Assignment 5:

Atharva Date - B22AI045

Problem 1

Introduction:

Image compression is a crucial task in image processing, aiming to reduce storage space while preserving visual quality. In this assignment, we explore K-means clustering for image compression, where each pixel is represented by the color of its nearest centroid.

(a) Implementation of computeCentroid Function:

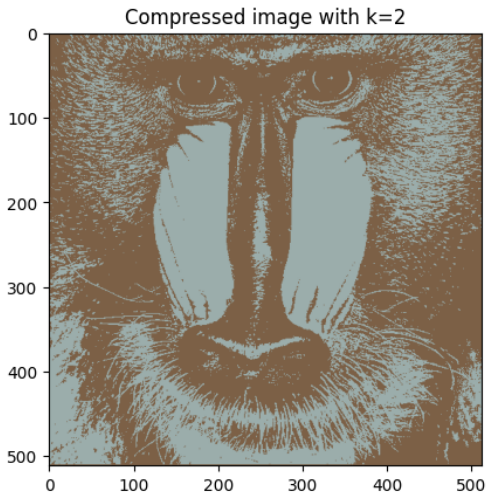
The computeCentroid function calculates the mean of n 3-dimensional features, crucial for computing cluster centroids in K-means.

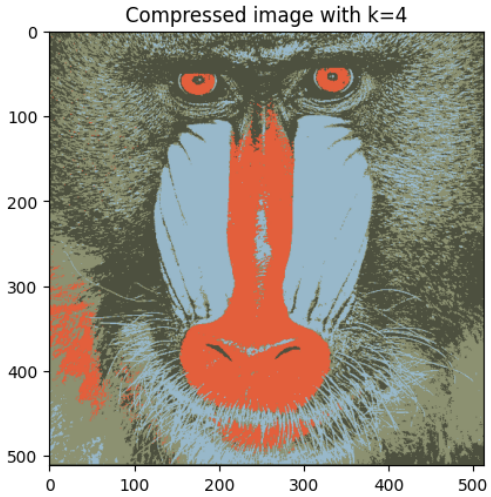
(b) Implementation of mykmeans Function:

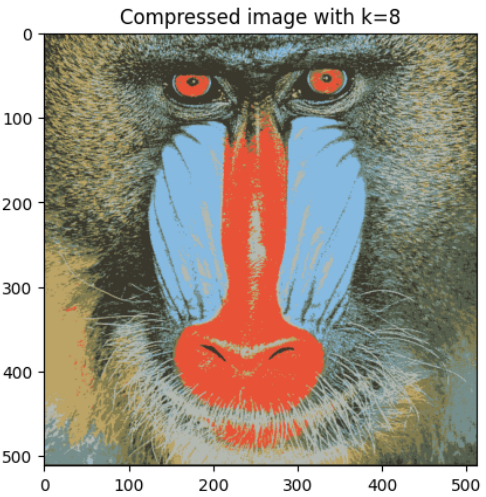
The mykmeans function is developed from scratch to perform K-means clustering on a data matrix X, iteratively updating centroids until convergence. Number of iterations was set to 100.

(c) Representing Pixels with Centroids:

Pixels in the image are represented by the nearest centroid's color, achieving image compression.

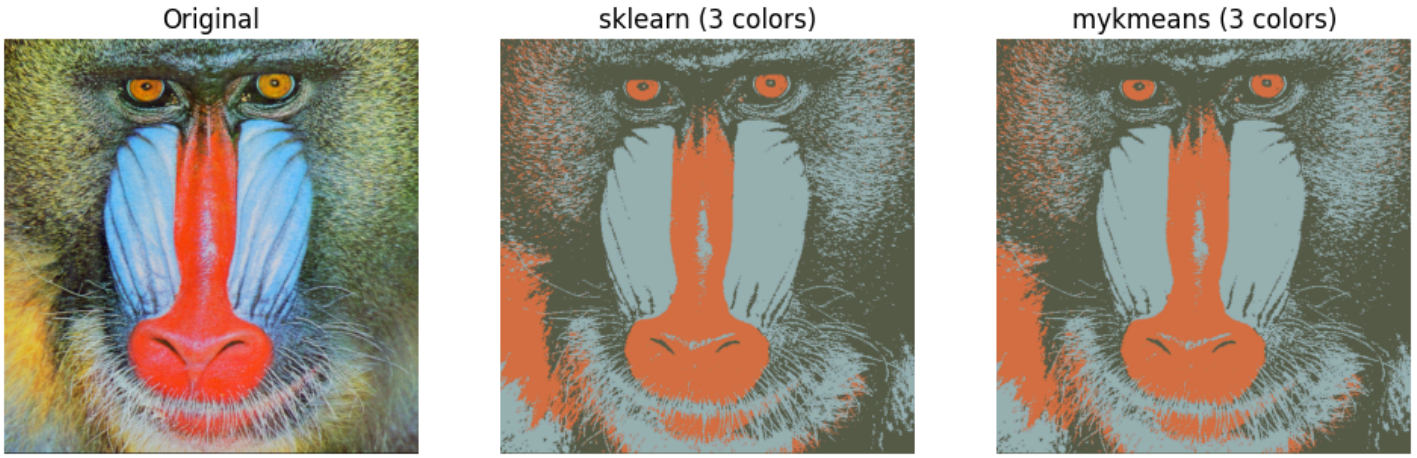




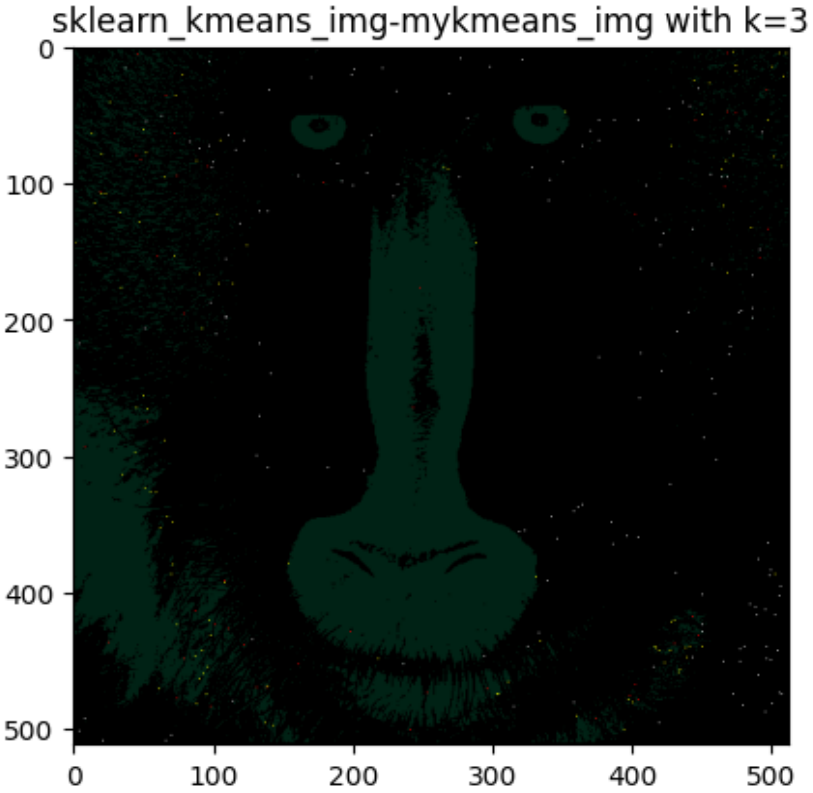


(d) Comparison with sklearn K-means Implementation:

Results of image compression using the implemented K-means algorithm are compared with sklearn's implementation, analyzing differences in compression quality and execution time.



The difference may not be visible but it present-




```
Time taken for compressing image with sklearn kmeans: 0.1982 seconds
Time taken for compressing image with mykmeans: 0.9247 seconds
```

Scikit-learn's KMeans is likely faster due to:

- **Early stopping:** Stops iterating when centroids barely change, reducing unnecessary computations.
- **Optimized structures:** Uses efficient data structures for faster distance calculations, a key KMeans operation.
- **Mini-batch option (large datasets):** Processes data in smaller chunks for significant speed improvements.

(e) Implementation of Spatial Coherence:

To enhance image compression with K-means clustering, we incorporate spatial coherence using a Gaussian filter. This filter smooths the image, emphasizing local structures and reducing noise.

1. **Kernel Construction:**
    - The Gaussian filter operates by convolving the input image with a Gaussian kernel.
    - This kernel is a matrix of weights that represents the shape of a Gaussian distribution which determines the extent of smoothing applied to the image.
    - Each element of the kernel represents the weight assigned to the corresponding pixel in the neighborhood of the current pixel being processed.
2. **Weighted Averaging:**
    - During convolution, the Gaussian filter computes a weighted average of the pixel values in the neighborhood of each pixel.
    - The weights assigned to each pixel are determined by the Gaussian function, with higher weights assigned to pixels closer to the center of the kernel and lower weights to those farther away.
    - This weighted averaging process effectively blurs the image, with neighboring pixels contributing more to the smoothed value than distant pixels.
3. **Kernel Size and Sigma Parameter:**
    - The kernel size determines the size of the neighborhood considered for smoothing. A larger kernel size means a wider area is considered, resulting in stronger smoothing.

- The sigma parameter controls the spread of the Gaussian distribution. A higher sigma value results in a wider and smoother distribution, leading to more significant smoothing effects.

4. **Implementation:**

- Done by scipy library

1/16

1	2	1
2	4	2
1	2	1

1/273

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

1/1003

0	0	1	2	1	0	0
0	3	13	22	13	3	0
1	13	59	97	59	13	1
2	22	97	159	97	22	2
1	13	59	97	59	13	1
0	3	13	22	13	3	0
0	0	1	2	1	0	0

Original

Compressed Image with Gaussian filtering

sklearn (3 colors)

mykmeans (3 colors)

```
Time taken for compressing image with Gaussian filter: 0.2166 seconds
Time taken for compressing image with sklearn: 0.1987 seconds
Time taken for compressing image with mykmeans: 1.3004 seconds
Time taken for blurring is : 0.0179 seconds
```

The Gaussian filter is effective in reducing noise in the image, as it smoothes out rapid intensity variations caused by noise. Importantly, it preserves edges in the image by adjusting the weights in such a way that regions with sharp intensity transitions are not overly smoothed. This edge preservation property ensures that important features in the image, such as object boundaries and textures, are retained despite the smoothing process.

Conclusion:

K-means clustering offers an efficient method for image compression. By implementing K-means from scratch and incorporating spatial coherence, we achieve improved compression results.

Problem 2

Introduction:

Support Vector Machines (SVM) are powerful models for classification and regression tasks. In this assignment, we explore SVM decision boundaries using different kernels and learn about hyperparameter tuning for optimal performance.

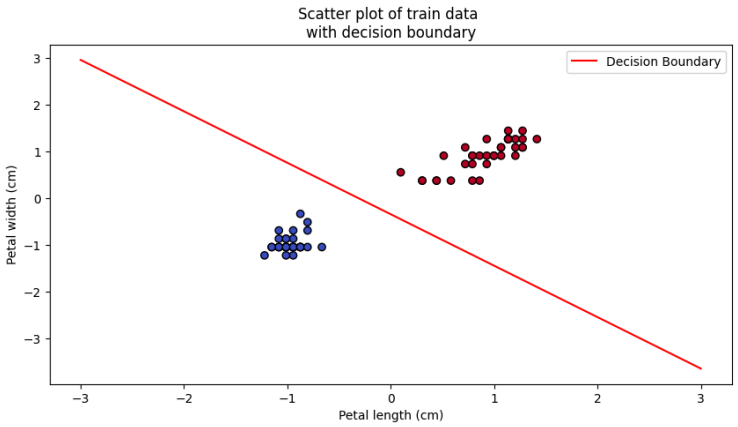
Task 1: SVM on Iris Dataset

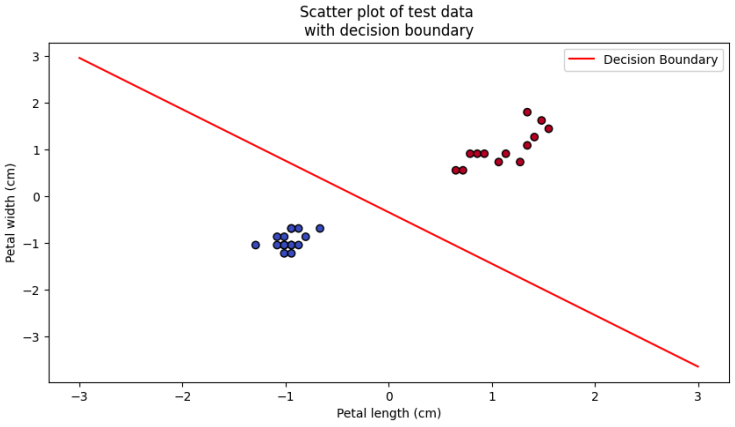
(a) Loading and Preprocessing Iris Dataset:

The Iris dataset is loaded, focusing on petal length and width features. Data is normalized and split into training and testing sets.

(b) Training Linear SVM and Plotting Decision Boundary:

A Linear Support Vector Classifier (LinearSVC) is trained and its decision boundary plotted along with the test data.





Task 2: SVM on Synthetic Dataset

(a) Generating Synthetic Dataset:

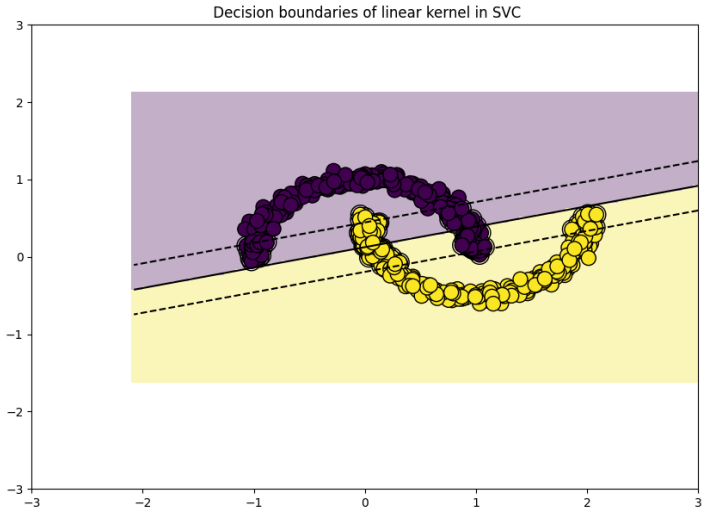
A synthetic dataset with added noise is generated using make_moons().

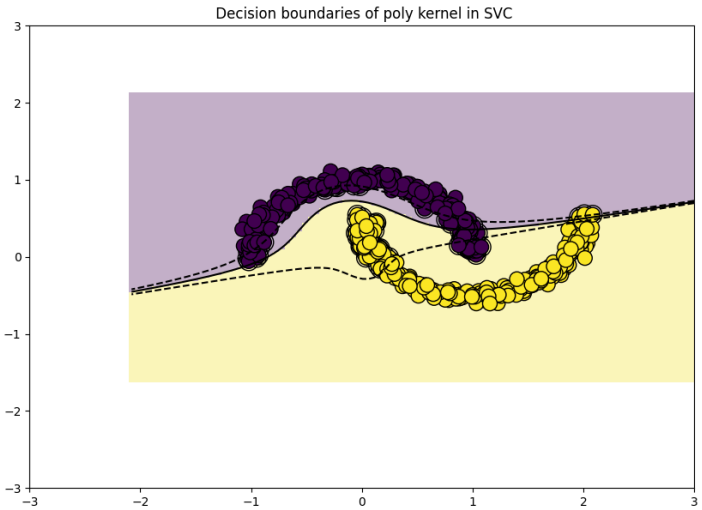
(b) Implementing SVM Models with Different Kernels:

SVM models with Linear, Polynomial, and RBF kernels are implemented. Decision boundaries for each kernel are analyzed. Gamma was taken as 1.

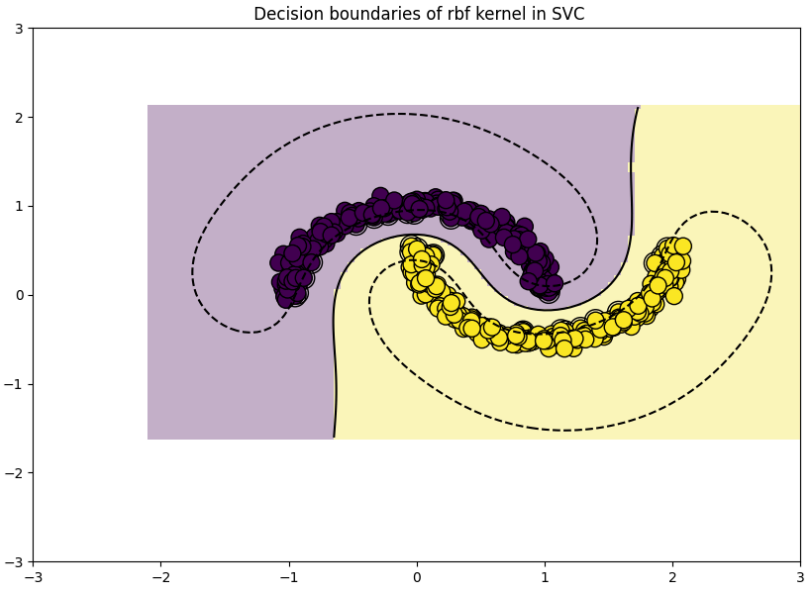
- **Linear Kernel:** Efficient for linearly separable data, offering interpretable decision boundaries. However, its applicability is limited to problems with inherent linearity.

- **Polynomial Kernel:** Provides increased flexibility for non-linear data by approximating class separations with polynomial curves. High-degree polynomials can lead to overfitting.
- **RBF Kernel:** Renowned for its versatility in handling complex, non-linear relationships. It dynamically adapts to the data distribution, resulting in flexible decision boundaries. Compared to high-degree polynomials, RBF kernels exhibit less susceptibility to overfitting, but necessitate careful hyperparameter tuning, particularly the gamma parameter.





(c) Hyperparameter Tuning for RBF



Kernel SVM with RandomizedSearchCV:

Hyperparameter tuning is performed to find the best gamma and C values for the RBF kernel SVM using RandomizedSearchCV with iterations set to 50.

Randomized search cross-validation (RandomizedSearchCV) is a method used to find the best combination of hyperparameters for a machine learning model. Instead of

trying every possible combination, which can be time-consuming, it randomly selects a certain number of combinations to evaluate.

```
Best hyperparameters: {'C': 1.3292918943162166, 'gamma': 0.711447600934342}
```

C:

- **Impact:** Balances between maximizing the margin and minimizing classification error.
- **High C:** Narrows margin, risking overfitting by closely fitting training data.
- **Low C:** Widens margin, encouraging generalization but may misclassify some examples.
- **Decision Boundary:** High C leads to complex boundaries fitting data closely; low C results in simpler boundaries with more tolerance for misclassifications.

Gamma:

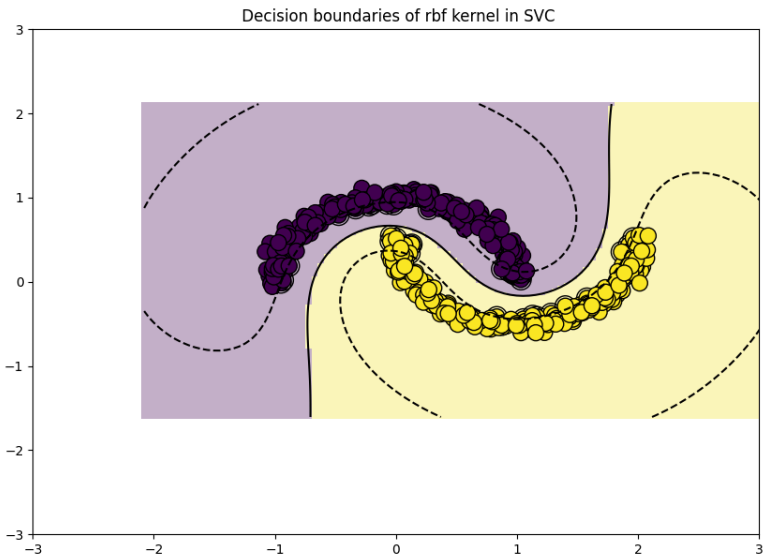
- **Impact:** Determines the influence of a single training example.
- **High Gamma:** Creates localized decision boundary, limited to immediate vicinity of support vectors.
- **Low Gamma:** Produces smoother boundary, with influence of support vectors extending further.
- **Decision Boundary:** High gamma yields intricate boundaries closely adapting to training data, possibly overfitting; low gamma results in smoother boundaries, less sensitive to individual examples, and better generalization to unseen data

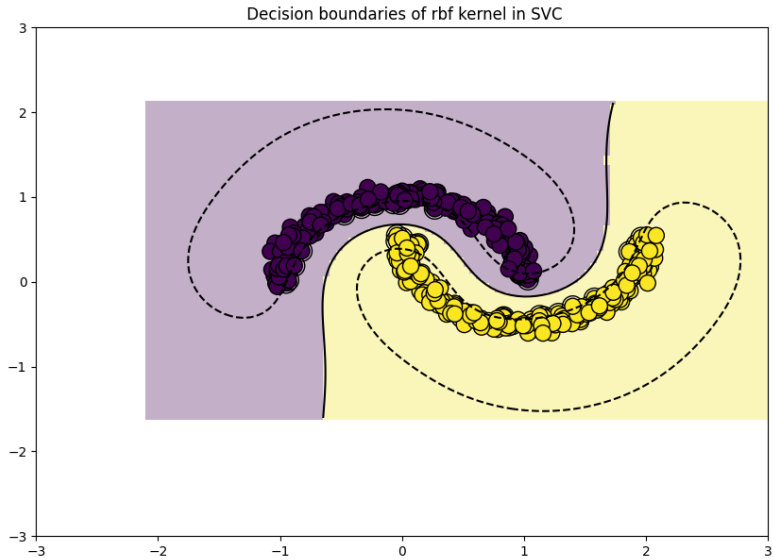
(d) Plotting Decision Boundary for RBF Kernel SVM with Best Hyperparameters:

The decision boundary for RBF kernel SVM with optimal hyperparameters is plotted, explaining their impact on model performance.

Not much changes can be observed between the best parameter decision boundary and decision boundary with gamma = 2.

best parameter (**Left**) and previous parameter **(Right)**





Conclusion:

SVM classification is robust. Through experimenting with kernel changes and hyperparameter adjustments, we can learn more about how support vector machines (SVMs) work, which in turn helps us better understand how they can be used in the real world.

The link for the assignment - B22AI045_all.ipynb