PRML PA-5

K-Means Clustering | Support Vector Machines

Colab Link

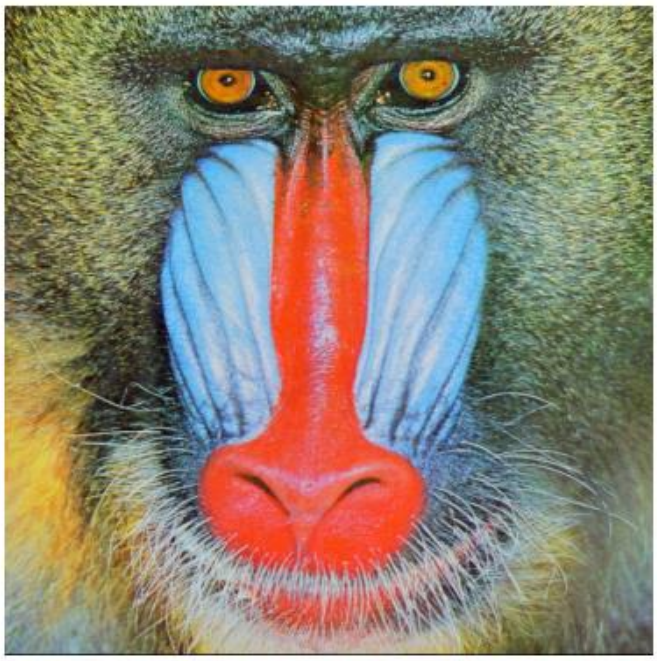
Rhythm Patni (B22CS043)

Question 1 : K-Means Clustering

Given an RGB image, we have to compress the image using K-means Clustering.

Task 1 : Implementing computeCentroid Function

The computeCentroid() function takes ‘n’ 3 dimensional data points as input and returns the mean vector of the data. The input data points are essentially the pixels of the image with the three features as the Red Shade, Green Shade and Blue shade of that pixel. Here's the original image.



Task 2 : Implementation of K-means Clustering.

The function mykmeans() is essentially the implementation of the k-means clustering algorithm. It takes the image, number of clusters, and number of epochs as the input and returns the cluster centers generated by the k-means clustering algorithm.

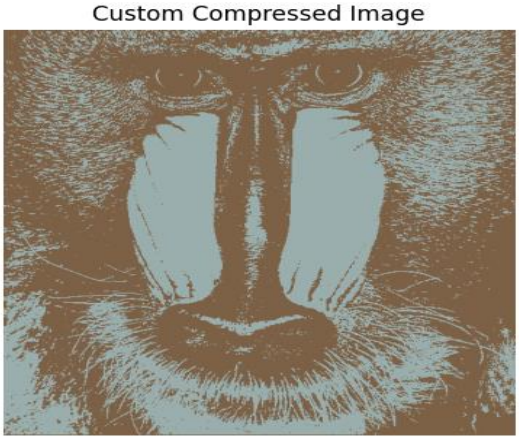
Task 3 : Result Analysis and Comparison with sk-learn library.

Here are the results of image compression,values of k ranging from 1-5 and also k = 10.

Custom Compressed Image

Compressed Image (SK-Learn) (1 colors)

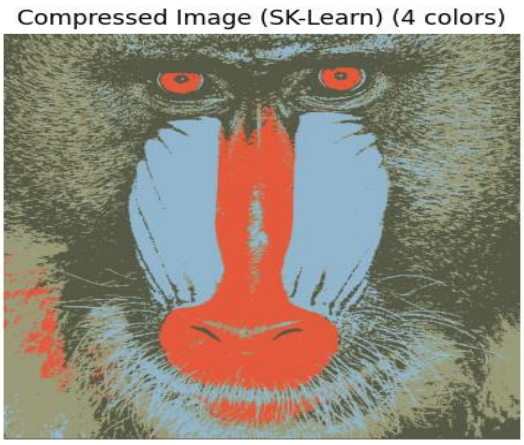
Custom Compressed Image

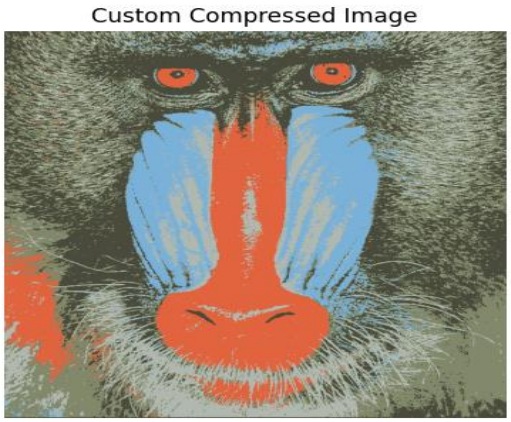


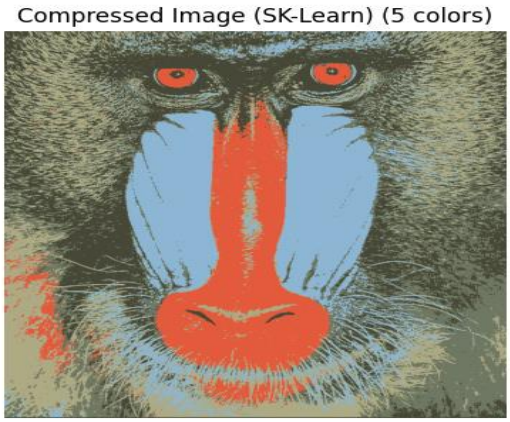
Compressed Image (SK-Learn) (2 colors)

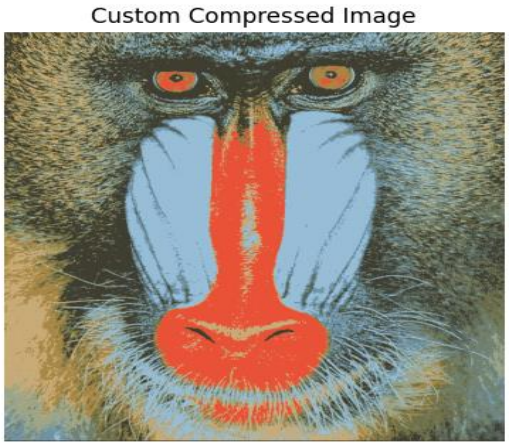
Custom Compressed Image

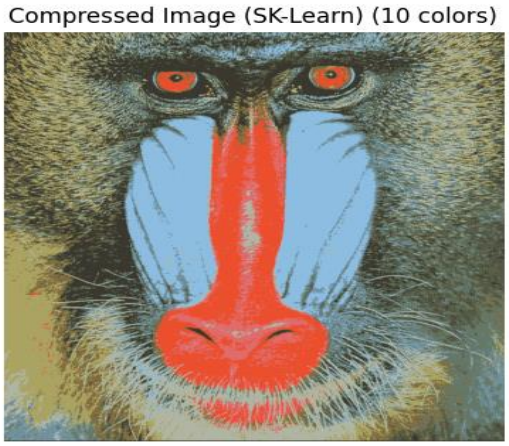
Compressed Image (SK-Learn) (3 colors)











Here are some Observations:

For all values of k > 4, the change in the image quality is insignificant and the results of custom code and the sk learn library are nearly the same.

The given image has essentially 4 colors, so, if we take more clusters, the difference will be insignificant.

Task 4: Spatial Coherence

We know that the pixels which are closer to each other are more likely to belong to the same clusters. To incorporate this into our implementation, we modify the implementation of the distance calculating functions. Now the contribution of both the color distance and the spatial distance will be added to the distance term. This change has been made in the code.

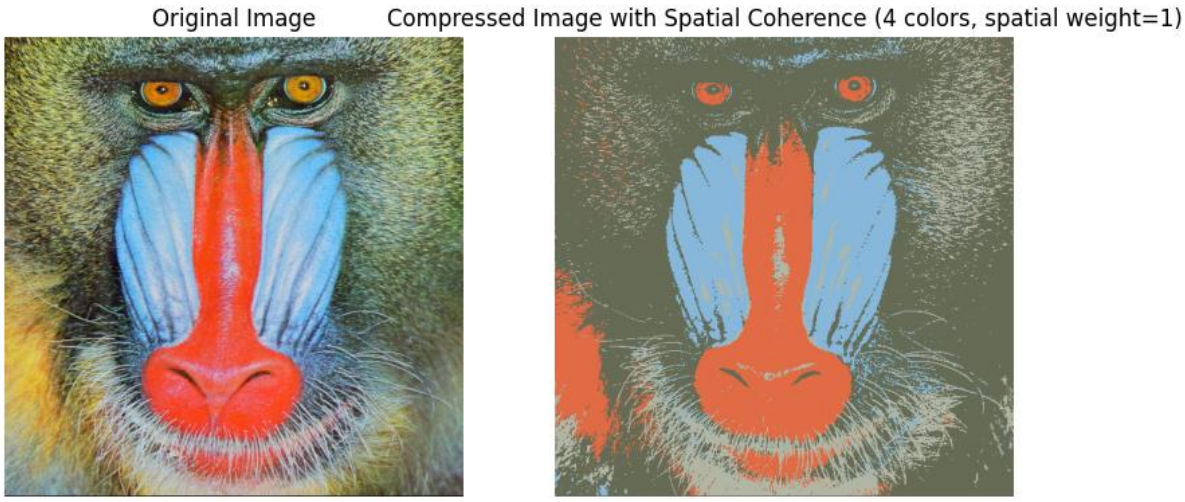
Here are some observations:

After incorporating spatial coherence into the image compression algorithm using k-means clustering, several observations can be made:

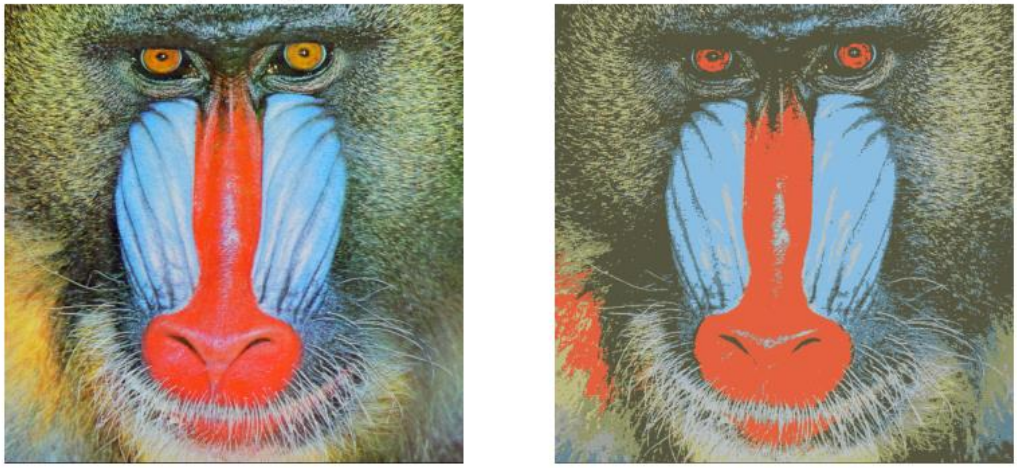
Preservation of Local Structures: Spatial coherence helps in maintaining local structures in the image. Pixels that are spatially close in the original image are more likely to be assigned to the same cluster. This results in better preservation of edges, textures, and other local details.

Controlled Compression: The degree of spatial coherence can be adjusted by varying the spatial weight parameter. Increasing the spatial weight puts more emphasis on spatial proximity, leading to stronger spatial coherence but potentially higher computational cost. Conversely, decreasing the spatial weight may result in less spatially coherent compression but faster processing.

Below is the comparison of the compressed images and given images for different values of k



Original Image Compressed Image with Spatial Coherence (8 colors, spatial weight=1)



Original Image Compressed Image with Spatial Coherence (12 colors, spatial weight=1)

Original Image Compressed Image with Spatial Coherence (16 colors, spatial weight=1)

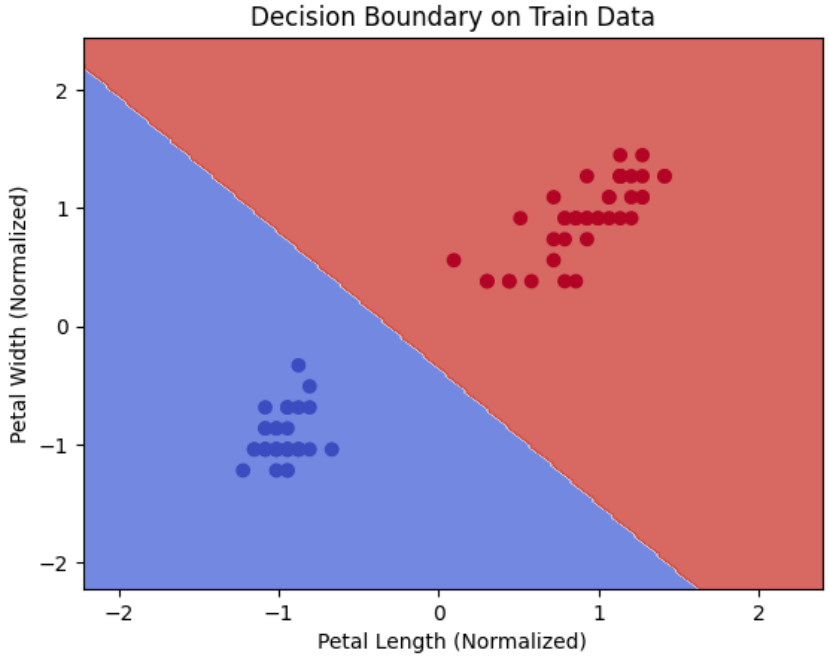
Question 2: Support Vector Machines

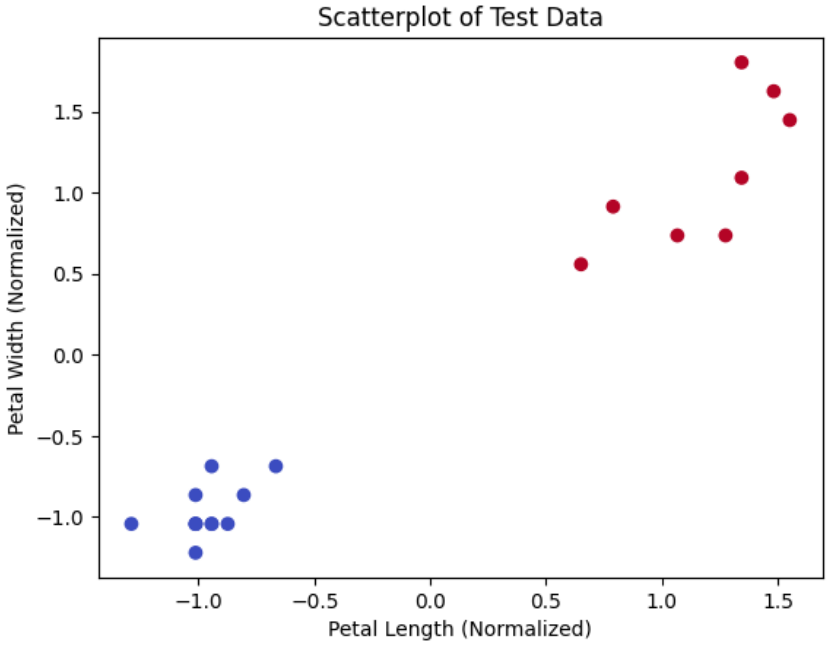
Task 1: Loading Dataset and Pre - Processing

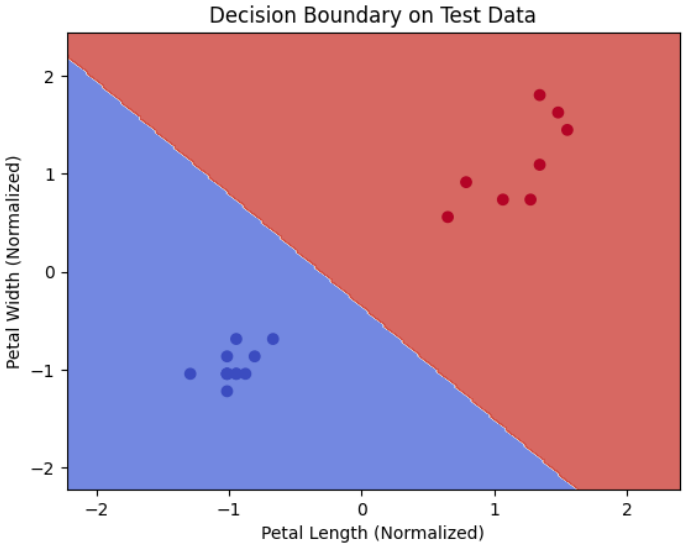
The iris dataset was loaded and pre- processed as mentioned in the task.

Task 2: Training Linear SVC

On the training dataset, the results obtained are shown as follows:

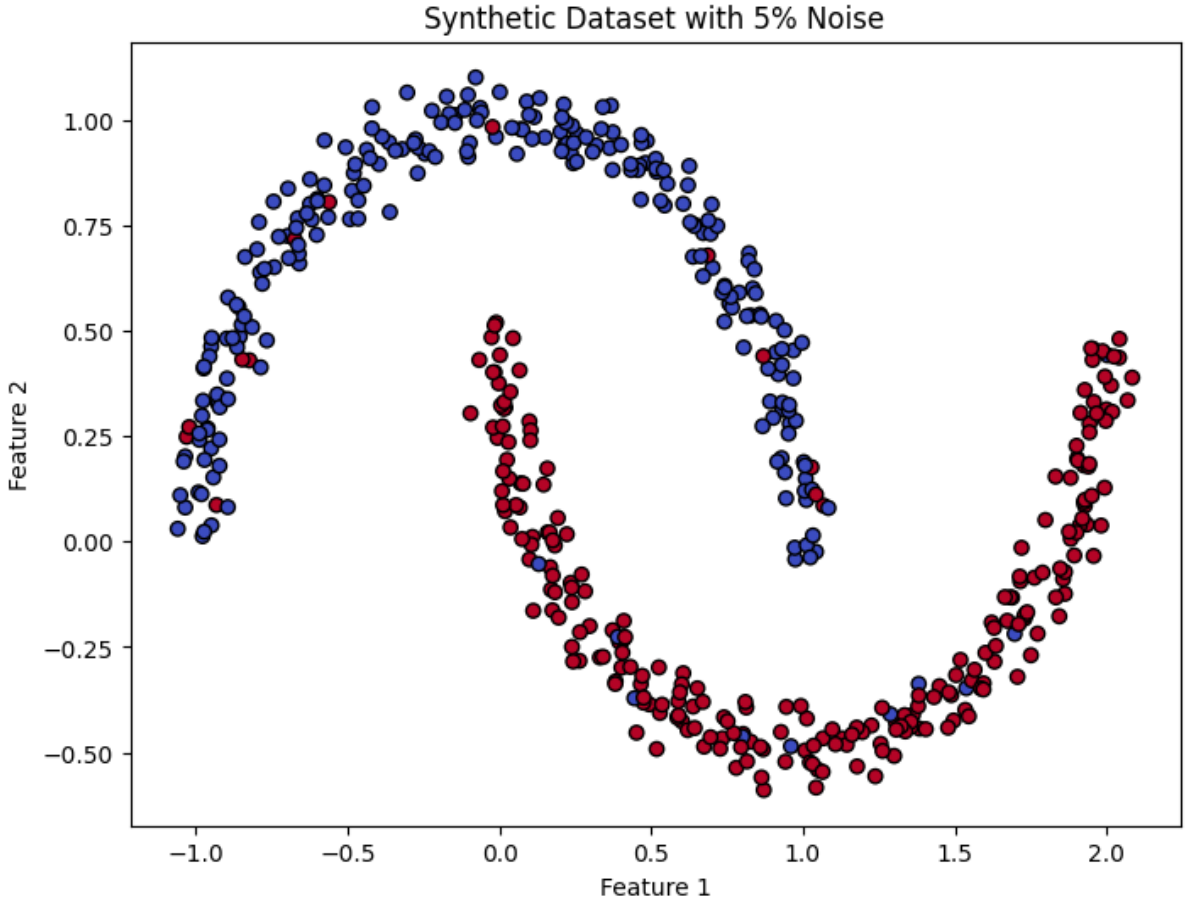






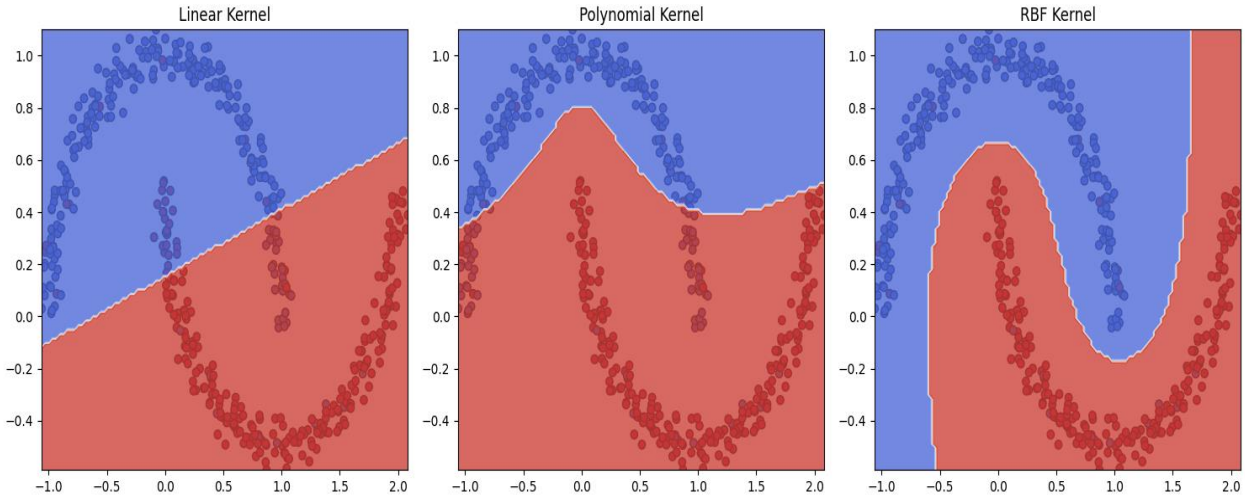
Task 3: Generating a synthetic moon dataset.

Here is the dataset generated as mentioned in the task:



Task 4: Training SVM models on the moon dataset.

Here are the results obtained on training SVM models on the dataset using 3 different Kernels, namely Linear, Polynomial, RBF.

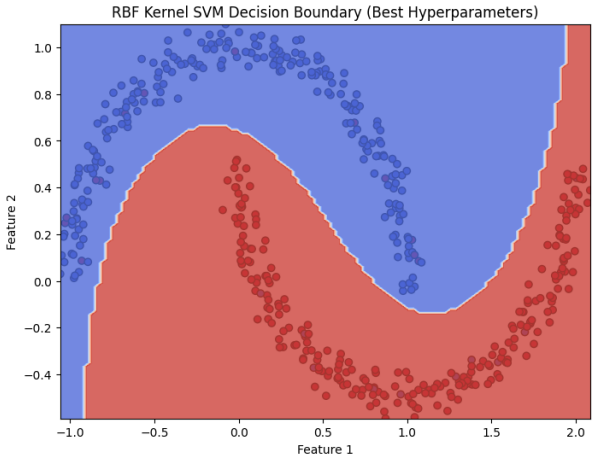


Here are some observations:

1. The dataset is not linearly separable so the linear kernel is not suited for this dataset since it misclassifies a lot of points.
2. The polynomial kernel does a better job at classifying the points, yet it is not perfect.
3. The RBF kernel performs classification perfectly, since it supports highly flexible decision boundaries, it does the job perfectly.

Task 5: Hyperparameter Tuning for RBF model:

I used Grid Search to perform Hyperparameter tuning, using 4 different sets of parameters for C and gamma. The GridSearch Function found the best parameter, which was then plotted as shown. The best hyperparameters were found to be C = 100 and gamma = 0.1



Here are the impacts of the hyper-parameters:

Gamma: Low values indicate "far," while high values indicate "close." Gamma describes the extent to which the influence of a single training example extends. Gamma values can cause underfitting or overfitting depending on how complicated the decision boundaries are; low values can cause underfitting. Gamma regulates the Gaussian kernel's width in relation to the RBF kernel. The Gaussian distribution is larger and the decision boundaries are smoother when the gamma is less.
C: The regularisation parameter, C regulates the trade-off between accurately identifying training points and having a smooth decision boundary. A bigger value of C produces a sharper margin that penalises misclassifications more severely and may result in narrower decision boundaries. A smaller value of C indicates a softer margin that permits certain misclassifications in favour of wider decision limits.