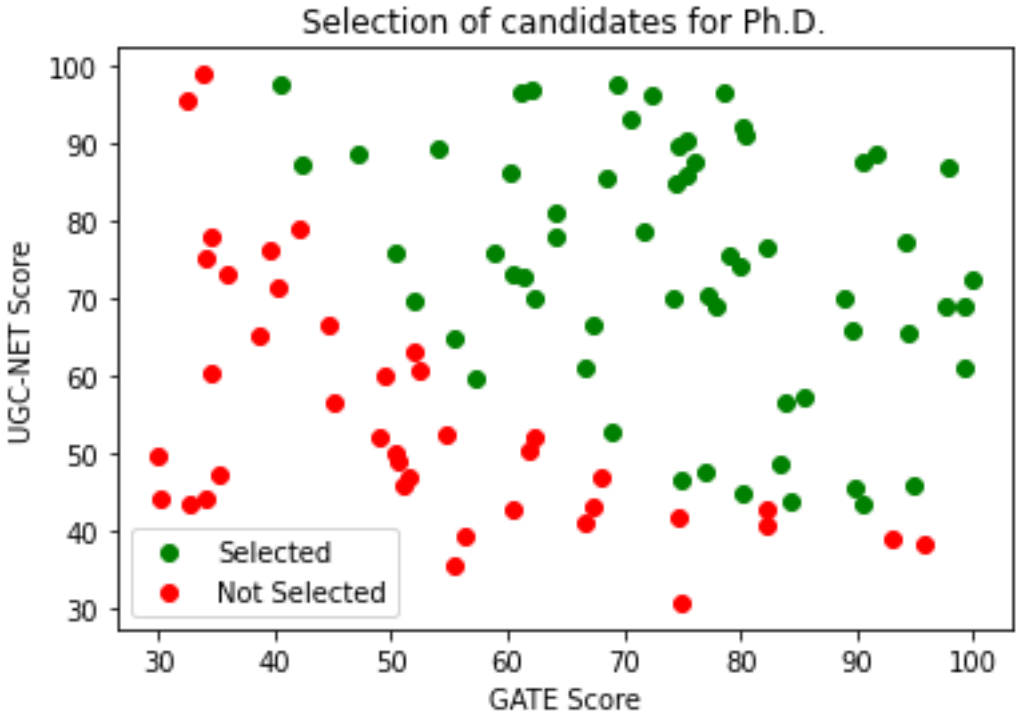
Name: Shahnawaz Khan
Entry Number: 2020CSB1123

Machine Learning(CS503)
Assignment 2

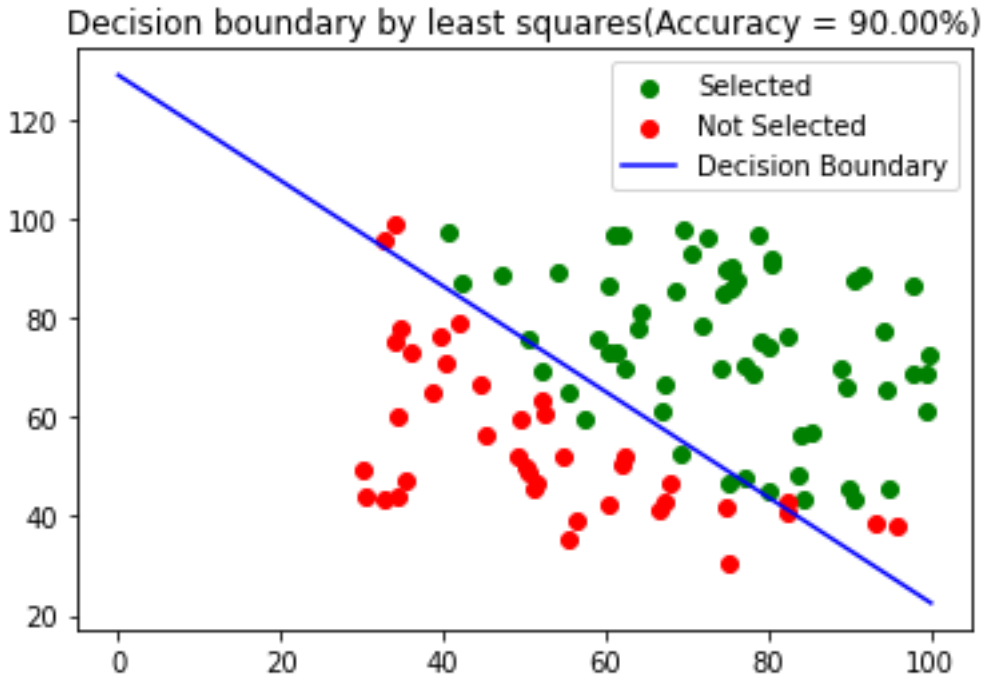
---

1. Task 1

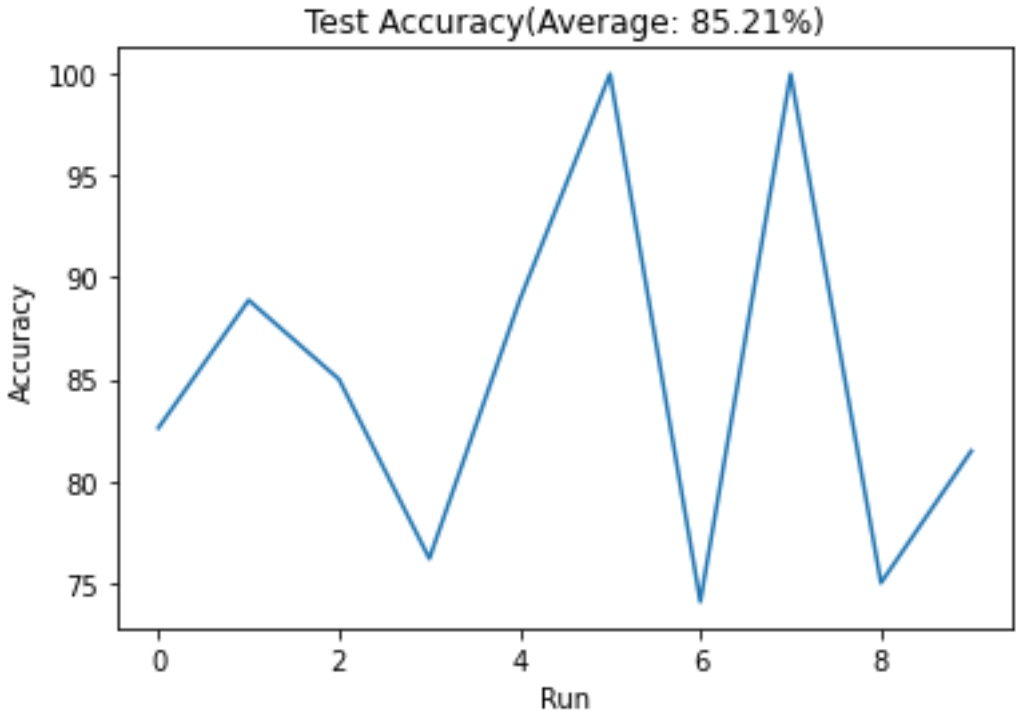
   (a) The dataset given is plotted as follows:

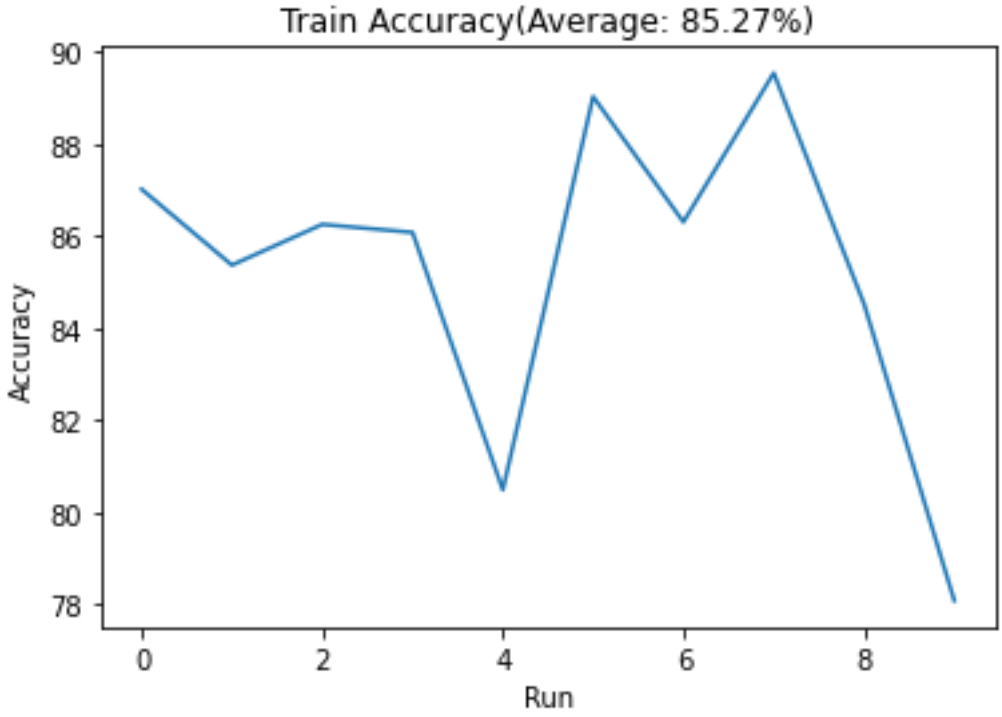


   (b) On applying least squares on the classification dataset, we get the following plot. The decision boundary cannot classify points correctly because it also takes into account the distance of points from the boundary.

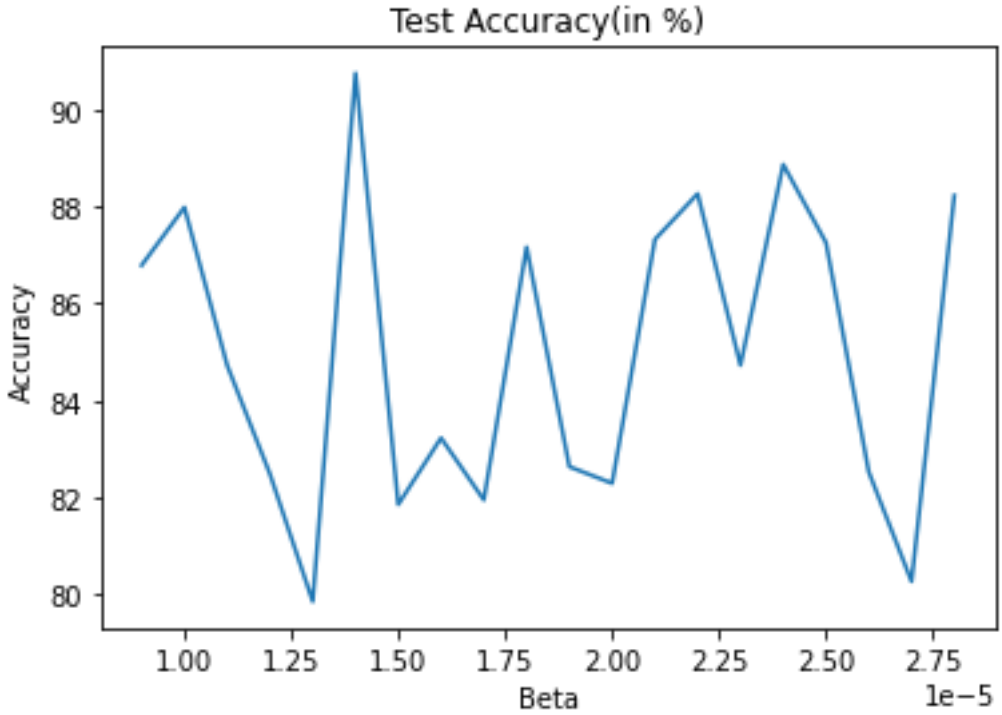


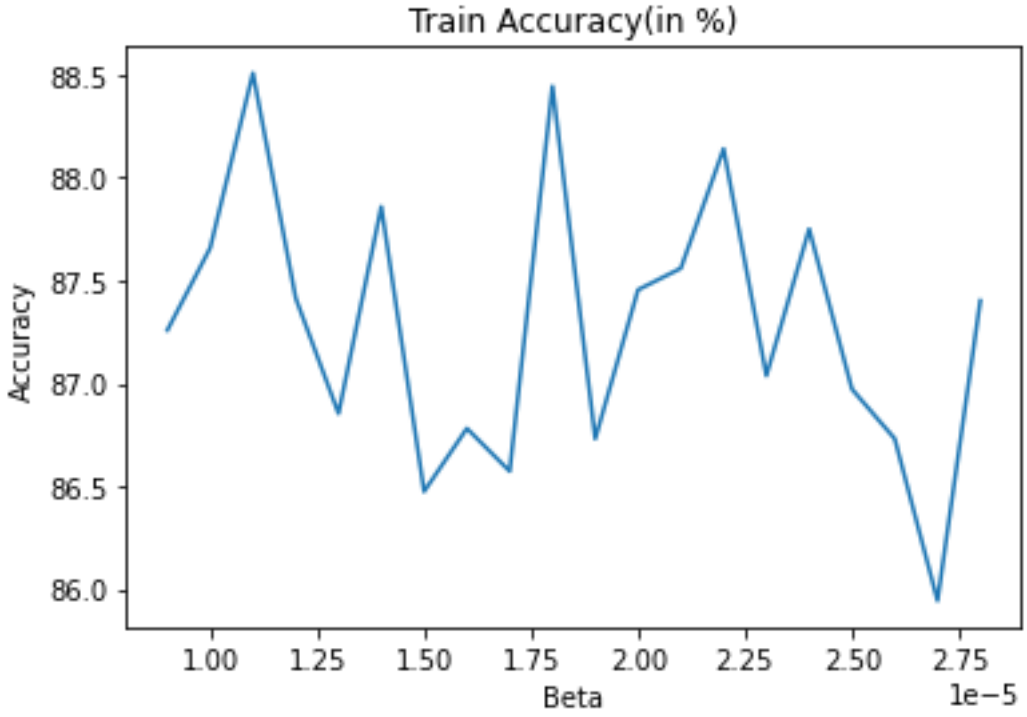
(c) On taking beta as 0.00001 and running gradient descend at 100000 iterations, we get the following test and train errors:



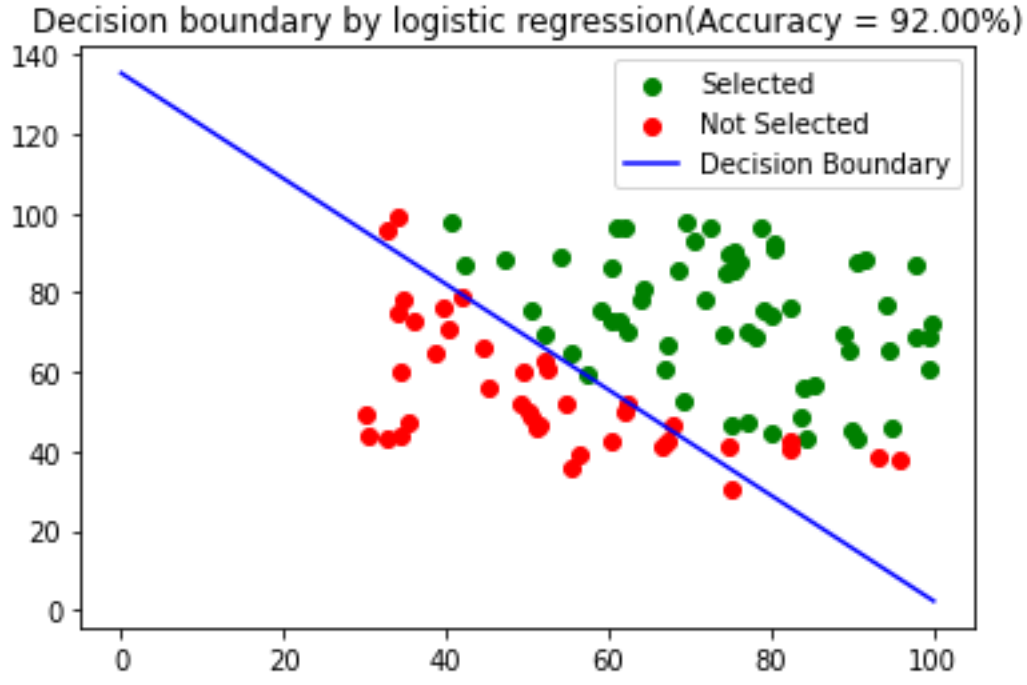


Since the splits are random, we get some random order of accuracy at subsequent runs. The model gives an accuracy of around 85% at the learning rate and iterations mentioned above. On different values of beta, we get the following curve. The curve is not very sensitive to the learning rate beta.



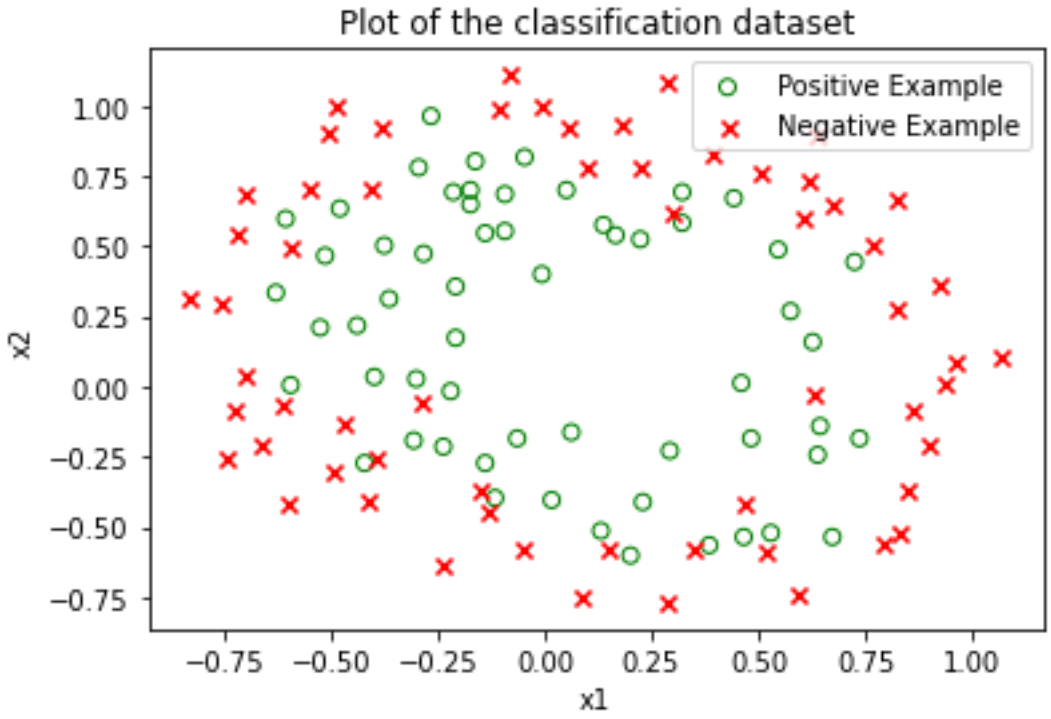


On implementing logistic regression on the dataset, it fits more compared to the previous approach and gives a better accuracy on the training dataset.

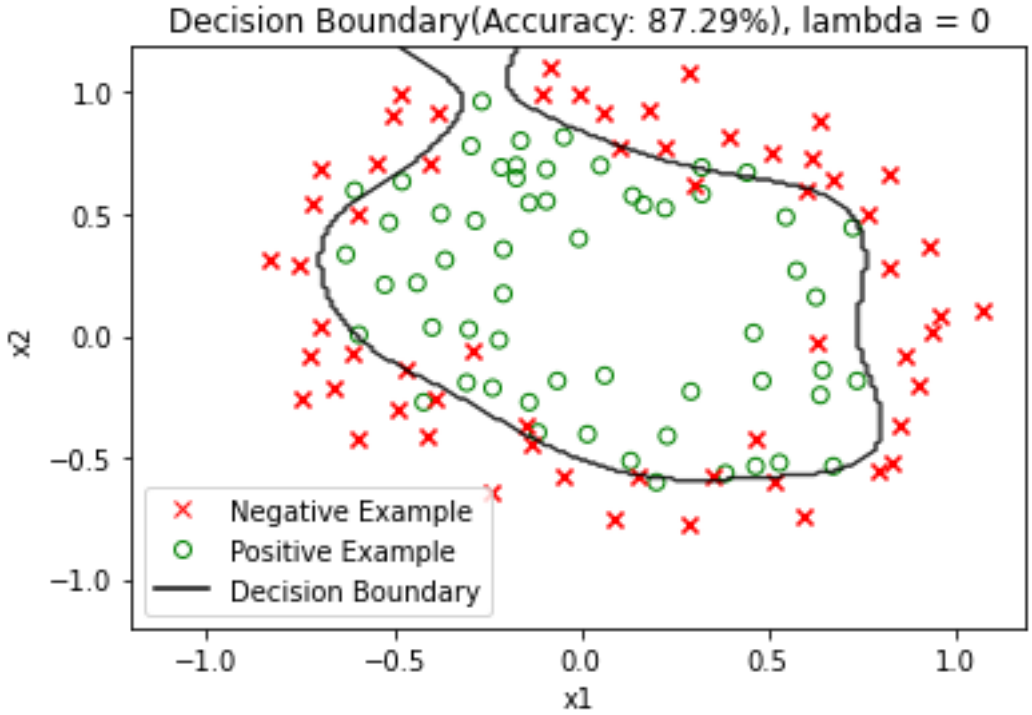


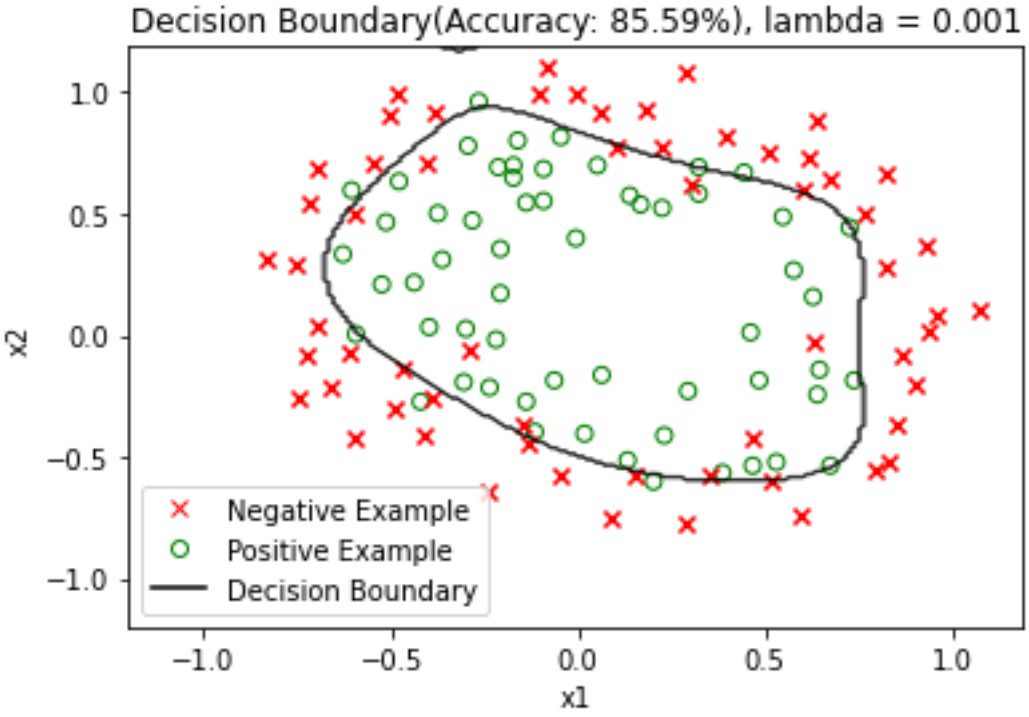
2. Task 2

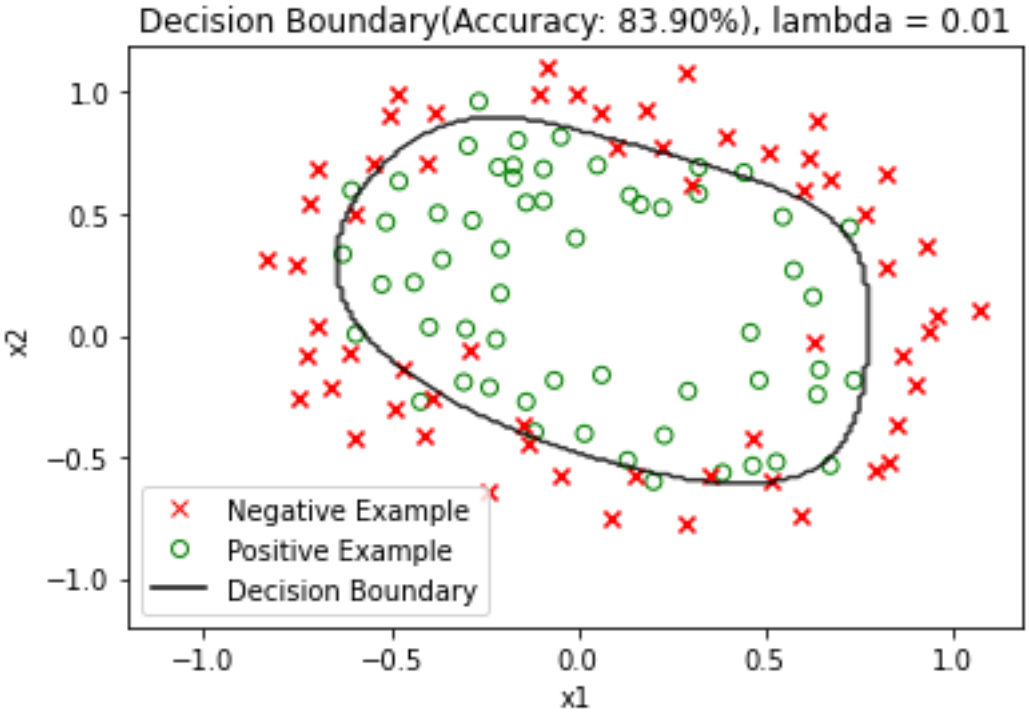
   (a) The dataset is plotted as follows:

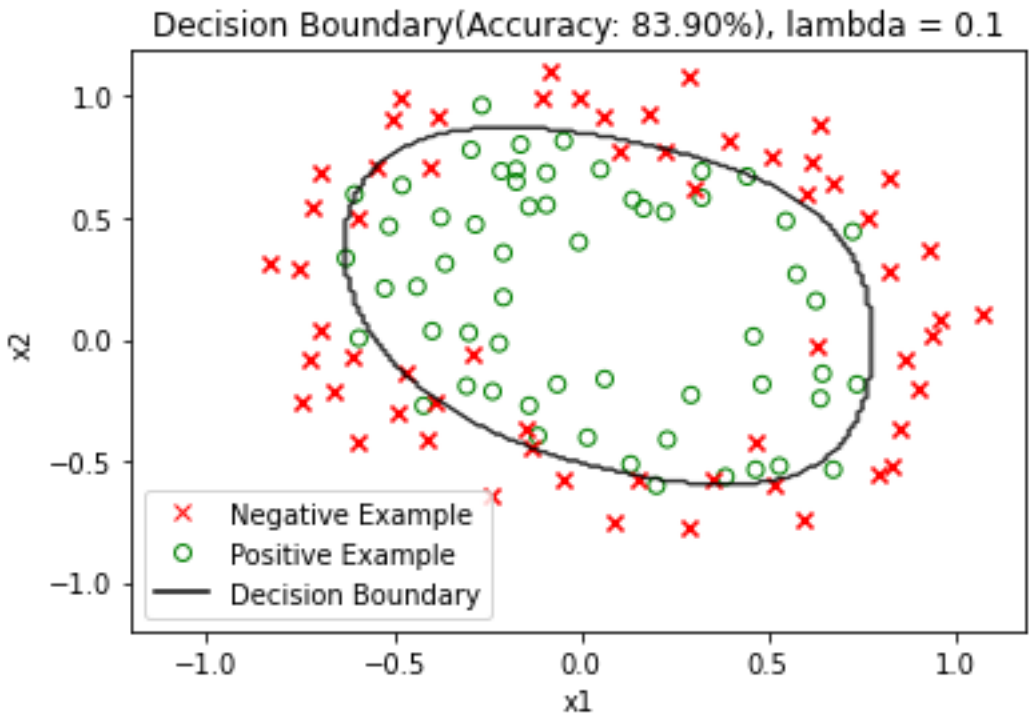


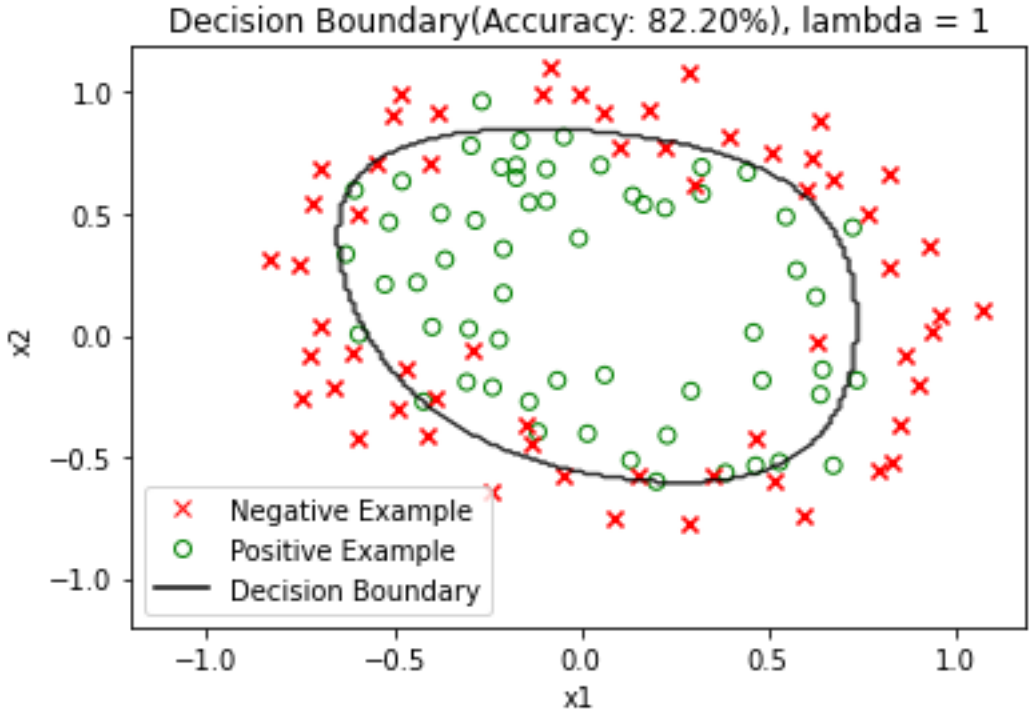
(b) On different values of λ, we can see that the model is over-fitting at lower values of λ. However, as we increase λ, the training accuracy decreases, and it shows that the over-fitting problem is avoided by increasing the values of λ.

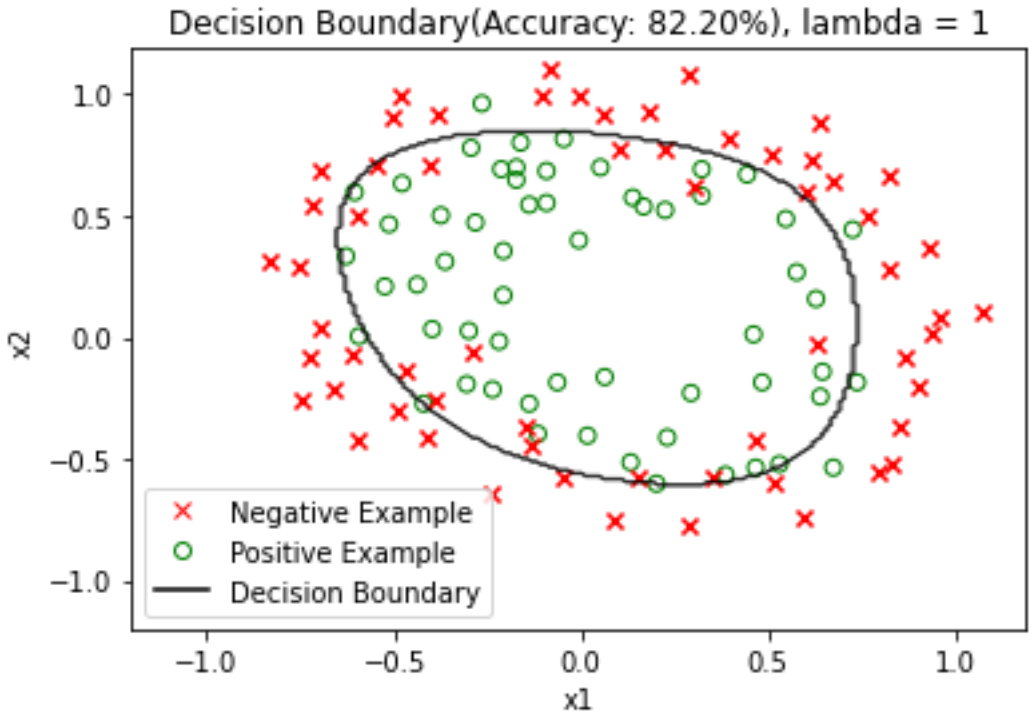



Decision Boundary(Accuracy: 85.59%), lambda = 0.001
x2
x1
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
Negative Example
Positive Example
Decision Boundary


Decision Boundary(Accuracy: 83.90%), lambda = 0.01
x2
x1
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
Negative Example
Positive Example
Decision Boundary

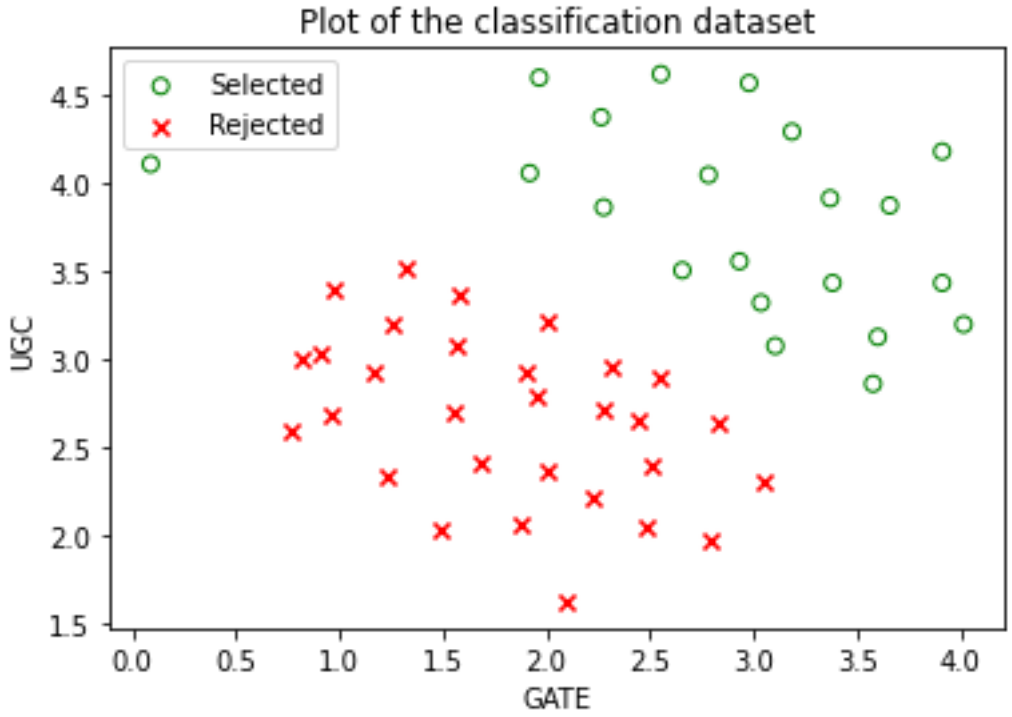

Decision Boundary(Accuracy: 83.90%), lambda = 0.1
Negative Example
Positive Example
Decision Boundary
x1
x2
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0


Decision Boundary(Accuracy: 82.20%), lambda = 1
Negative Example
Positive Example
Decision Boundary
x1
x2
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0

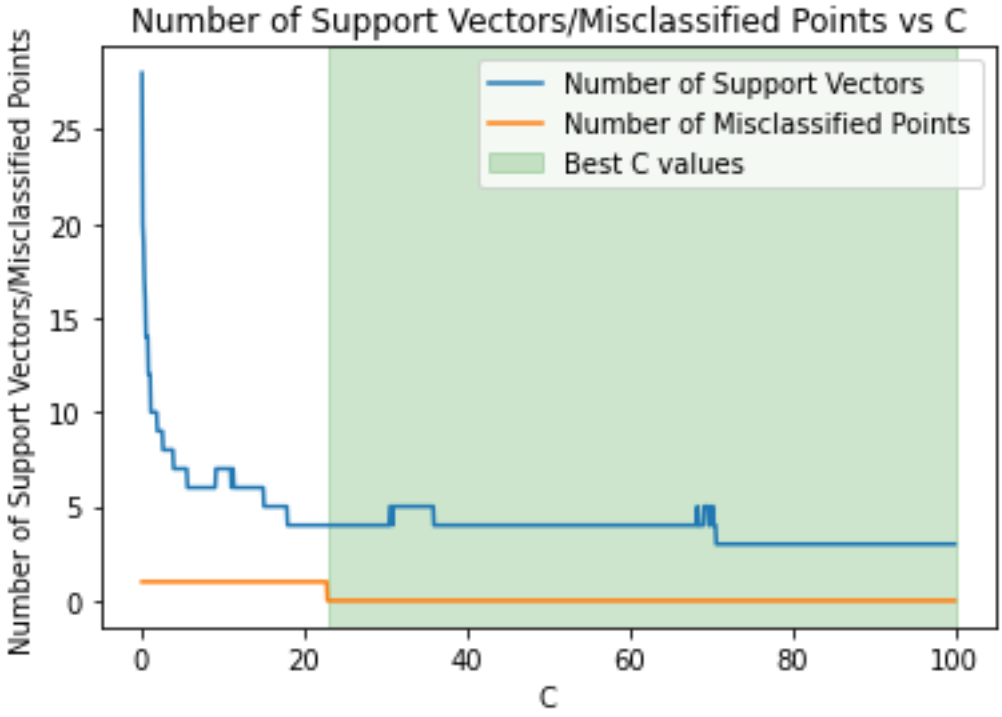


3. Task 2

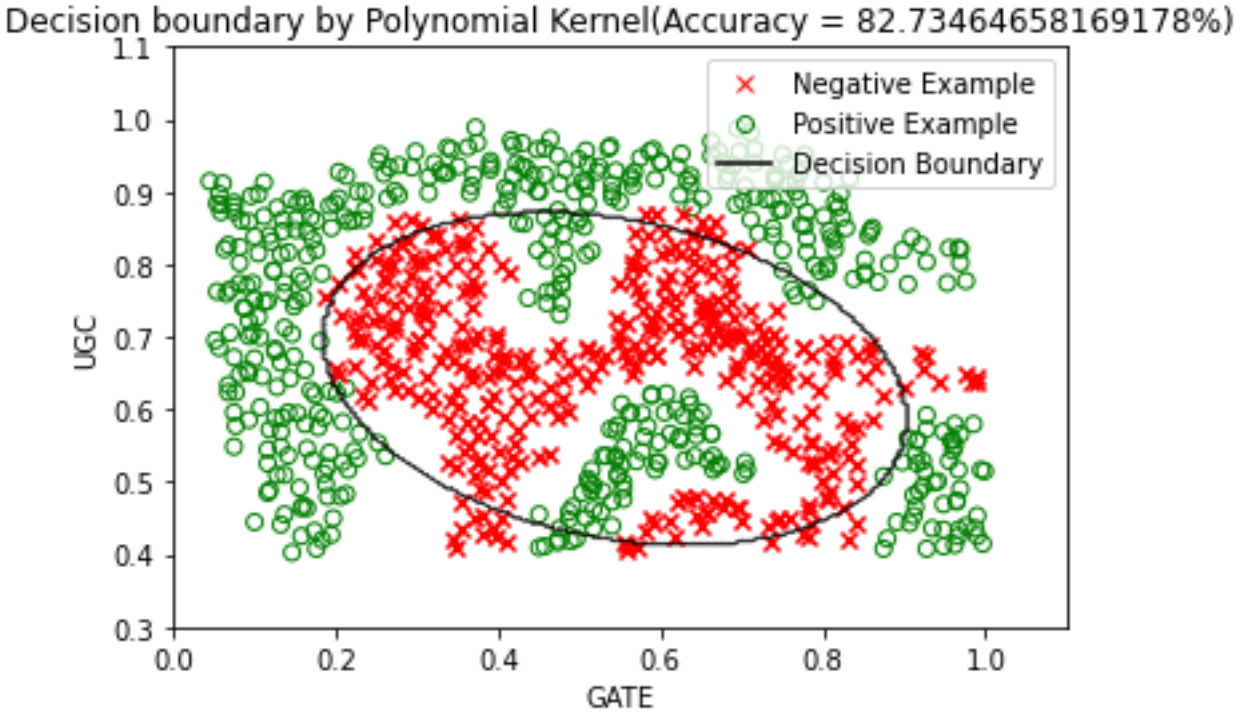
(a) The data-set is plotted below

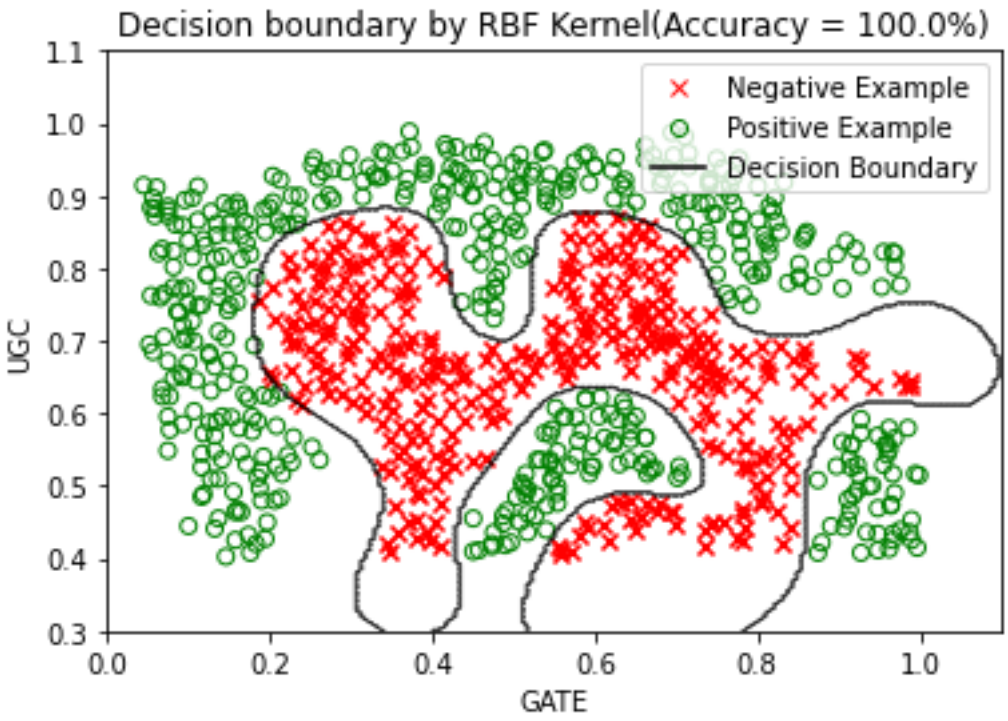


(b) As we increase the value of C, the number of misclassified and support vector points decreases. This is because the soft SVM approaches Hard SVM at higher values of C. On this run, we get $C >= 22$ as the best value for the SVM.



(c) We get the following decision boundary for the classification data set given.

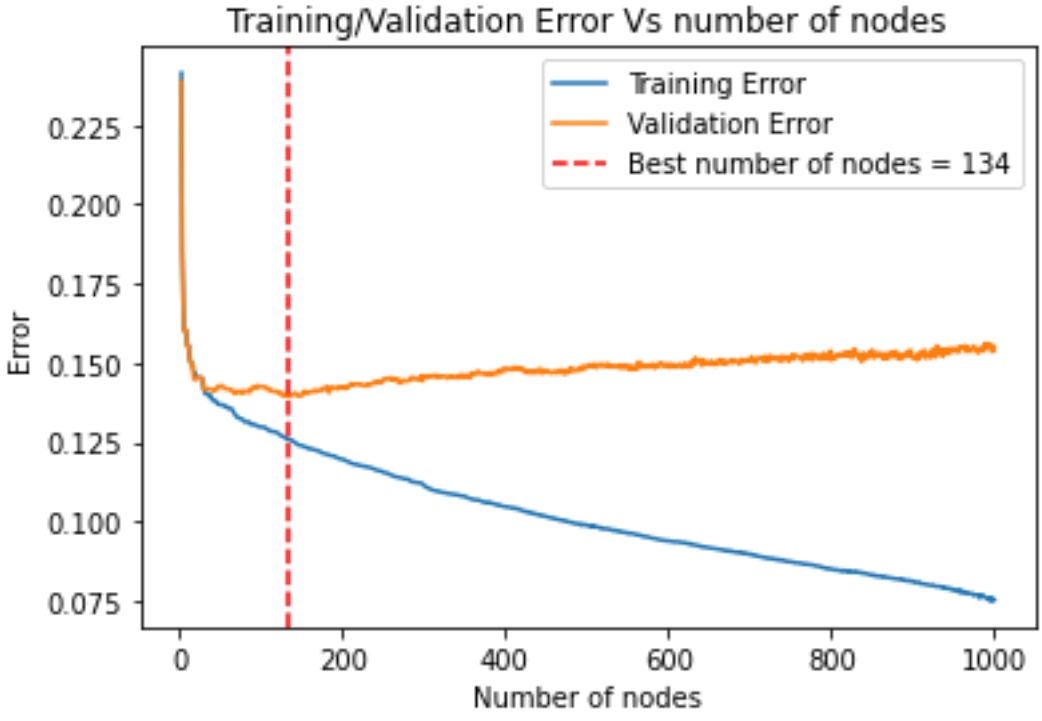




As we can see, the RBF kernel leads to over-fitting because it is giving 100% accuracy on the training set. On the other hand, the polynomial kernel is not giving good training accuracy, but it may perform better at test data compared to RBF kernel. Because of its kernel function, the polynomial kernel gives some elliptical shape decision boundary.

4. Task 4

   (a) On an increasing number of decision tree nodes, the training error is decreasing, but the test error is increasing. This is because the decision tree is over-fitting at a higher number of nodes.



(b) By feature and instance bagging, we get the error at around 14.75%. This error has slightly increased but may be due to the dataset type and random seed value. In general, we use this technique to avoid over-fitting get better test error.

(c) After applying PCA, we select features that explain more than 90% of the variance. By this method, we get the optimal number of features to be 12(which may change based on our tolerance value). Projecting this data into a subspace spanned by 12 eigenvectors corresponding to the 12 highest eigenvalues, we get the error on the validation set to be around 17.04%. This error has increased because we have mapped the data into a lower dimension.