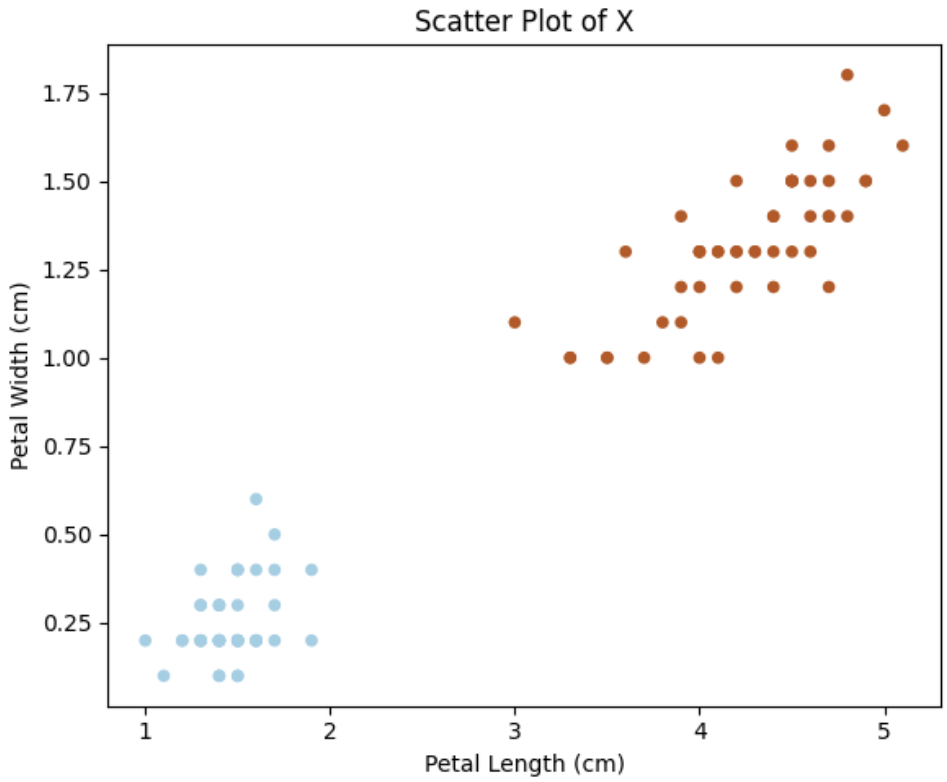
2 Support Vector Machines (SVM)

2.1 Task-1(a)

As per categorical encoding for target y :

- 0 - setosa
- 1 - versicolor
- 2 - virginica



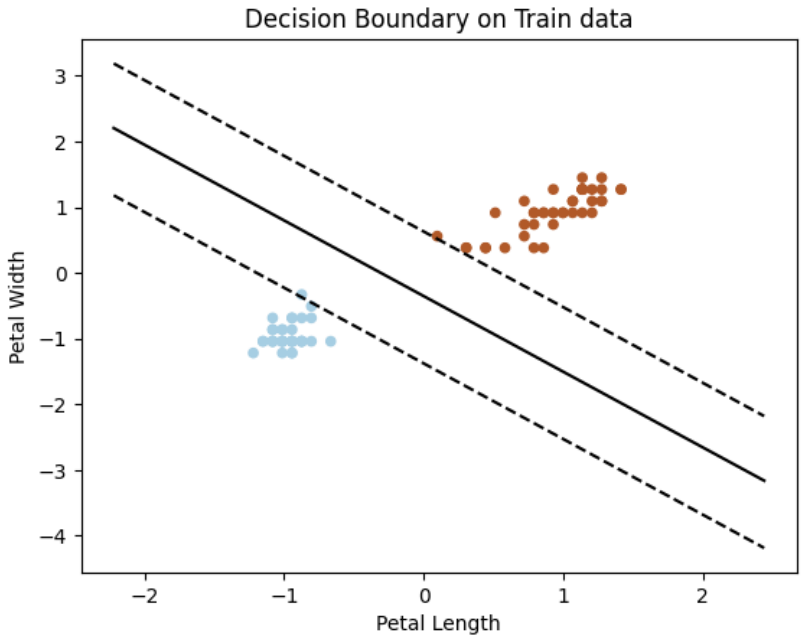
1. Although there are 100 data points in X, I don't see that many in the scatter plot
2. After checking there were 42 duplicates

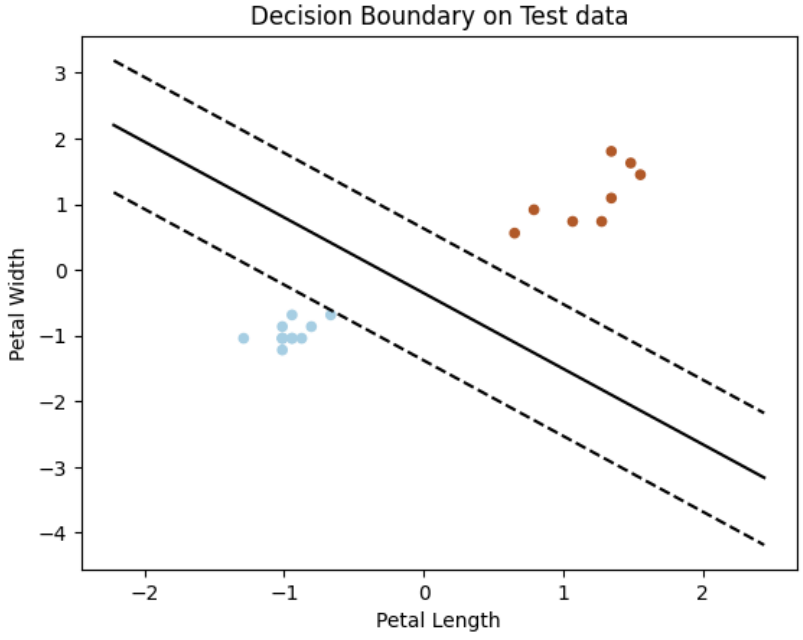
2.2 Task-1(b)

- The seperating Hyperplane has an equation : ax1 + bx2 + c = 0
- a, b are given coefficients given by clf.coef_
- c is the intercept given by clf.intercept_

Support Vectors :

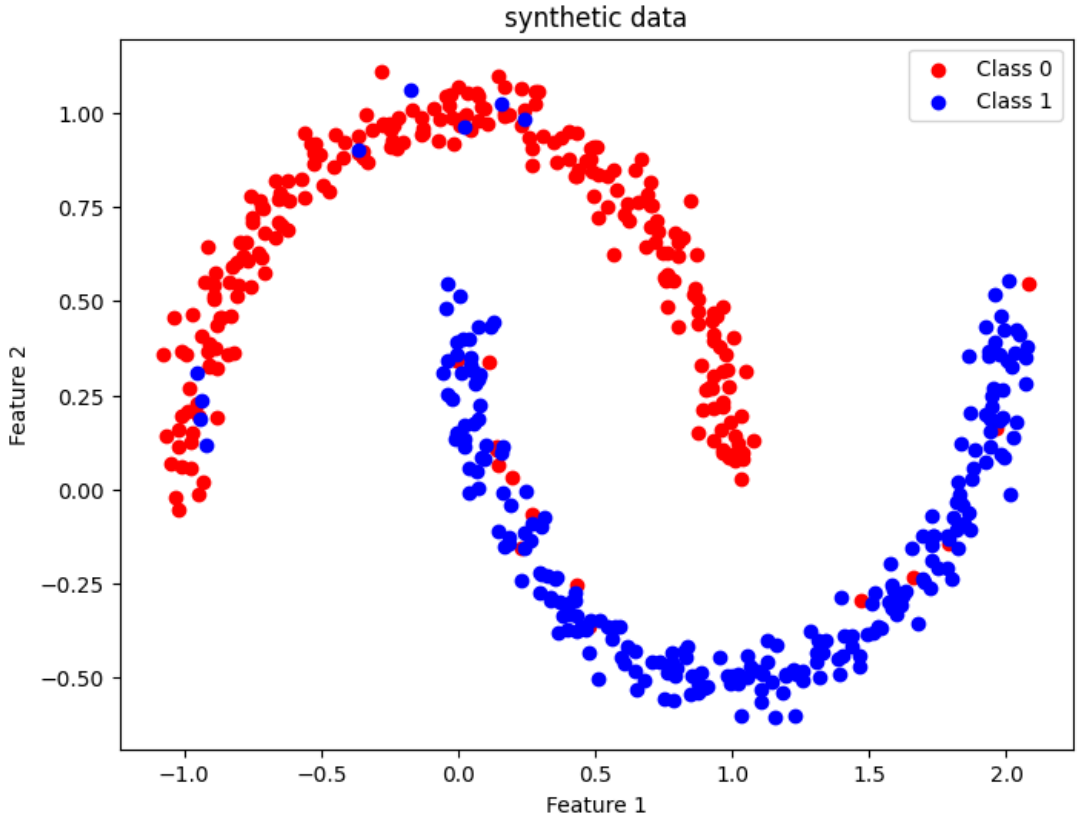
1. To include the support vectors of each class I am going to use the help of decision_function which outputs the distance od each input/training data along with sign corresponding to which class it belongs
2. Then I will sort them and get the nearest points
3. Now that the support vectors are parallel to hyperplane and distance between two parallel lines is mod(c1 - c1) just add the distance to original hyperplane
4. I have included the hyperplane using - and the support vectors with –


Decision Boundary on Train data
Petal Width
Petal Length
3
2
1
0
−1
−2
−3
−4
−2
−1
0
1
2


Decision Boundary on Test data
Petal Width
Petal Length
3
2
1
0
−1
−2
−3
−4
−2
−1
0
1
2

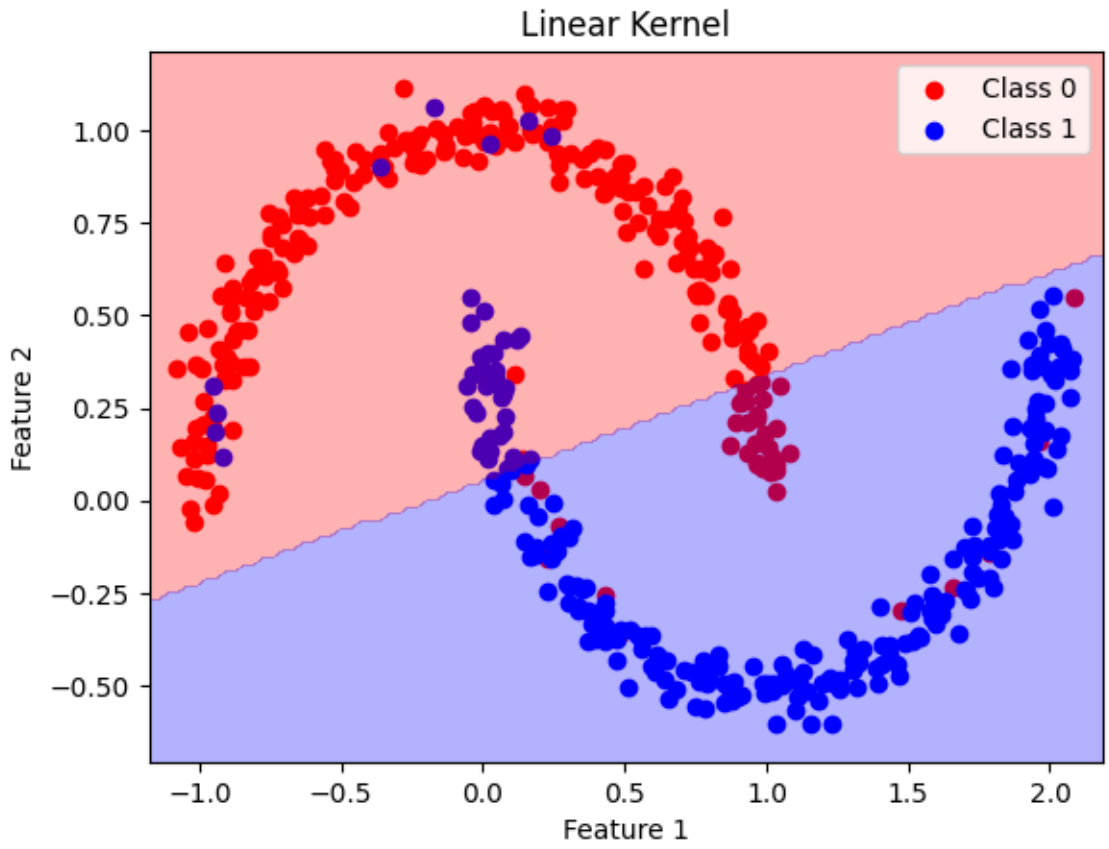
2.3 Task-2 (a)

For noise(misclassfication) I have randomly selected indices using np.random.choice and then I flipped the class label by replacing it with 1 - label



2.4 Task-2 (b)

2.4.1 Linear Kernel



1. With a linear kernel, the decision boundary is a straight line
2. May not perform well if the data is not linearly separable

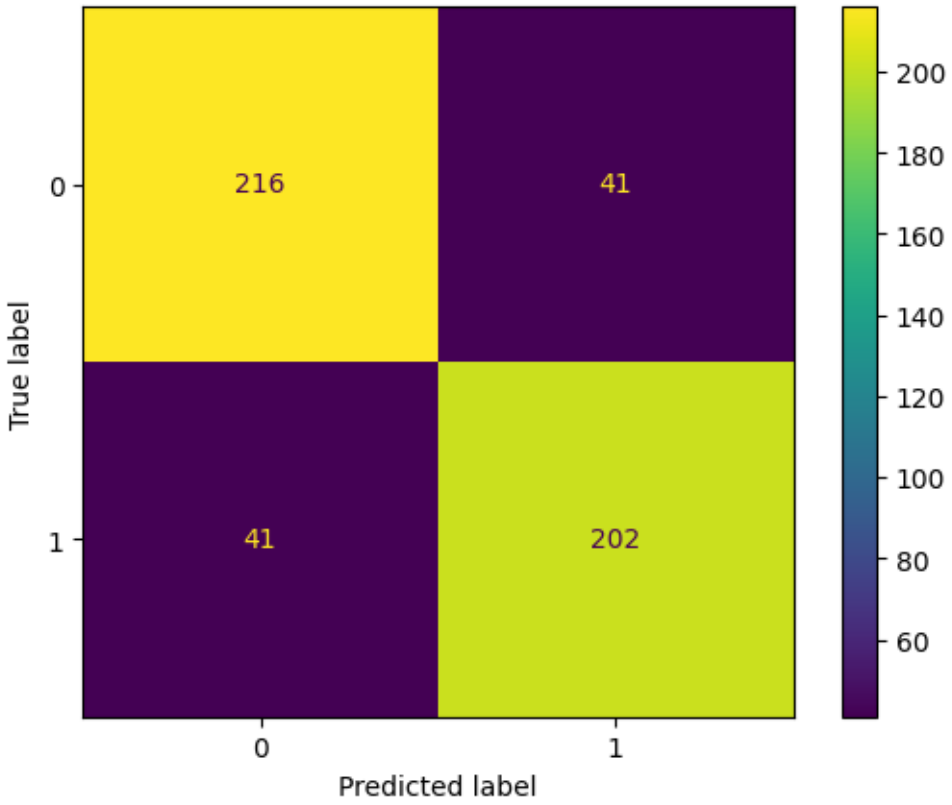
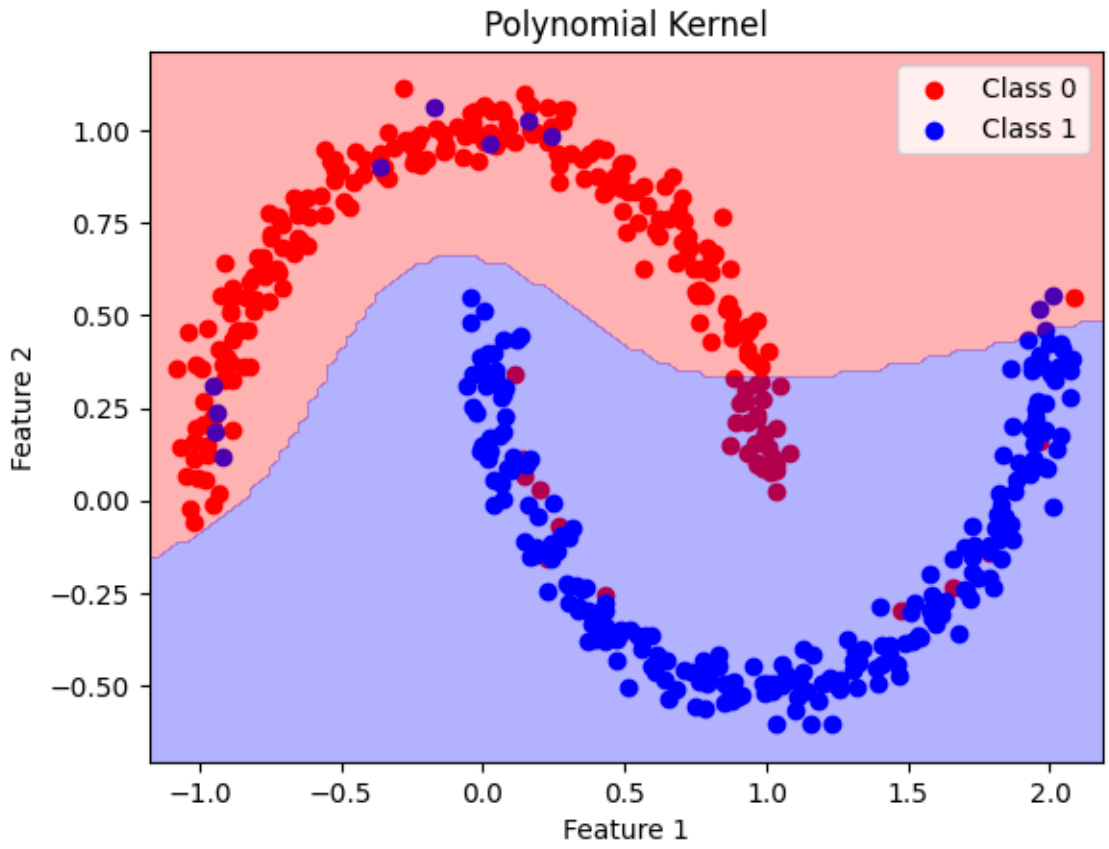


Figure 19: Confusion Matrix - linear

	precision	recall	f1-score	support
0	0.84	0.84	0.84	257
1	0.83	0.83	0.83	243

Figure 20: Classification Report - linear

2.4.2 Polynomial Kernel



1. With a polynomial kernel, the decision boundary can be nonlinear
2. May perform well for moderately complex data that cannot be separated linearly

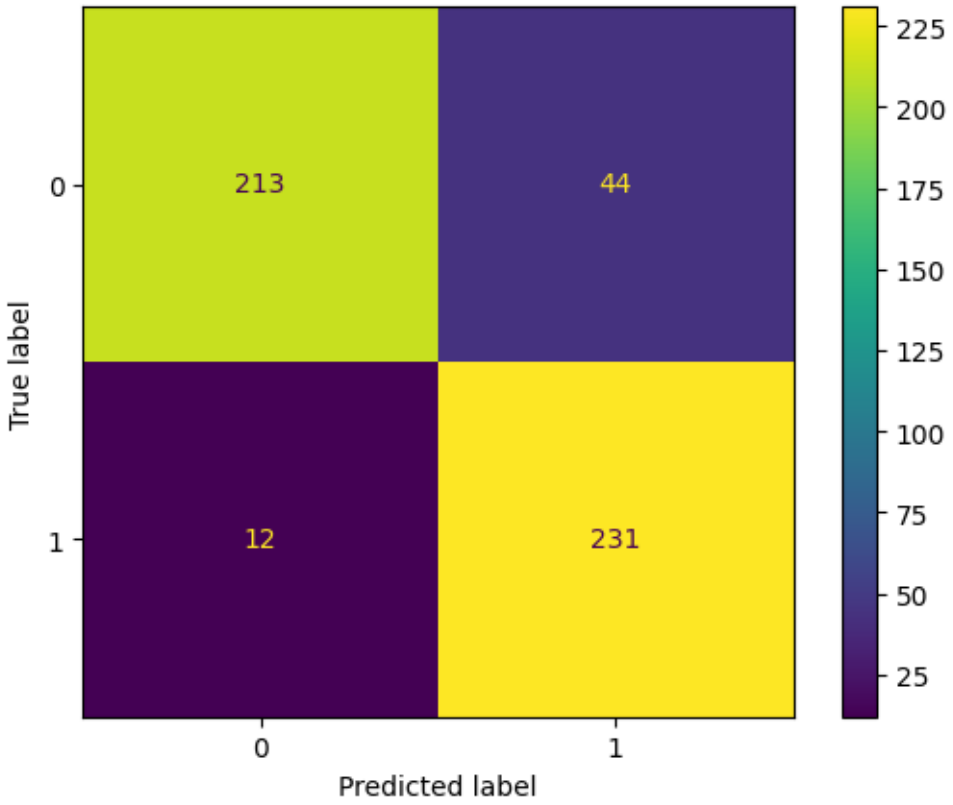
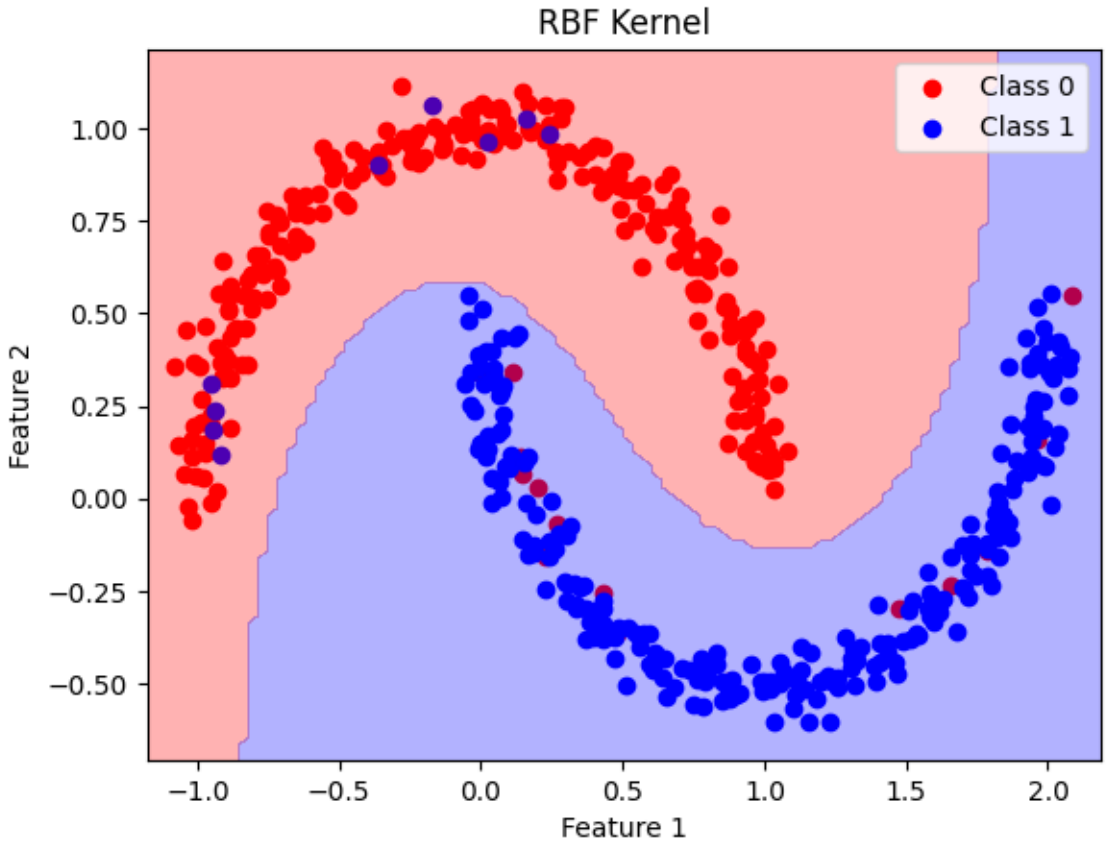


Figure 21: Confusion Matrix - Polynomial

	precision	recall	f1-score	support
0	0.95	0.83	0.88	257
1	0.84	0.95	0.89	243

Figure 22: Classification Report - Polynomial

2.4.3 RBF Kernel



1. With an RBF kernel, the decision boundary is non-linear and can be highly flexible

2. May perform well for highly non-linear data but can be sensitive to the choice of gamma parameter

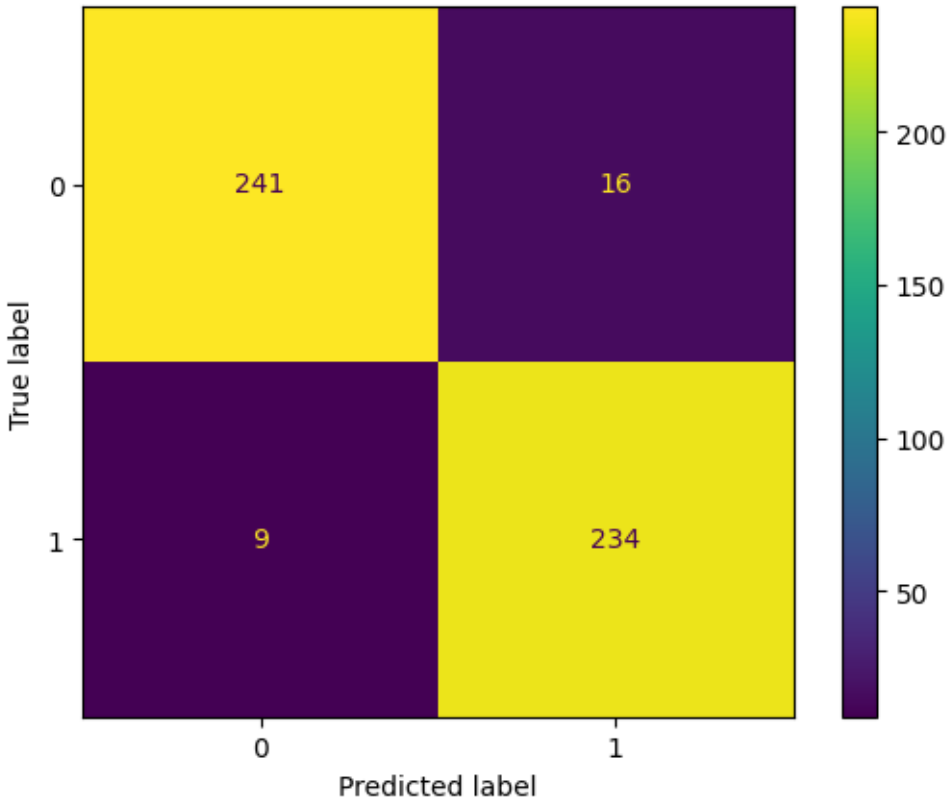


Figure 23: Confusion Matrix - RBF

	precision	recall	f1-score	support
0	0.96	0.94	0.95	257
1	0.94	0.96	0.95	243

Figure 24: Classification Report - RBF

2.5 Task-2 (c)

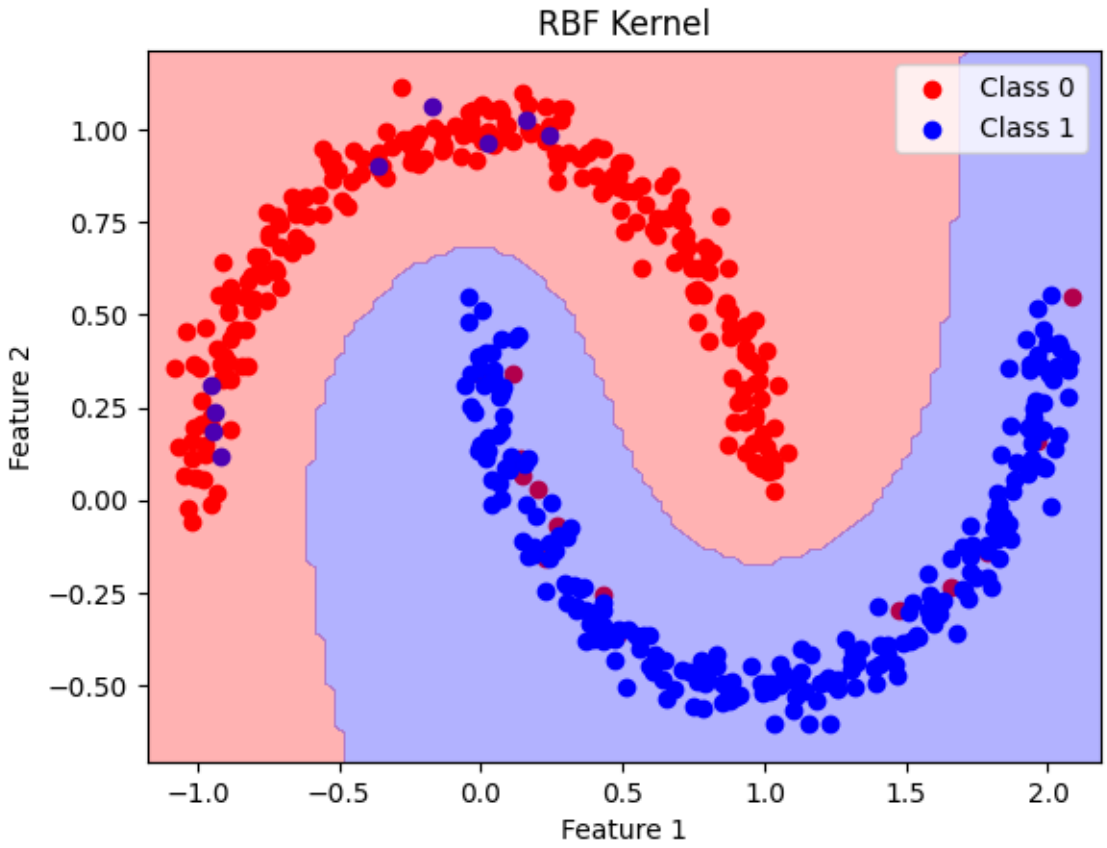
1. I have used grid search for the case of hyper parameter tuning of gamma and c in RBF kernel. parameter grid is as follows
   - C : [0.1, 1, 10, 100]
   - gamma : [0.1, 0.5, 1, 5, 10]
2. cv(cross validation) = 5 meaning data is divided into 5 equal parts, and the model is trained and evaluated 5 times, each time using a different combination of 4 folds for training and 1 fold for validation.
3. Finally scoring='accuracy' meaning tuning is compared via accuracy

```
All accuracies:
Mean accuracy: 0.808, Std: 0.057411, Parameters: {'C': 0.1, 'gamma': 0.1}
Mean accuracy: 0.882, Std: 0.047074, Parameters: {'C': 0.1, 'gamma': 0.5}
Mean accuracy: 0.938, Std: 0.027857, Parameters: {'C': 0.1, 'gamma': 1}
Mean accuracy: 0.950, Std: 0.022804, Parameters: {'C': 0.1, 'gamma': 5}
Mean accuracy: 0.950, Std: 0.022804, Parameters: {'C': 0.1, 'gamma': 10}
Mean accuracy: 0.834, Std: 0.054626, Parameters: {'C': 1, 'gamma': 0.1}
Mean accuracy: 0.950, Std: 0.022804, Parameters: {'C': 1, 'gamma': 0.5}
Mean accuracy: 0.950, Std: 0.022804, Parameters: {'C': 1, 'gamma': 1}
Mean accuracy: 0.950, Std: 0.022804, Parameters: {'C': 1, 'gamma': 5}
Mean accuracy: 0.950, Std: 0.022804, Parameters: {'C': 1, 'gamma': 10}
Mean accuracy: 0.914, Std: 0.040299, Parameters: {'C': 10, 'gamma': 0.1}
Mean accuracy: 0.950, Std: 0.022804, Parameters: {'C': 10, 'gamma': 0.5}
Mean accuracy: 0.950, Std: 0.022804, Parameters: {'C': 10, 'gamma': 1}
Mean accuracy: 0.950, Std: 0.022804, Parameters: {'C': 10, 'gamma': 5}
Mean accuracy: 0.950, Std: 0.022804, Parameters: {'C': 10, 'gamma': 10}
Mean accuracy: 0.950, Std: 0.022804, Parameters: {'C': 100, 'gamma': 0.1}
Mean accuracy: 0.950, Std: 0.022804, Parameters: {'C': 100, 'gamma': 0.5}
Mean accuracy: 0.950, Std: 0.022804, Parameters: {'C': 100, 'gamma': 1}
Mean accuracy: 0.950, Std: 0.022804, Parameters: {'C': 100, 'gamma': 5}
Mean accuracy: 0.942, Std: 0.017205, Parameters: {'C': 100, 'gamma': 10}
```

Figure 25: Hyper parameter tuning results

Best hyperparameters found are C = 0.1, gamma = 5

2.6 Task-2 (d)



*** Although the classification is done good, the accuracy will not be 100 percent because of the induced 5 percent noise in the data**

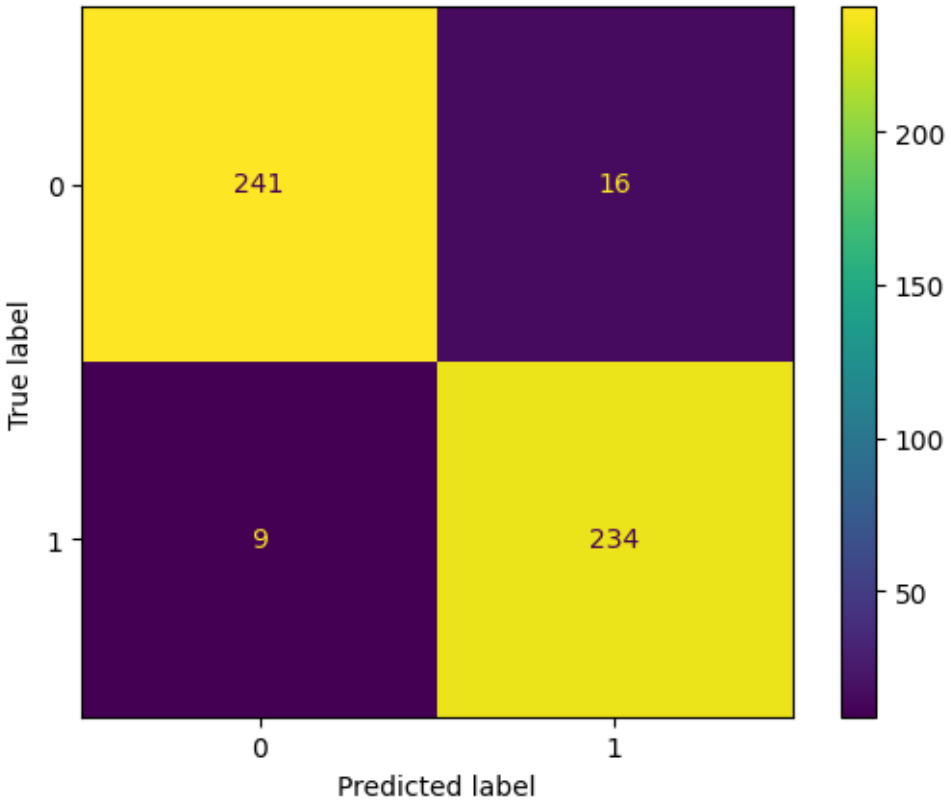


Figure 26: Confusion Matrix

	precision	recall	f1-score	support
0	0.96	0.94	0.95	257
1	0.94	0.96	0.95	243

Figure 27: Classification Report