Pattern Recognition & Machine Learning
Lab-11 Assignment

Name: Tanish Pagaria
Roll No.: B21AI040

Loading the dataset

Performed using Pandas.

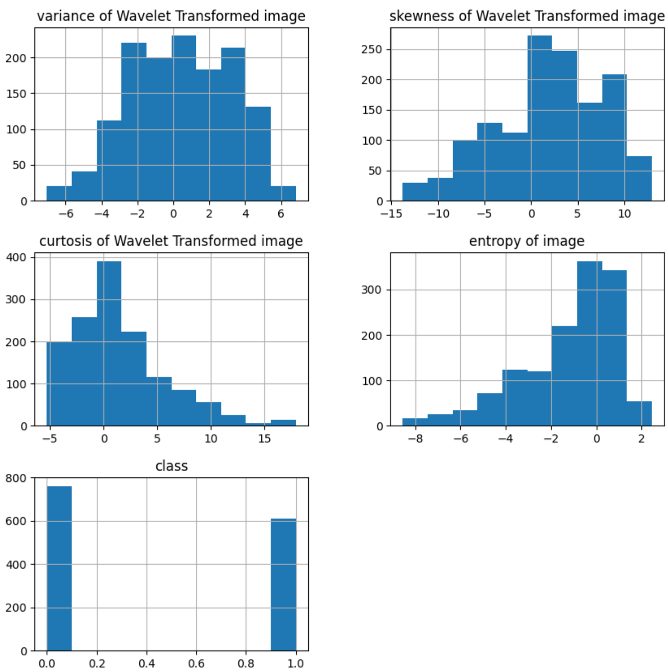
	variance of Wavelet Transformed image	skewness of Wavelet Transformed image	curtosis of Wavelet Transformed image	entropy of image	class
0	3.62160	8.66610	-2.8073	-0.44699	0
1	4.54590	8.16740	-2.4586	-1.46210	0
2	3.86600	-2.63830	1.9242	0.10645	0
3	3.45660	9.52280	-4.0112	-3.59440	0
4	0.32924	-4.45520	4.5718	-0.98880	0
...	...	...	...	...	...
1367	0.40614	1.34920	-1.4501	-0.55949	1
1368	-1.38870	-4.87730	6.4774	0.34179	1
1369	-3.75030	-13.45860	17.5932	-2.77710	1
1370	-3.56370	-8.38270	12.3930	-1.28230	1
1371	-2.54190	-0.65804	2.6842	1.19520	1

1372 rows × 5 columns

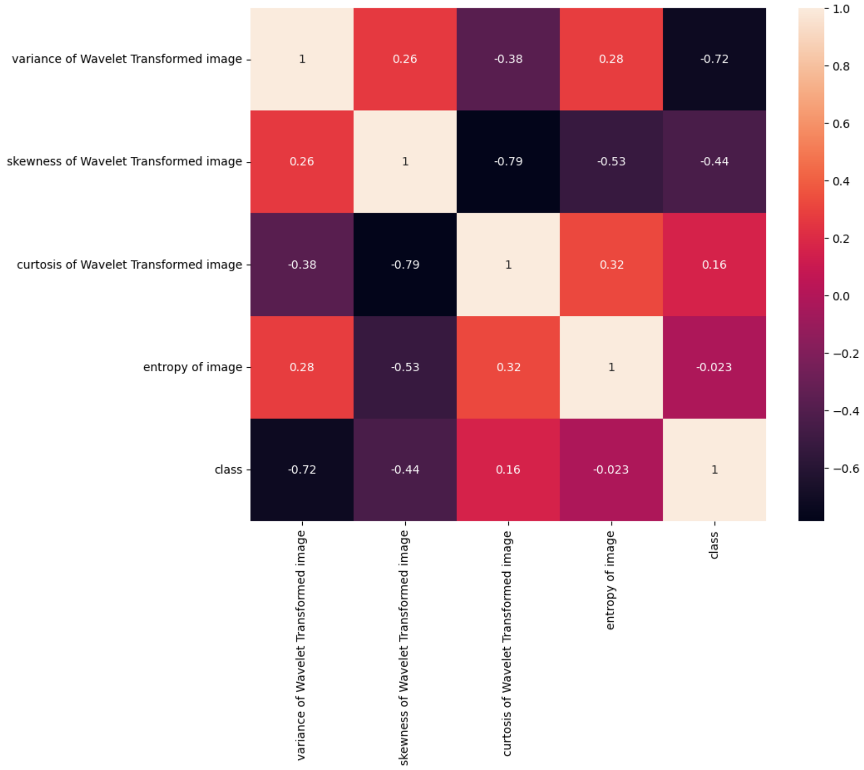
pandas dataframe for the dataset

Exploratory Analysis and Dataset Visualization

The dataset did not contain any NULL values or NaN entries. There were two classes in the dataset.



Histogram representation of the dataset



Correlation heatmap for the dataset

Dataset Preprocessing and Train-Test-Validation Splitting

Preprocessing included only performing standard scaling on the features. It was performed using `sklearn.preprocessing.StandardScaler`.

The train-test-validation was performed on the dataset in the ratio 70:20:10 using `sklearn.model_selection.train_test_split`.

Training SVM Classifier models and Plotting Decision Boundaries

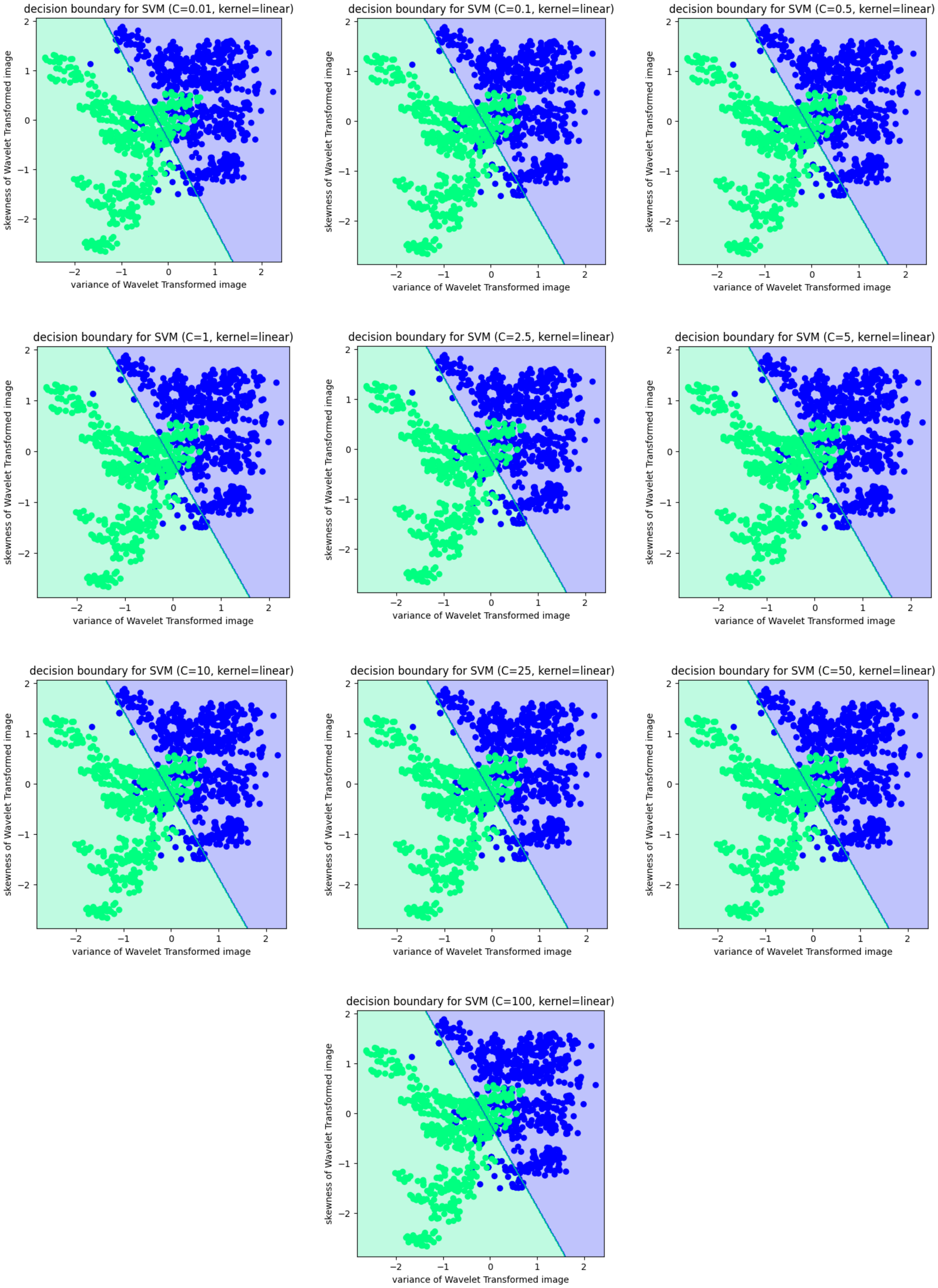
The SVM classifier models were created using the `sklearn.svm.SVC` class. The hyperparameters C and kernel were varied, and decision boundaries for the same were plotted on the dataset while considering the two features selected using the correlation heatmap previously. The test accuracy scores were also reported.

The two features: **variance of Wavelet Transformed image** and the **skewness of Wavelet Transformed image** were used while plotting the decision boundaries for the classifier models.

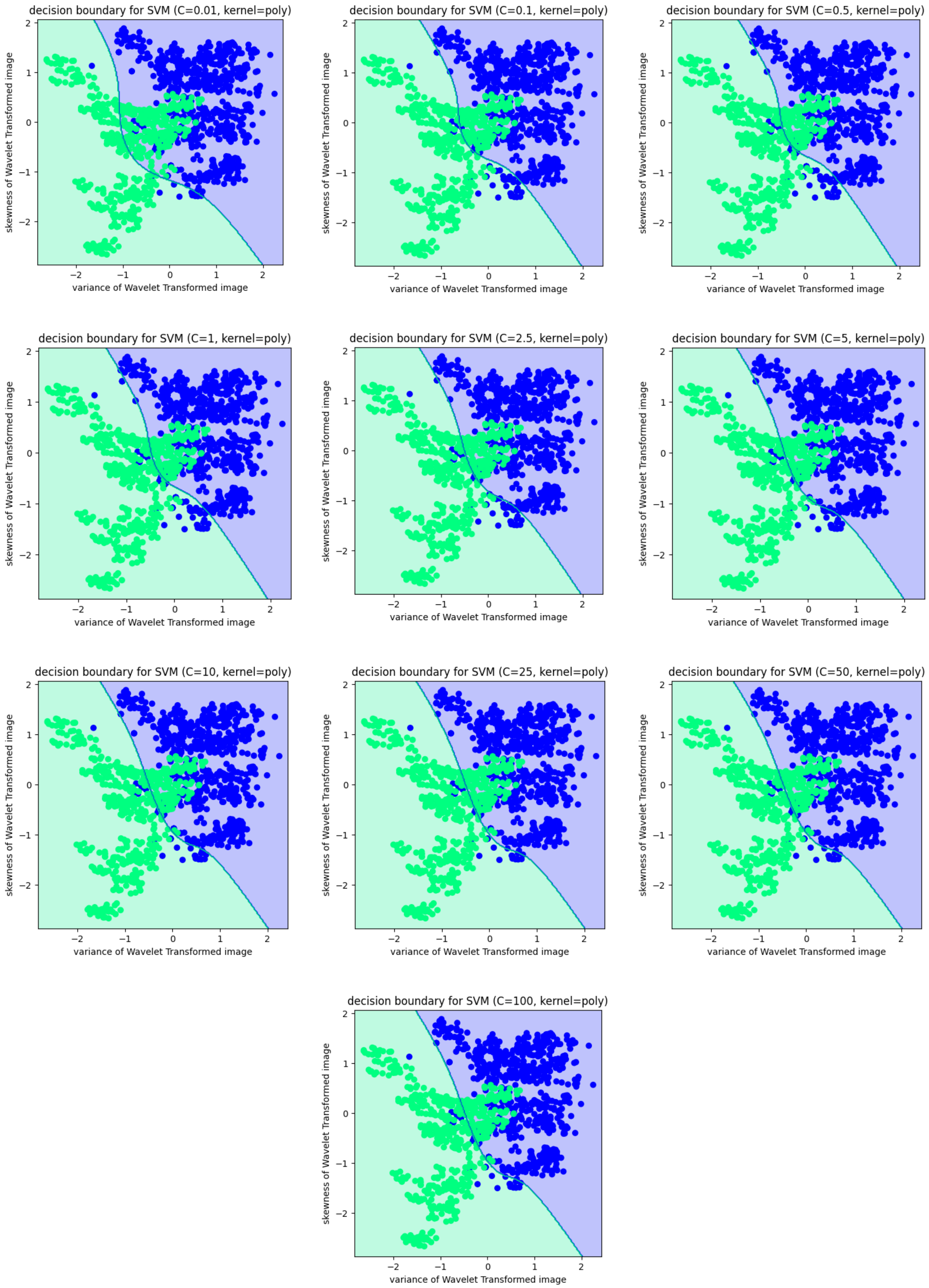
Test accuracy scores:-

```
test accuracy of SVM (C=0.01, kernel=linear): 0.9710144927536232
test accuracy of SVM (C=0.1, kernel=linear): 0.9818840579710145
test accuracy of SVM (C=0.5, kernel=linear): 0.9855072463768116
test accuracy of SVM (C=1, kernel=linear): 0.9855072463768116
test accuracy of SVM (C=2.5, kernel=linear): 0.9855072463768116
test accuracy of SVM (C=5, kernel=linear): 0.9891304347826086
test accuracy of SVM (C=10, kernel=linear): 0.9891304347826086
test accuracy of SVM (C=25, kernel=linear): 0.9891304347826086
test accuracy of SVM (C=50, kernel=linear): 0.9855072463768116
test accuracy of SVM (C=100, kernel=linear): 0.9855072463768116

test accuracy of SVM (C=0.01, kernel=poly): 0.7246376811594203
test accuracy of SVM (C=0.1, kernel=poly): 0.9384057971014492
test accuracy of SVM (C=0.5, kernel=poly): 0.9818840579710145
test accuracy of SVM (C=1, kernel=poly): 0.9746376811594203
test accuracy of SVM (C=2.5, kernel=poly): 0.9782608695652174
test accuracy of SVM (C=5, kernel=poly): 0.9891304347826086
test accuracy of SVM (C=10, kernel=poly): 0.9855072463768116
test accuracy of SVM (C=25, kernel=poly): 0.9963768115942029
test accuracy of SVM (C=50, kernel=poly): 1.0
test accuracy of SVM (C=100, kernel=poly): 1.0

test accuracy of SVM (C=0.01, kernel=rbf): 0.9130434782608695
test accuracy of SVM (C=0.1, kernel=rbf): 0.9927536231884058
test accuracy of SVM (C=0.5, kernel=rbf): 1.0
test accuracy of SVM (C=1, kernel=rbf): 1.0
test accuracy of SVM (C=2.5, kernel=rbf): 1.0
test accuracy of SVM (C=5, kernel=rbf): 1.0
test accuracy of SVM (C=10, kernel=rbf): 1.0
test accuracy of SVM (C=25, kernel=rbf): 1.0
test accuracy of SVM (C=50, kernel=rbf): 1.0
test accuracy of SVM (C=100, kernel=rbf): 1.0

test accuracy of SVM (C=0.01, kernel=sigmoid): 0.9021739130434783
test accuracy of SVM (C=0.1, kernel=sigmoid): 0.8297101449275363
test accuracy of SVM (C=0.5, kernel=sigmoid): 0.782608695652174
test accuracy of SVM (C=1, kernel=sigmoid): 0.7681159420289855
test accuracy of SVM (C=2.5, kernel=sigmoid): 0.7644927536231884
test accuracy of SVM (C=5, kernel=sigmoid): 0.7644927536231884
test accuracy of SVM (C=10, kernel=sigmoid): 0.7644927536231884
test accuracy of SVM (C=25, kernel=sigmoid): 0.7608695652173914
test accuracy of SVM (C=50, kernel=sigmoid): 0.7608695652173914
test accuracy of SVM (C=100, kernel=sigmoid): 0.7608695652173914
```

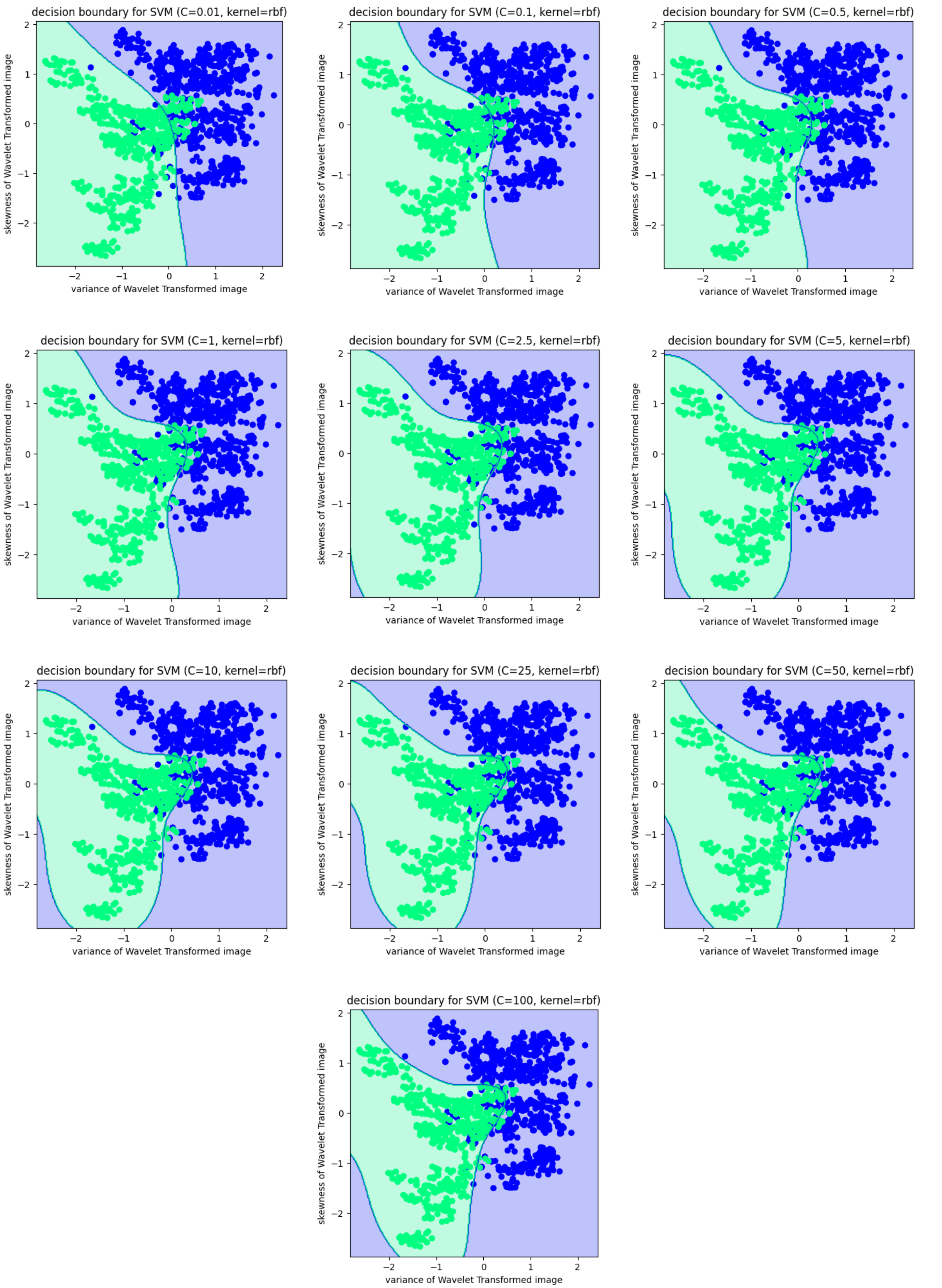
Decision boundary plots for kernel='`linear`' (for different C values)



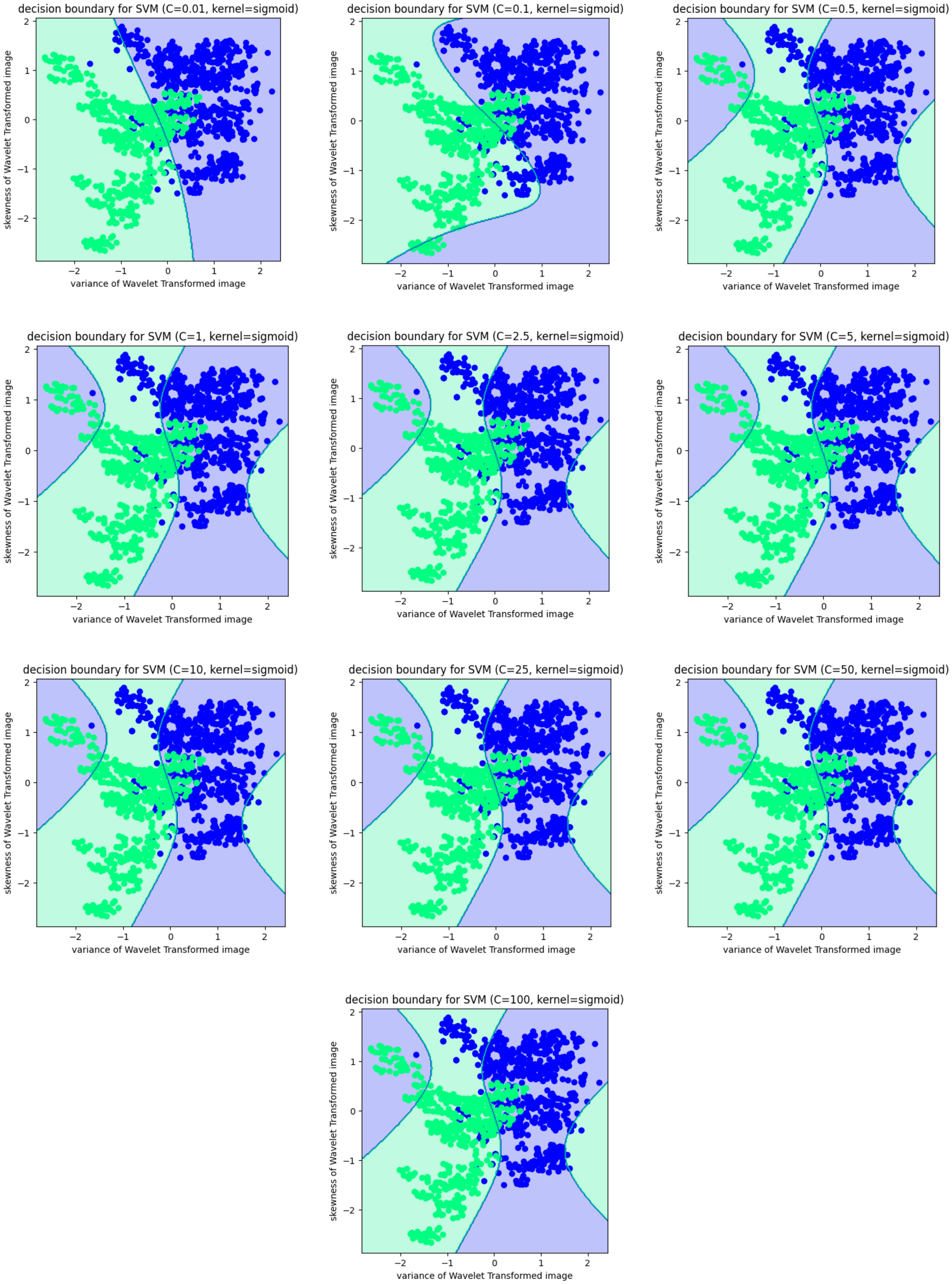
Decision boundary plots for kernel='poly' (for different C values)



Decision boundary plots for kernel=`'rbf'` (for different C values)

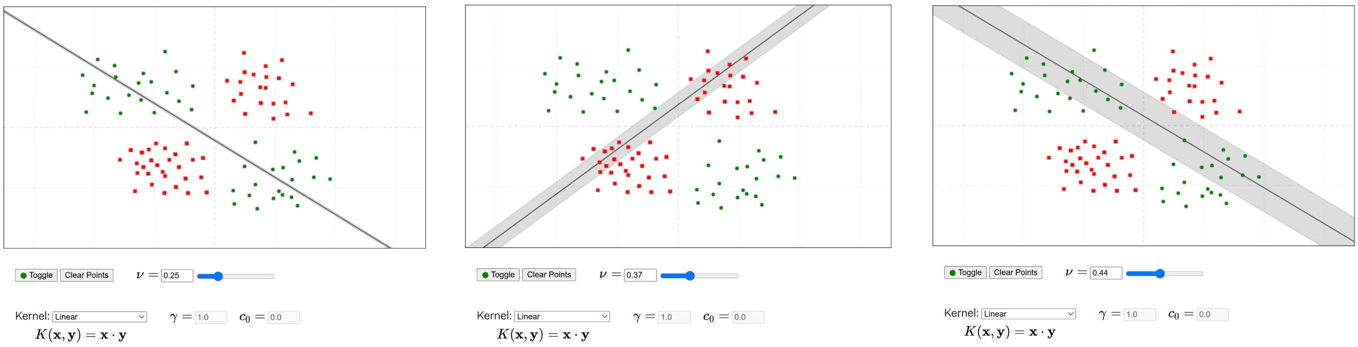


Decision boundary plots for kernel=`'sigmoid'` (for different C values)

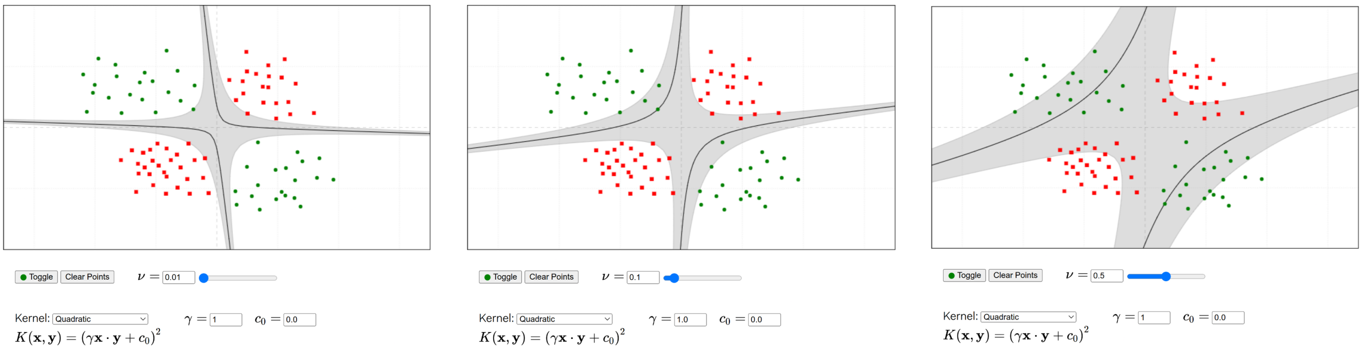


Interactive SVM visualizer

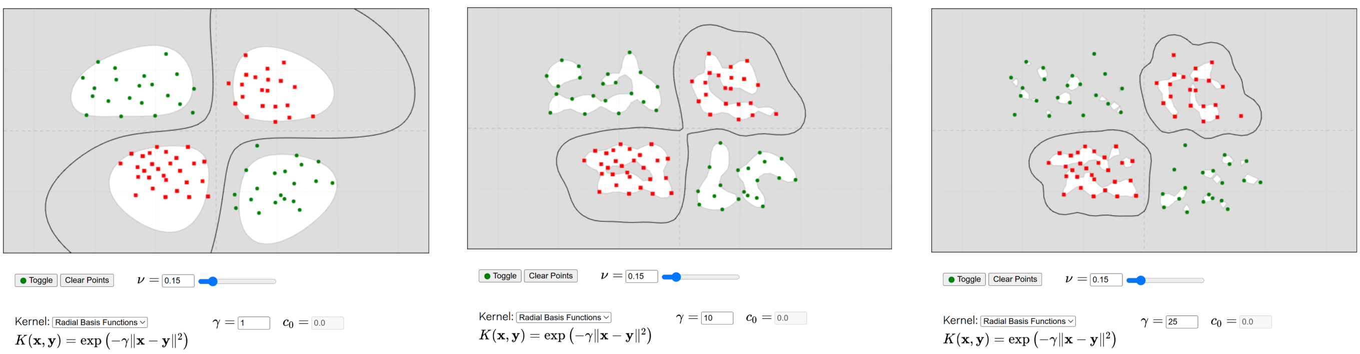
<u>Dataset-1</u>



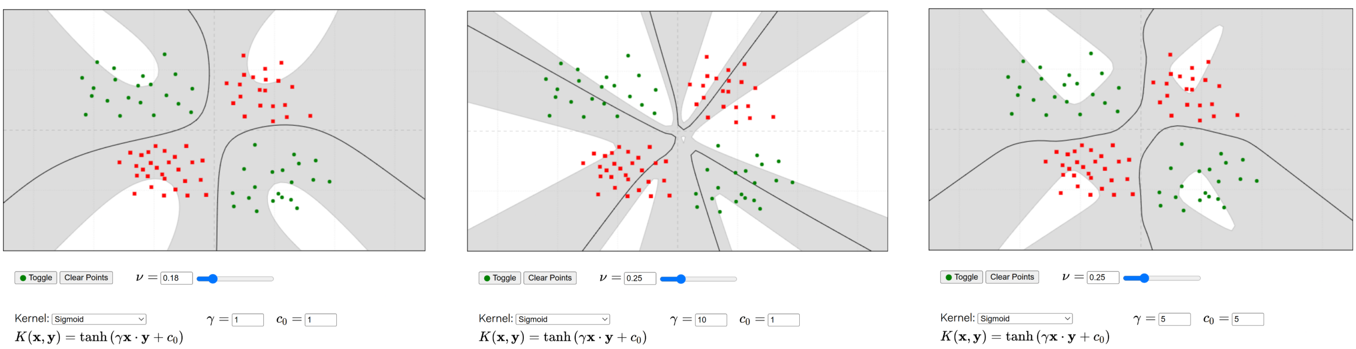
Linear kernel did not show any good results with this dataset since the classes are not exactly linearly separable.



Quadratic kernel showed good results with the dataset. Changing the regularization parameters generated decent decision boundaries, but after a certain point, the boundaries started to get worse. Changing the value of C also did not give any

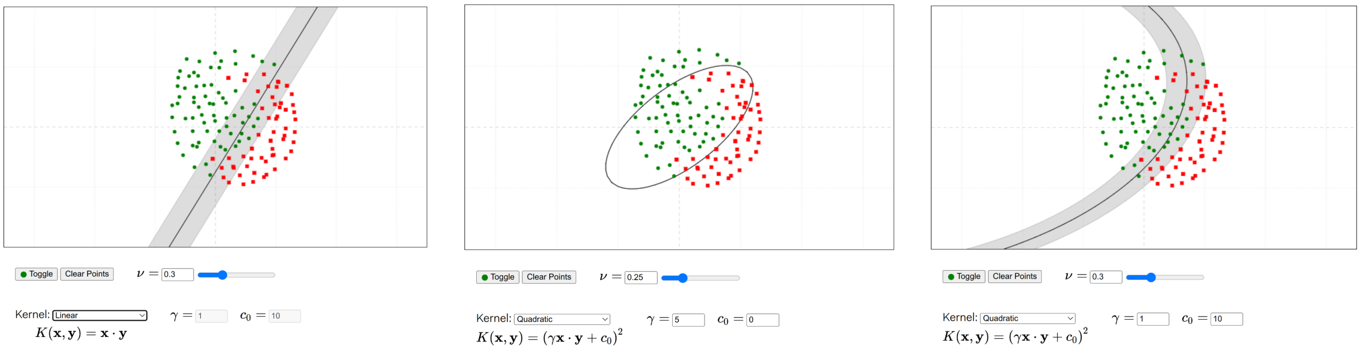


Radial Basis Functions kernel showed a good decision boundary, but with increasing values of gamma, it started to overfit the data points.

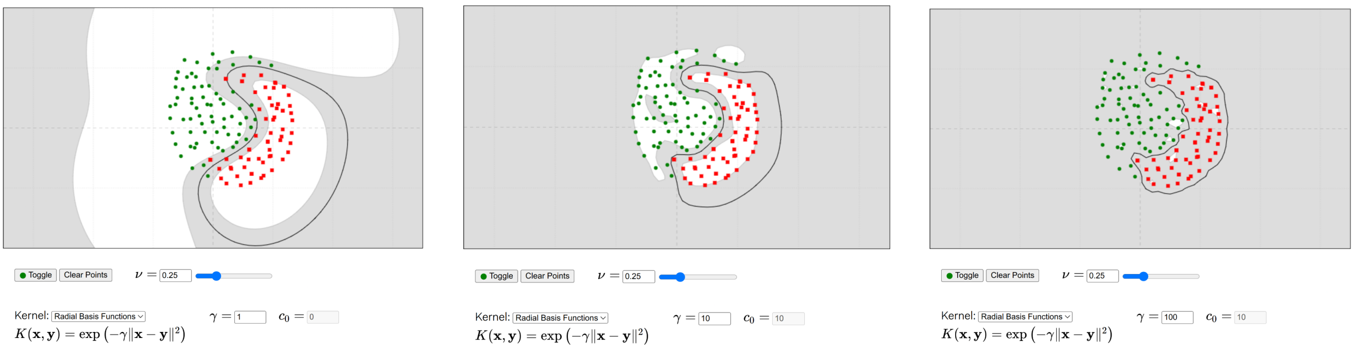


Sigmoid kernel gave varied results when varying the hyperparameters. It was able to plot the decision boundary.

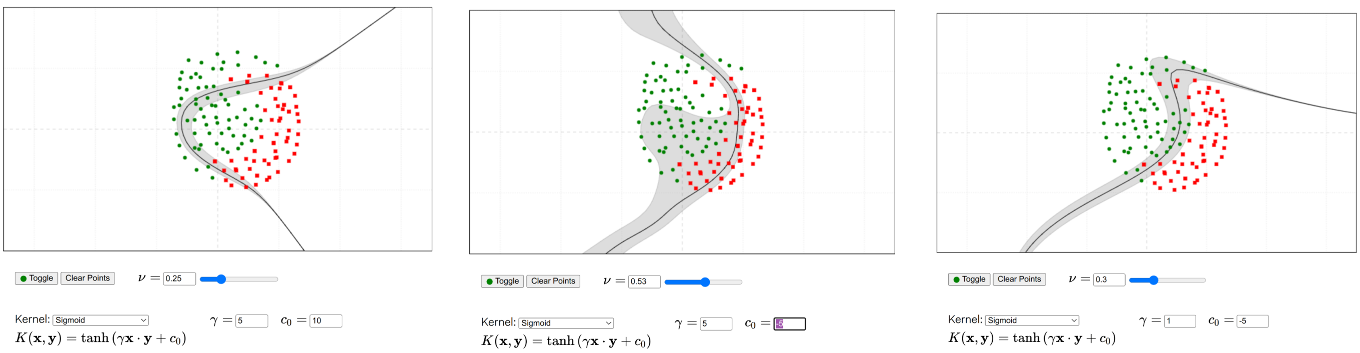
Dataset-2



Linear kernel was unable to perform well for this dataset. *Quadratic* kernel, after hyperparameter tuning, was able to generate the above shown (rightmost) decision boundary.



Radial Basis Functions kernel was able to optimally generate the decision boundary for the dataset.



Sigmoid kernel showed varied decision boundaries during hyperparameter tuning. By decreasing the value of C, we were able to capture the above shown (rightmost) decision boundary.

The choice of hyperparameters can greatly affect the performance of an SVM model. The optimal hyperparameters depend on the complexity and nonlinearity of the data. Generally, a linear kernel works well for linearly separable data, while an RBF kernel works well for nonlinear data. The value of C should be chosen carefully to prevent overfitting or underfitting. A small value of C can lead to more errors, while a large value of C can lead to overfitting. The value of gamma should also be chosen carefully to prevent overfitting or underfitting. A small value of gamma can lead to a smoother decision boundary, while a large value of gamma can lead to a more complex decision boundary.