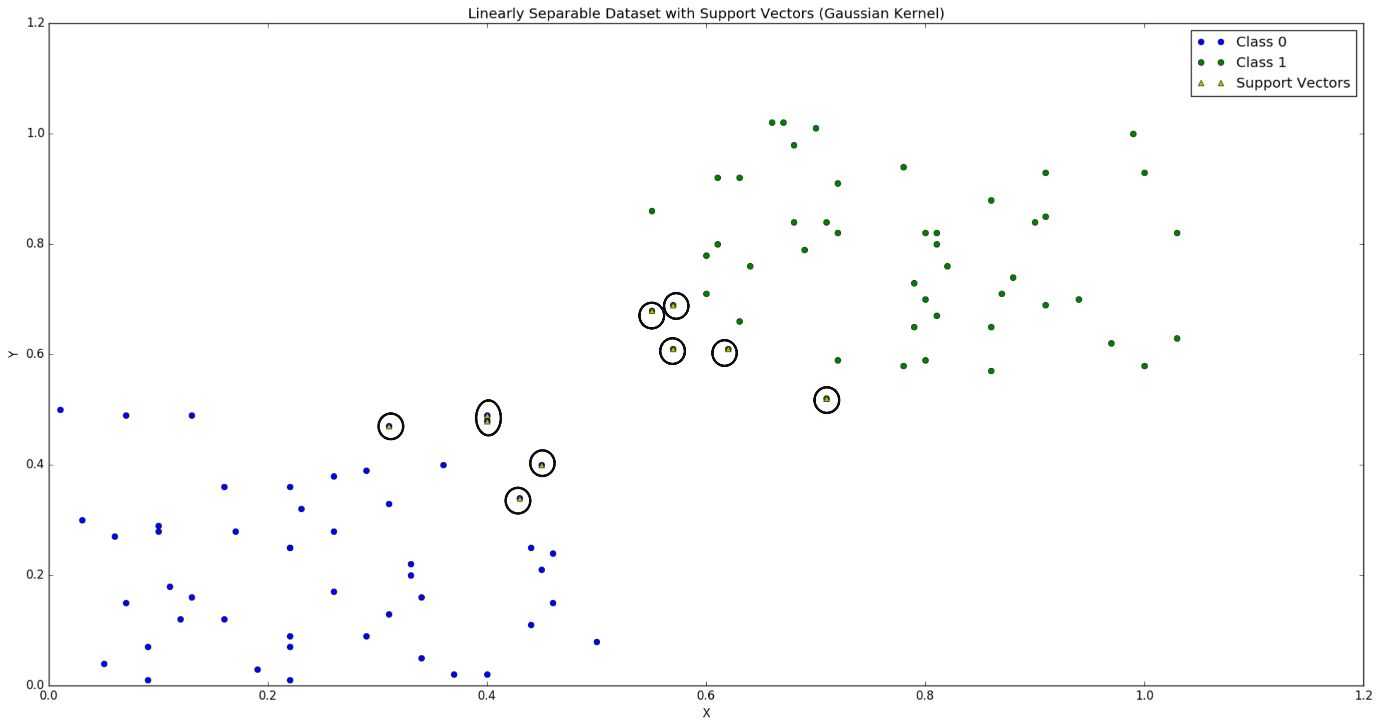
```
Separable Dataset Metrics (Soft Margin with c = 30 )-
Accuracy = 0.98
Precision = [ 0.96291209  1.        ]
Recall = [ 1.          0.9599359]
F-Score = [ 0.98074074  0.97913043]
Confusion Matrix -
[[ 12.5   0. ]
 [  0.5  12. ]]

Non-Separable Dataset Metrics (Soft Margin with c = 30 )-
Accuracy = 0.75
Precision = [ 0.6768883  1.        ]
Recall = [ 1.          0.5357906]
F-Score = [ 0.79531567  0.67589286]
Confusion Matrix -
[[ 12.5    0.  ]
[  6.25   6.25]]
```

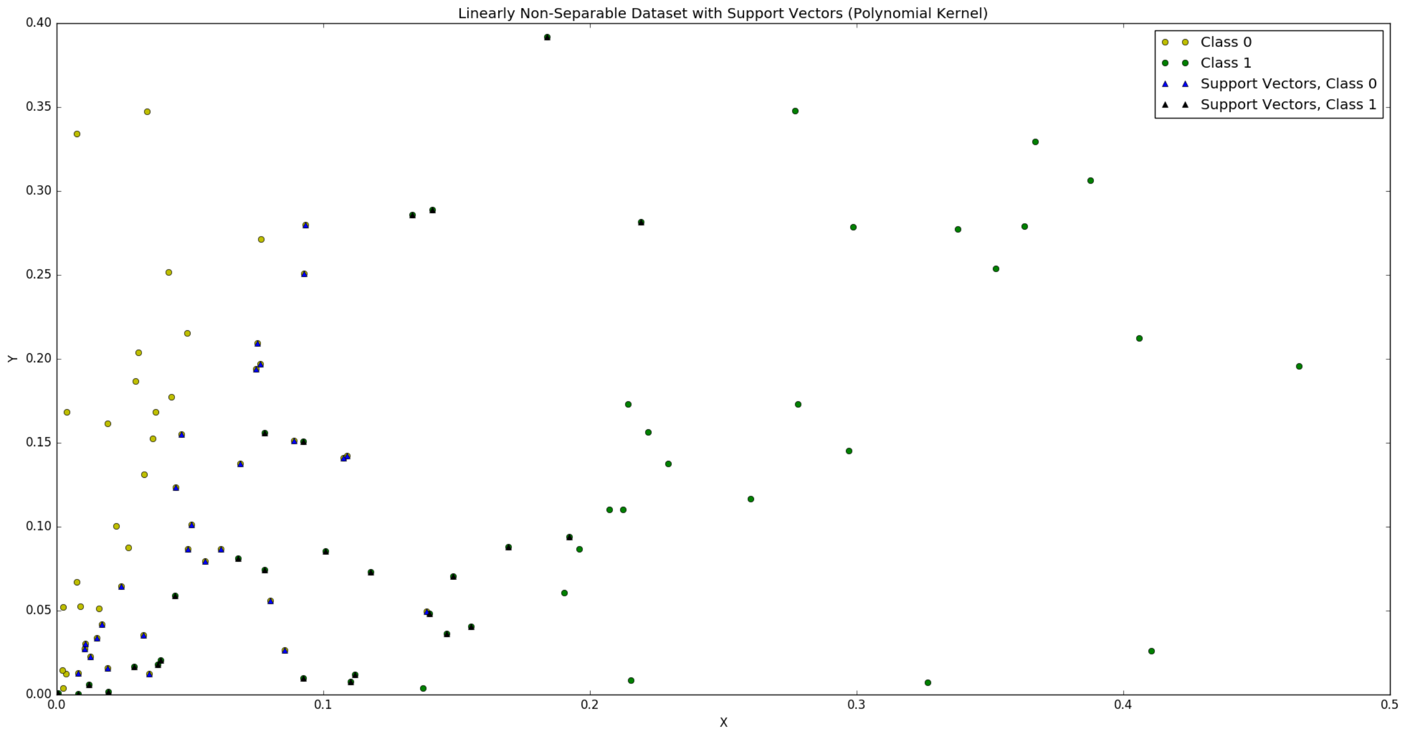
Here, the value of c (which is 30) is basically the slack. As can be observed, the accuracy (and other metrics) for the non-separable dataset has risen to 75% from 50% in the hard margin case. The performance in the case of the separable dataset remains more or less the same.

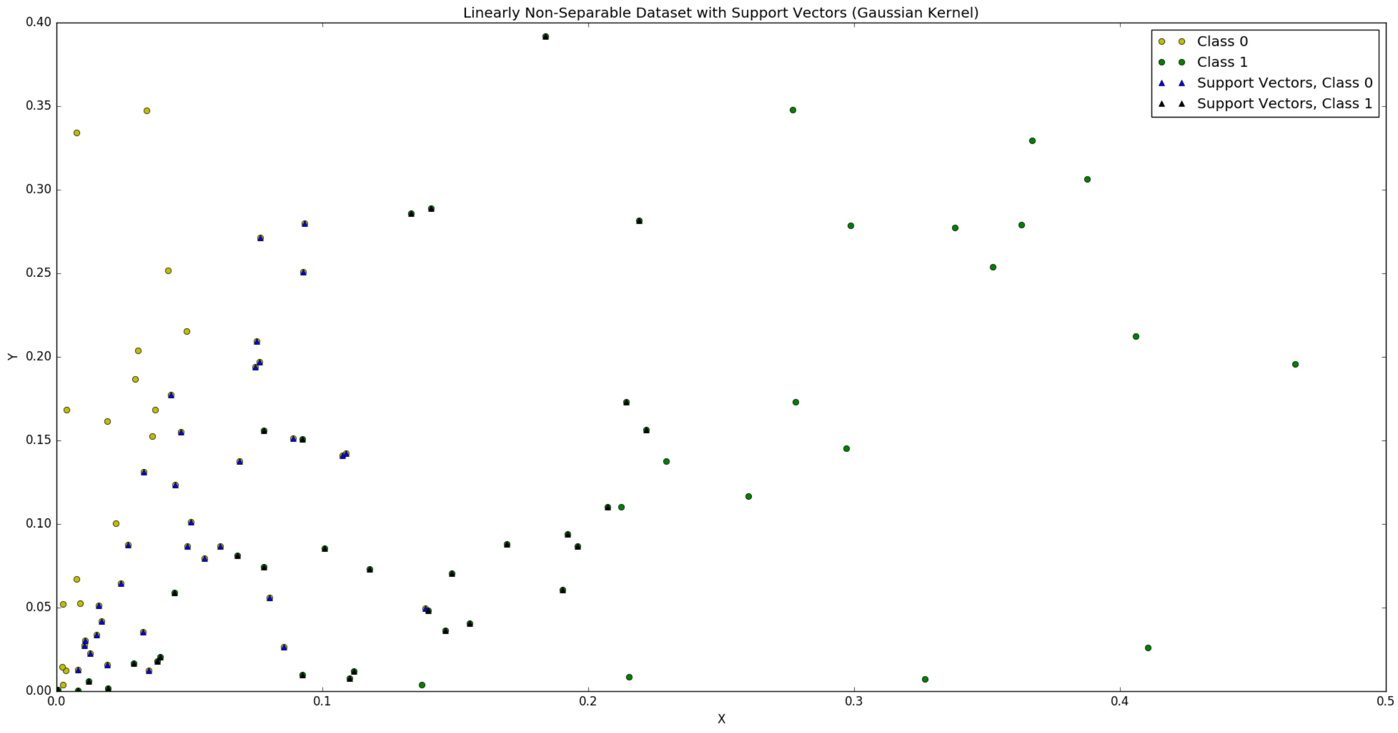
Answer 5-

Kernel based SVMs using Polynomial and Gaussian kernels were implemented on both the generated and the external datasets. The support vectors identified in the generated datasets are illustrated as follows –



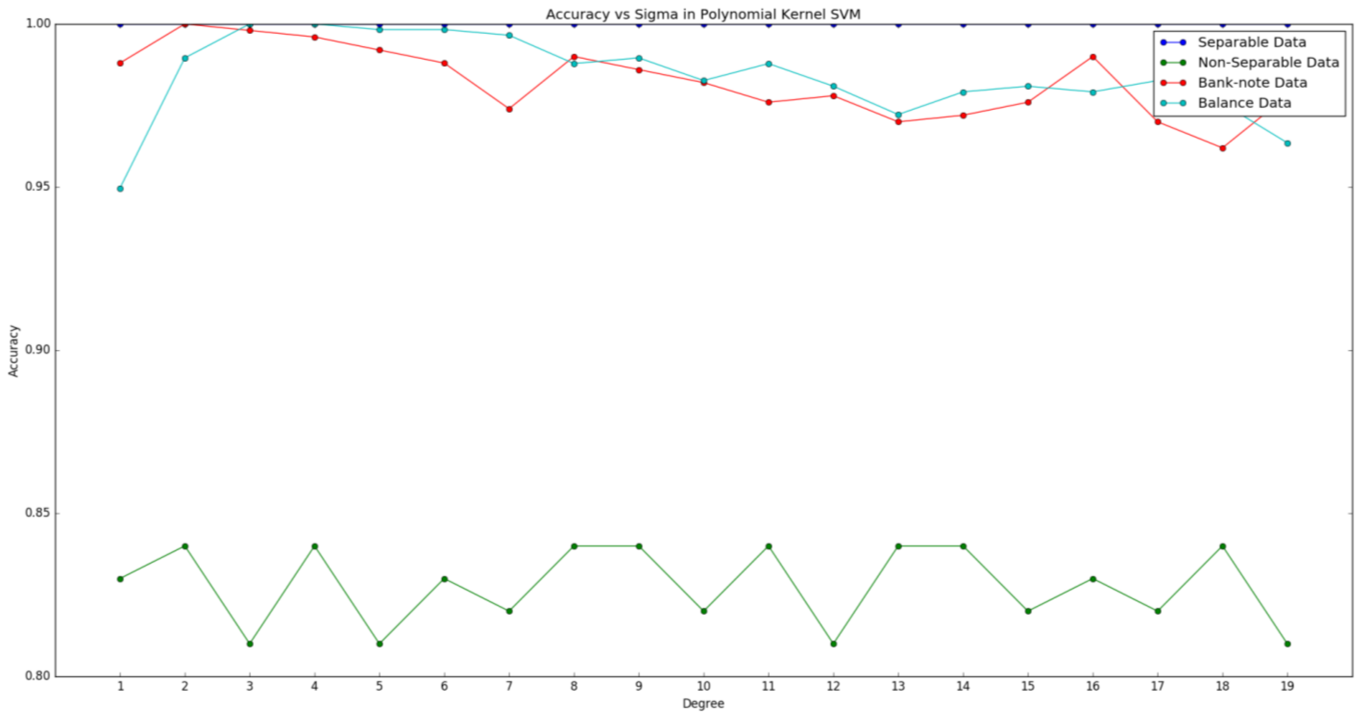


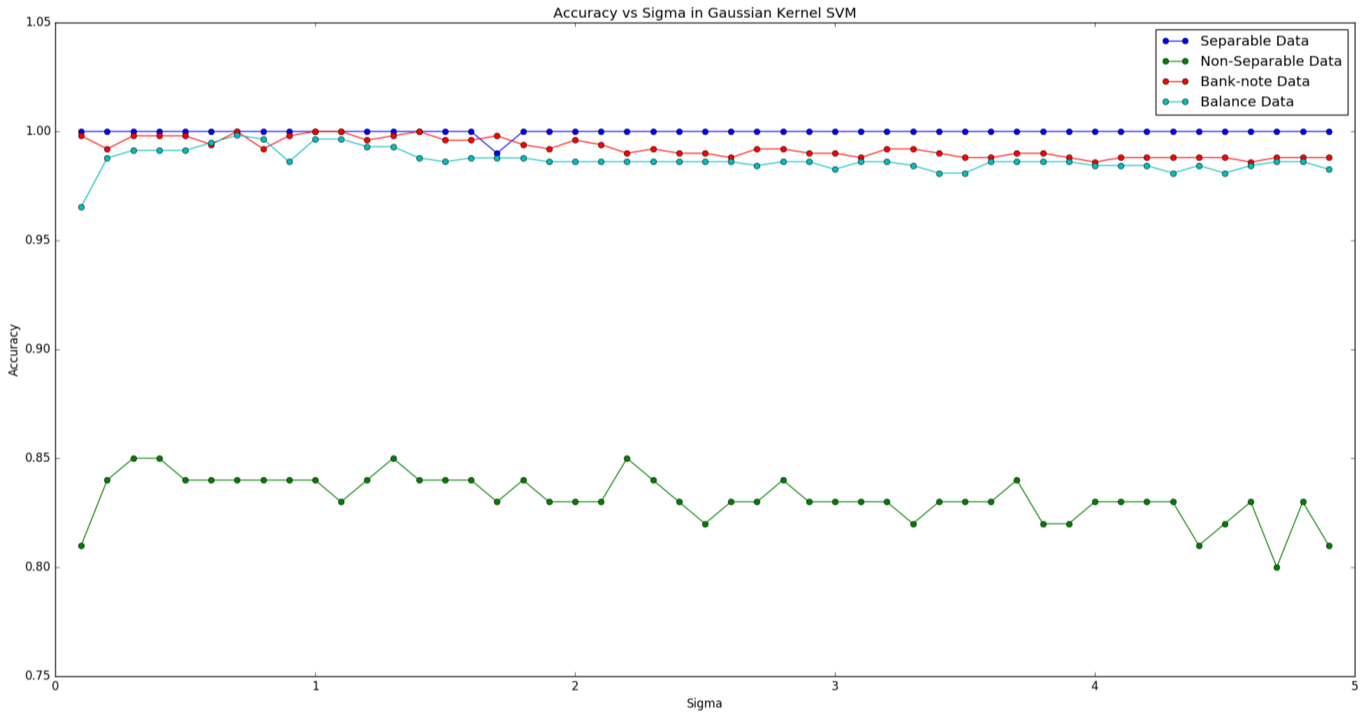

Linearly Non-Separable Dataset with Support Vectors (Polynomial Kernel)
Class 0
Class 1
Support Vectors, Class 0
Support Vectors, Class 1
X
Y
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.0
0.1
0.2
0.3
0.4
0.5


Linearly Non-Separable Dataset with Support Vectors (Gaussian Kernel)
Class 0
Class 1
Support Vectors, Class 0
Support Vectors, Class 1
X
Y
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.0
0.1
0.2
0.3
0.4
0.5

As we can we can see from the plots above, the Gaussian kernel tends to capture more points as support vectors in this case.

The SVM algorithms were tested on the generated datasets and on external datasets. The plot of the performance vs SVM parameters are presented below –





In both the cases, the term “sigma” corresponds to the model parameter. In the case of the polynomial kernel SVM, it the power corresponding to the kernel function and in the case of the Gaussian kernel SVM, it is the denominator of the power of the exponential.

As can be observed from the graphs above, no significant change in the performance occurs by changing the parameters for most of the datasets. However, it can be observed that the accuracy of the polynomial kernel SVM over the Bank note dataset somewhat decreases with the increase in sigma (power).

A sample of the result is presented as follows –

Polynomial Kernel-

```
Separable Dataset Metrics (Polynomial Kernel with degree = 13 )-
Accuracy = 1.0
Precision = [ 1.  1.]
Recall = [ 1.  1.]
F-Score = [ 1.  1.]
Confusion Matrix -
[[ 12.5   0. ]
 [  0.   12.5]]

Non-Separable Dataset Metrics (Polynomial Kernel with degree = 13 )-
Accuracy = 0.84
Precision = [ 0.80059524  0.88881119]
```

```
Recall = [ 0.89621212  0.78681319]
F-Score = [ 0.84422767  0.83268921]
Confusion Matrix -
[[ 11.25   1.25]
 [  2.75   9.75]]

Banknote Dataset Metrics (Polynomial Kernel with degree = 13 )-
Accuracy = 0.97
Precision = [ 0.96758273  0.9735437 ]
Recall = [ 0.97772436  0.95998151]
F-Score = [ 0.97256597  0.96662082]
Confusion Matrix -
[[ 66.     1.5 ]
 [  2.25  55.25]]

Balance Dataset Metrics (Polynomial Kernel with degree = 13 )-
Accuracy = 0.972222222222
Precision = [ 0.98493503  0.95912538]
Recall = [ 0.95836855  0.98685049]
F-Score = [ 0.97140199  0.97271346]
Confusion Matrix -
[[ 69.   3.]
 [  1.  71.]]
```

Gaussian Kernel-

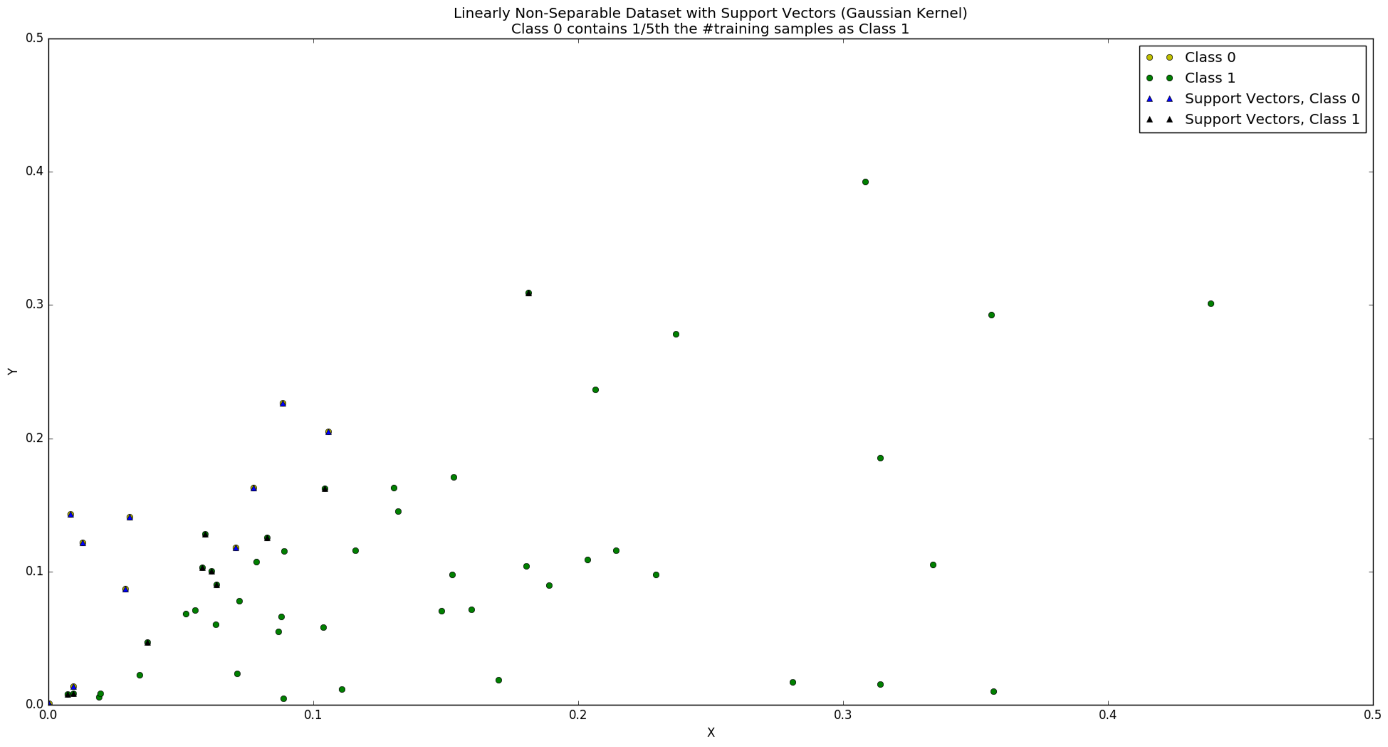
```
Separable Dataset Metrics (Gaussian Kernel with sigma = 4.8 )-
Accuracy = 1.0
Precision = [ 1.  1.]
Recall = [ 1.  1.]
F-Score = [ 1.  1.]
Confusion Matrix -
[[ 12.5   0. ]
 [  0.   12.5]]

Non-Separable Dataset Metrics (Gaussian Kernel with sigma = 4.8 )-
Accuracy = 0.83
Precision = [ 0.77470238  0.90681818]
Recall = [ 0.91396104  0.74837662]
F-Score = [ 0.83747126  0.81928571]
Confusion Matrix -
[[ 11.5    1.  ]
 [  3.25   9.25]]

Banknote Dataset Metrics (Gaussian Kernel with sigma = 4.8 )-
Accuracy = 0.988
Precision = [ 1.          0.9721921]
Recall = [ 0.97969341  1.        ]
F-Score = [ 0.98964314  0.98568572]
Confusion Matrix -
[[ 68.5   1.5]
 [  0.   55. ]]

Balance Dataset Metrics (Gaussian Kernel with sigma = 4.8 )-
Accuracy = 0.986111111111
Precision = [ 0.9859099  0.9862049]
```

```
Recall = [ 0.98615868  0.98624496]
F-Score = [ 0.98600927  0.98620255]
Confusion Matrix -
[[ 71.   1.]
 [  1.  71.]]
```

Upon looking at the performance of both the SVM models, it can be posited that their performances are more or less similar to each other.

Answer 6-

The datapoints from class 0 of the generated were reduced to 1/5th of the training datapoints of class 1 and a Gaussian Kernel SVM was trained and tested on it. The support vectors so identified are as follows –



As can be seen from the plot above, since class 0 contains much fewer datapoints than class 1, most of the support vectors identified belong to class 0 (the blue ones). The test performance results averaged over the cross-validation iterations are as follows-

```
Non-Separable Dataset Metrics (Gaussian Kernel with sigma = 0.5 )-
Class 0 contains 1/5th the #training samples as Class 1
Accuracy = 0.85
```

```
Precision = [ 0.58333333  0.87797619]
Recall = [ 0.5         0.96153846]
F-Score = [ 0.51666667  0.91074074]
Confusion Matrix -
[[  0.75   1.75]
 [  0.5   12.  ]]
```

As can be observed from the results above, although the accuracy is 85%, the precision and recall of class 0 is 58.3% and 50% respectively as compared to 87% precision and 96% recall of class 1. This has undoubtedly been caused by the difference in the number of training examples of each class.