COL341 Assignment-2 Report

Amaiya Singhal (2021CS50598)

2. Kernels

The following kernels were implemented in the `kernel.py` file.

- **Linear**
$$K(x, y) = x^T y$$
- **Polynomial**
$$K(x, y) = (x^T y + 1)^d$$
- **RBF**
$$K(x, y) = e^{-\gamma(||x-y||^2)}$$
- **Sigmoid**
$$K(x, y) = tanh(\gamma x^T y + r)$$
- **Laplacian**
$$K(x, y) = e^{-\gamma(||x-y||_1)}$$

3. Binary SVM

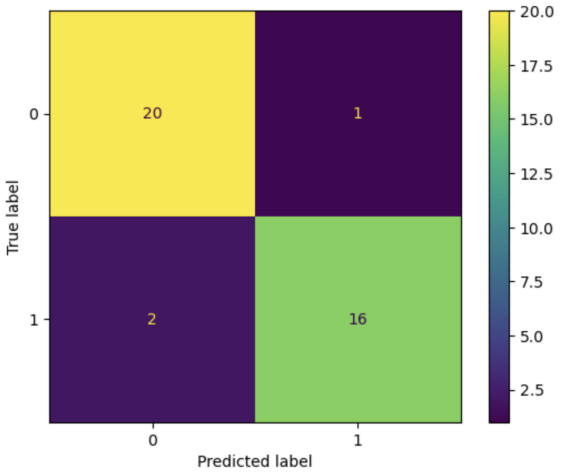
Overview

- In this part, the task was to create a soft margin support vector machine from scratch.
- I used the dual form which was derived as part of the Homework 2 to obtain the minimisation equation.
- The library `qpsolvers` was used for quadratic programming and the solver that I used was `"ecos"`.
- The provided `.csv` files contained y as 0 or 1 but the 0s had to be converted to -1 as the equation is derived using the possible values of y as +1 and -1.
- Upon observing the values of each α, I observed that the appropriate threshold for classifying the support vectors should be $\alpha > 10^{-4}$ because the largest values were around 10^{-3} and smaller values were of the order of 10^{-13} which could be approximated as 0.
- Linear and RBF kernels were used and the parameters C and γ (for RBF) were varied and observations were made.
- The obtained support vectors from training on the `train` set were used to obtain predictions on the `validation` set and the accuracy was calculated. The class `Trainer` also had to be modified to store attributes such as the value of y and α for the support vectors which were required to make the predictions.

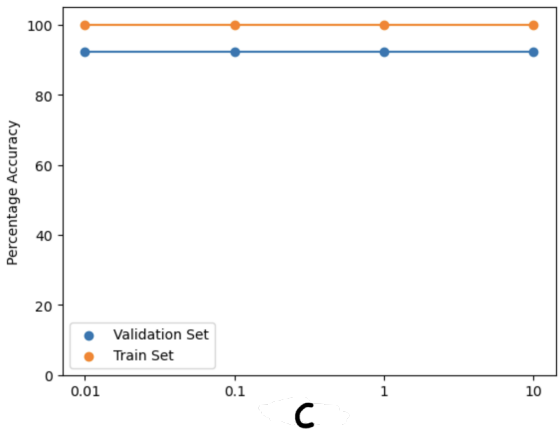
Linear Kernel

C	Accuracy on Train set	Accuracy on Validation set
0.01	100%	92.3076923076923%
0.1	100%	92.3076923076923%
1	100%	92.3076923076923%
10	100%	92.3076923076923%

Confusion Matrix on Validation Set



Same confusion matrix for all the 4 values of C



Variation with C

Observation

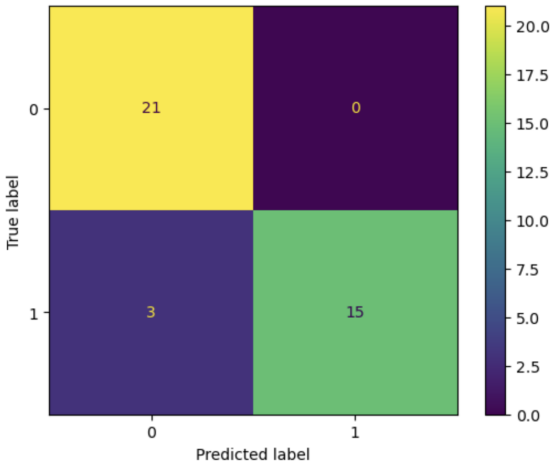
- We observe that the accuracy is very good $\approx 92.3\%$ is the same for all the 4 values of the parameter C.
- This can be explained by the fact that C is just an upper bound for the αs. Since even for the support vectors, the αs are coming out to be of the order 10^{-3}, increasing C from 0.01 to even 10 would not affect their values because they satisfy the upper bound in each of these cases.
- However on experimenting with different values of C, I observed that upon decreasing C to 0.001, we obtain a small increase in the accuracy to roughly about 94% because now the earlier values of α that were above 10^{-3} are no longer valid under this constraint.

RBF Kernel

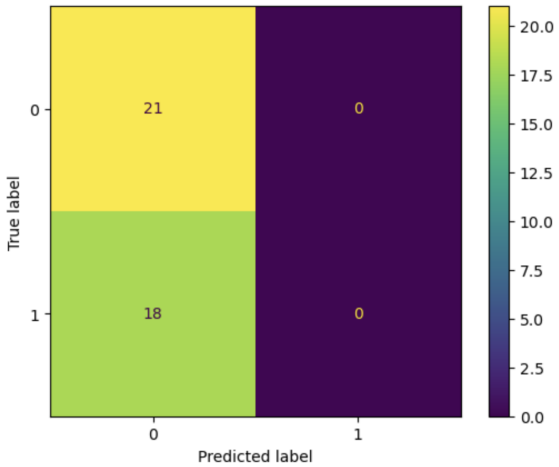
C = 0.01

γ	Accuracy on Train set	Accuracy on Validation set
0.001	92.50936329588015%	92.3076923076923%
0.01	61.4232209737827 7%	53.84615384615385%
0.1	61.4232209737827 7%	53.84615384615385%

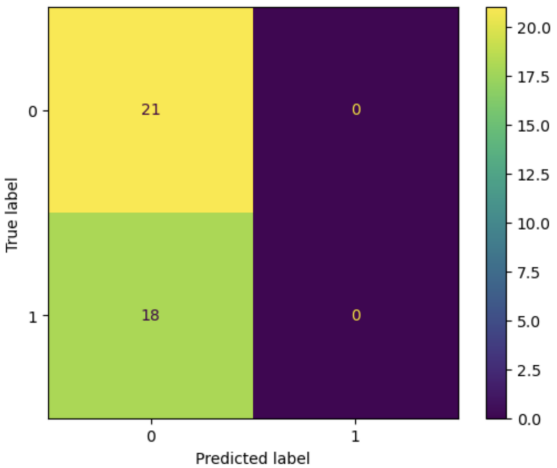
Confusion Matrix on Validation Set



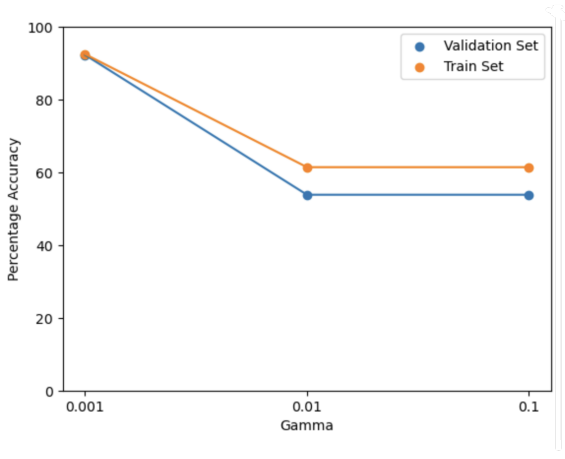
C = 0.01, γ = 0.001



C = 0.01, γ = 0.01



C = 0.01, γ = 0.001



Variation with γ, (C = 0.01)

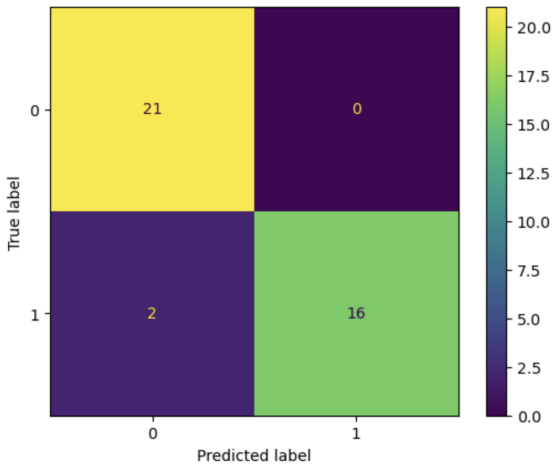
Observations

- The best results are obtained for $\gamma = 0.001$
- As γ increases, the accuracy decreases.

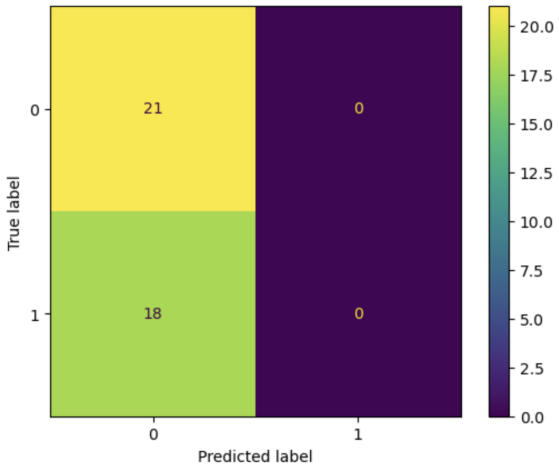
C = 0.1

γ	Accuracy on Train set	Accuracy on Validation set
0.001	93.2584269629213%	94.87179487179486%
0.01	67.04119850187266%	53.84615384615385%
0.1	61.42322097378277%	53.84615384615385%

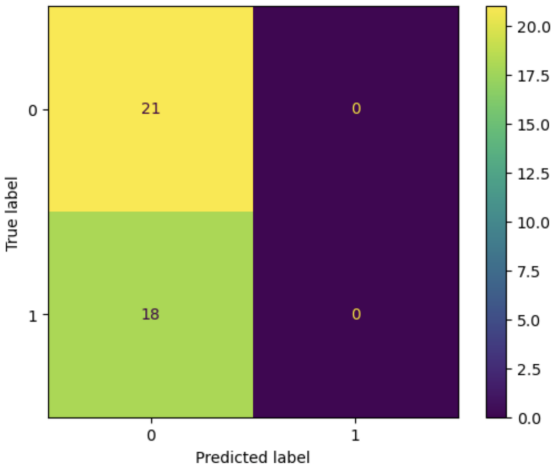
Confusion Matrix on Validation Set



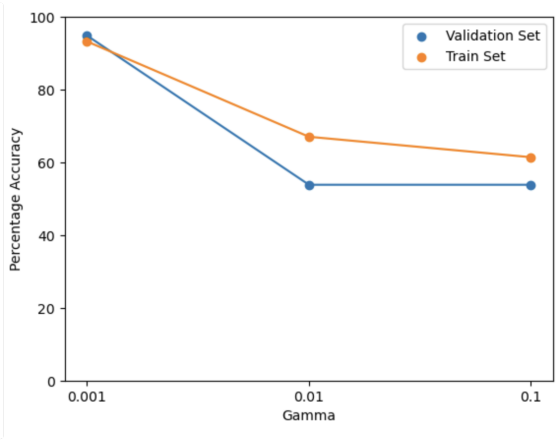
C = 0.1, γ = 0.001



C = 0.1, γ = 0.01



C = 0.1, γ = 0.001



Variation with γ, (C = 0.1)

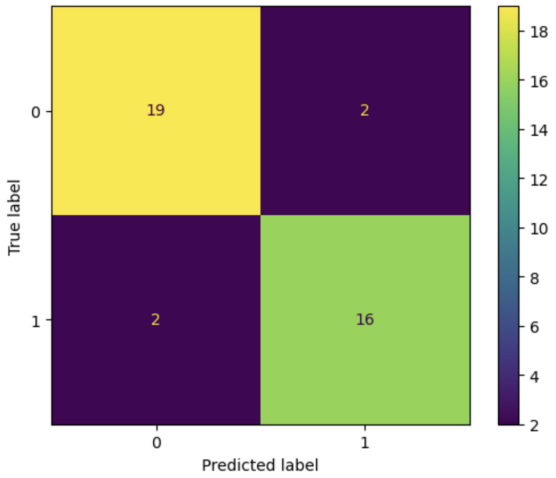
Observations

- The best results are obtained for $\gamma = 0.001$
- As γ increases, the accuracy decreases.

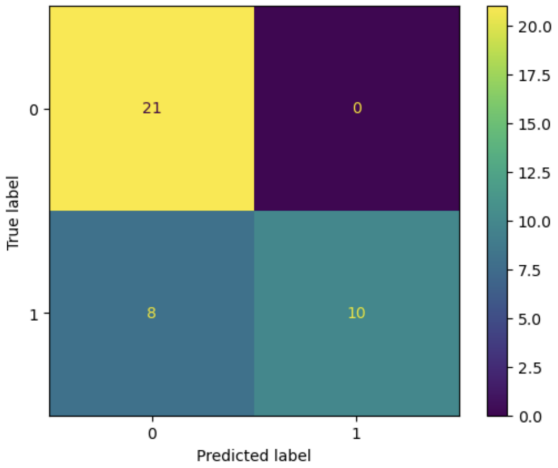
C = 1

γ	Accuracy on Train set	Accuracy on Validation set
0.001	98.50187265917603%	89.74358974358975%
0.01	100%	79.48717948717949%
0.1	100%	53.84615384615385%

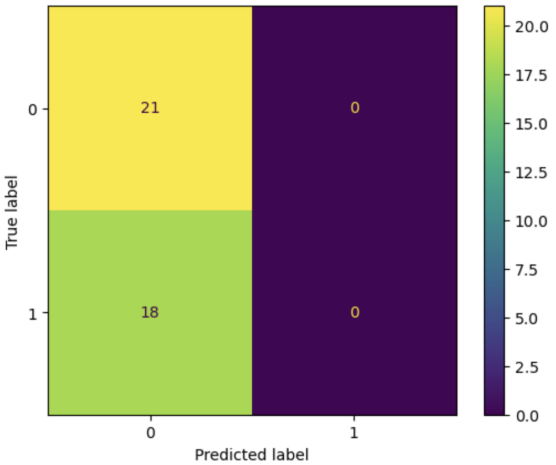
Confusion Matrix on Validation Set



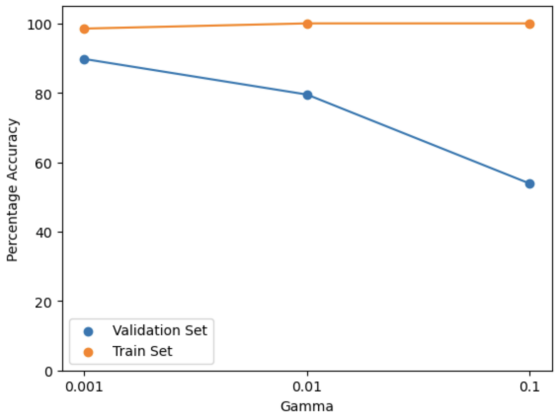
C = 1, $\gamma = 0.001$



C = 1, $\gamma = 0.01$



C = 1, $\gamma = 0.001$



Variation with γ, (C = 1)

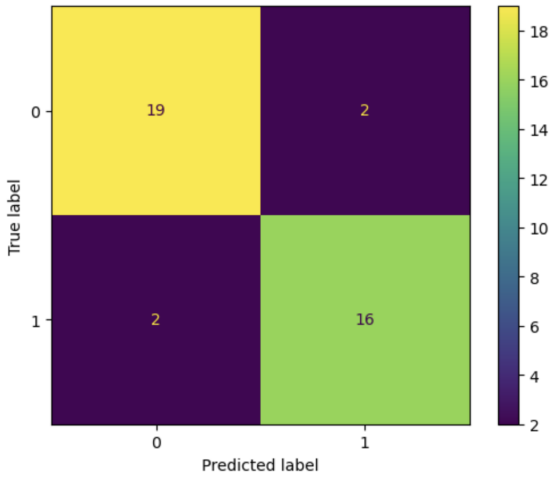
Observations

- The best results are obtained for $\gamma = 0.001$
- As γ increases, the accuracy decreases.

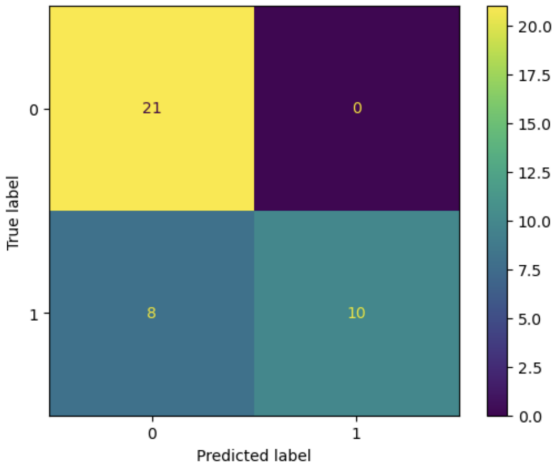
C = 10

γ	Accuracy on Train set	Accuracy on Validation set
0.001	100%	89.74358974358975%
0.01	100%	79.48717948717949%
0.1	100%	53.84615384615385%

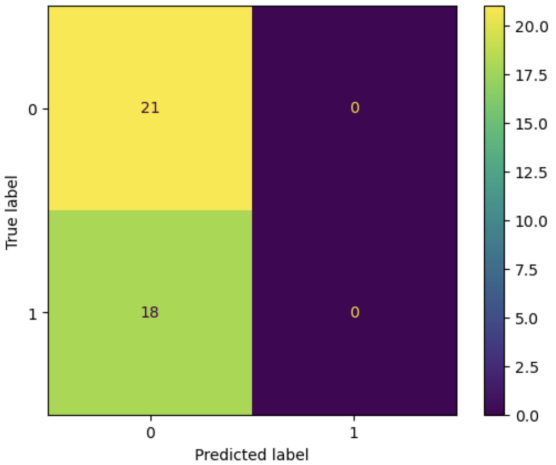
Confusion Matrix on Validation Set



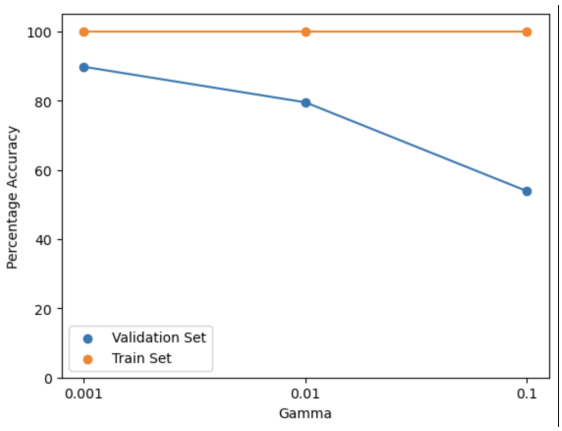
C = 10, $\gamma = 0.001$



C = 10, $\gamma = 0.01$



C = 10, $\gamma = 0.001$



Variation with γ, (C = 10)

Observations

- The best results are obtained for $\gamma = 0.001$
- As γ increases, the accuracy decreases.

Overall Observations

- For all the values of C, $\gamma = 0.001$ consistently yielded the best results.
- Highest accuracy $\approx 94.87\%$ was obtained for $C = 0.1$ and $\gamma = 0.001$ and this was even higher than that obtained on the train set. This shows that the model did not overfit and these are very optimum parameters.

4. Multi-class Classification

Overview

- In this part, the task was to use the soft margin soft vector machine implemented in the `svm_binary.py` for multi-class classification. We use both the **One vs All** and **One vs One** classification methods.
- In the **One vs All** Classification, n different models were trained for the n classes and predictions were made on each test data point using each of these classifiers.
- Instead of taking the sign of the obtained score, we use the score itself to obtain the prediction. The class corresponding to the maximum score was assigned to the data point.
- In the **One vs One** Classification, $n(n-1)/2$ different models were trained for each pair of the n classes and predictions were made on each test data point using each of these classifiers.
- Firstly for each pair of the classes, we filter out the train set to observations corresponding to the 2 labels of the pair. Then this subset is used to train the model.
- Using all these models, scores are calculated for each of the data point and absolute values for the scores for each datapoint corresponding to each class is added. The class corresponding to the maximum of these values is chosen as the prediction.

Analysis

One vs. All Classifier (RBF Kernel)

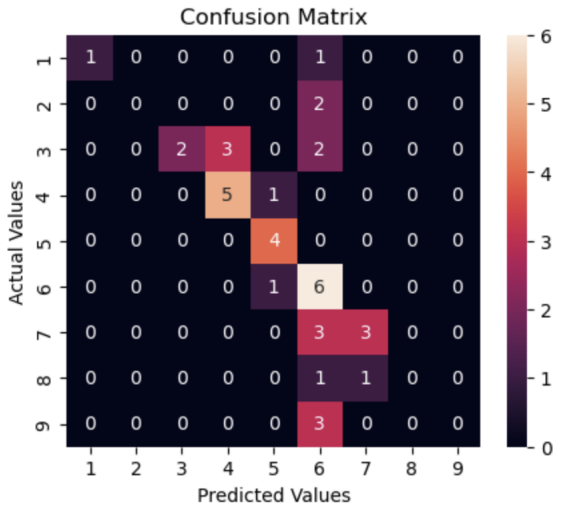
C = 0.1

γ	Accuracy on Train set	Accuracy on Validation set
0.1	76.40449438202248%	53.84615384615385%

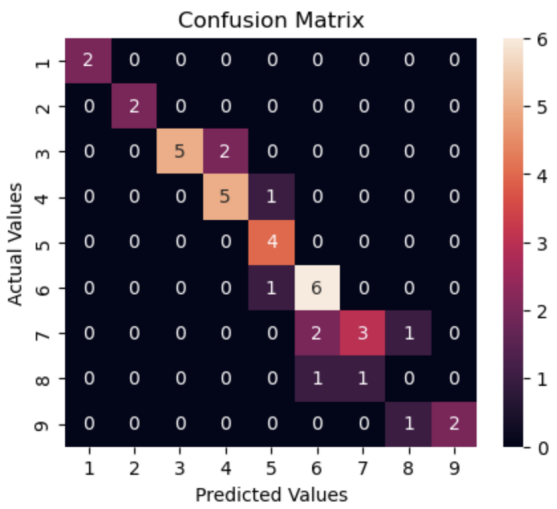
C = 1.0

γ	Accuracy on Train set	Accuracy on Validation set
0.1	98.50187265917603%	74.35897435897436%

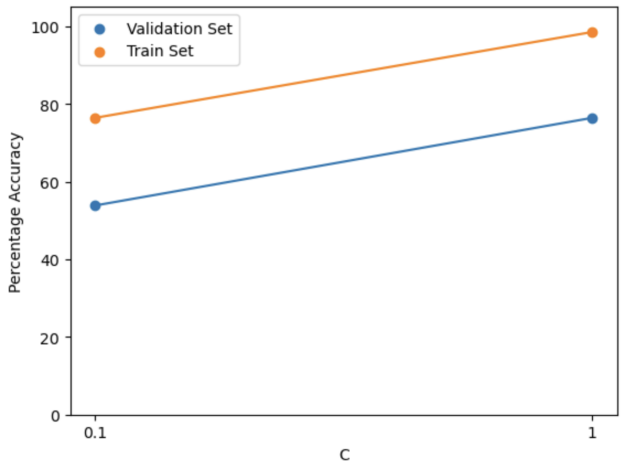
Confusion Matrix on Validation Set



$C = 0.1, \gamma = 0.1$



$C = 1.0, \gamma = 0.1$



Variation with C

Observation

- We observe that the accuracy both on the `train` and `validation` is much better for C = 1.
- The accuracy is not optimal for the case C = 0.1.
- This can be explained by the fact that C is an upper bound on the values of α, having a larger bound on C allows more flexibility on α and can allow training and better predictions.

One vs. One Classifier (RBF Kernel)

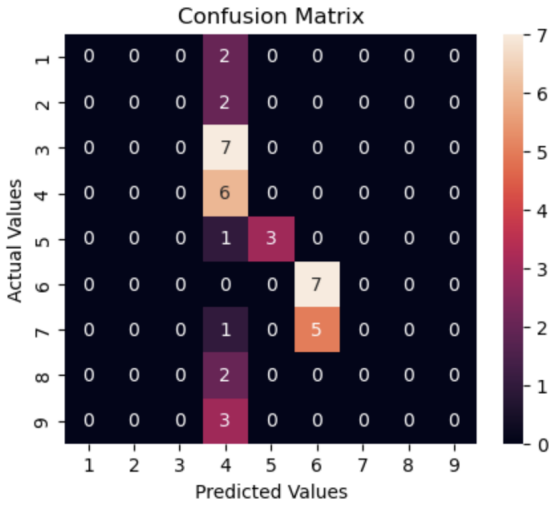
C = 0.1

γ	Accuracy on Train set	Accuracy on Validation set
0.1	55.80524344569288%	41.02564102564103%

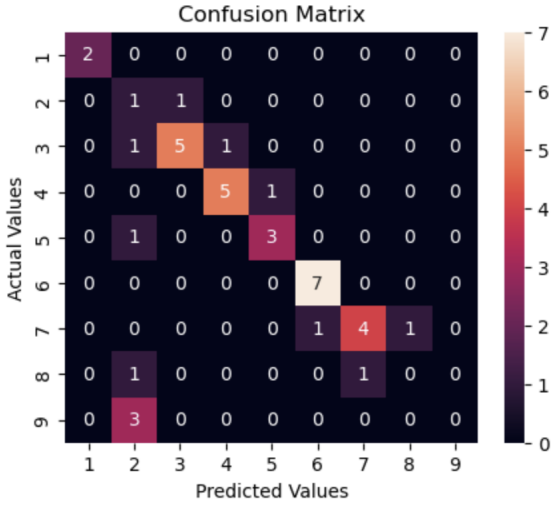
C = 1.0

γ	Accuracy on Train set	Accuracy on Validation set
0.1	97.00374531835206%	69.23076923076923%

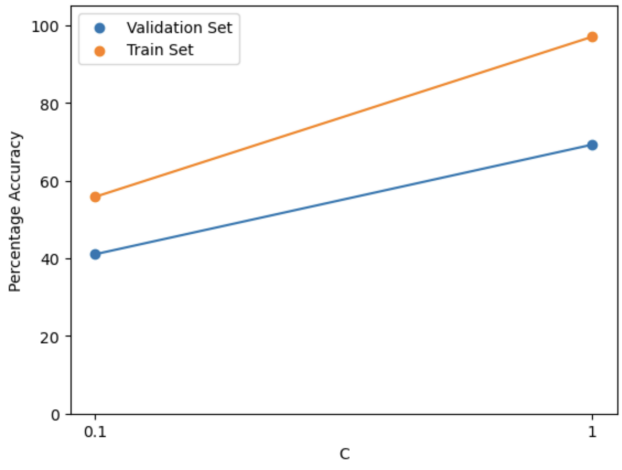
Confusion Matrix on Validation Set



C = 0.1, γ = 0.1



C = 1.0, γ = 0.1



Variation with C

Observation

- We observe that the accuracy both on the `train` and `validation` is much better for C = 1.

- The accuracy is not optimal for the case C = 0.1.
- This can be explained by the fact that C is an upper bound on the values of α, having a larger bound on C allows more flexibility on α and can allow training and better predictions.

Overall Observations

- The OVA classifier gives better results than the OVO classifier.
- The observations for both the classifiers are very similar.
- In both the cases, C = 1 gives much better results. This is due to the fact that having a smaller bound on the values of α might not be allowing the model to train good enough to give accurate predictions.

5. Competitive Part

- For **binary classification**, I have chosen the parameters, C = 0.1 and $\gamma = 0.001$
- For **multi-class classification**, I have chosen the parameters, C = 10 and $\gamma = 0.001$
- These have been chosen after testing on the validation set multiple times using varied set of hyper-parameters and observing which set of hyper-parameters yields the best results.