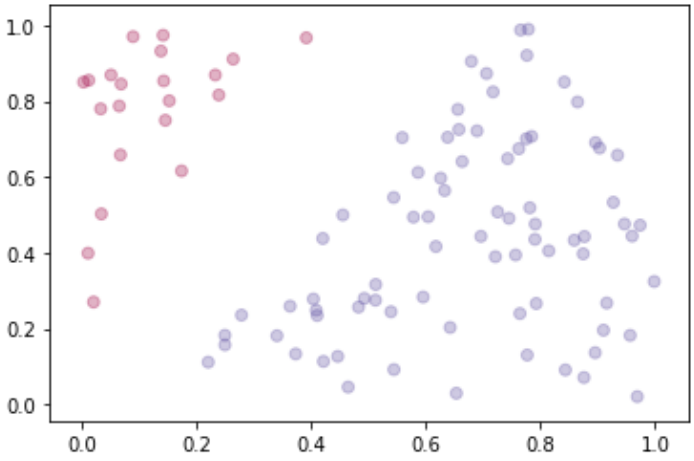
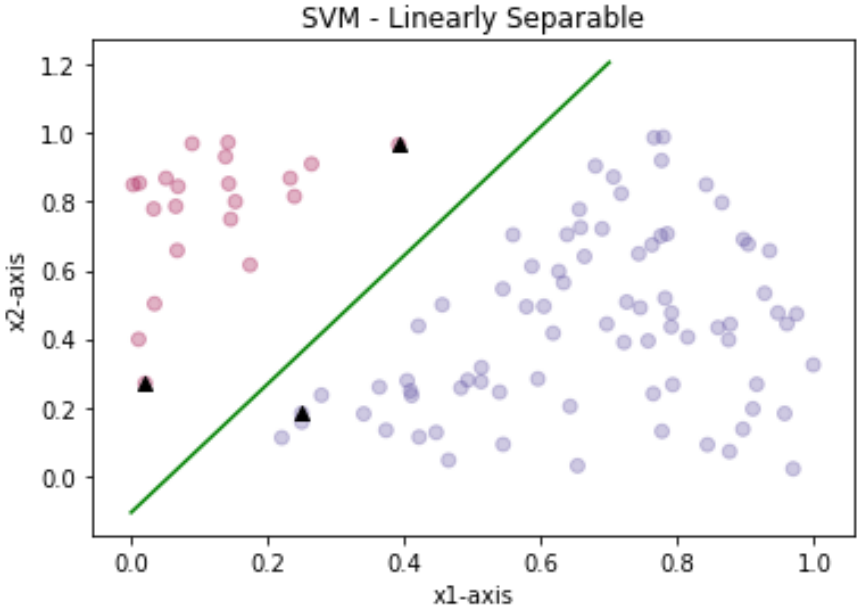
Assignment 6 Report

Part(a): Asked to find the fattest margin line that separates points in the linsep.txt file using a Quadratic Programming solver and to report the equation of the line.

A preliminary exploration of the data confirms the data within linsep.txt is indeed linearly separable.



The Q matrix is built for this Quadratic Programming problem and a QP solver from the cvxopt python library is invoked to find the W and b for the SVM hyperplane. 3 support vectors are identified and marked in black triangles with their coordinates listed below.




```
Support Vector 1:   [0.24979414 0.18230306]
Support Vector 2:   [0.3917889  0.96675591]
Support Vector 3:   [0.02066458 0.27003158]
```

After 8 iterations, the QP solver finds a weight matrix and b:

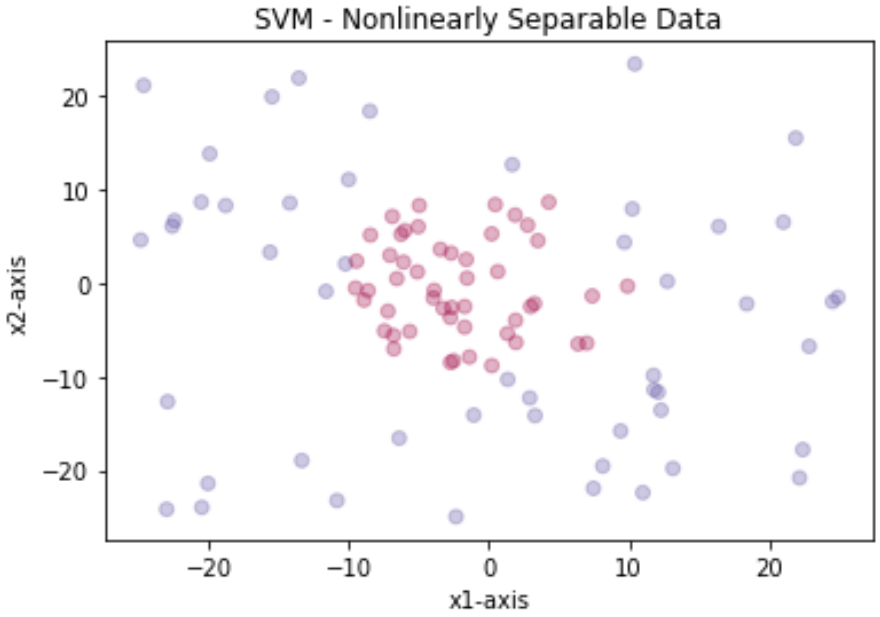
```
W:  [ 7.25005616 -3.86188932]
b:  -0.10698729032462029
```

The equation of the hyperplane is given by

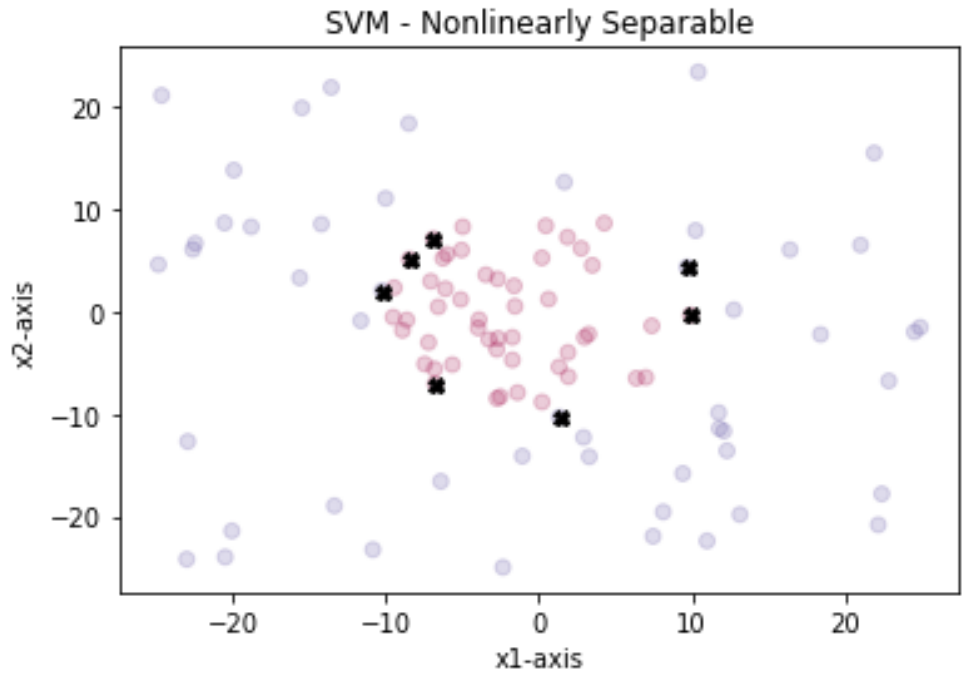
$$b + W^T x = 0$$
$$\Longrightarrow -0.1068 + (7.25) * X_1 + (-3.86) * X_2 = 0$$

Part(b) asks us to use a kernel function and the same QP-solver to find the curve that separates the points in nonlinsept.txt. Then we are asked to report the kernel function used and the support vectors.

Preliminary data exploration shows that the data in nonlinsep.txt is not linearly separable.



Modifying the SVM function, so that the Q-matrix is built using the Kernel function instead of the direct inner product of the x-vectors, the QP-solver is invoked and, after 13 iterations, identifies seven support vectors (shown in black).




```
Support Vector 1:  [-8.47422847  5.15621613]
Support Vector 2:  [-10.260969     2.07391791]
Support Vector 3:  [  1.3393313  -10.29098822]
Support Vector 4:  [9.67917724 4.3759541 ]
Support Vector 5:  [-6.90647562  7.14833849]
Support Vector 6:  [-6.80002274 -7.02384335]
Support Vector 7:  [ 9.90143538 -0.31483149]
```

The **kernel function used for this SVM was the polynomial kernel function that was described in lecture as $\boldsymbol{K(x, x') = (1 + +x^T x')^2}$.**

The equation of the curve is given by the summation:

$$\sum_{n \in SupportVector} \alpha_n y_n K(x_n, x) + b$$

Where:

```
alpha_n:
 [1.25011960e-02 1.49903971e-02 6.64212765e-03 4.66214862e-03
 6.87420856e-10 7.40836604e-03 6.38511049e-03]
y_n:
 [-1.  1.  1.  1. -1. -1. -1.]
b:
 -0.9161376238787018
x_n:
 [[ -8.47422847   5.15621613]
 [-10.260969     2.07391791]
 [  1.3393313  -10.29098822]
 [  9.67917724   4.3759541 ]
 [ -6.90647562   7.14833849]
 [ -6.80002274  -7.02384335]
 [  9.90143538  -0.31483149]]
```

Additional Descriptions:

Mostly numpy arrays, cvxopt matrices were used for this implementation. One of the challenges that I was not able to resolve for this assignment was the graph of the nonlinear SVM curve in the nonlinearly separable case, but the support vectors are still visible and are positioned in a way that is indicative of a nonlinear separating curve that was used due to the polynomial kernel. Another challenging area was mapping the symbolic SVM notation onto the format prescribed by the cvxopt QP-solver documentation. Code-level optimizations-wise I found that the Quadratic Programming solver gave many alpha values that were effectively zero but not exactly zero. So a threshold was defined for the SVMs such that a reasonable number of support vectors could be identified. For the linearly separable case, I used a threshold value of 1e-3, below which, the alphas were regarded as zero. In the nonlinearly separable case, a threshold value of 1e-10 was used.