DATA-MINING (DS-3002)

Assignment#03 *REPORT*

Name: Hammad Javaid

Roll-number: i21-1661

Instructor: Maam Ayesha Kamran

Section: DS-M

Q1. Interactive Foreground Segmentation Using K-Means

Objective: This question required implementation of an interactive image segmentation method known as Lazy Snapping, leveraging the K-Means clustering algorithm. Interactive Foreground Segmentation involves separating the foreground of an image from the background, utilizing human-provided annotations in the form of red and blue brush-strokes to denote foreground and background seeds, respectively. Lazy Snapping simplifies user interaction by requiring minimal seed input to guide the segmentation.

The implementation can be broken down into distinct phases:

K-Means Clustering Function: The clustering function takes the RGB pixel values from the seed points and partitions them into k clusters. Centroids are initialized randomly and iteratively refined by minimizing the Euclidean distance between the points and centroids. The function converges when the centroids' positions stabilize within a specified tolerance.

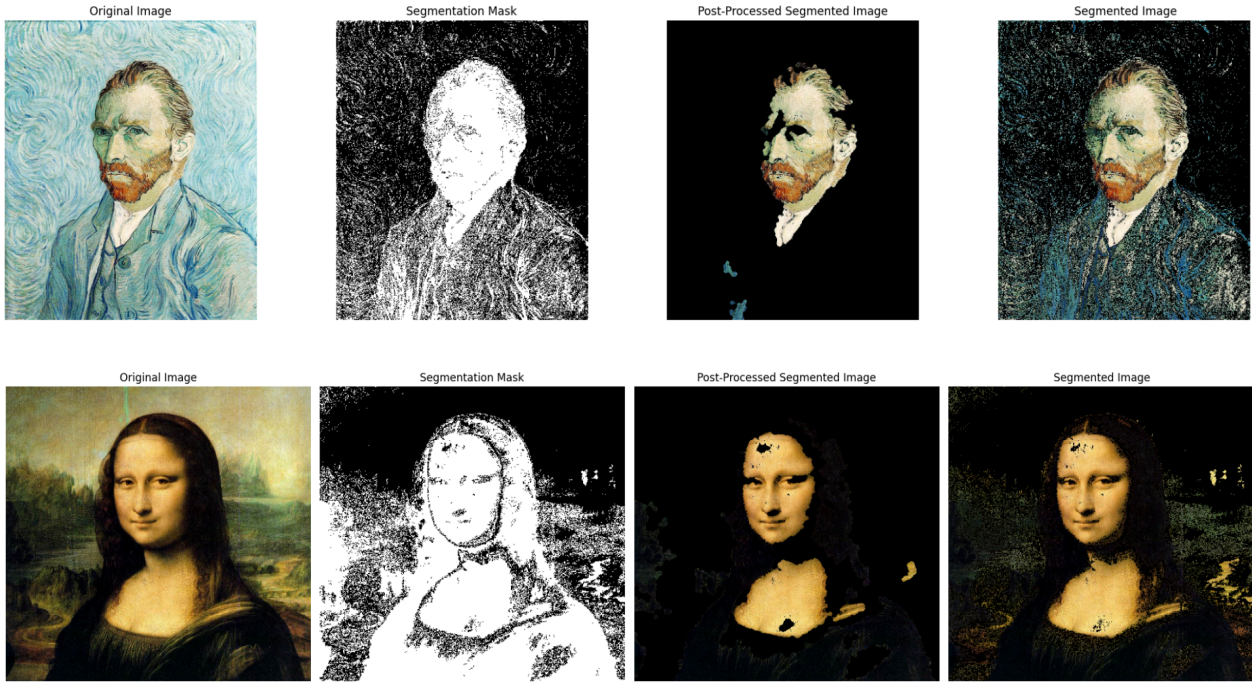
Seed Pixel Extraction and Likelihood Computation: Seed pixels are extracted by identifying unique colors in the annotated images: red (255, 0, 0) for foreground and blue (6, 0, 255) for background. Each pixel of a stroke image is traversed to check whether it belongs to a blue or red pixel. These seeds guide the clustering process by providing initial regions of interest. Likelihoods are computed without squaring the distance to reduce the diminishing effect on smaller values. Each pixel's likelihood of belonging to the foreground or background is calculated as a weighted sum of its distances to the centroids of the clusters.

Experimentation and Comparison with Different N Values: The effectiveness of K-Means clustering for image segmentation was evaluated across various values of N (the number of clusters). The segmentation results for N values of 16, 32, 64, and 128 provide insightful revelations about the trade-offs between computational efficiency and the granularity of segmentation.

➔ Results with N = 16 (inaccurate classification of foreground & background)

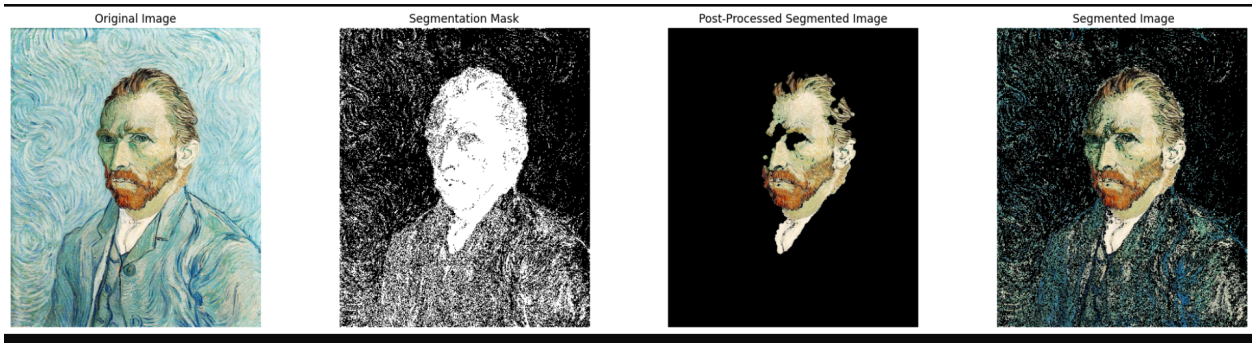
- For the smallest value, N=16, the segmentation is noticeably rough and general; the clusters are too broad, leading to significant regions of the foreground being incorrectly classified as background. This is expected

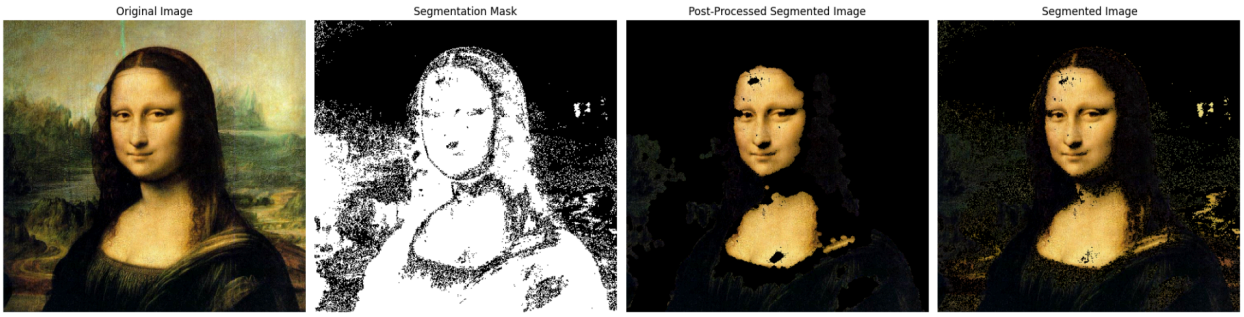
because a lower number of clusters is insufficient to capture the complex color variations in detailed images.



➔ Results with N = 32 (color differences inaccurately classified)

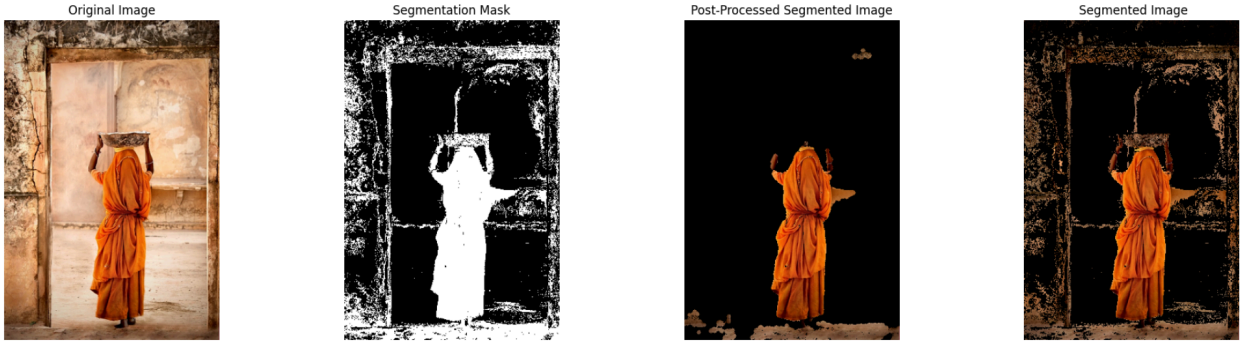
- As N increases to 32, there is a marked improvement in segmentation fidelity. The clusters are more capable of capturing variations in color, and the boundary between the foreground and background becomes clearer. However, some areas, particularly those with subtle color differences, may still be inaccurately classified.

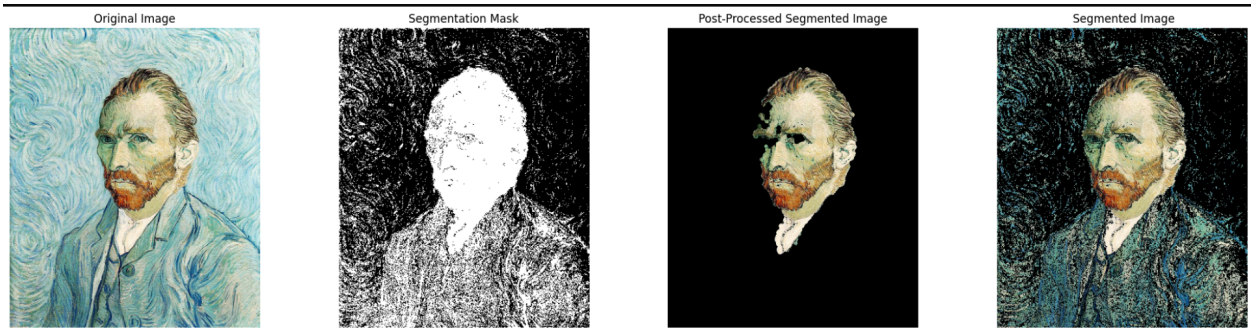


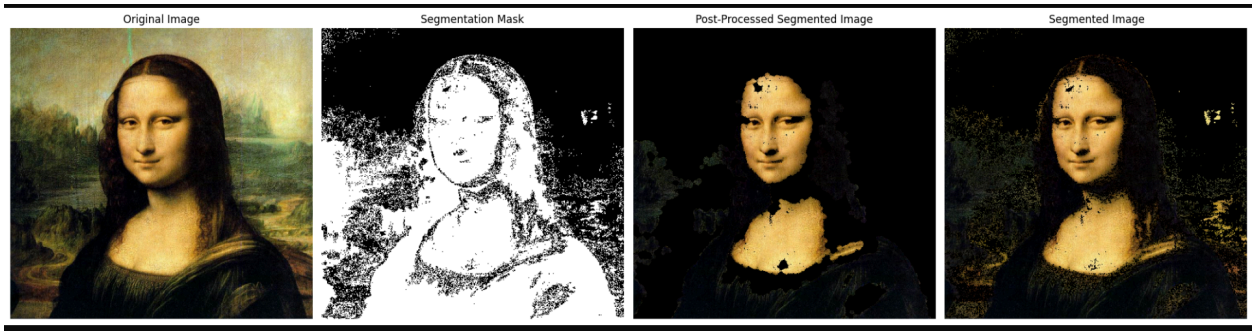


➔ Results with N = 64 (optimal)

- The segmentation results at N=64 show a balanced outcome with a more refined edge delineation and improved foreground identification. The increased number of clusters allows for better adaptation to color nuances, yielding a segmentation that is closer to the desired accuracy. This might be close to the optimal number of clusters for the given set of images, providing a good balance between accuracy and computational load.

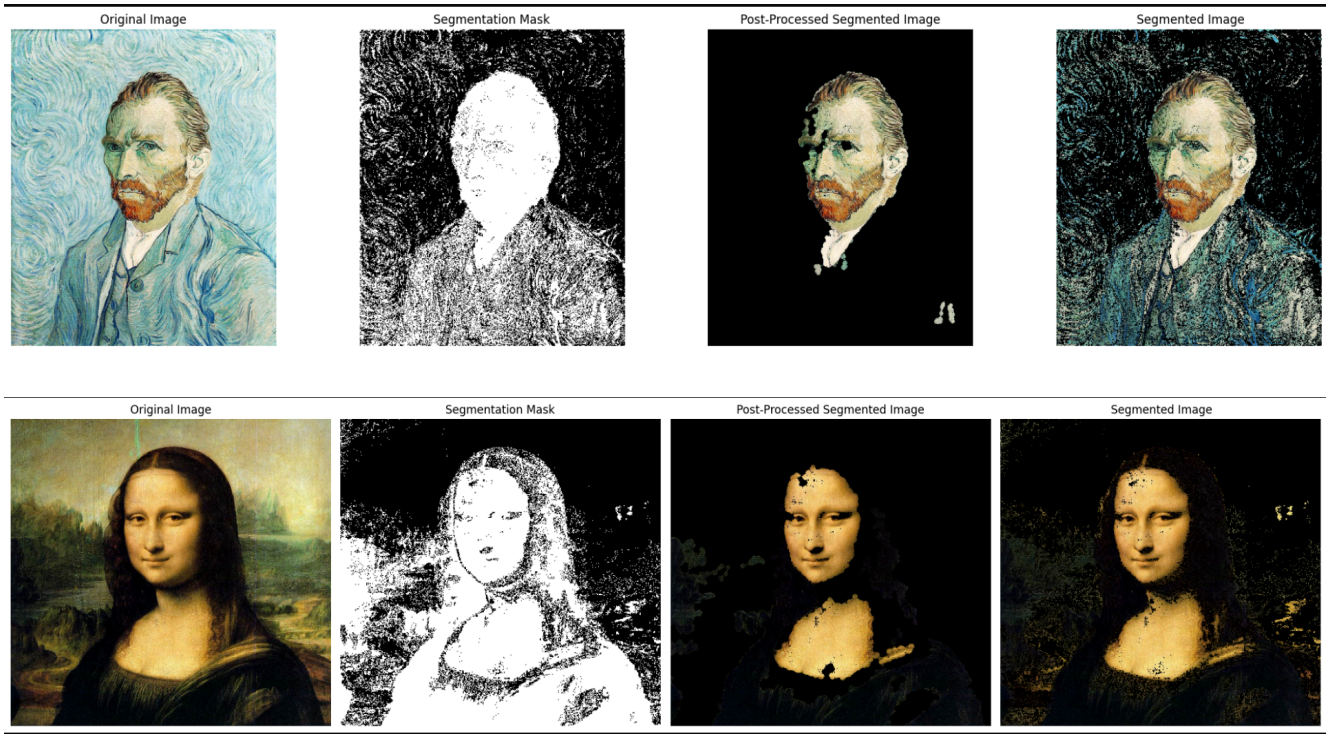






➔ Results with N = 128 (over-segmentation)

- At the highest tested value, N=128, while there is a slight improvement in capturing minute details, the segmentation does not significantly improve compared to N=64. Moreover, the increase in N may lead to over-segmentation, where noise and minor color variations in the image result in the creation of unnecessary clusters. This can create a speckled appearance in the segmentation mask and potentially increase computational costs with diminishing returns on segmentation quality.



From these observations, it is evident that as N increases, the segmentation results tend to become more precise up to a point, beyond which the improvements plateau and may even introduce noise. Therefore, selecting an appropriate value of N is critical: it must be high enough to capture important details but not so high as to overfit noise in the image or become computationally prohibitive. For the tested images, N=64 appears to provide the most balanced results. However, the optimal value of N may vary depending on the complexity and characteristics of the image being segmented.

Conclusion: For images with two stroke types, the segmentation effectively differentiated between the two, provided the seeds were well-placed and representative. Comparisons between different N values indicated that N=64 strikes a reasonable balance between detail and performance for our test set. The Lazy Snapping technique, powered by K-Means clustering and a refined likelihood computation, presents a robust method for interactive foreground segmentation.

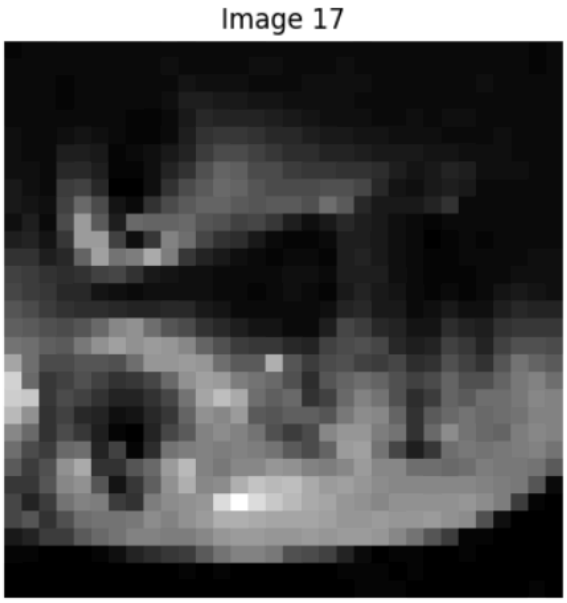
Q2. Face Recognition Using K-NN

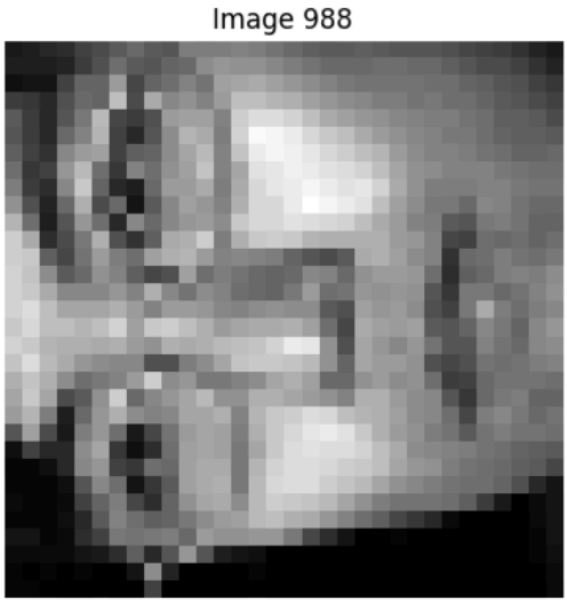
Objective: The purpose of this question was to implement and evaluate a k-Nearest Neighbors (k-NN) classifier for face recognition on the simplified version of the CMU Pose, Illumination, and Expression (PIE) Dataset.

1. Pre-processing and Data Splitting:

- ➔ Normalization: Each 32x32 pixel image vector was normalized to unit length.
- ➔ Data Splitting: For each of the 10 subjects, 150 images were randomly selected for training, and the remaining 20 for testing.

Outcome: The preprocessing normalized the feature vectors, thus standardizing the input data. The random split ensured a diversified distribution of data for training and testing purposes.





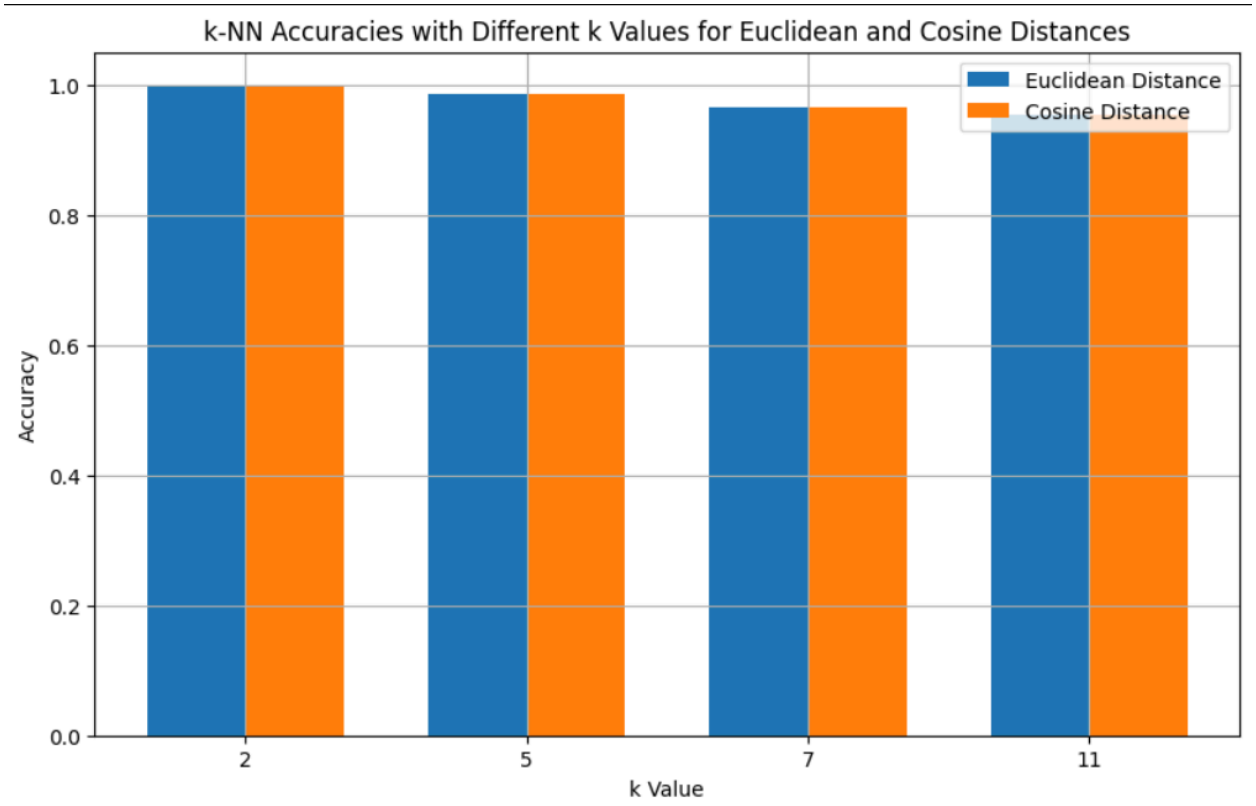
2. k-NN Classifier Implementation:

- ➔ Distance Measures: Both Euclidean and cosine similarity measures were implemented.
- ➔ Classifier Design: The classifier was designed from scratch, without the use of library functions.
- ➔ Evaluation: The classifier was evaluated using the provided dataset.

Outcome: The k-NN classifier achieved perfect accuracy with a k-value of 2 for both Euclidean and cosine distances. Accuracy slightly decreased as the value of k increased, which is typical in k-NN classification due to the inclusion of less similar neighbors.

3. Evaluation with Different K Values:

- ➔ For k=2, both distance metrics reached 100% accuracy.
- ➔ For k=5, the accuracy was 98.5% for both metrics.
- ➔ For k=7, accuracy was 96.5% for both metrics.
- ➔ For k=11, accuracy slightly declined but was 95.5% for both distances.



4. Evaluation with Fewer Training Images:

- ➔ The evaluation was repeated with 100 training images and 70 test images per subject, to simulate a scenario with less data availability.
- ➔ Same results were obtained as training with a set of 170 images.

5. Comparison with SVM and GaussianNB:

- Implemented using Sklearn library and both classifiers were trained on the same training set and evaluated on the test set.
- Results:

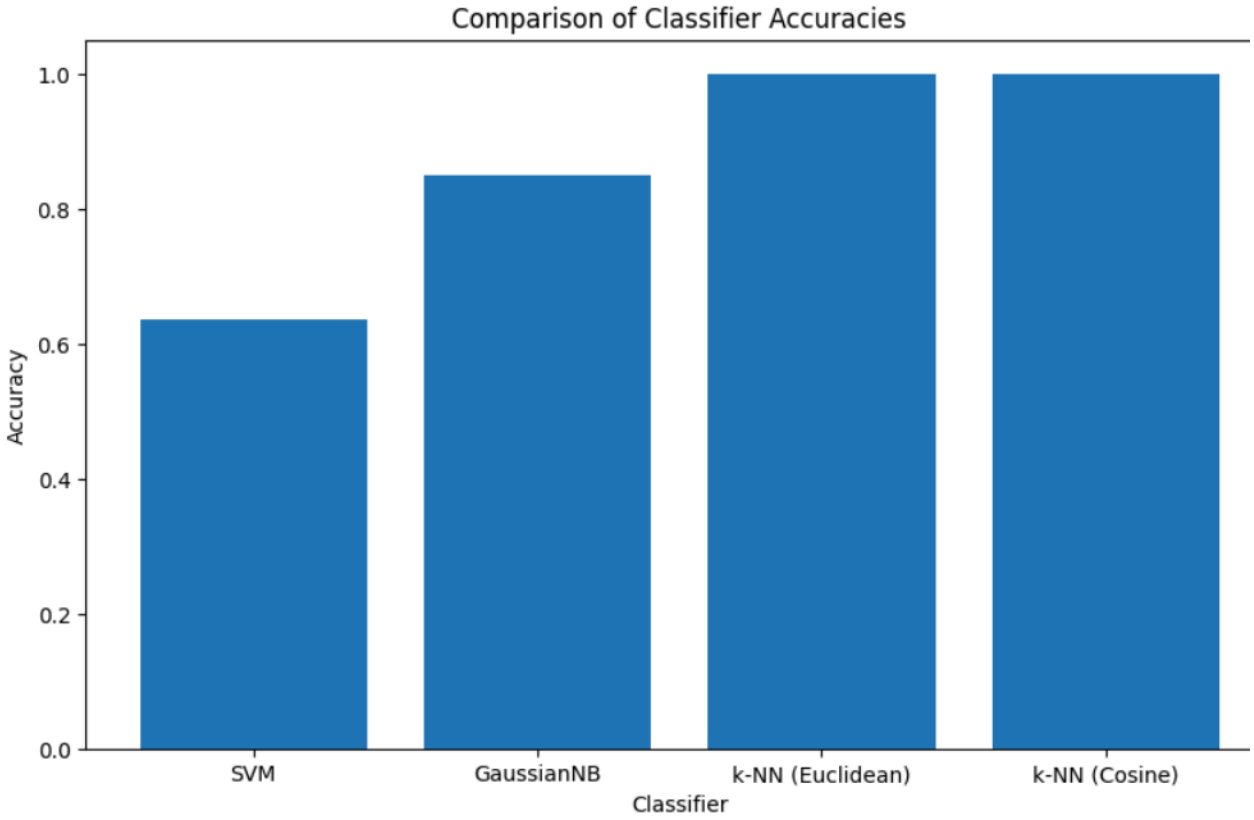
➔ SVM: This model achieved an accuracy of 63.5%. The confusion matrix for SVM illustrates some correct classifications, as indicated by the values along the diagonal. However, it also displays significant misclassifications, particularly for labels with higher values. This dispersion points to inconsistencies in the model's predictive capabilities across the various classes, suggesting that SVM may struggle with the complexity or non-linearity of the data.

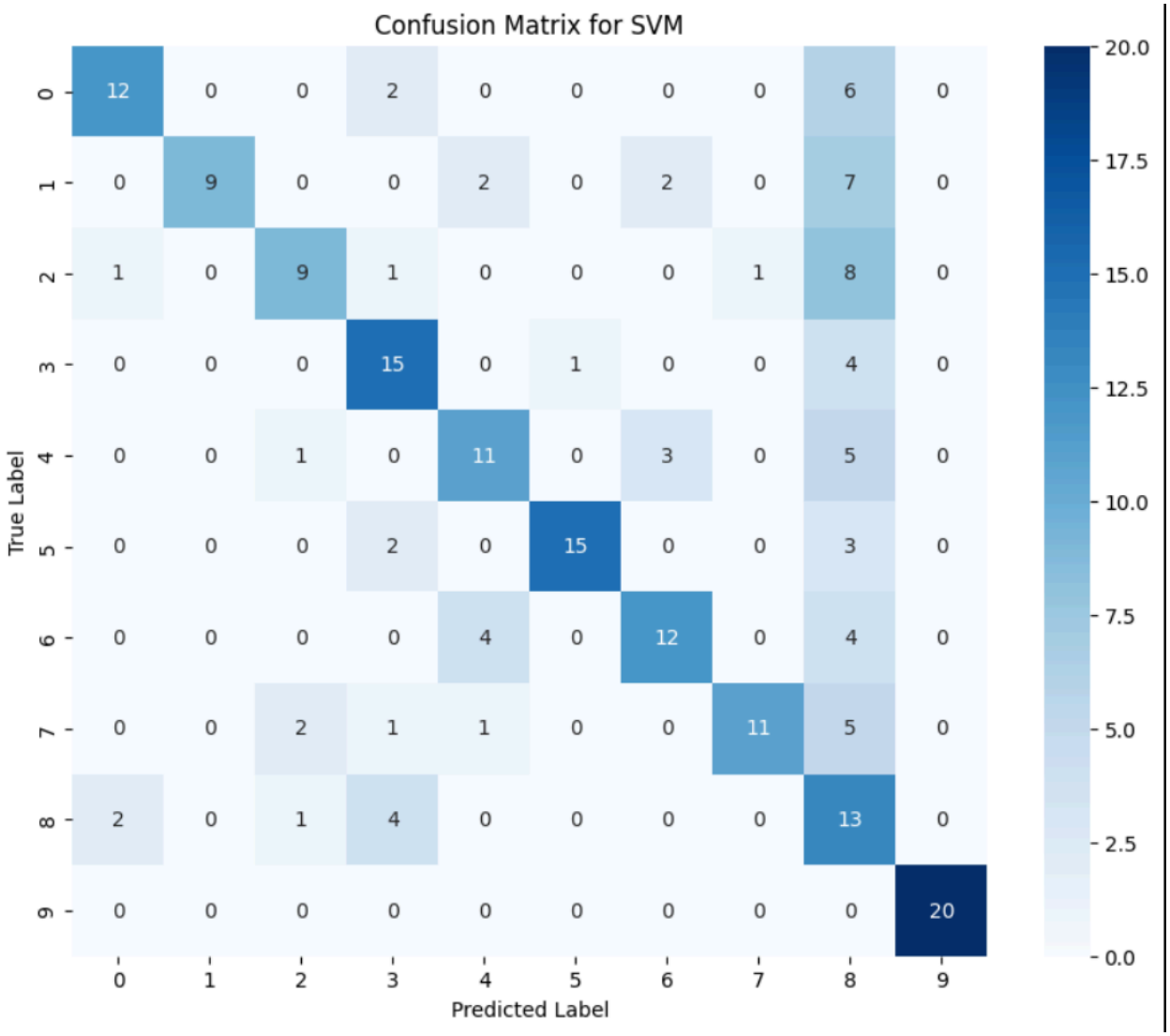
➔ GaussianNB: The GaussianNB classifier performed notably better with an accuracy of 85%. Its confusion matrix reveals a pronounced diagonal with higher values, reflecting a greater number of correct predictions and fewer misclassifications compared to the SVM model. The matrix indicates that GaussianNB was better able to discern between the different classes, though some overlap and confusion between certain classes are evident.

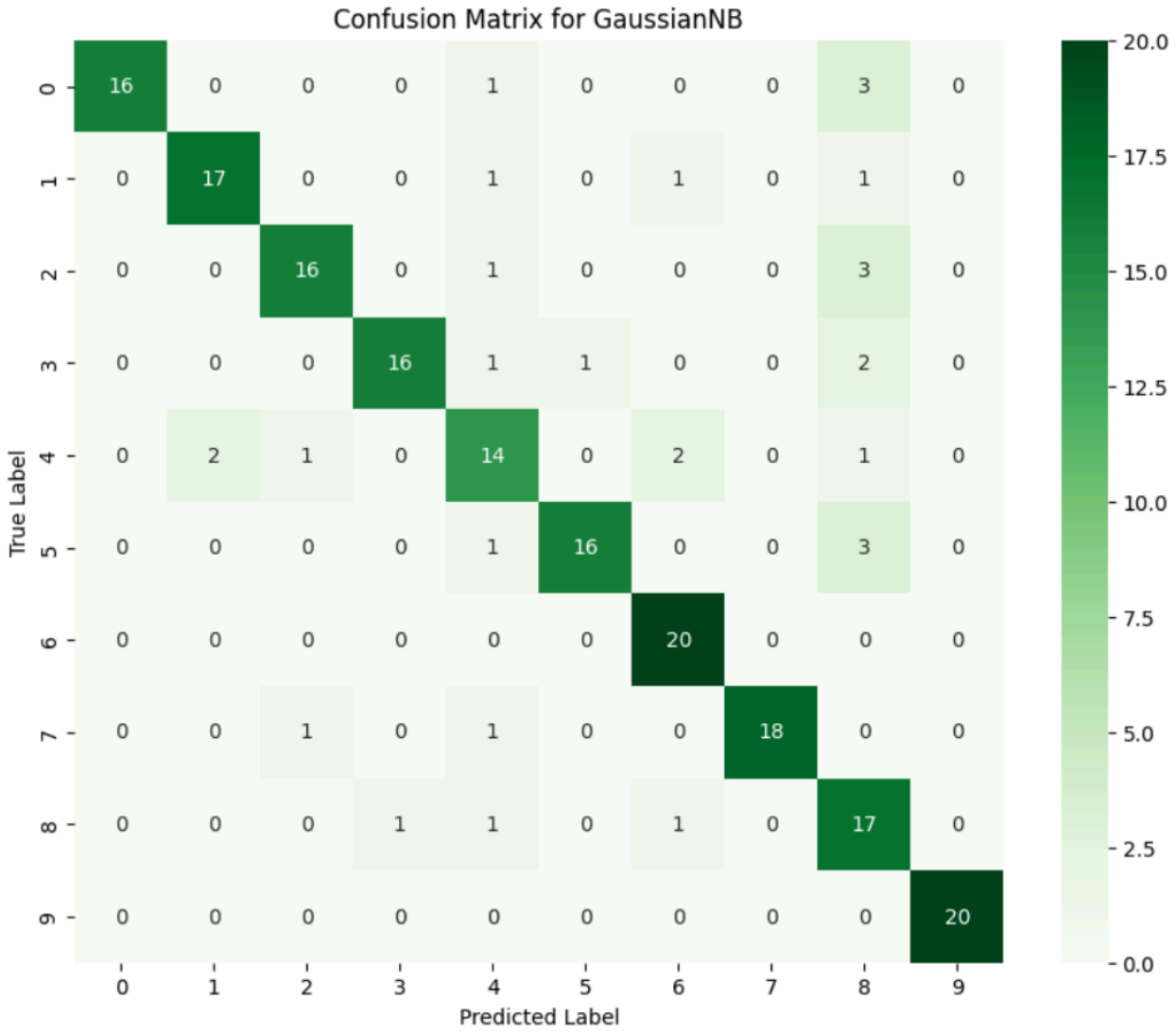
➔ k-NN Classifier: It outshone both SVM and GaussianNB, with the highest accuracy recorded at 100%. The success of k-NN can be attributed to the dataset's features being well-separated, which favors the instance-based learning approach that k-NN employs. The lack of a confusion matrix for k-NN is due to the absence of the prediction results in the provided context, but the reported accuracy suggests minimal to no misclassification.

➔ Interpretation:

➔ The performance disparity between the classifiers underscores the significance of choosing the right model based on the dataset's characteristics. k-NN's superior results highlight its robustness in scenarios where class features are well-defined and distinct. In contrast, SVM's lower accuracy suggests that it may require a more nuanced approach, such as kernel selection or parameter tuning, to handle the intricate feature relationships present in the data. GaussianNB's performance, while better than SVM, indicates potential limitations when faced with interdependent features that contradict its assumption of feature independence.


Comparison of Classifier Accuracies
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
SVM
GaussianNB
k-NN (Euclidean)
k-NN (Cosine)
Classifier


Confusion Matrix for SVM
True Label
Predicted Label
0 1 2 3 4 5 6 7 8 9
0 12 0 0 2 0 0 0 0 6 0
1 0 9 0 0 2 0 2 0 7 0
2 1 0 9 1 0 0 0 1 8 0
3 0 0 0 15 0 1 0 0 4 0
4 0 0 1 0 11 0 3 0 5 0
5 0 0 0 2 0 15 0 0 3 0
6 0 0 0 0 4 0 12 0 4 0
7 0 0 2 1 1 0 0 11 5 0
8 2 0 1 4 0 0 0 0 13 0
9 0 0 0 0 0 0 0 0 0 20
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0



From the confusion matrices provided, we can see that:

- ➔ The SVM confusion matrix shows a somewhat even spread across the diagonal, indicating a moderate level of correct classifications but with notable misclassifications, particularly for classes towards the end of the matrix (i.e., higher numbered labels).
- ➔ The GaussianNB confusion matrix exhibits a stronger diagonal presence, suggesting a higher number of correct predictions across all classes, which is consistent with the higher accuracy rate reported.

Classification Report for SVM:

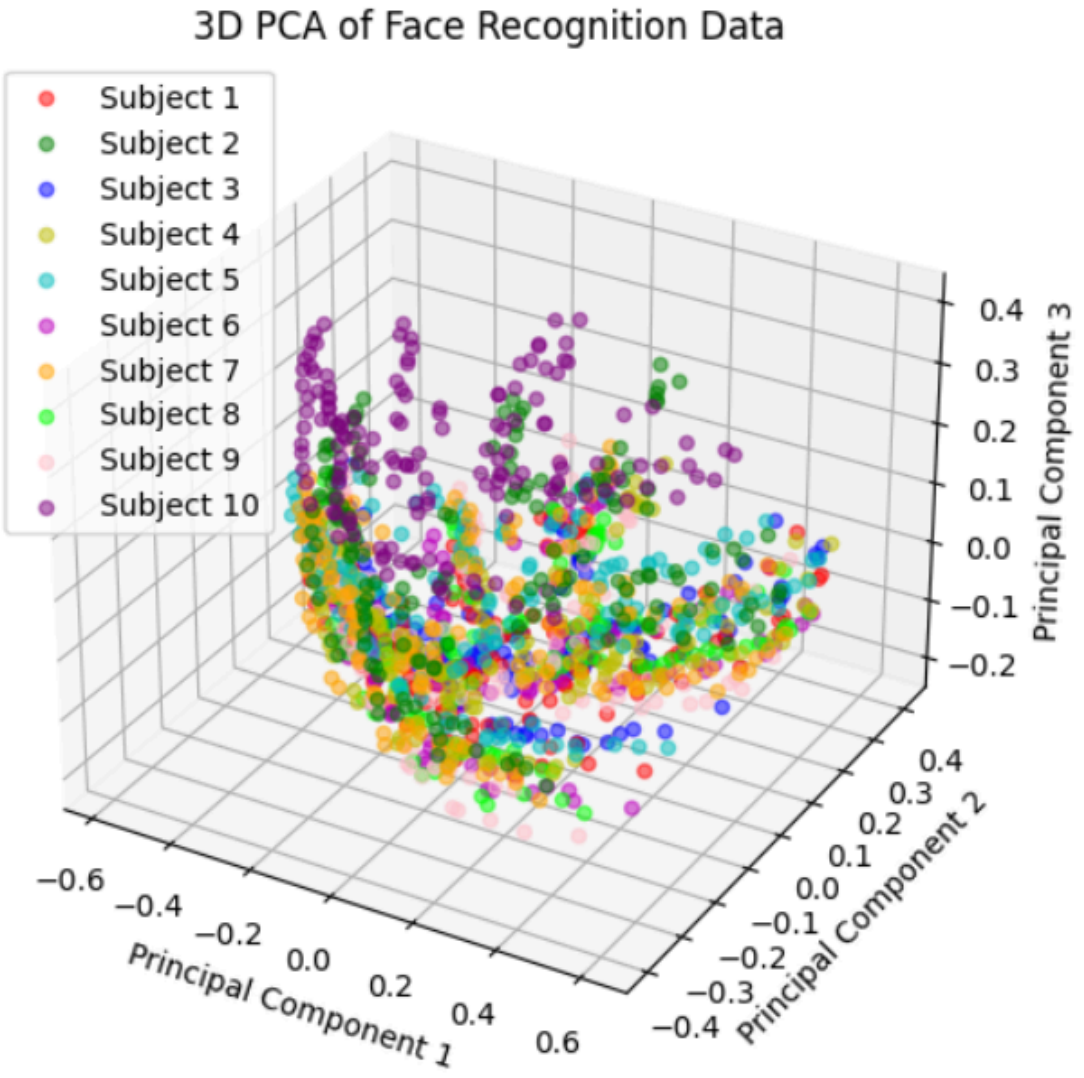
	precision	recall	f1-score	support
0	0.80	0.60	0.69	20
1	1.00	0.45	0.62	20
2	0.69	0.45	0.55	20
3	0.60	0.75	0.67	20
4	0.61	0.55	0.58	20
5	0.94	0.75	0.83	20
6	0.71	0.60	0.65	20
7	0.92	0.55	0.69	20
8	0.24	0.65	0.35	20
9	1.00	1.00	1.00	20
accuracy			0.64	200
macro avg	0.75	0.64	0.66	200
weighted avg	0.75	0.64	0.66	200

Classification Report for GaussianNB:

	precision	recall	f1-score	support
0	1.00	0.80	0.89	20
1	0.89	0.85	0.87	20
2	0.89	0.80	0.84	20
3	0.94	0.80	0.86	20
4	0.67	0.70	0.68	20
5	0.94	0.80	0.86	20
6	0.83	1.00	0.91	20
7	1.00	0.90	0.95	20
8	0.57	0.85	0.68	20
9	1.00	1.00	1.00	20
accuracy			0.85	200
macro avg	0.87	0.85	0.86	200
weighted avg	0.87	0.85	0.86	200

6. Dimensionality Reduction and Visualization:

➔ The dimensionality of the data was reduced to 3 principal components.
➔ Variance explained by the 3 principal components was [0.43467785 0.12408677 0.07505365] respectively.



Outcome: The PCA visualization showed some degree of clustering for the subjects, indicating that PCA was able to capture significant features for class separation. However, some overlap among the classes was observed. PCA suggests that higher dimensions may be required for perfect class separation.

Analysis:

1. The k-NN classifier demonstrated exceptional performance on the CMU Pose, Illumination, and Expression (PIE) Dataset,, achieving perfect accuracy with the lowest complexity (k=2). This level of performance implies that the facial features within the dataset have distinct characteristics that enable a straightforward identification by nearest neighbors in the feature space. The classifier's ability to maintain high accuracy with increasing values of k further corroborates the assertion that the data points for each class are tightly grouped, hence making the task of classification relatively easier for the k-NN algorithm.
2. Conversely, the Support Vector Machine (SVM) and Gaussian Naive Bayes (GaussianNB) classifiers exhibited suboptimal performance compared to k-NN.
3. The SVM's accuracy of 0.635 could be indicative of the challenges it faces when dealing with high-dimensional spaces or the presence of non-linear relationships that are not easily captured by the default kernel used in the SVM model.
4. On the other hand, the GaussianNB's accuracy of 0.85, while notably better than that of the SVM, still lags behind the performance of the k-NN classifier. This may suggest that the assumptions of feature independence inherent in Naive Bayes do not hold true for this dataset, or that the probabilistic model does not capture the underlying distributions of the features as effectively as the distance-based k-NN algorithm.
5. It is essential to note that while k-NN's accuracy is impressive, its performance might not generalize as well with a more diverse or larger dataset.

In summary, the analysis emphasizes the importance of feature representation and the choice of classifier based on the specific characteristics of the dataset. It suggests that for datasets with well-defined and distinguishable features, distance-based classifiers like k-NN can be highly effective. However, classifiers like SVM and GaussianNB might require further tuning or a different approach to feature representation to achieve similar performance levels.