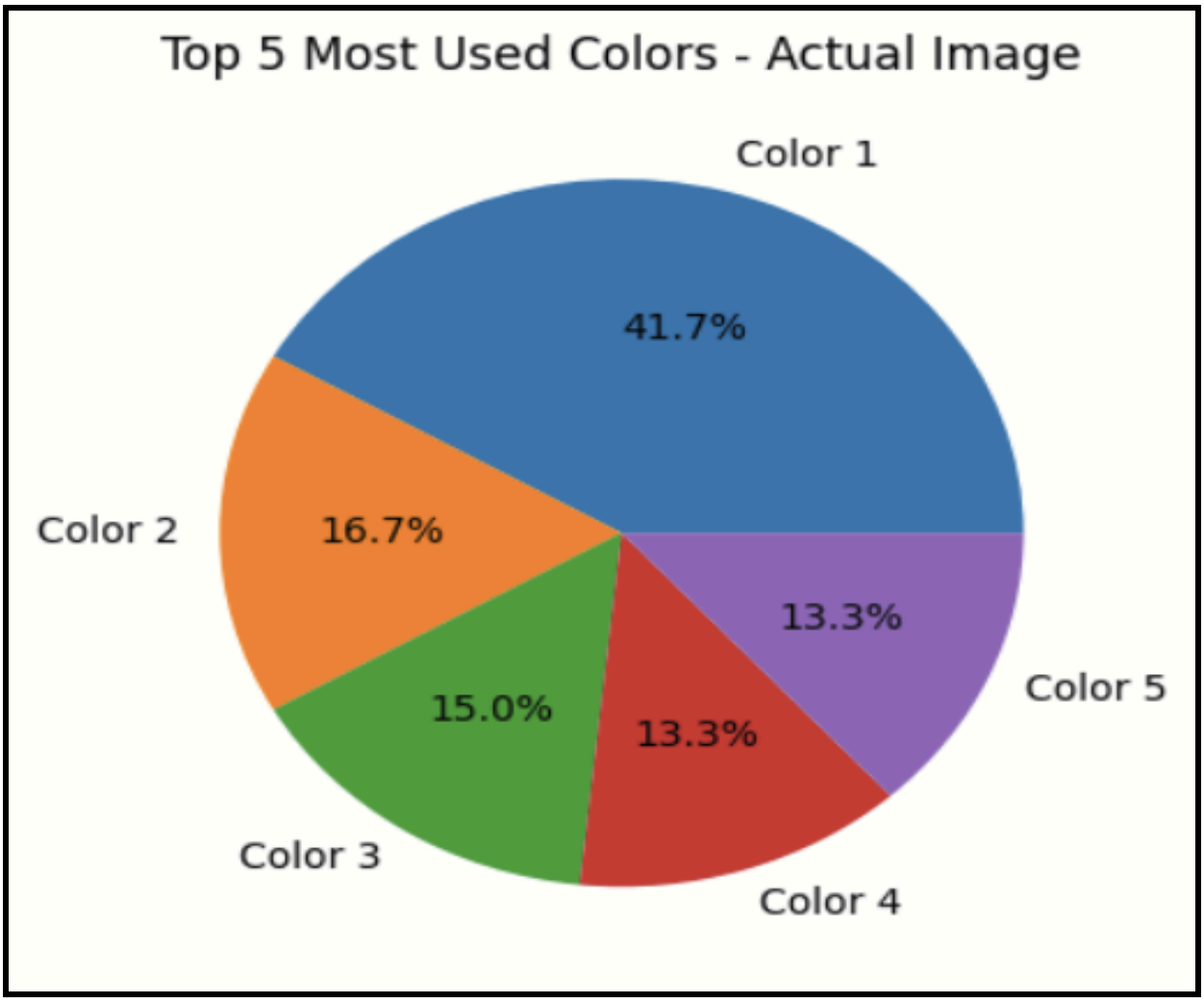
Pattern Recognition and Machine Learning 2024
Lab - 9 & 10

Question 1: Image Compression using K-means

In order to keep digital photos on less storage space while maintaining their vital visual information, image compression is an essential approach. K-means clustering is a popular compression technique that compresses photos efficiently and simply by using the color of the pixel that is closest to it to represent each pixel. In this study, we investigate the application of K-means clustering for picture compression, focusing on the tasks mentioned in the question.



Task-wise Implementation

(a) Implementing computeCentroid Function

The computeCentroid function calculates the mean of 3-dimensional features, effectively finding the centroid of a set of points in a multi-dimensional space. This centroid serves as the representative color for a cluster of pixels in the image.

```
[137.3913345336914, 128.8587760925293, 113.11710739135742]
```

(b) Implementing mykmeans Function

The K-means clustering process has been written from scratch (mykmeans function). This function iteratively assigns pixels to their nearest centroids and updates the centroids until convergence, producing the final cluster centers, given a data matrix X of pixel characteristics and the required number of clusters, k.

(c) Displaying Compressed Images for Different Values of k

We use the centroids from K-means clustering to represent the image's pixels using the colors of their respective centroids. We produce compressed images with varied degrees of compression by changing the number of clusters, k. This enables us to see how image quality and compression ratio are traded off.

```
value of k is 1
```

```
value of k is 2
```

```
value of k is 4
```

```
value of k is 5
```

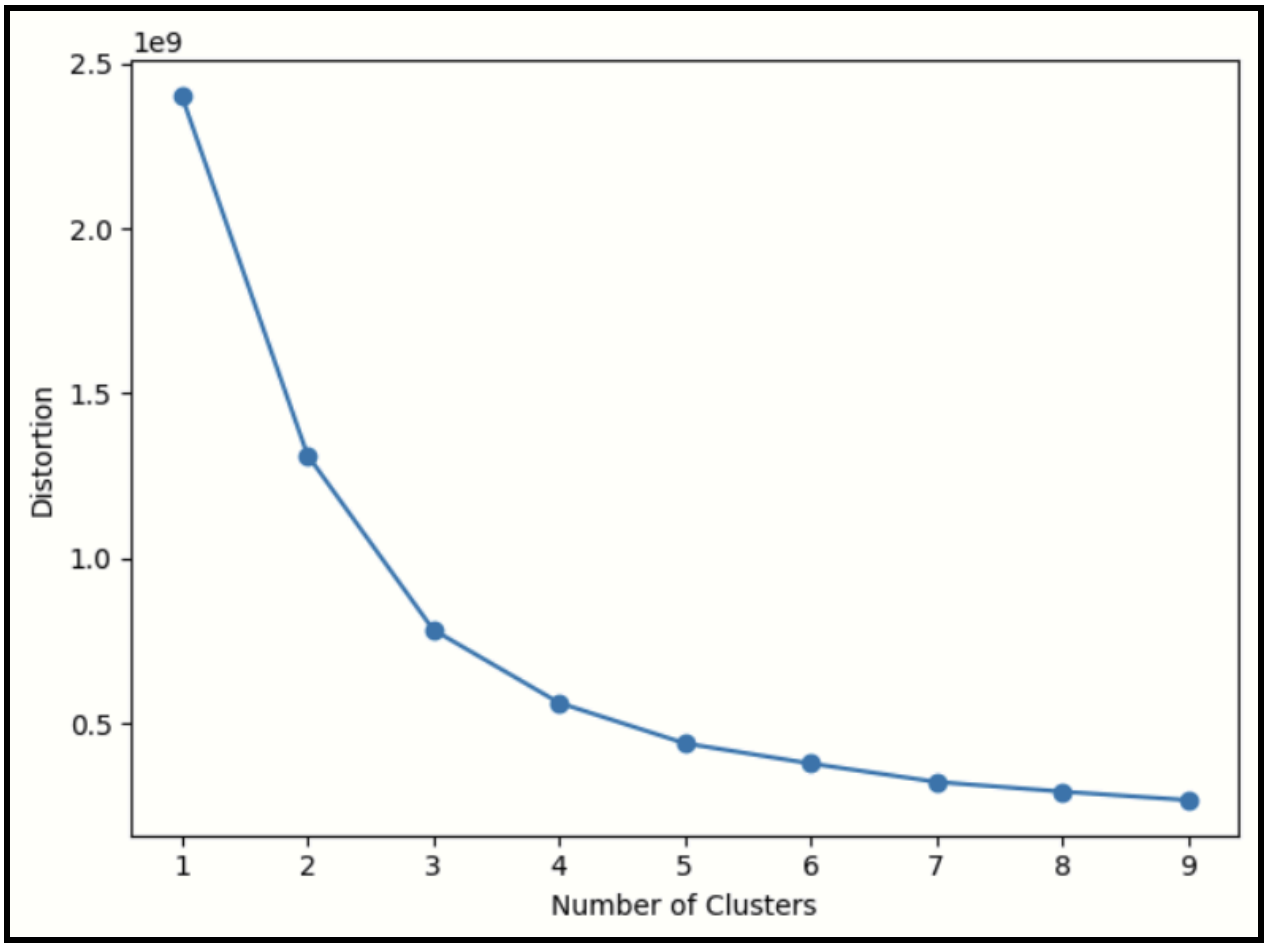
```
value of k is 7
```

```
value of k is 9
```

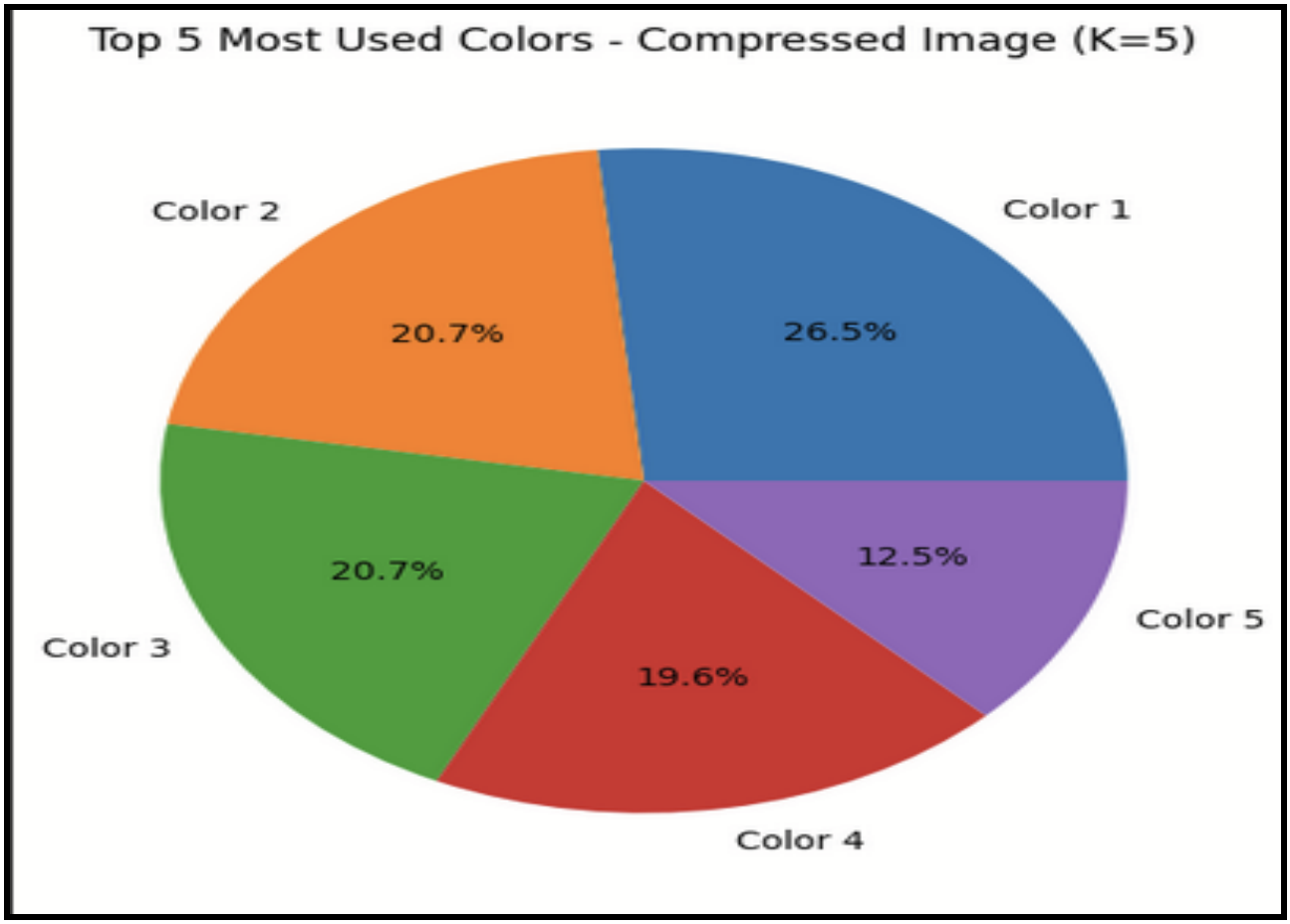
```
value of k is 10
```

(d) Comparing Results with sklearn's K-means Implementation

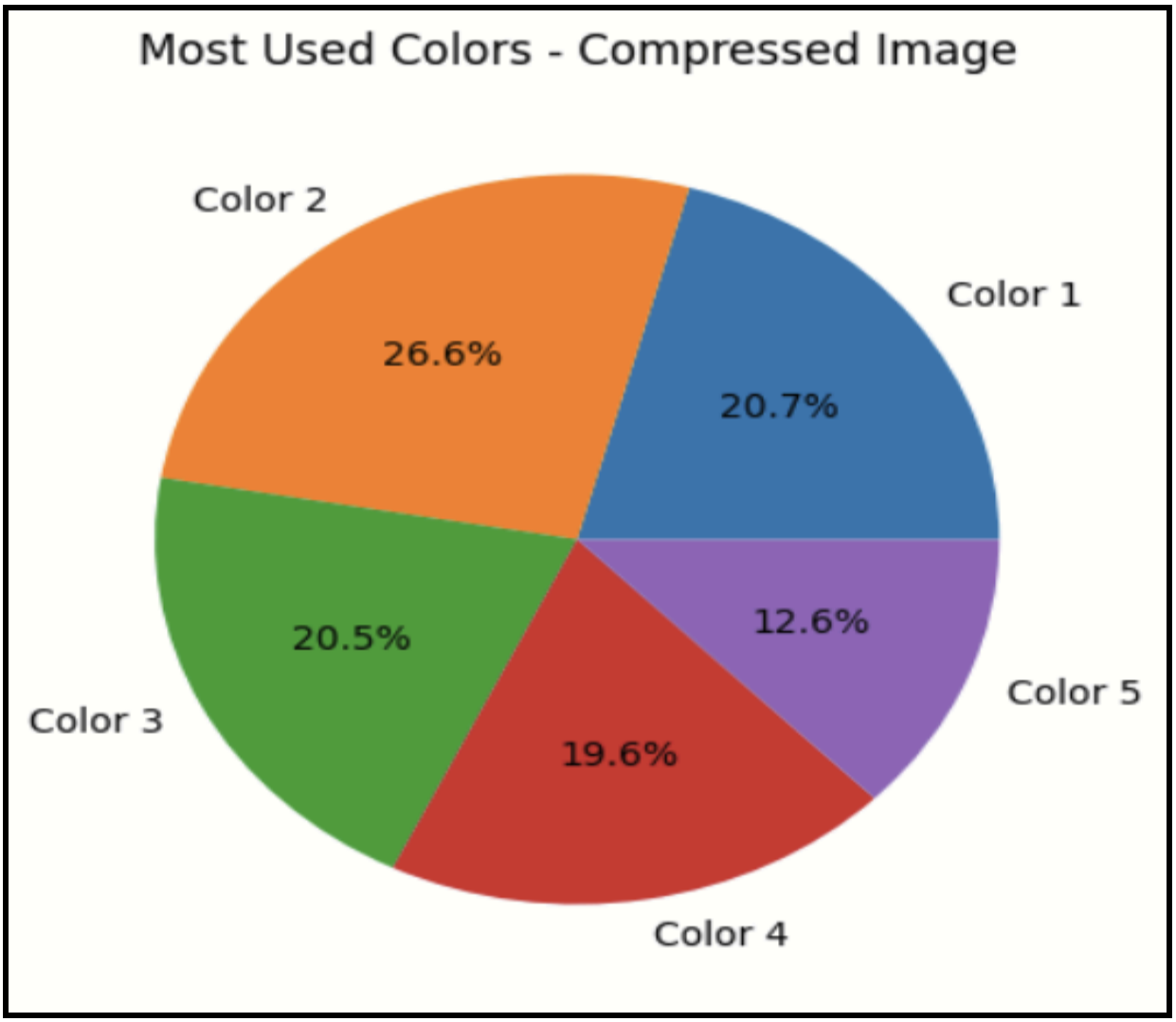
We evaluate the compressed images obtained using our custom K-means implementation against those generated using the K-means implementation provided by the sk-learn library. Differences in image quality, compression ratio, and computational efficiency may be observed between the two approaches.



Elbow method to check the best value of k



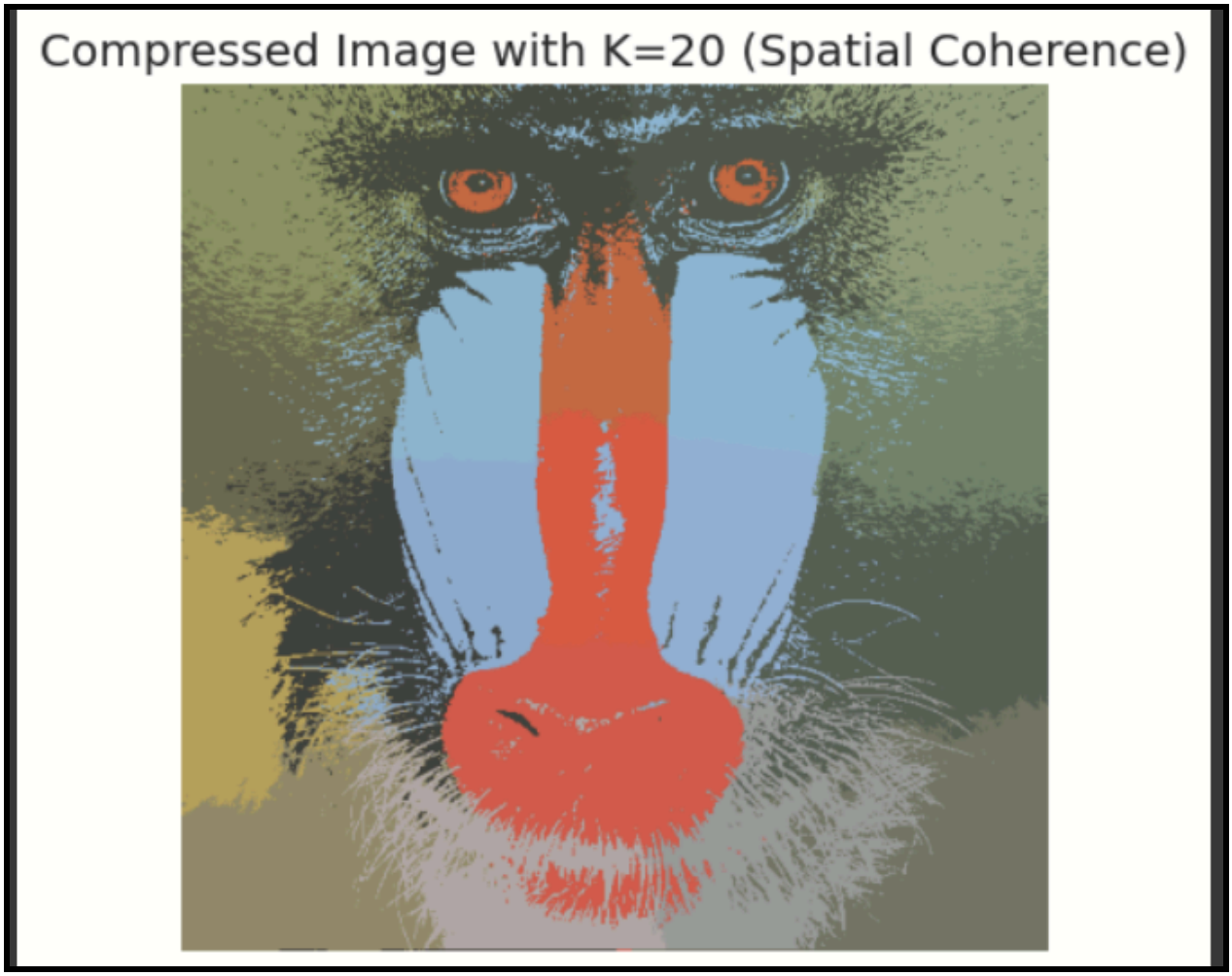
Through mykmeans function



Through sklearn library

(e) Incorporating Spatial Coherence

We suggest adding spatial information to the clustering procedure in order to preserve spatial coherence in the compressed image. Our approach guarantees that adjacent pixels in the original image have a higher probability of being assigned to the same cluster by taking into account both the spatial closeness and color similarity of individual pixels. By preserving local structures, the compressed image's artifacts like color bleeding and noise are lessened. Before executing K-means clustering, spatial coordinates must be added to the pixel characteristics in order to implement spatial coherence.



Conclusion

To sum up, K-means clustering picture compression offers a practical way to minimize digital image storage space requirements without sacrificing image quality. We showed how to apply K-means clustering for image compression by completing the tasks listed in the question, which included computing the centroid, implementing the clustering algorithm, and evaluating the compressed photos. Spatial coherence reduces artifacts and preserves local structures, improving compressed image quality even further.

Question 2: the decision boundaries of Support Vector Machines (SVM)

Introduction

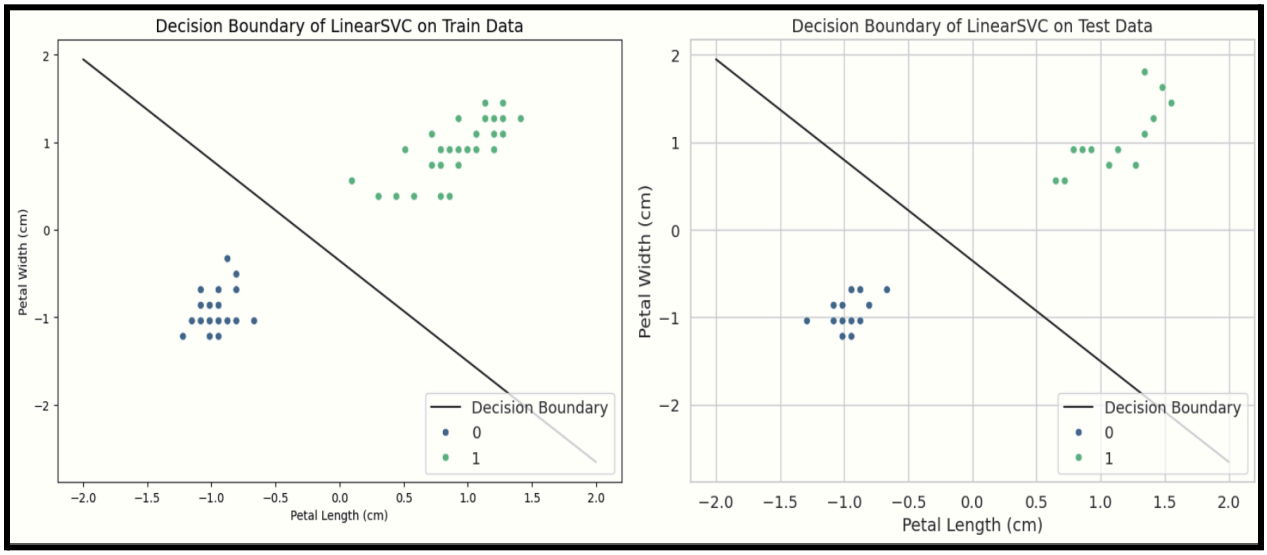
Robust supervised learning models, Support Vector Machines (SVMs) are extensively employed in classification problems. They function by locating the ideal hyperplane in the feature space that best divides several classes. In this study, we examine SVM analysis with a particular emphasis on optimizing hyperparameters for better performance and utilizing various kernels to explore decision limits.

Task 1(a): Data Preparation

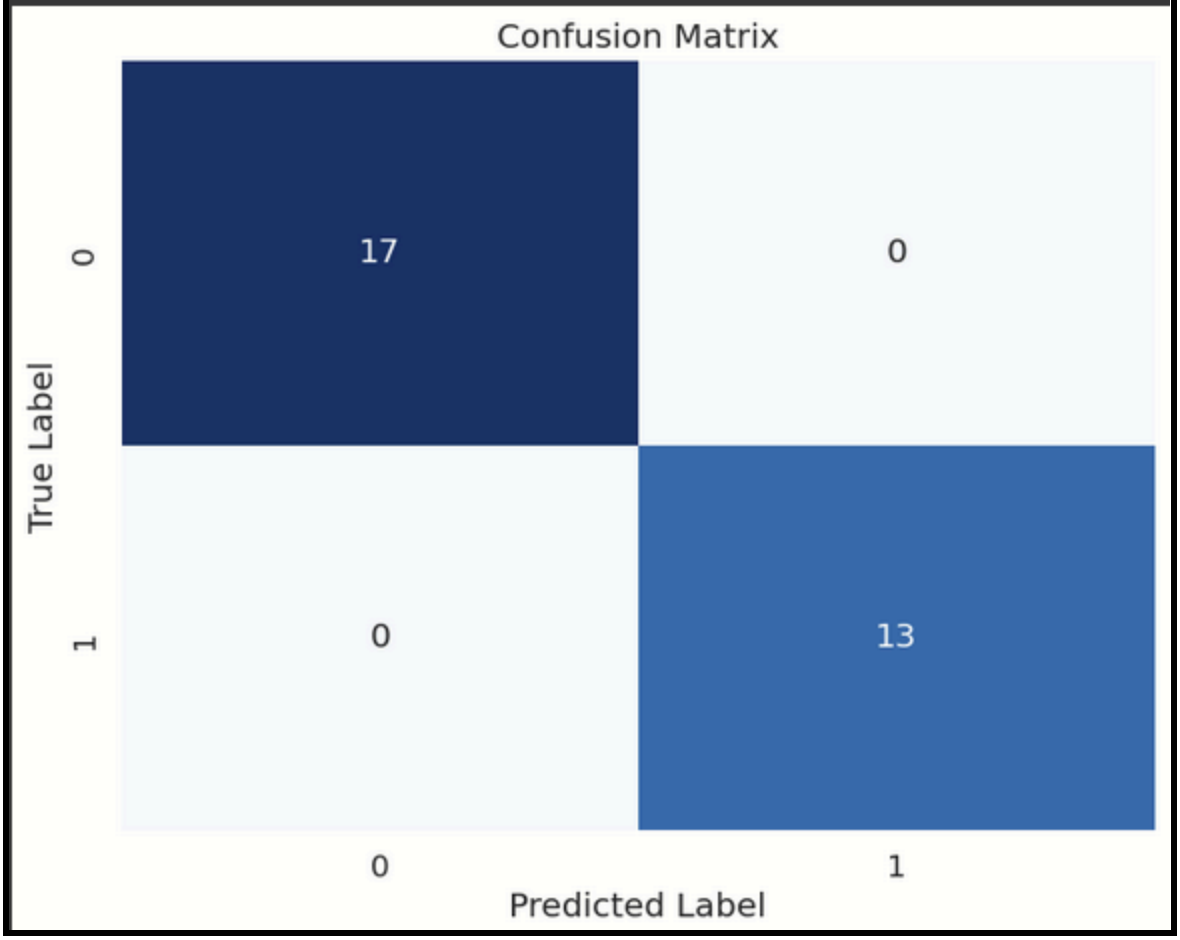
The dataset must be ready for analysis as the initial job. First, we import the Iris dataset and restrict our selection to the petal width and length attributes. This subset preserves discriminative characteristics while simplifying the task. The classifications "setosa" and "versicolor" are then chosen in order to produce a binary classification dataset. The dataset is divided into training and testing subsets after normalization is conducted to guarantee consistent feature scales.

Task 1(b): Training Linear Support Vector Classifier (LinearSVC)

For this assignment, we will be training a Linear Support Vector Classifier (LinearSVC) using the preprocessed training data. The goal is to acquire knowledge of a decision boundary that efficiently distinguishes between the two classes. We generate a visual representation of the decision boundary using the training data and assess the model's ability to generalize by placing the test data alongside the original decision boundary.



```
Test Accuracy: 1.0
Training Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       1.00      1.00      1.00        13

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```



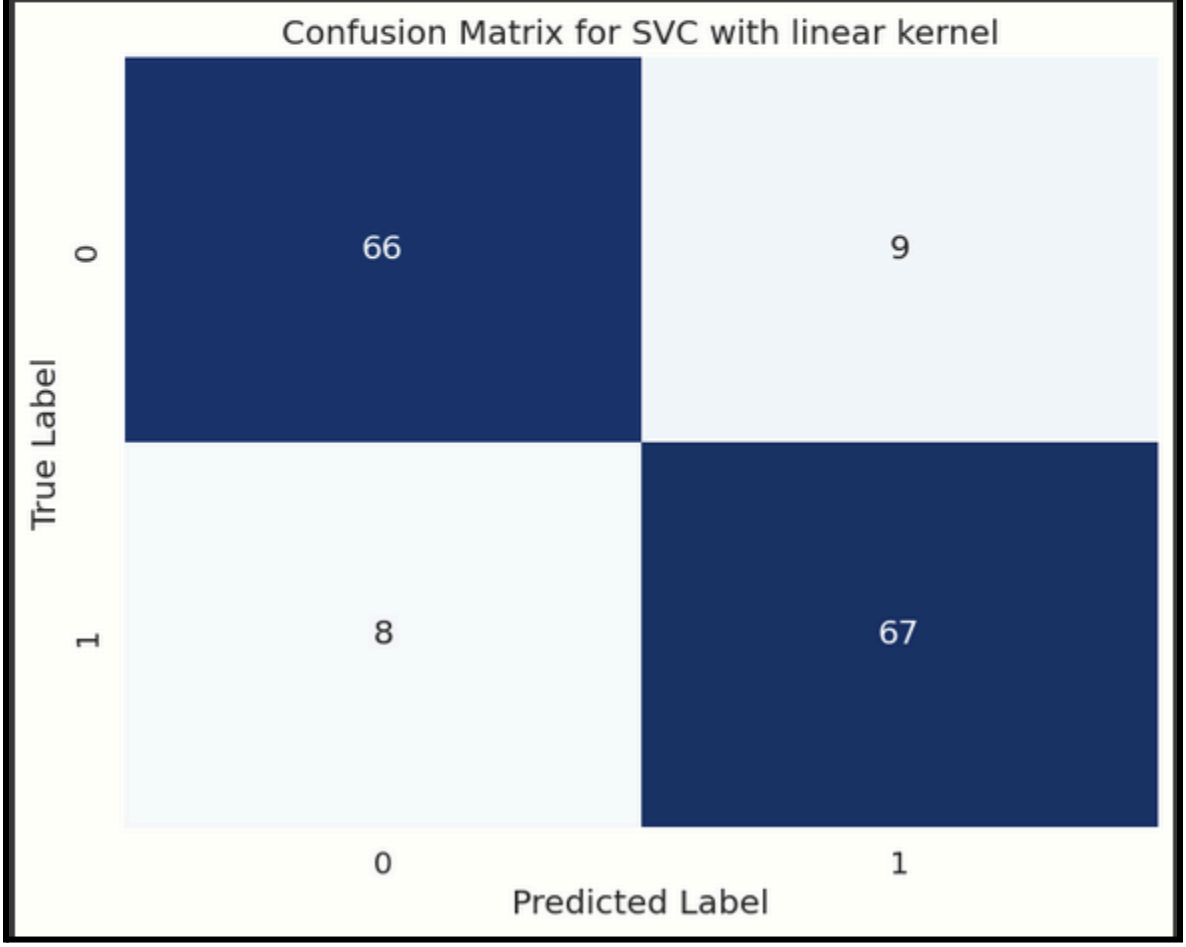
Task 2(a): Generating Synthetic Dataset

Task 2 entails creating a synthetic dataset by utilizing the make_moons() tool and introducing more noise. Around 500 data points are produced with a noise level of 5%, resulting in a more difficult classification task to evaluate the effectiveness of SVM under different scenarios.

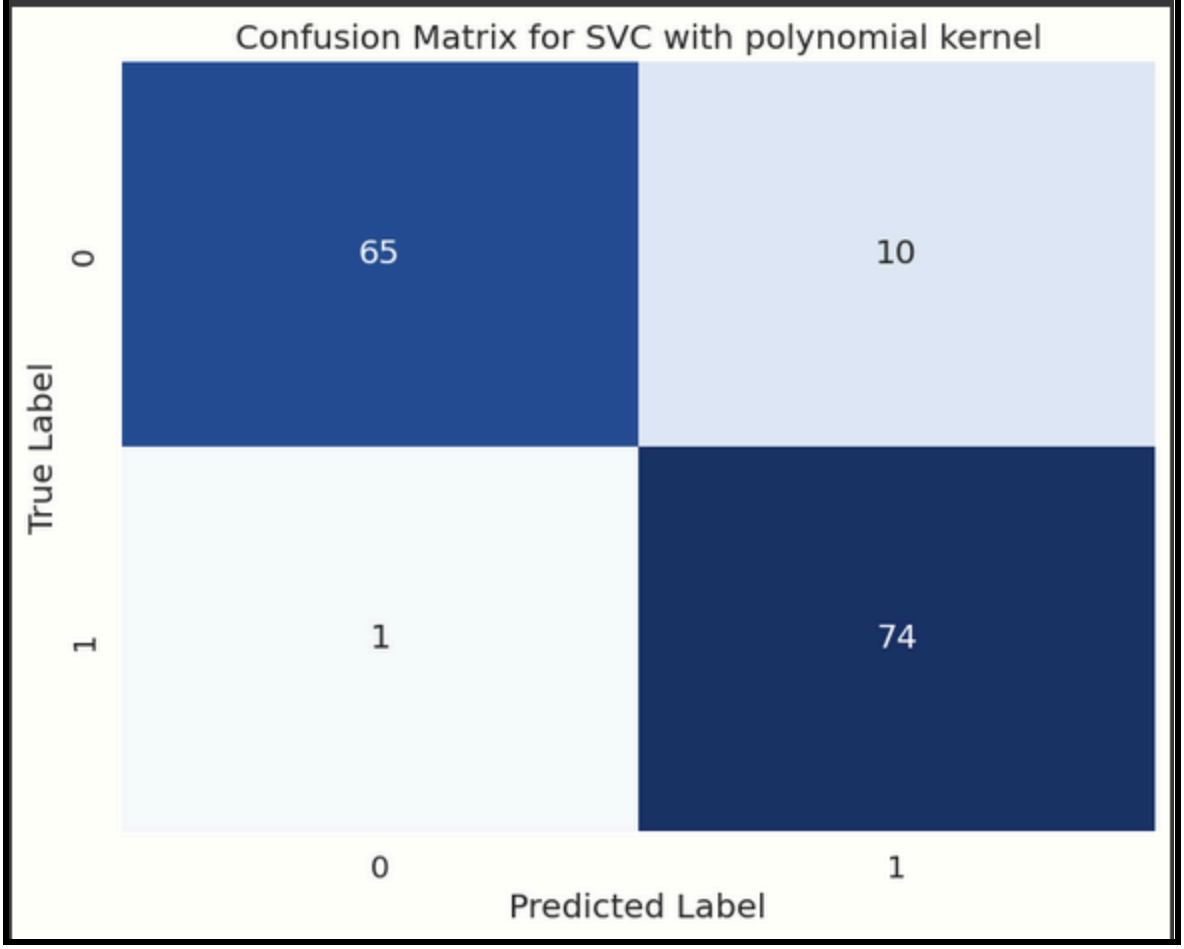
Task 2(b): Implementing SVM Models with Different Kernels

This work involves implementing SVM models with three distinct kernels: Linear, Polynomial, and Radial Basis Function (RBF). The decision boundaries generated by each kernel on the synthetic dataset are graphed and examined to get insight into their adaptability and capacity to generalize.

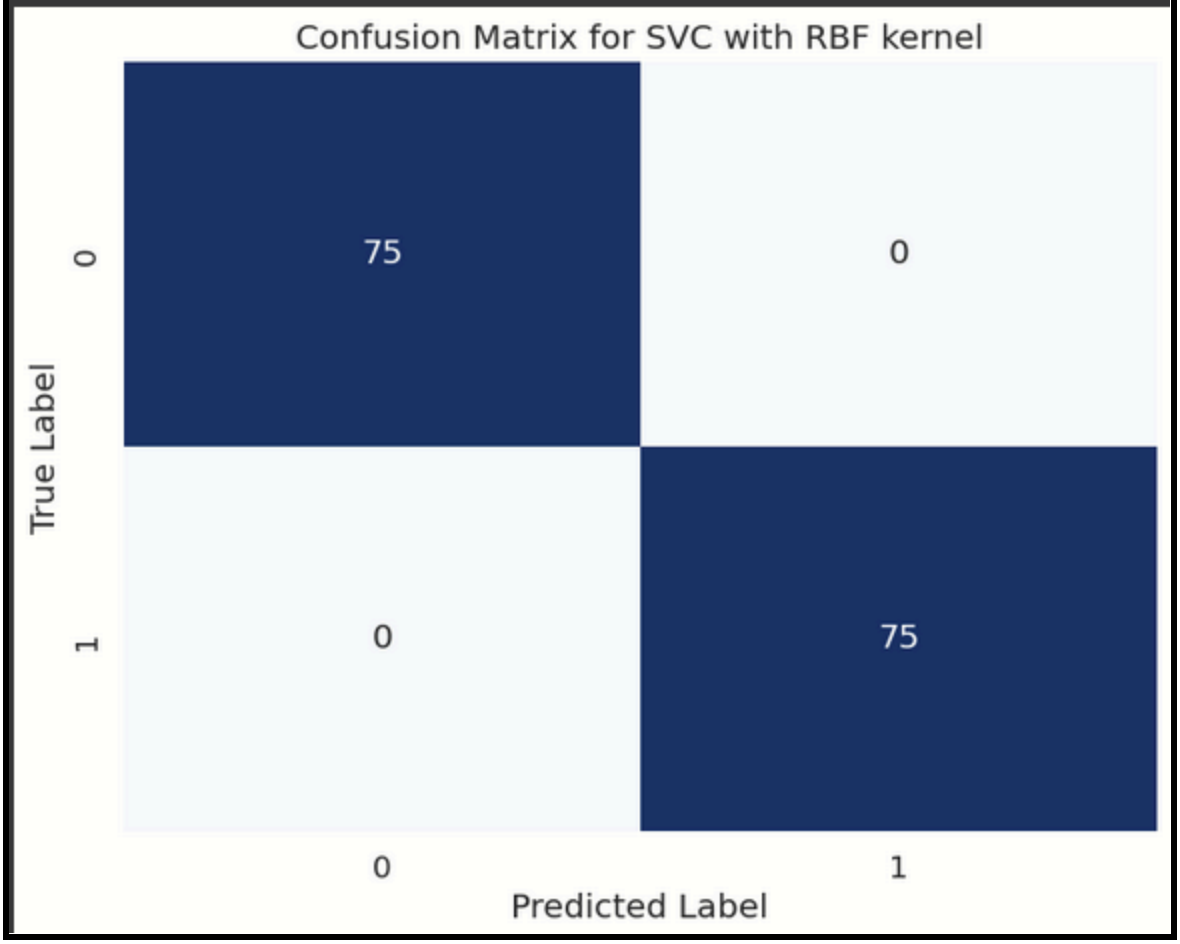
```
Accuracy for SVC with linear kernel: 0.89
Training Accuracy: 0.8771428571428571
Classification Report for SVC with linear kernel:
              precision    recall  f1-score   support

           0       0.89      0.88      0.89        75
           1       0.88      0.89      0.89        75

    accuracy                           0.89       150
   macro avg       0.89      0.89      0.89       150
weighted avg       0.89      0.89      0.89       150
```

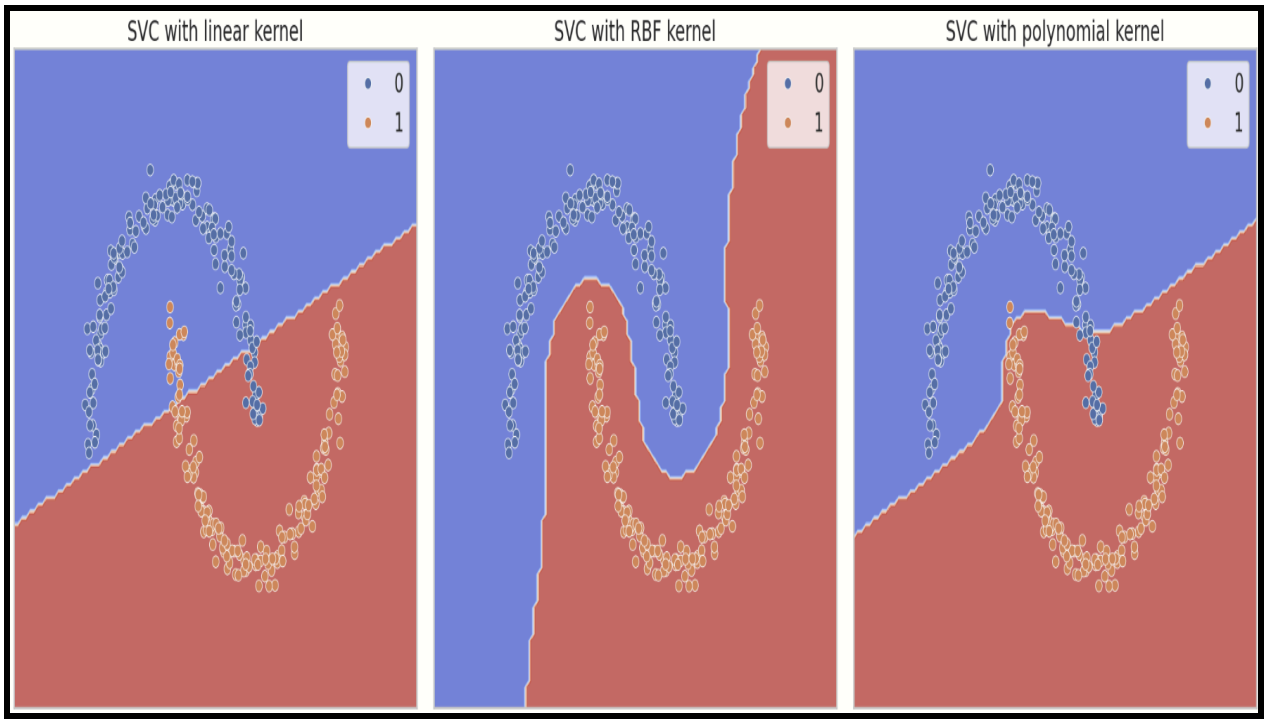


```
Accuracy for SVC with polynomial kernel: 0.93
Training Accuracy: 0.9142857142857143
Classification Report for SVC with polynomial kernel:
              precision    recall  f1-score   support

           0       0.98      0.87      0.92        75
           1       0.88      0.99      0.93        75

    accuracy                           0.93       150
   macro avg       0.93      0.93      0.93       150
weighted avg       0.93      0.93      0.93       150
```



```
Accuracy for SVC with RBF kernel: 1.00
Training Accuracy: 1.0
Classification Report for SVC with RBF kernel:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        75
           1       1.00      1.00      1.00        75

    accuracy                           1.00       150
   macro avg       1.00      1.00      1.00       150
weighted avg       1.00      1.00      1.00       150
```





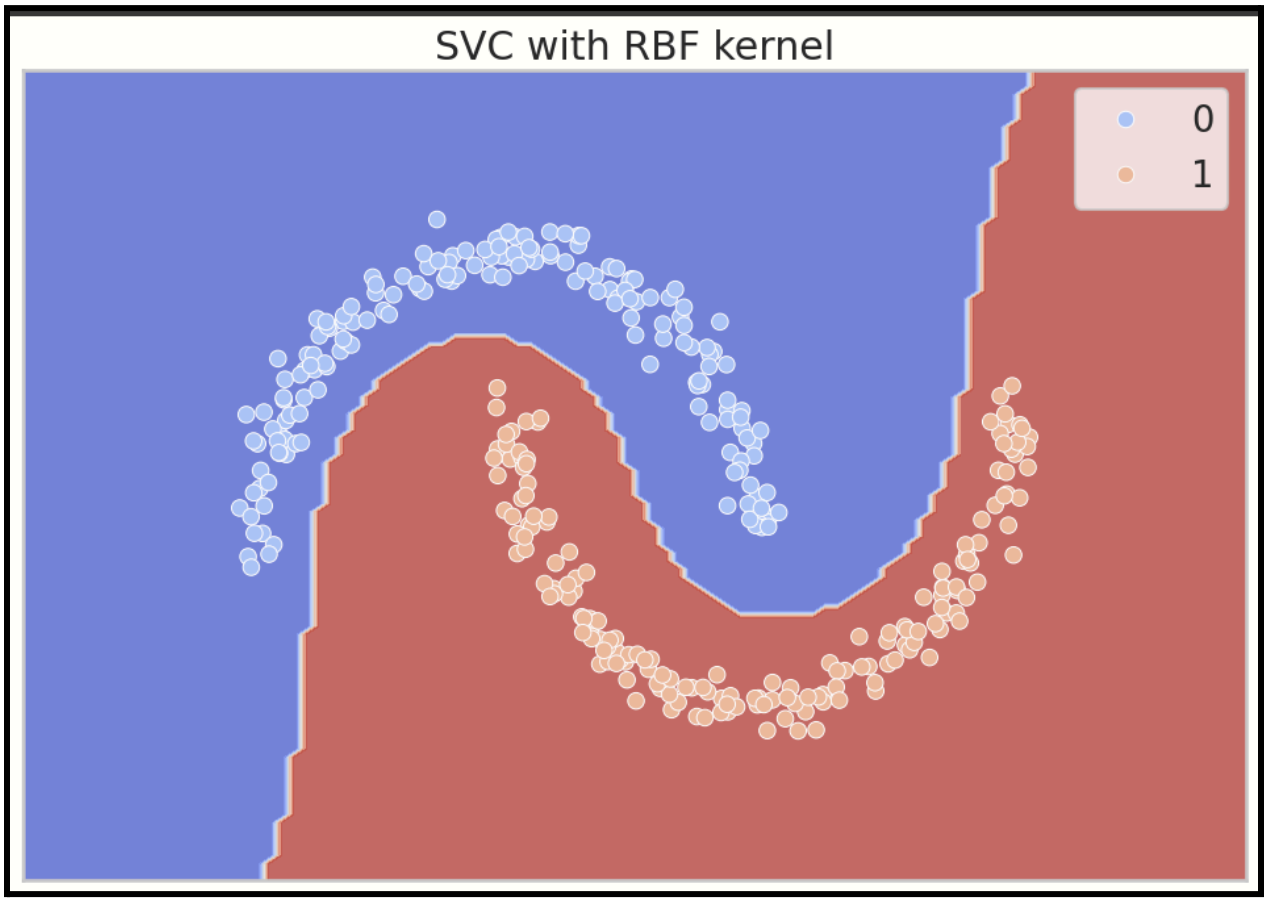
Task 2(c): Hyperparameter Tuning for RBF Kernel SVM

Task 2(c) is centered around optimizing the hyperparameters of the RBF kernel SVM model. Grid search isutilized to discover the ideal values of the gamma and C parameters, with the goal of improving the performance of the model.

```
{'C': 100, 'gamma': 0.1}
```

Task 2(d): Plotting Decision Boundary for Optimized RBF Kernel SVM

The ultimate objective is creating a visual representation of the decision boundary for the Support Vector Machine (SVM) model with a Radial Basis Function (RBF) kernel, using hyperparameters that have been improved. An analysis is conducted on the influence of specific gamma and C values on the performance of the model and the shape of its decision boundary. This analysis provides valuable insights into the importance of adjusting hyperparameters.



Conclusion

To summarize, this report has conducted a thorough investigation of Support Vector Machines, encompassing the analysis of decision boundaries using various kernels and the optimization of hyperparameters. By conducting methodical experiments and analysis, we have acquired useful knowledge about the behavior of SVM models and their impact on classification tasks. The results emphasize the significance of comprehending kernel functions and fine-tuning hyperparameters to attain the best possible performance of Support Vector Machines (SVM) in practical scenarios.