PRML ASSIGNMENT - 5

Name : Aditya Sahani
Roll No : B22CS003
Google Colab :
https://colab.research.google.com/drive/1qkenQ_flyXP7Hq6ymGLbRkjkqilXM-jJ?usp=sharing

K-Means Clustering

Task A

In this task, the necessary libraries are imported, and an image (test.png) is loaded using OpenCV. The image is then converted from BGR to RGB format. The dimensions of the image are extracted, and the pixels of the image are stored in an array named rgb_data.
computeCentroid() : function is written to calculate the centroid of the pixels

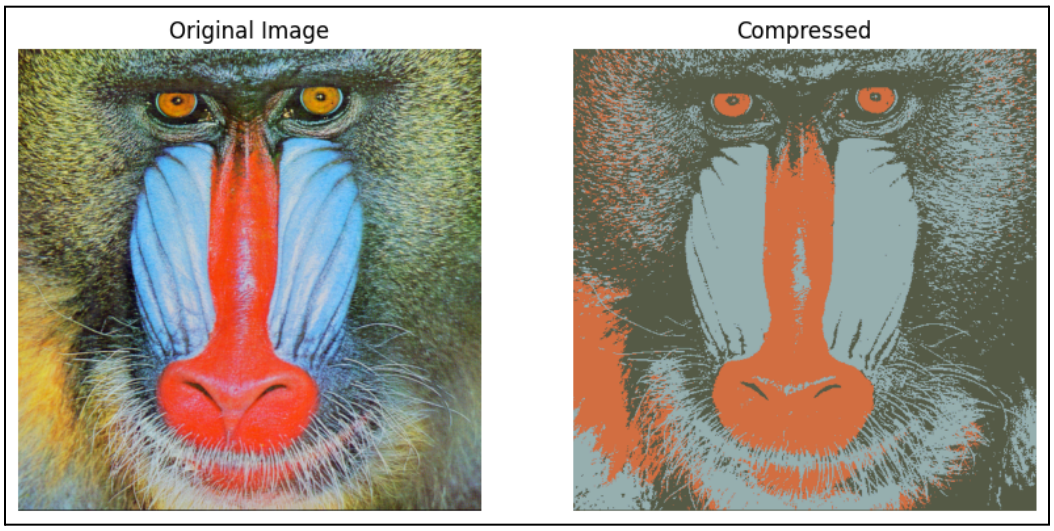
Task B

A custom implementation of the K-Means algorithm is created. The functions computeCentroid, group_clusters, and new_centroids are defined to compute centroids, assign data points to clusters, and update centroids, respectively. The main function mykmeans runs the K-Means algorithm for a specified number of clusters (no_of_clusters) and epochs (epochs).

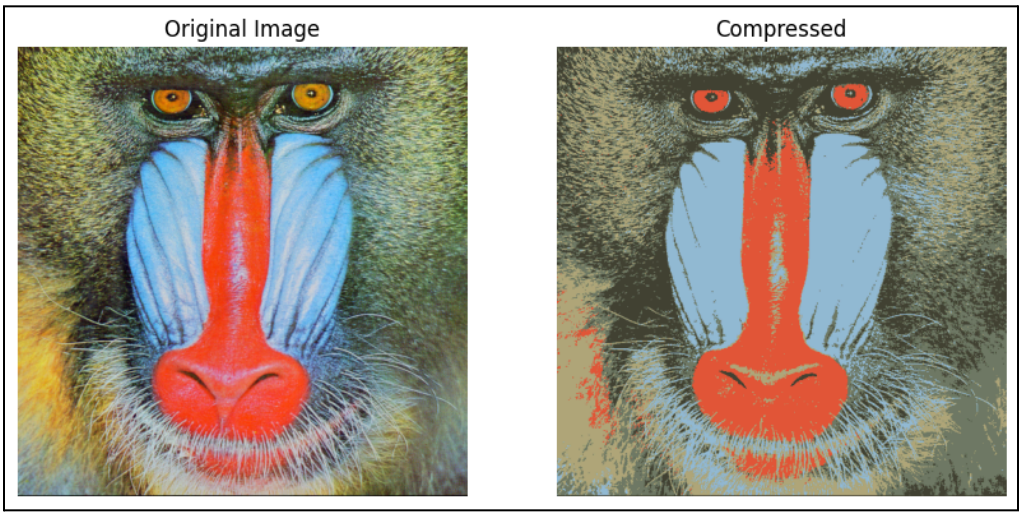
Task C

The K-Means algorithm is applied to the image data with different values of k (number of clusters). For each value of k, the image is compressed using K-Means clustering, and the original and compressed images are displayed side by side. Additionally, the Within-Cluster Sum of Squares (WCSS) is calculated for each value of k, and an elbow plot is generated to determine the optimal number of clusters.

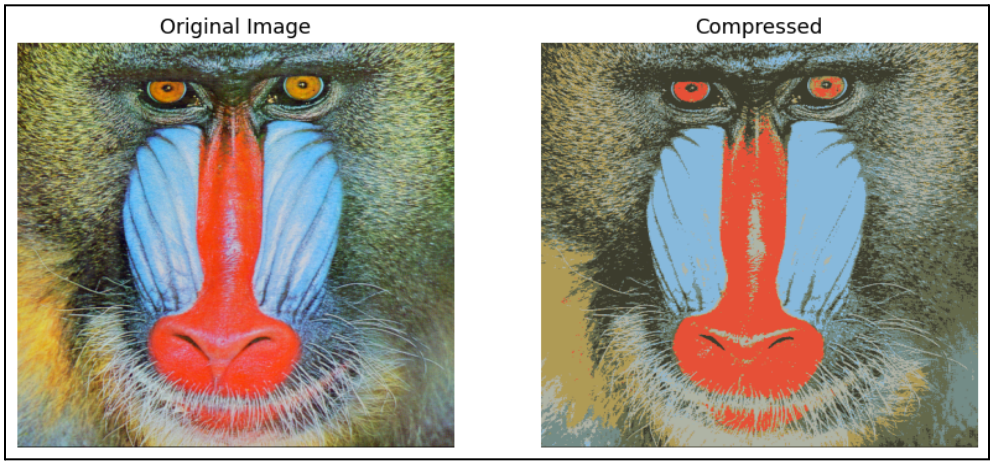
Number of Clusters = 3



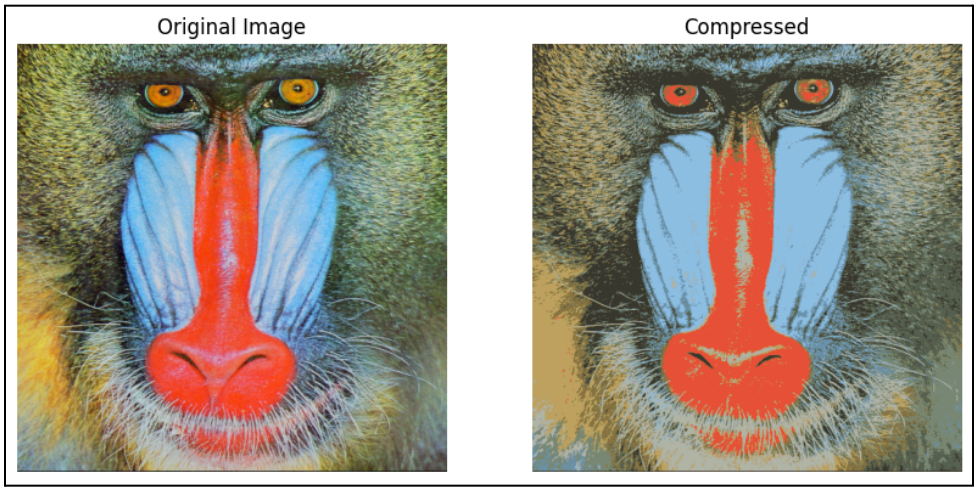
Number of Clusters = 5



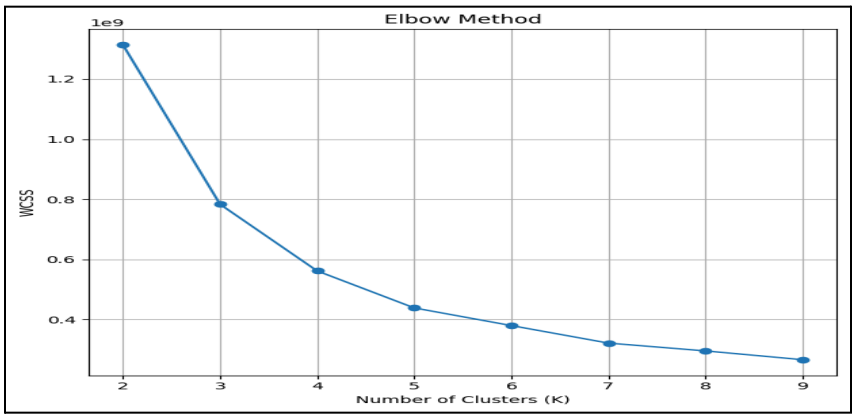
Number of Clusters = 7



Number of Clusters = 9



Elbow method is used to find the optimum number of clusters.

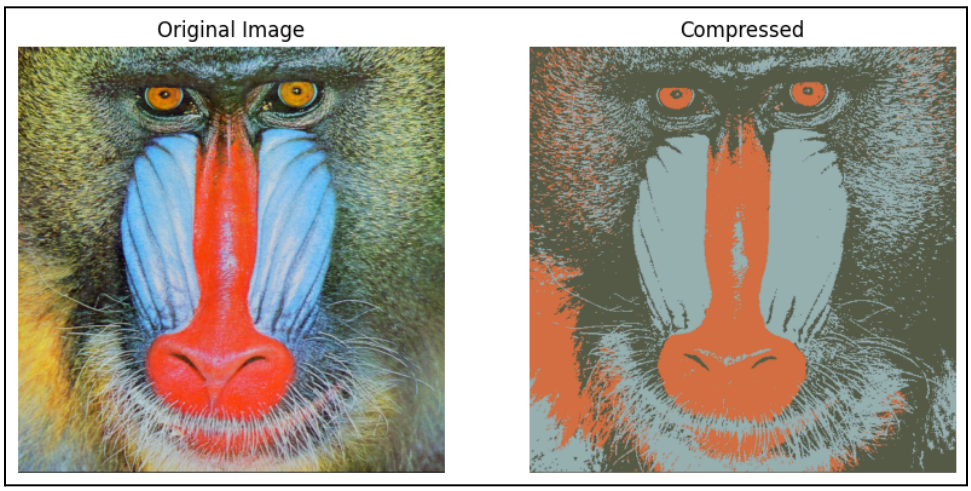


Optimum Number of Clusters **should be 6** because graphs tend to form the elbow or flatten at 6.

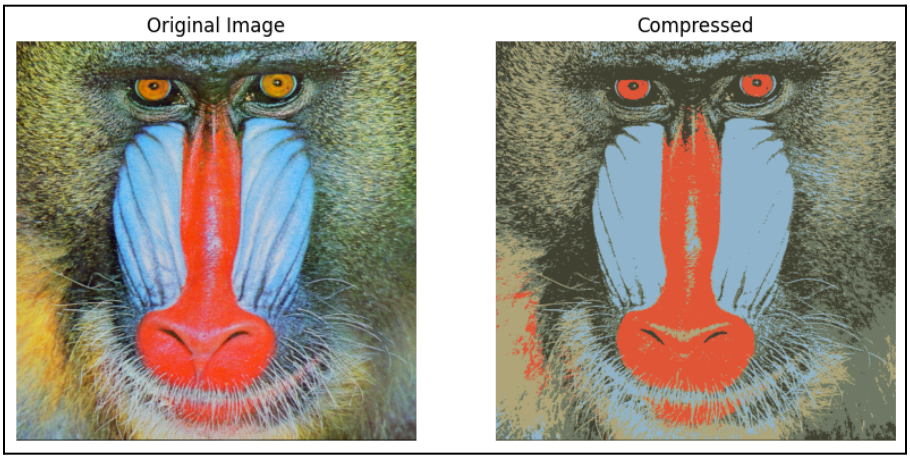
Task D

The K-Means algorithm is implemented using the KMeans class from the sklearn.cluster module. The algorithm is applied to the image data with different values of k, and the compressed images are displayed alongside the original images. An inertia plot is generated to determine the optimal number of clusters.

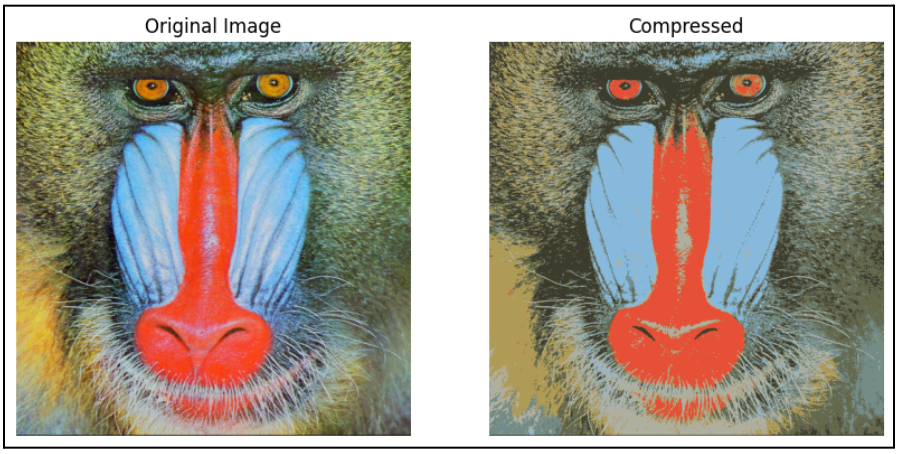
Number of Clusters = 3



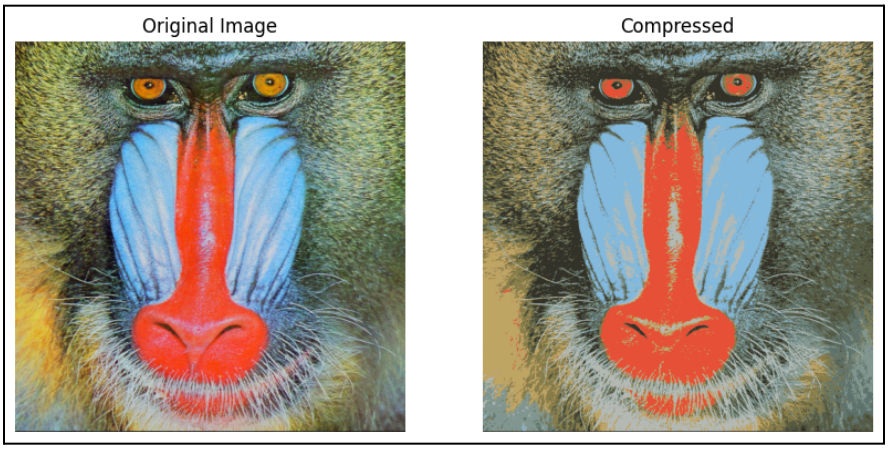
Number of Clusters = 5



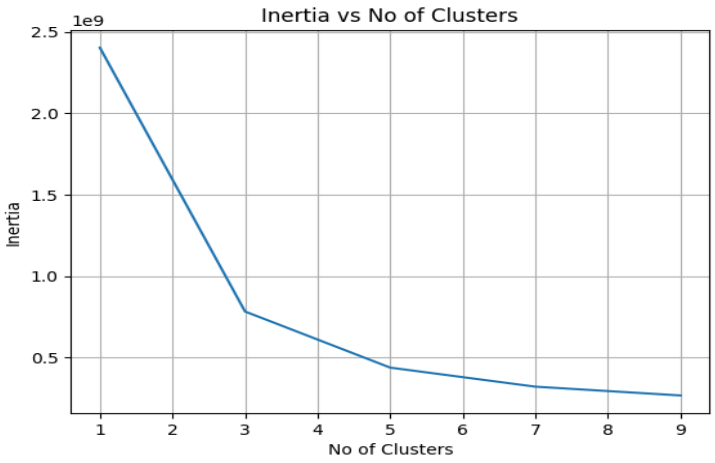
Number of Clusters = 7



Number of Clusters = 9



Elbow method is used to find the optimum number of clusters.

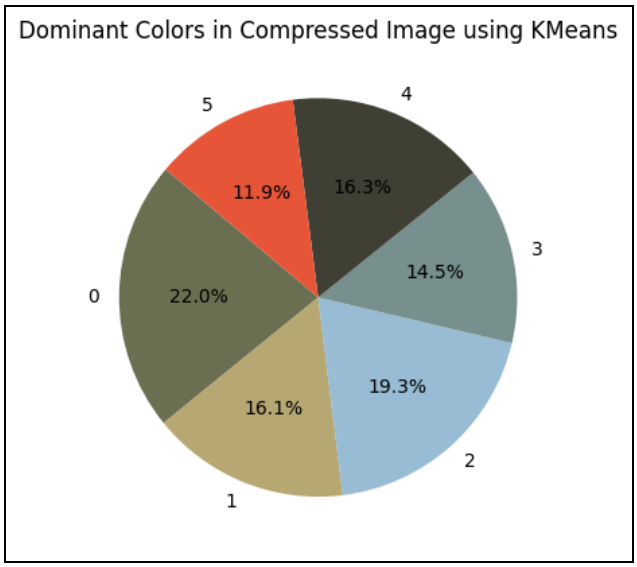


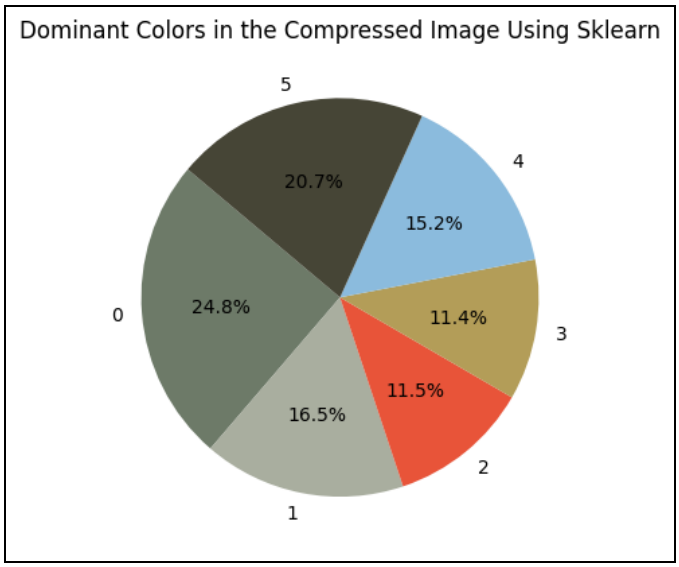
Optimum Number of Clusters **should be 6** because graphs tend to form the elbow or flatten at 6 using sklearn.

Comparison :

USING SKLEARN

USING K-MEANS SCRATCH





From this pie chart, one can clearly see that the **percentage of colors differ between compressed images using mykmeans and sklearn**, hence there will be little difference between the two compressed images.

```
array([[ 69.95916929,  69.21887632,  54.06656777],
       [108.37745106, 121.37527577, 103.58612444],
       [178.57155405, 156.53388514,  86.48226351],
       [139.37256664, 186.63626834, 220.84735949],
       [168.58771124, 173.25500551, 157.8771813 ],
       [232.18120961,  83.57945319,  57.60258492]])
```

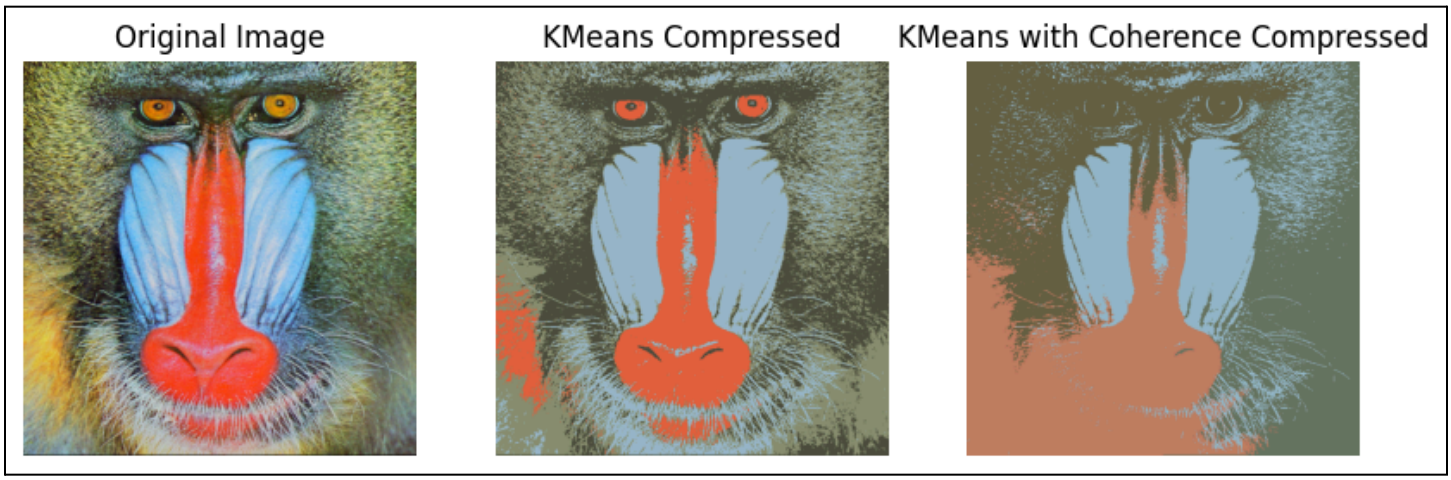
```
array([[169.26305684, 173.11451167, 157.0357249 ],
       [108.24730747, 121.62999735, 104.35993391],
       [177.55531602, 155.98525134,  84.71436882],
       [139.58801535, 186.63278445, 220.62144361],
       [232.31176568,  83.60503061,  58.01303988],
       [ 70.21716197,  69.47703589,  54.19130323]])
```

There is **no significant difference in the colors because the centroids for k = 6 are almost the same.**

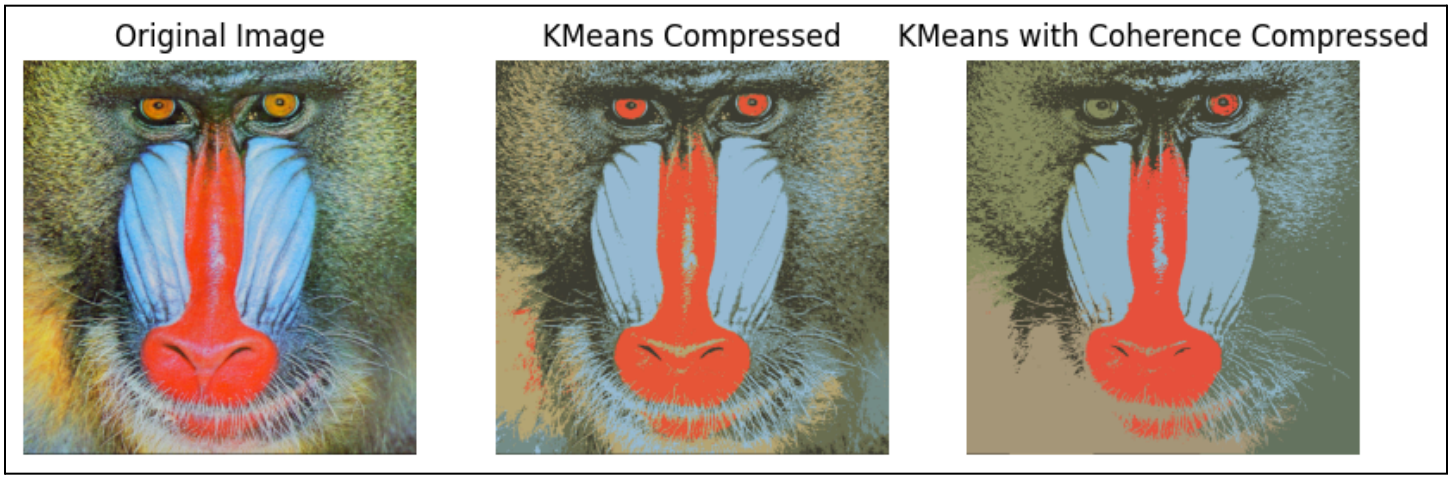
Task E

A new implementation of K-Means with spatial coherence is introduced. This version of K-Means considers both color similarity and spatial proximity when clustering pixels. The algorithm is applied to the image data with different values of k, and the compressed images are displayed alongside the original images. A comparison is made between the results of standard K-Means clustering and K-Means clustering with spatial coherence.

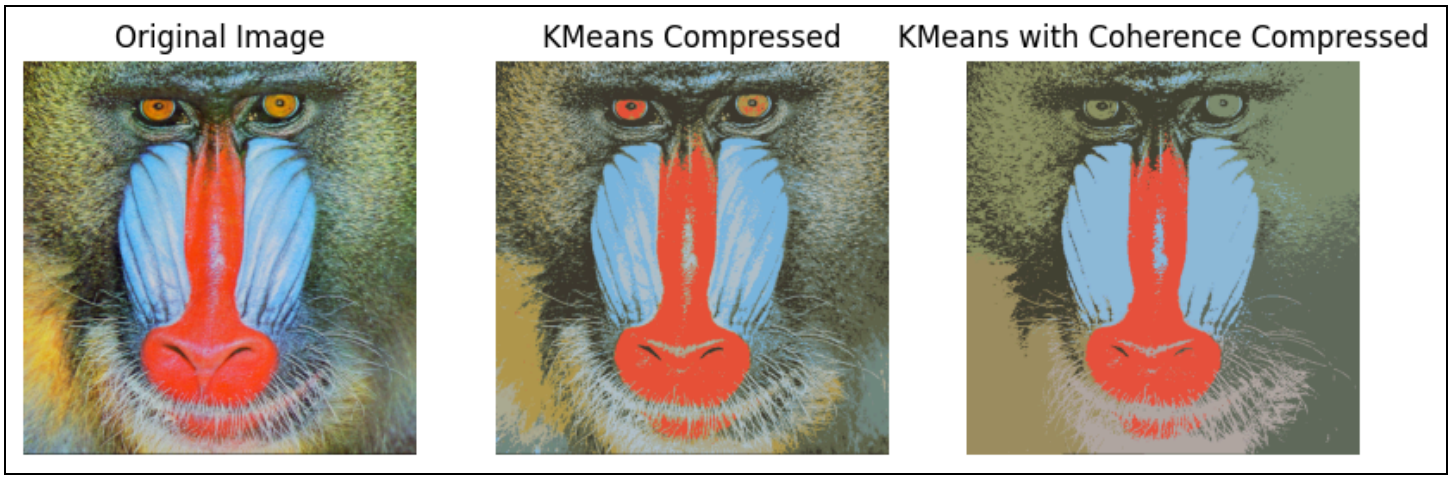
Number of Clusters = 4



Number of Clusters = 6



Number of Clusters = 8



From the above pictures, we can see that when we are doing compression using spatial coherence , **the image is dominantly affected by its neighboring pixels.** Hence, we are not getting the proper compression.

SUPPORT VECTOR MACHINE

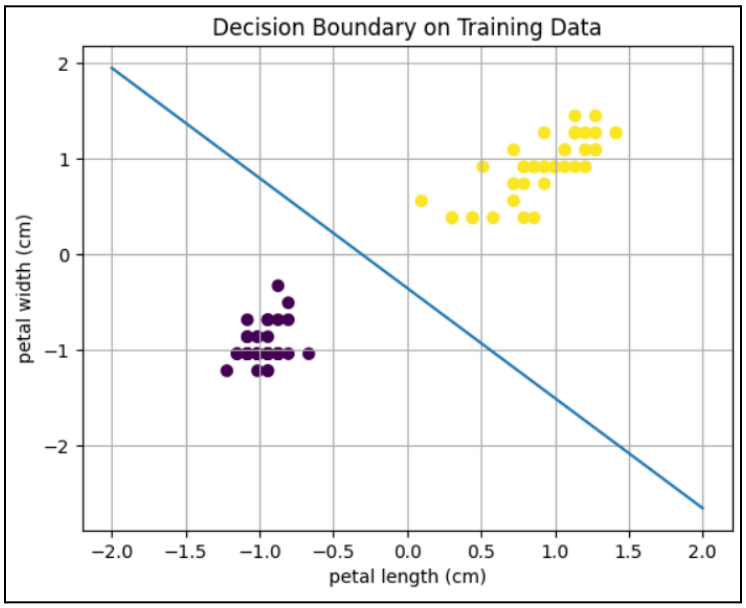
Task - 1(a): Support Vector Machine (SVM) with Iris Dataset

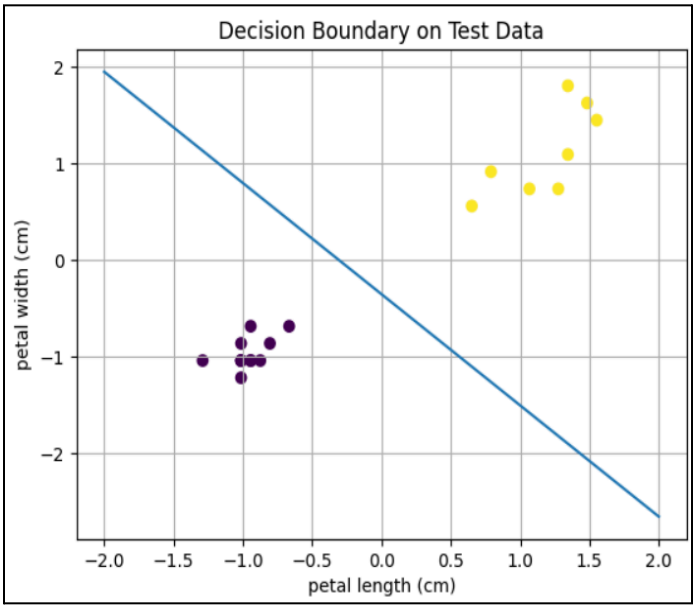
- Importing Necessary Libraries: The code begins by importing the required libraries such as NumPy, pandas, and matplotlib.
- Loading and Preprocessing the Dataset: The Iris dataset is loaded using scikit-learn's datasets module. The features (X) and target (y) are extracted from the dataset. The shape of X and y is printed, and both X and y are converted to NumPy arrays. The data is then concatenated into a single array and converted to a pandas DataFrame for further analysis.
  Shape of X : (150, 4)
  Shape of y: (150, 1)

- Data Exploration: The DataFrame information is printed to inspect the dataset's structure and data types. Additionally, a subset of the dataset containing only two classes (target 0 and target 1) is created and displayed.
- Feature Selection: Two features, "petal length (cm)" and "petal width (cm)", are selected for modeling, and a new DataFrame containing only these features and the target variable is created, as per the question.
- Data Scaling: The features are standardized using the **StandardScaler** to ensure that each feature has a mean of 0 and a standard deviation of 1.
- Train-Test Split: The dataset is split into training and testing sets using an 80:20 ratio.

Task - 1(b): Training Linear SVM and Visualizing Decision Boundary

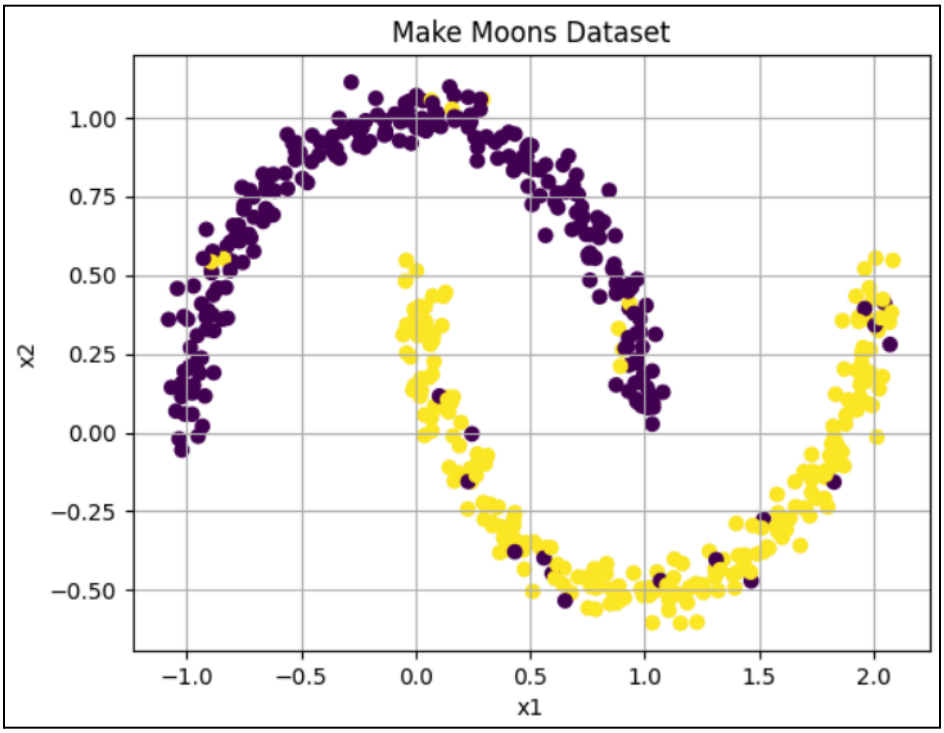
- Linear SVM Training: A Linear Support Vector Classifier (LinearSVC) is initialized and trained on the training data.
- Prediction and Evaluation: The trained model is used to make predictions on the test data, and the accuracy of the model is computed.
  Accuracy : 100 %
- Decision Boundary Visualization: The decision boundary of the trained linear SVM is plotted on both the training and test data.





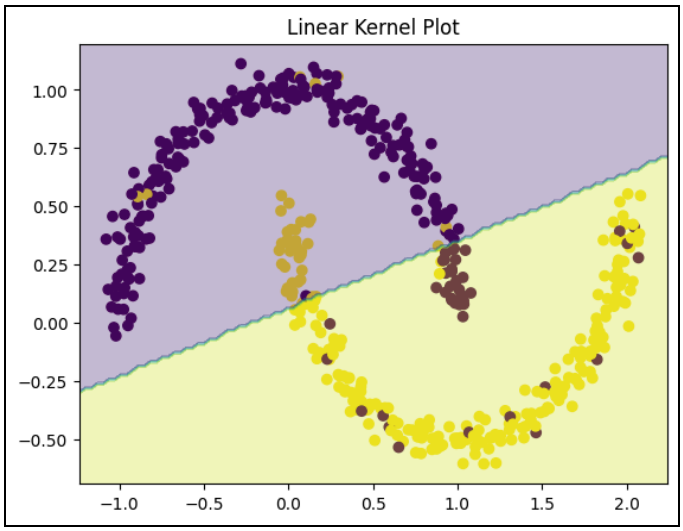
Task - 2(a): Generating Moon-Shaped Dataset

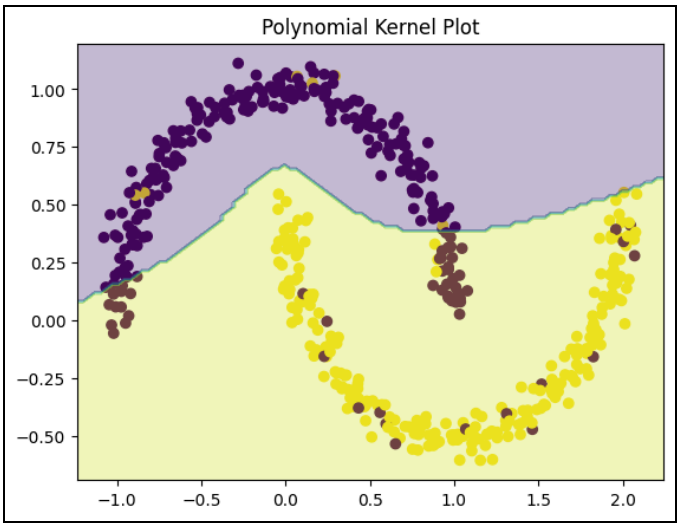
- Generating Dataset: A moon-shaped dataset with 500 samples and 5% noise is generated using the make_moons() function from scikit-learn.
- Adding Misclassified Points: 5% of the points are randomly selected and their labels are flipped to introduce misclassification.
- Dataset Visualization: The generated dataset is visualized using a scatter plot.

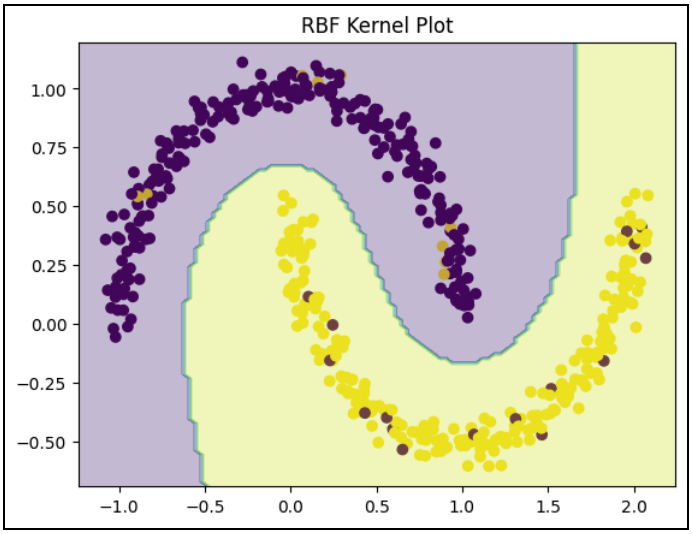


Task - 2(b): Training SVMs with Different Kernels and Visualizing Decision Boundaries

- SVM Initialization and Training: Three SVM models with different kernels (linear, polynomial, and radial basis function - RBF) are initialized and trained on the generated dataset.
- Decision Boundary Plotting: Decision boundaries for each SVM model are plotted along with the dataset.







OBSERVATIONS ON THE DECISION BOUNDARY :

Linear Kernel:

- The decision boundary is a straight line, as the linear kernel constructs a linear decision boundary.
- It separates the two moon-shaped clusters with a straight line, resulting in a simple boundary.

Polynomial Kernel:

- The decision boundary is more complex and non-linear compared to the linear kernel.

- It captures the curvature of the moon-shaped clusters more accurately than the linear kernel.
- With a polynomial kernel of degree 3, the decision boundary becomes a curve.

RBF Kernel:

- The decision boundary is highly non-linear and flexible.
- It can adapt to the irregular shapes of the moon-shaped clusters effectively.
- The boundary contours closely follow the data distribution, providing a flexible separation.

Task - 2(c): Hyperparameter Tuning with GridSearchCV and RandomizedSearchCV

- GridSearchCV: Hyperparameter tuning is performed using GridSearchCV to find the best combination of parameters (C and gamma) for the RBF kernel SVM.
- RandomizedSearchCV: Hyperparameter tuning is performed using RandomizedSearchCV with a specified parameter distribution.
- Model Evaluation: The best models obtained from GridSearchCV and RandomizedSearchCV are evaluated on the test set, and their accuracies are printed.

GridSearchCV :
Best parameters: **{'C': 0.5, 'gamma': 0.7}**
Best cross-validation score: 0.95
Test set accuracy: 0.95

RandomizedSearchCV :
Best parameters: **{'C': 1.3292918943162166, 'gamma': 0.711447600934342}**
Best cross-validation score: 0.95
Test set accuracy: 0.95

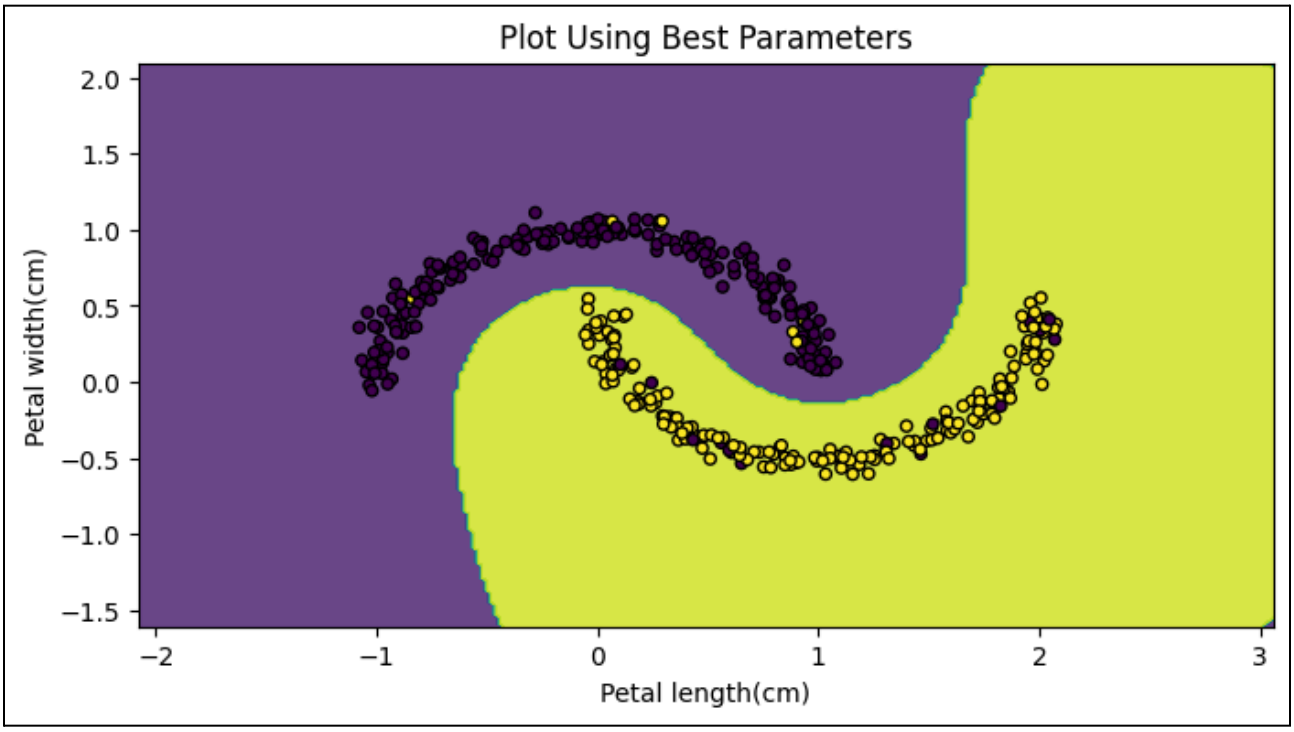
Task - 2(d): Plotting Decision Boundary with Best Parameters

- Plotting Decision Boundary: Decision boundaries for the best models obtained from GridSearchCV and RandomizedSearchCV are plotted along with the training data.

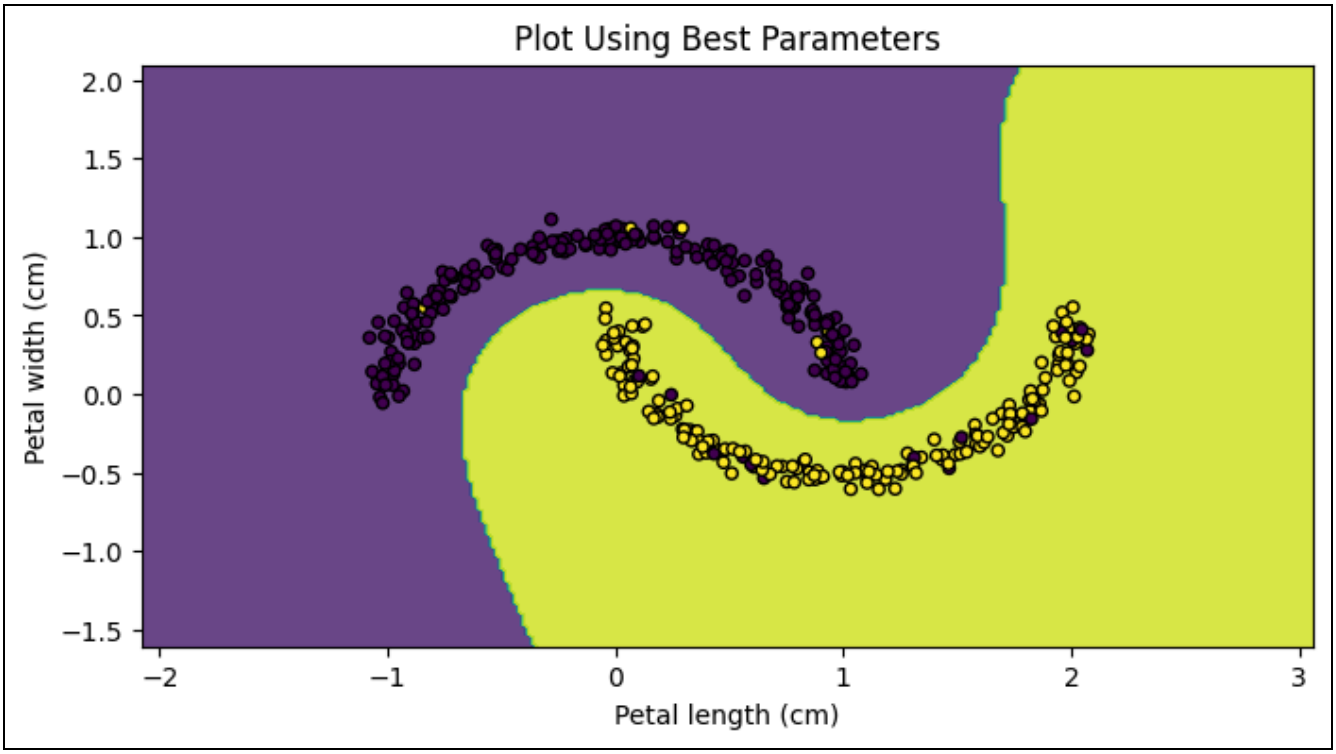
Impact of Hyperparameters :

- We analyze the effect of the selected gamma and C values on the model's performance and decision boundary.
- Higher values of gamma lead to a more complex decision boundary, which can result in overfitting.
- Larger values of C result in a narrower margin, potentially leading to overfitting if the data is noisy.

GridSearchCV:



RandomizedSearchCV:


Plot Using Best Parameters
Petal width (cm)
Petal length (cm)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1
2
3