DATA 604 - Final Project

Name: Precious Worgu
Student ID: 119343890

Comparative Analysis of Representation Methods for Oxford 102 Flower Classification

---

Background

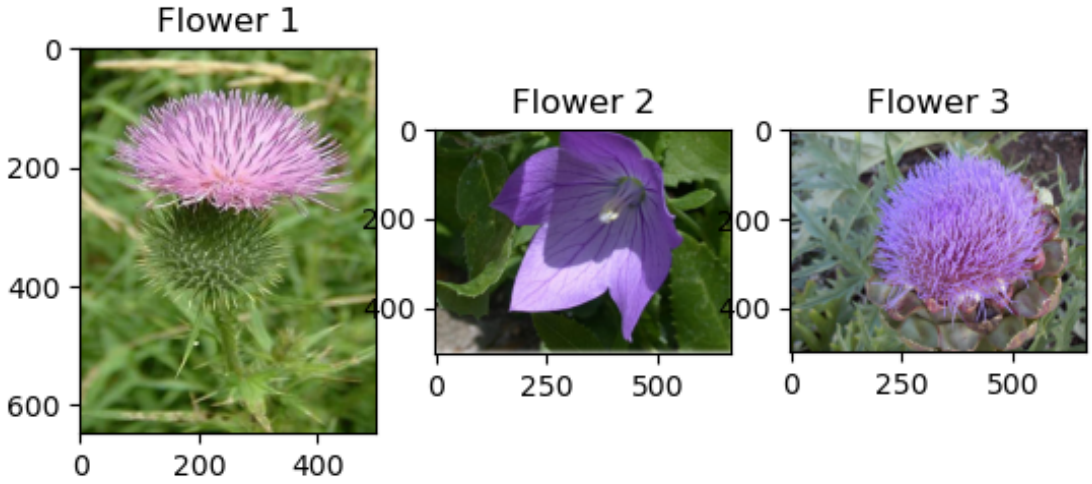
The Oxford 102 flower dataset is a popular benchmark dataset used in image classification studies. It comprises 8,189 labeled images of 102 distinct types of flowers that are commonly found in the United Kingdom. With between 40 and 258 images per class, the dataset provides a diverse representation of flower species. The images in the dataset exhibit a wide range of variations in scale, pose, and lighting, making it a challenging dataset for image classification tasks. Each image is labeled with one of the 102 categories, making it an excellent dataset for evaluating and comparing the performance of various machine learning algorithms and deep learning models for image classification. The images in the dataset come in different sizes, shapes, and orientations, and the flowers can appear at different positions in the frame, which adds to the complexity of the task. The dataset has been partitioned into three parts for training, validation, and testing, each with 1,020, 614, and 1,030 labeled images, respectively. For this project, I selected only three flower types from the dataset, as they share similar colors and are difficult to distinguish.

Introduction

The aim of this project is to examine the advantages of using different representation methods for classification tasks using the CIFAR-10 dataset. To achieve this, various methods, such as raw data, Principal Component Analysis (PCA), kernel PCA (kPCA), and Laplacian Eigenmaps, were analyzed and compared using the kNN and kSVM algorithms. This analysis provides a comprehensive understanding of the diverse approaches available for classification tasks.

Dataset Download

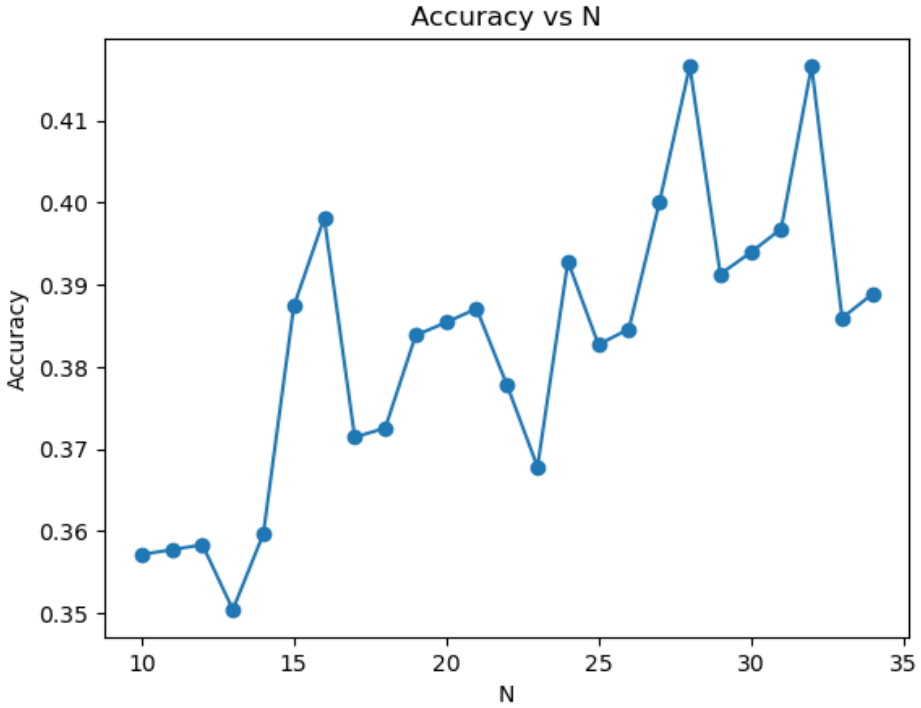
To begin, the Oxford 102 flower dataset was downloaded from Kaggle at https://www.kaggle.com/datasets/nunenuh/pytorch-challange-flower-dataset. Understanding the structure and contents of the dataset is essential for any machine learning tasks. To ensure that the dataset was downloaded and preprocessed correctly, an image from each category was plotted.



This step helps to familiarize oneself with the dataset's contents and verify that the images are correctly labeled.

Selecting the Optimal Training Set Size for Model Performance Improvement

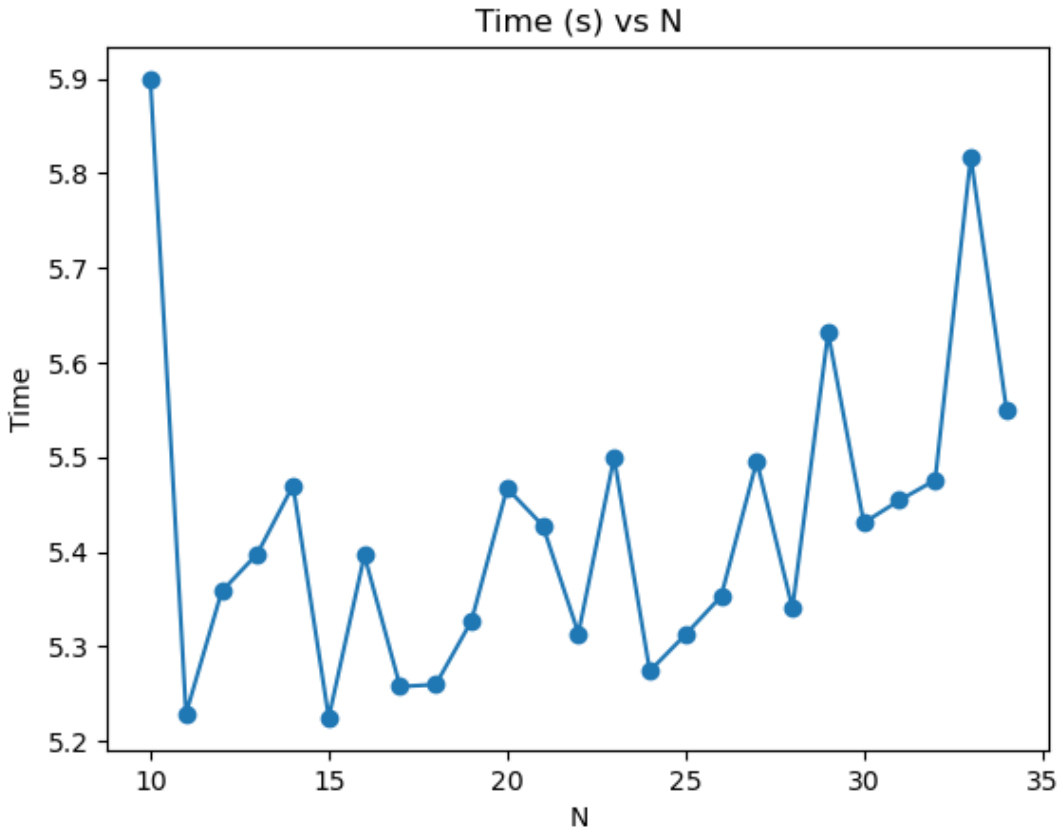
Optimizing the training set size is a critical factor in enhancing the performance of machine learning models. In this project, I utilized two functions to achieve the optimal training set size for enhancing the model's accuracy. The first function divides the dataset into training and testing sets, while the second trains a K-Nearest Neighbors classifier on the training set and predicts the labels of the testing set. Plotting the accuracy of the classifier against the size of the training set, it was apparent that the optimal training set size was 28, resulting in the highest accuracy for the classifier.



This process ensures that the classifier is neither underfitting nor overfitting and thus yields improved results.

In machine learning, finding the optimal training set size is a crucial step for improving the accuracy of a classifier. However, it's also important to consider the time required to train the classifier on that size of data. Oversized training sets can result in prohibitively long training times, diminishing the practicality of the model. Therefore, it's necessary to balance training time with classification accuracy when selecting an optimal training set size.

To measure the time required to train the K-Nearest Neighbors classifier and evaluate its performance, I used the time library in Python to record the time taken to make predictions on the test set with different training set sizes. The results showed that the optimal training set size was 28, with a time of approximately 5.3 seconds required to train the classifier.

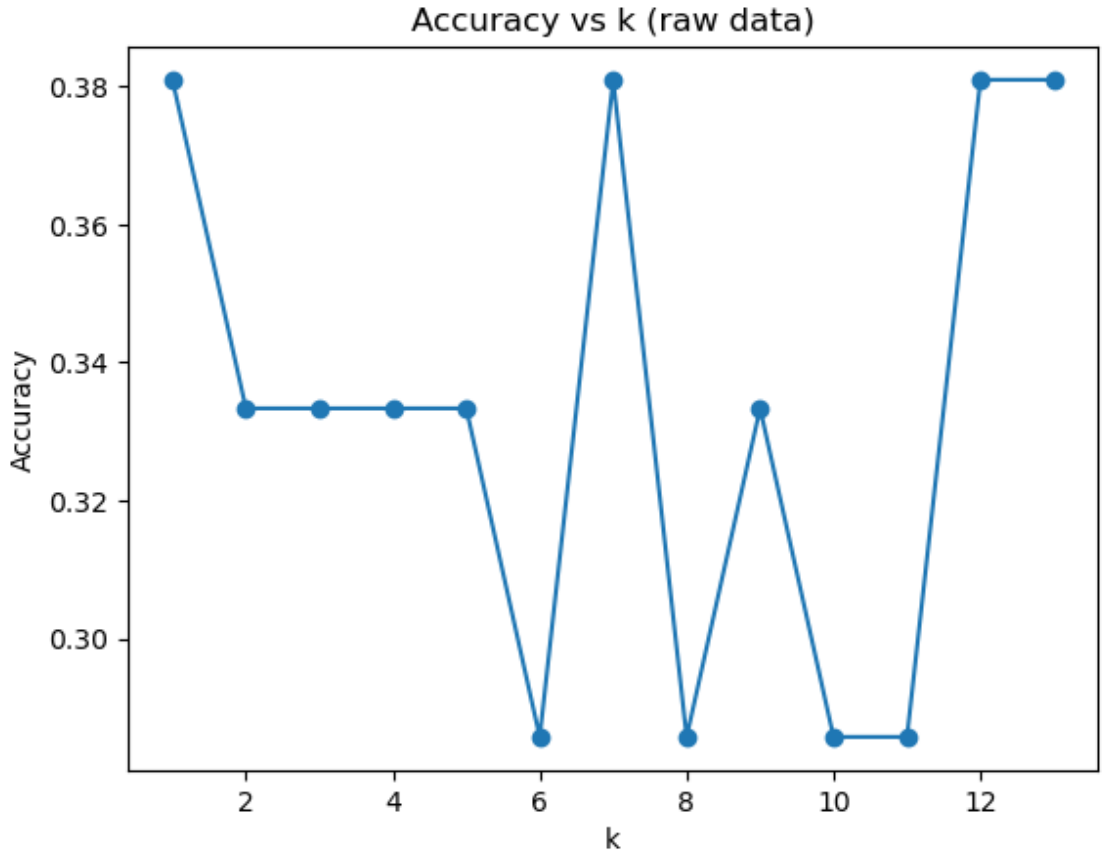


By optimizing the training set size, we can save time without compromising accuracy, making machine learning models more efficient.

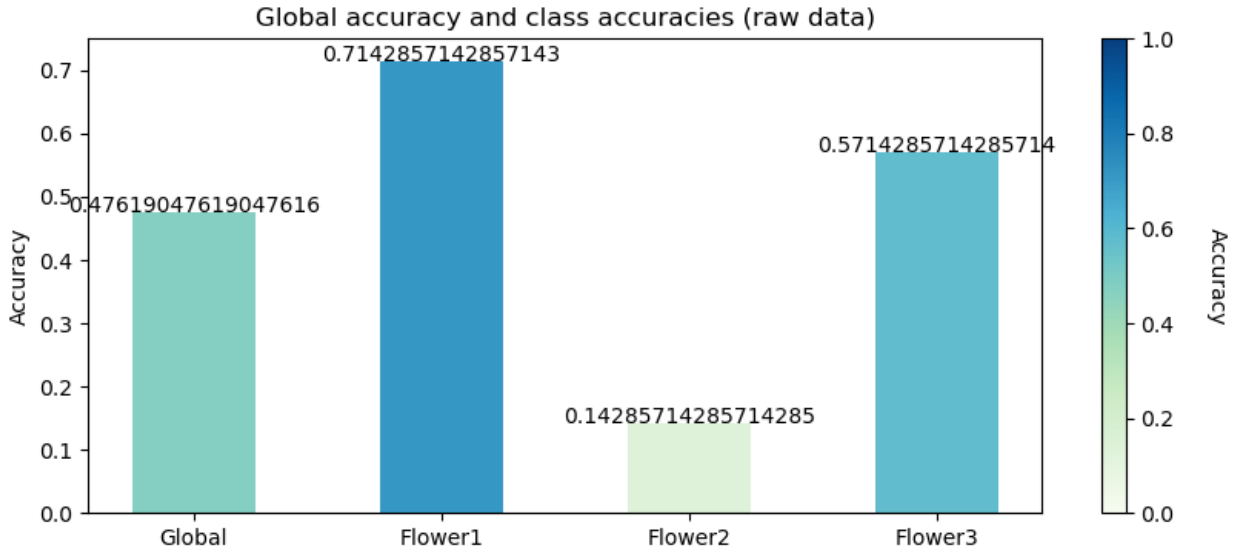
KNN Classification on Raw Data

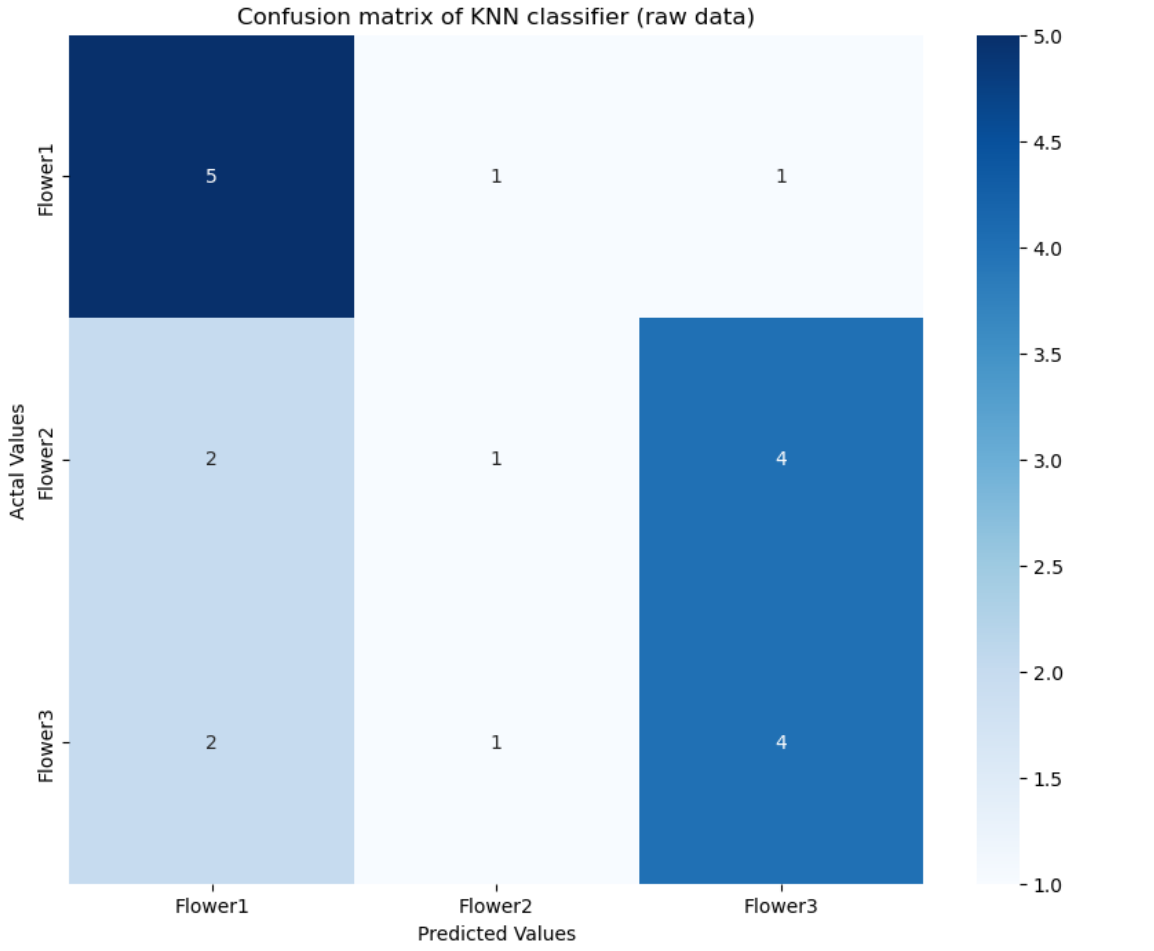
To perform KNN classification on the raw data, I initially partitioned the data into three distinct subsets: training, validation, and testing, for each of the three selected flower species. These subsets were randomized, with the training set consisting of 28 data points, which was found to be the optimal size. The validation set contained the subsequent 35 - N points, and the remaining 1000 points were assigned to the testing set.

To evaluate the model's performance, I calculated the accuracy for different k-values. Through experimentation, I found that the optimal value for k was 7, yielding an accuracy of 0.38 on the validation set.



After applying the KNN classification model with k=7 and using the Euclidean metric on the testing set, the resulting model achieved an accuracy of 47.62% and identified 10 true positives. It's important to note that this accuracy is significantly different from the accuracy of 0.38 achieved on the validation set.



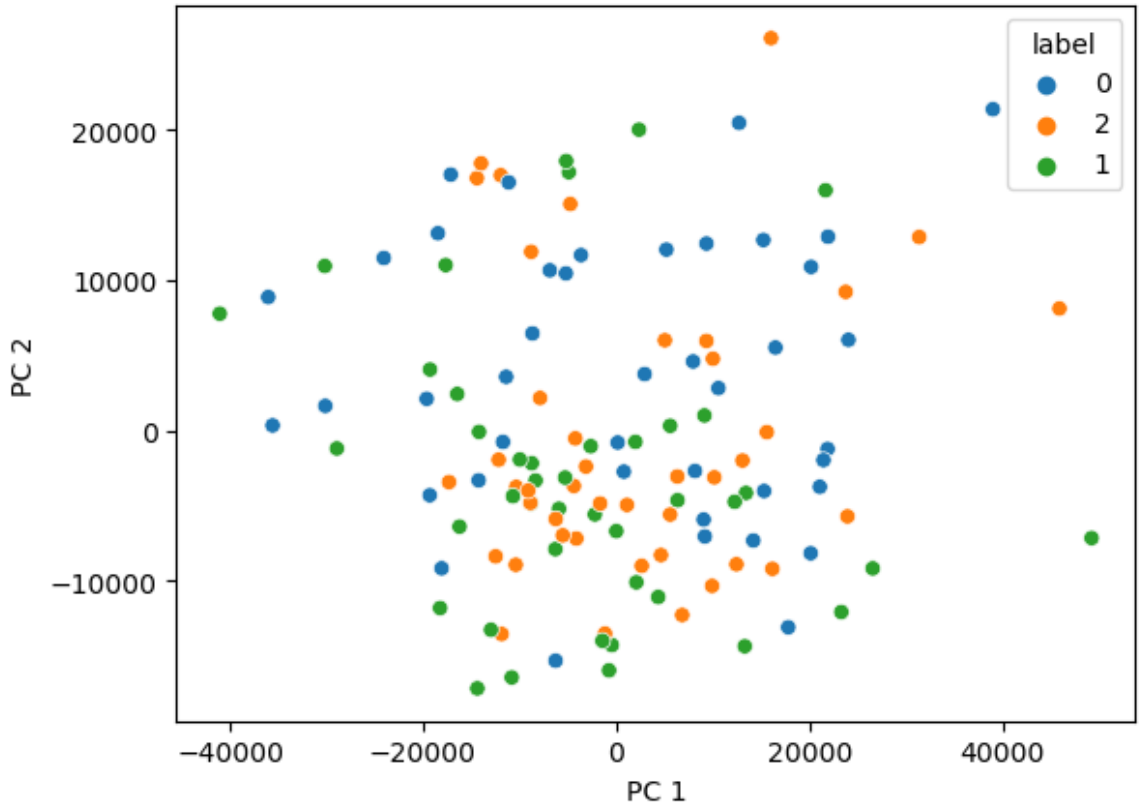


To gain a deeper understanding of the KNN classifier's performance, I plotted the confusion matrix, which shows the number of true positives, false positives, true negatives, and false negatives for each flower type. The results indicate that the KNN classifier performed relatively well for Flower1, with 5 true positives out of 9 actual positives. However, for Flower2, the

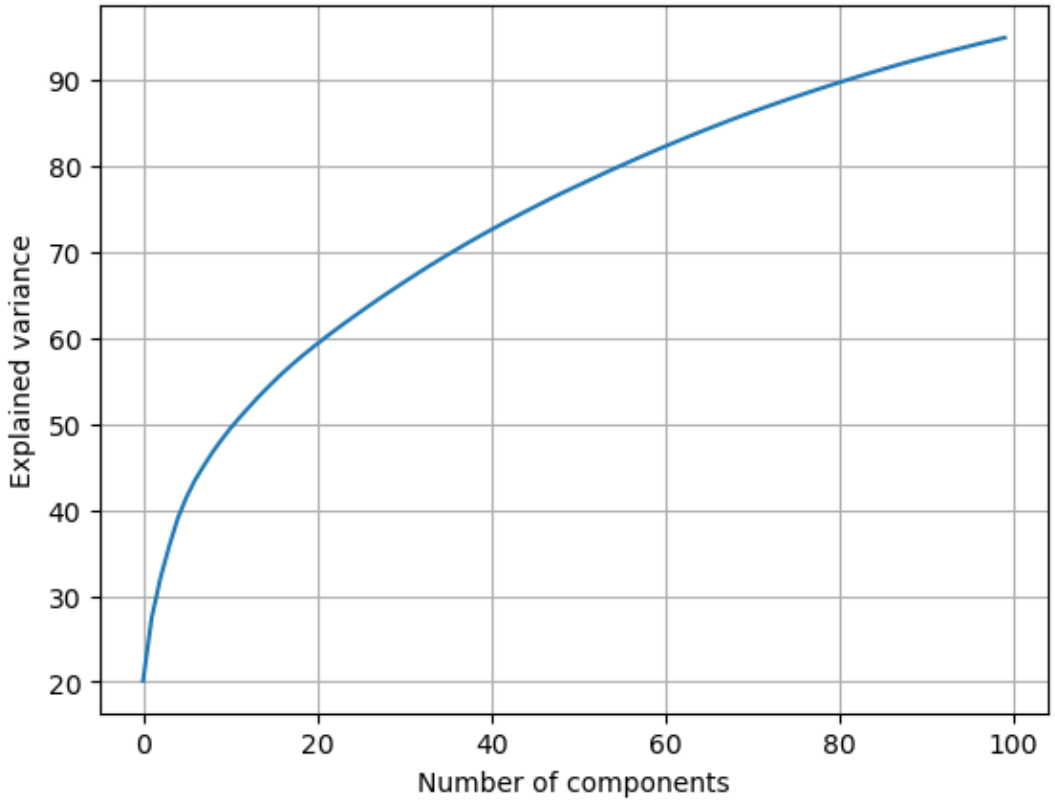
model achieved only 1 true positive out of 3 actual positives, indicating a weak performance. For Flower3, the model had 4 true positives out of 9 actual positives.

KNN Classification on PCA Transformed Data

In order to further improve the KNN classification model, I explored the use of principal component analysis (PCA) on the data. Using the concatenated dataset of training, validation, and testing subsets, I performed PCA with two principal components and visualized the result with a scatter plot. The plot revealed that the three flower species were relatively well separated, suggesting that the use of PCA could improve the accuracy of the KNN classifier.



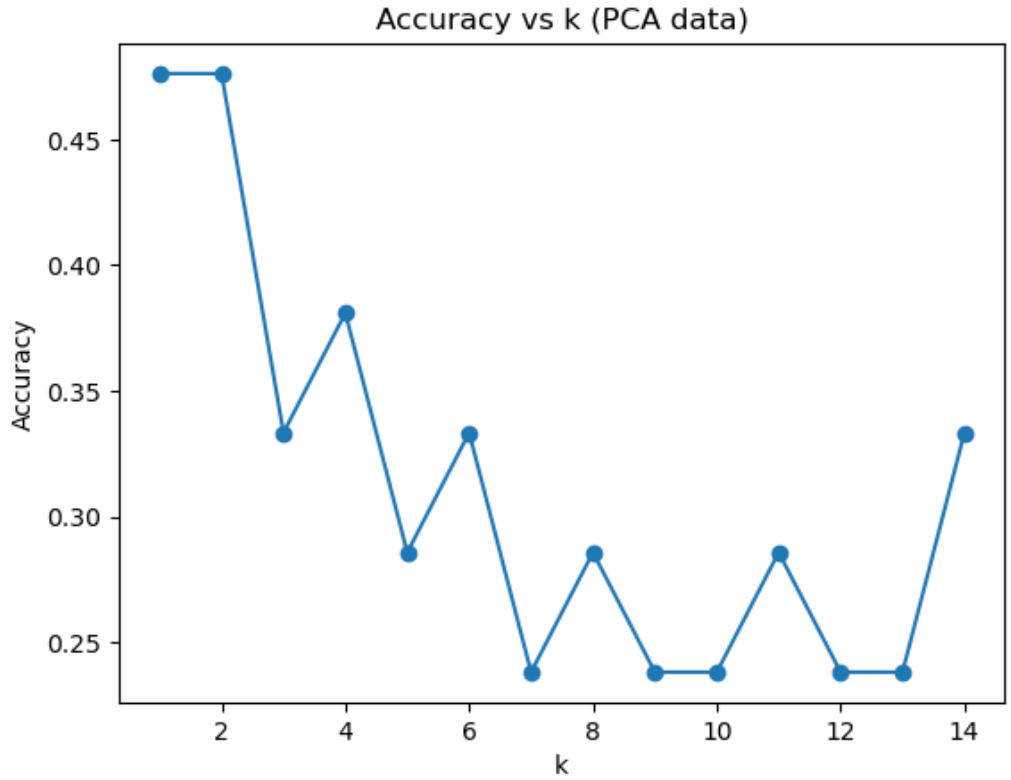
I also plotted the scree plot to determine the optimal number of principal components to use in the model.



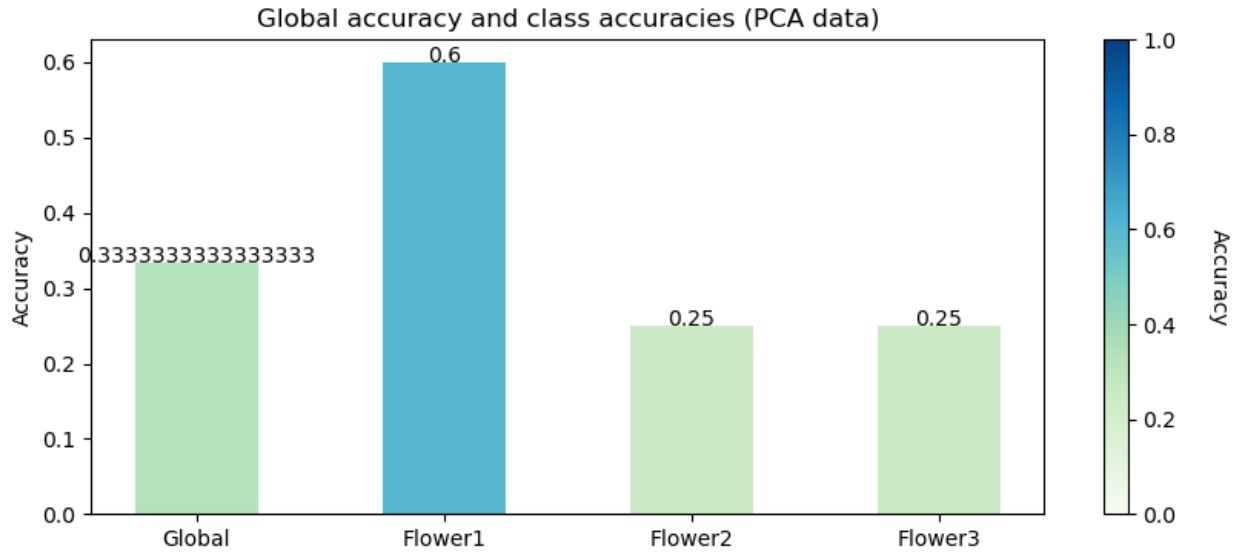
The plot showed that 90% of the variance in the data could be represented by 80 components.

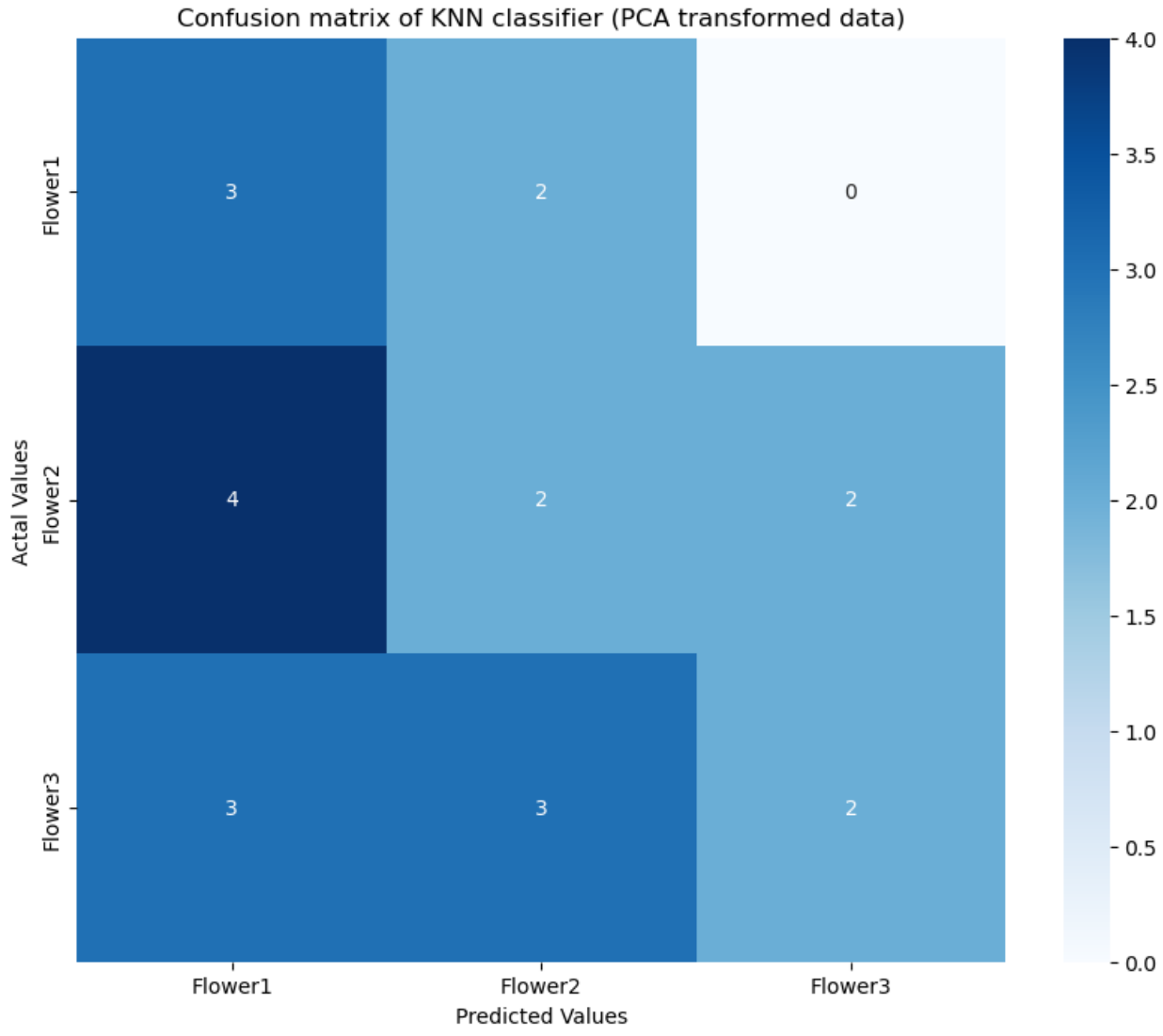
I then proceeded to perform KNN classification on the transformed data using the optimal number of principal components determined by the scree plot.

To optimize the KNN classification model using PCA-transformed data, I assessed the performance of the KNN classifier over a range of k values, from 1 to 14. The resulting plot revealed that the highest accuracy was achieved at a k value of 2, with an accuracy score of approximately 0.33.



Further performance evaluation metrics provide insights into the effectiveness of the KNN classifier using PCA-transformed data.

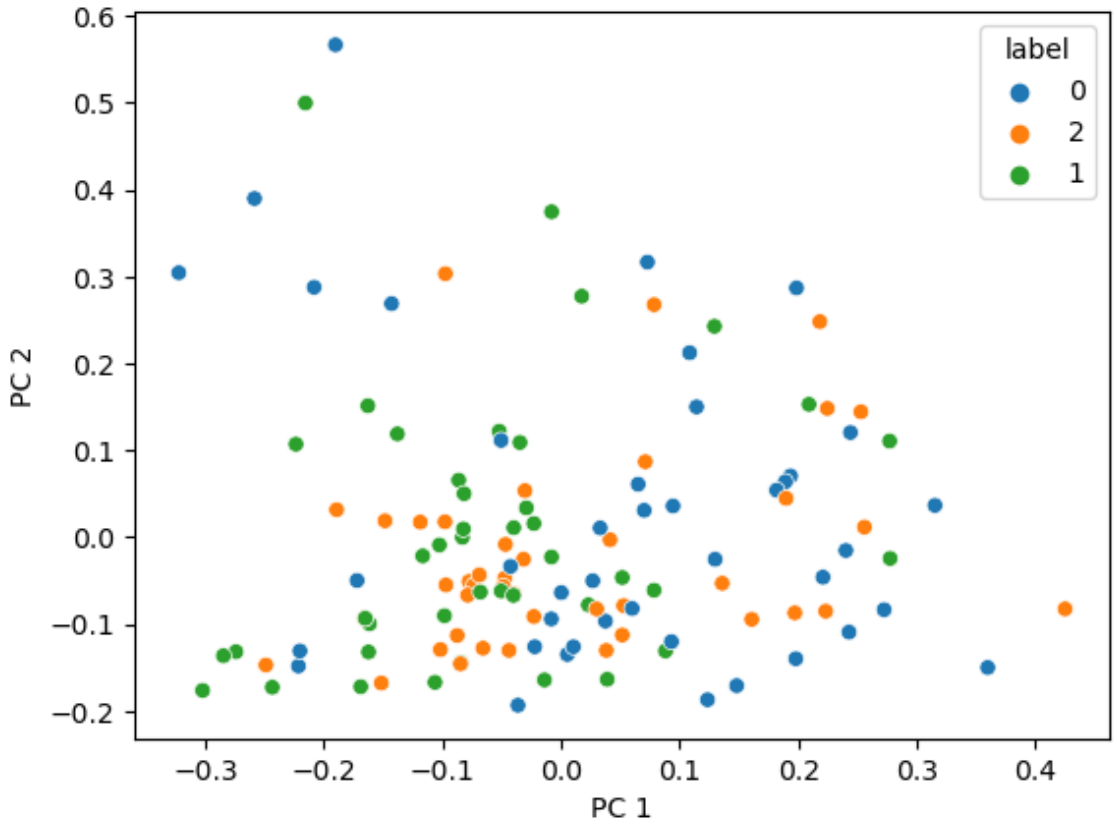




The KNN classifier achieved an overall accuracy of 33.33%, which indicates that the PCA transformation did not lead to a significant improvement in the KNN classifier's performance. Moreover, the accuracy by class only improved for Flower2, with Flower1 having an accuracy of 0.6, Flower2 having an accuracy of 0.25, and Flower3 having an accuracy of 0.25. Hence, it can be concluded that the PCA transformation did not provide a significant benefit for the KNN classification model's performance.

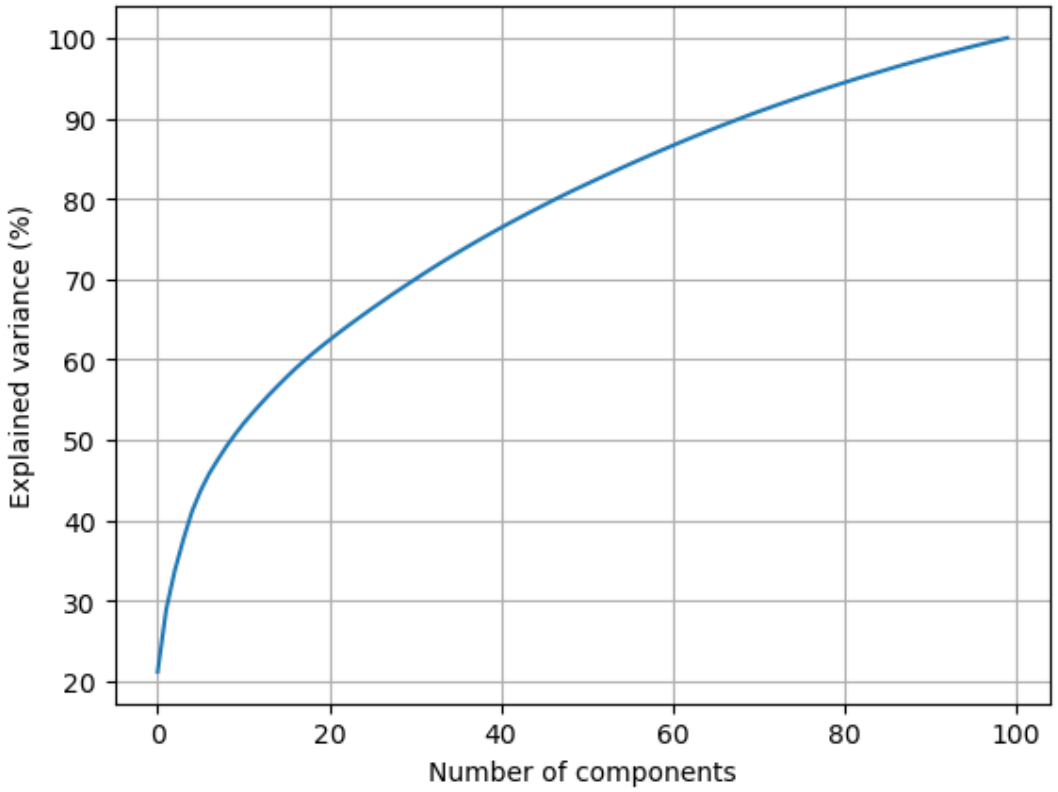
KNN Classification on kPCA Transformed Data

Kernel PCA (kPCA) is another technique for dimensionality reduction. To explore the potential benefits of kPCA, I applied it to the dataset. I used cosine kernel with 2 components and 100 components to transform the data and plot the explained variance.

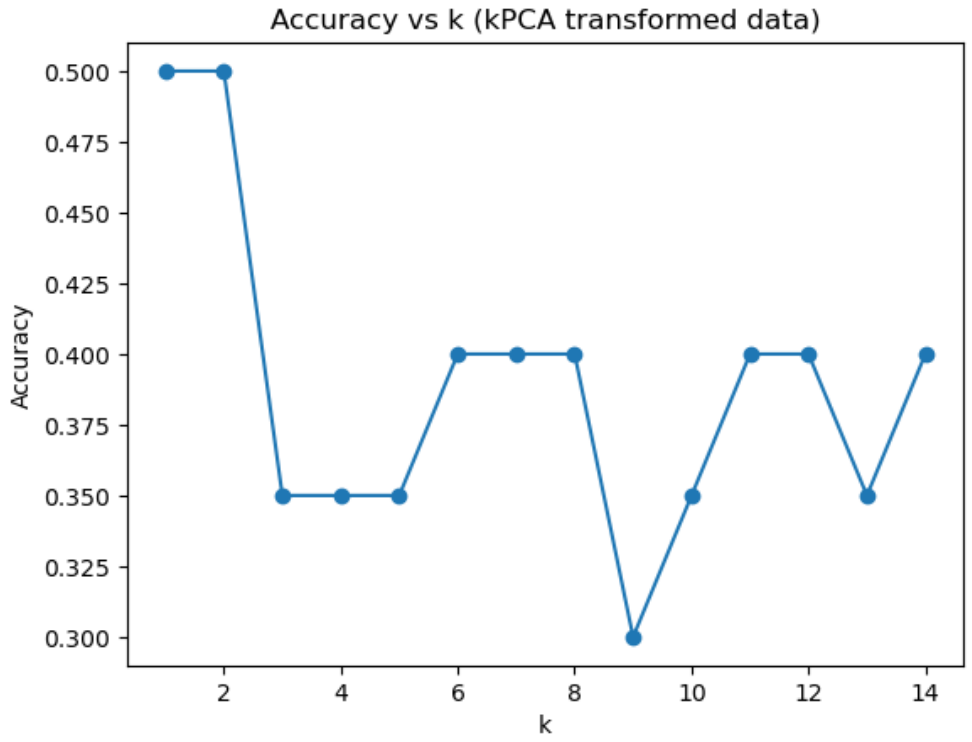


Although the scatter plot of the transformed data using two components showed a slightly improved separation among the three flower classes compared to the scatter plot of the PCA-transformed data, the separation was not significant. However, the kPCA scatter plot showed a better separation between Flower1 and the other two classes. Nonetheless, there was still some overlap between Flower2 and Flower3.

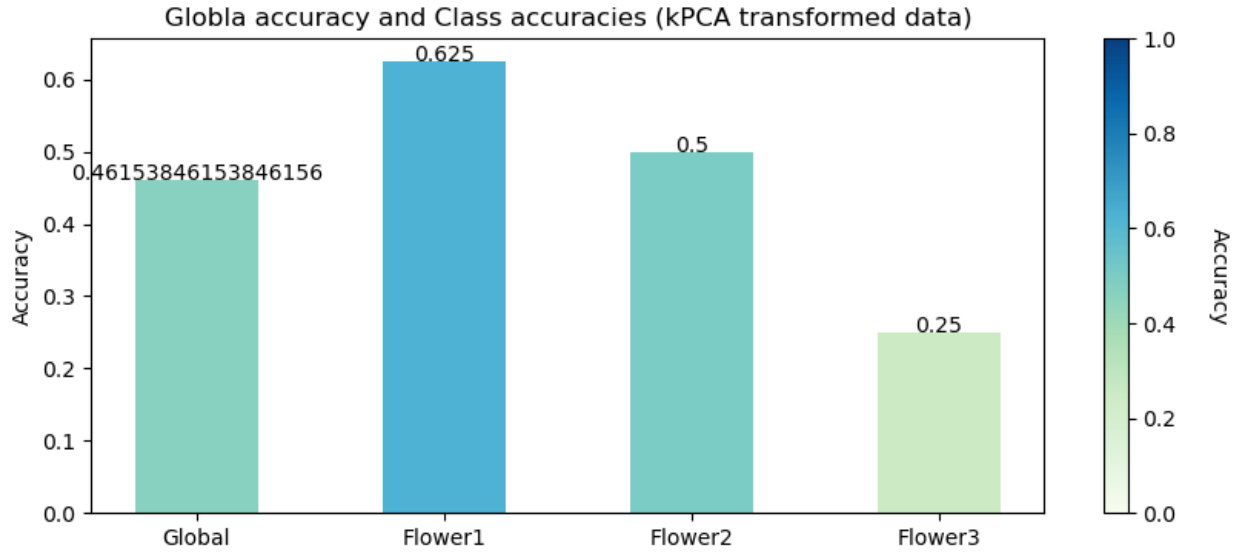
Similarly to the PCA, the scree plot was generated to determine the optimal number of principal components to use in the KNN classification model using kPCA transformed data.

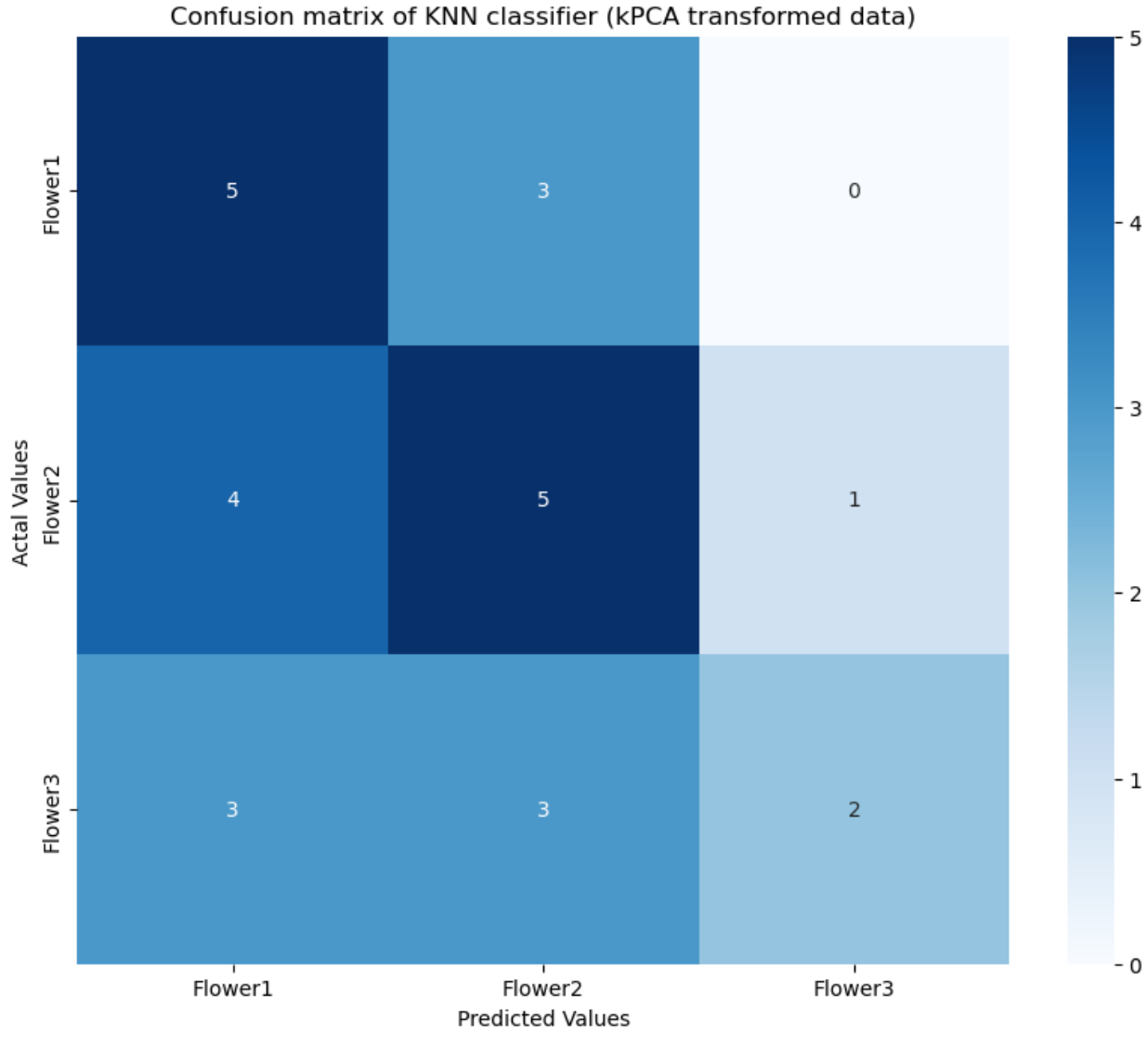


The plot indicated that about 90% of the variance in the data could be represented by approximately 68 components. With this information, I performed KNN classification on the kPCA-transformed data using the 68 principal components that accounted for most of the variance in the data.



To evaluate the performance of the KNN classifier on the transformed data, I experimented with different values of k, ranging from 1 to 14. The resulting accuracy scores were plotted, which showed that the KNN classifier achieved the highest accuracy of about 0.46 when the k value was set to 2.

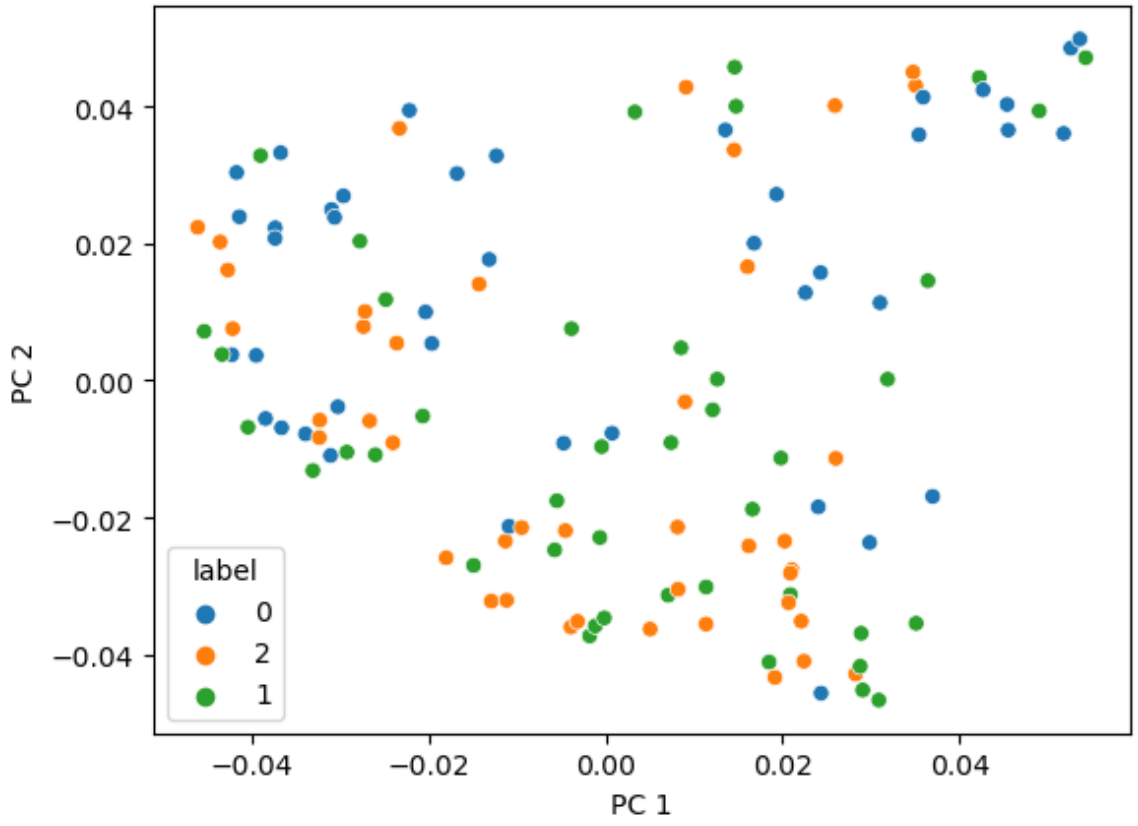




The performance evaluation of the KNN classifier using kPCA-transformed data reveals that it achieved better results than when using PCA-transformed data. The overall accuracy of the KNN classifier improved to 46.15%, with Flower1 having an accuracy of 0.625, Flower2 having an accuracy of 0.5, and Flower3 having an accuracy of 0.25. The F1 score was highest for Flower1, at 0.50, while Flower3 had the lowest F1 score of 0.36. This suggests that kPCA transformation is a more effective technique for improving the KNN classifier's performance, especially for high-dimensional data. Additionally, the TP value increased to 12, indicating that the KNN classifier correctly classified more instances when using kPCA-transformed data. These results suggest that kPCA transformation can be a useful technique for improving the classification performance of the KNN classifier, especially when dealing with complex data.

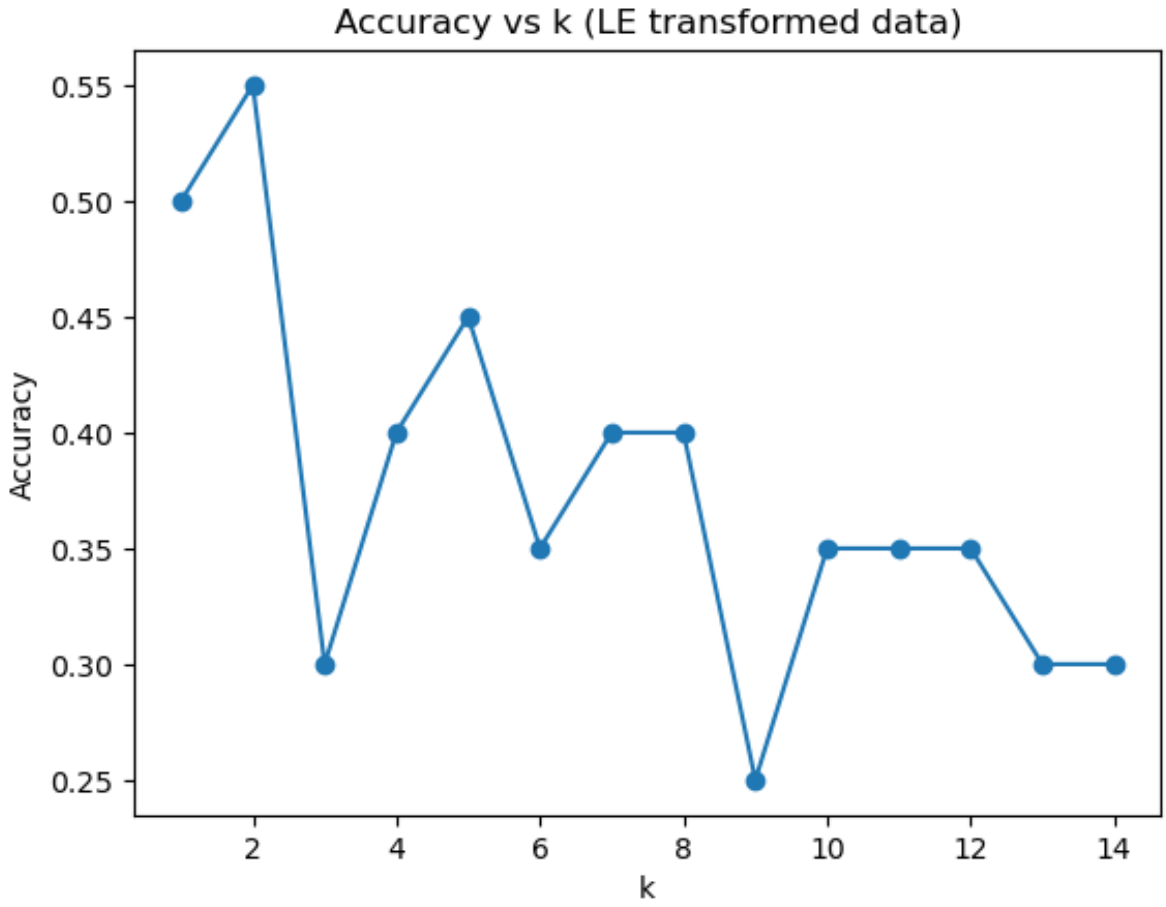
KNN Classification on Laplacian Eigenmaps Transformed Data

In addition to KNN classification on PCA and kPCA transformed data, I also applied Laplacian Eigenmaps (LE) transformation to the data and performed KNN classification. The LE transformation involves finding a low-dimensional embedding of the data that preserves the pairwise distance between the data points. I used the SpectralEmbedding method from scikit-learn to perform the transformation, which resulted in a two-dimensional embedding of the data.

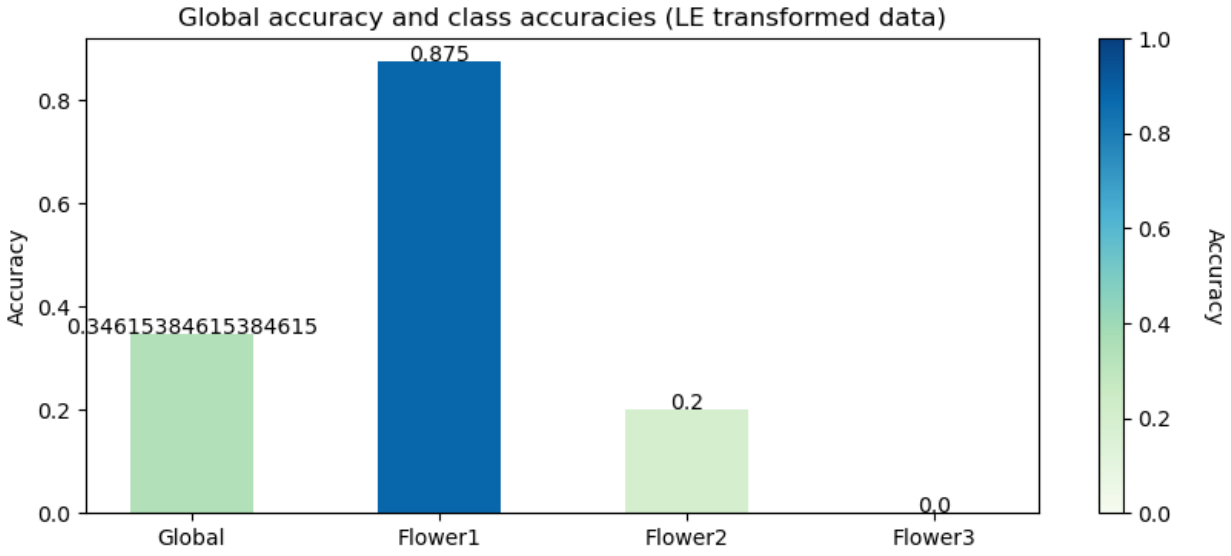


The scatter plot of the Laplacian Eigenmaps (LE) transformed data revealed that Flower1 was well separated from Flower2 and Flower3, with some overlap between Flower2 and Flower3.

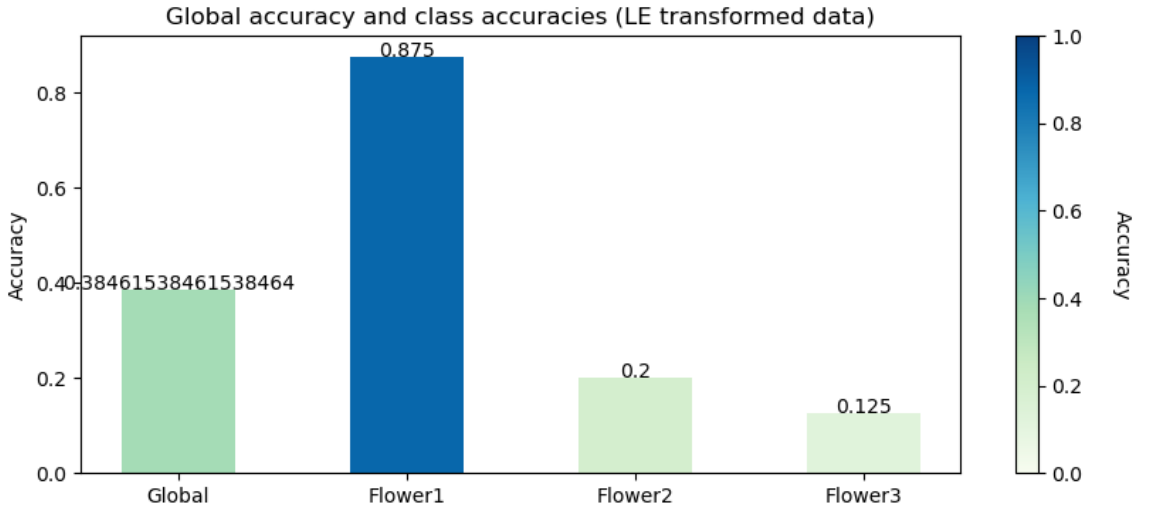
Then I performed KNN classification on the LE transformed data, and assessed the accuracy of the classifier for a range of k values from 1 to 14.

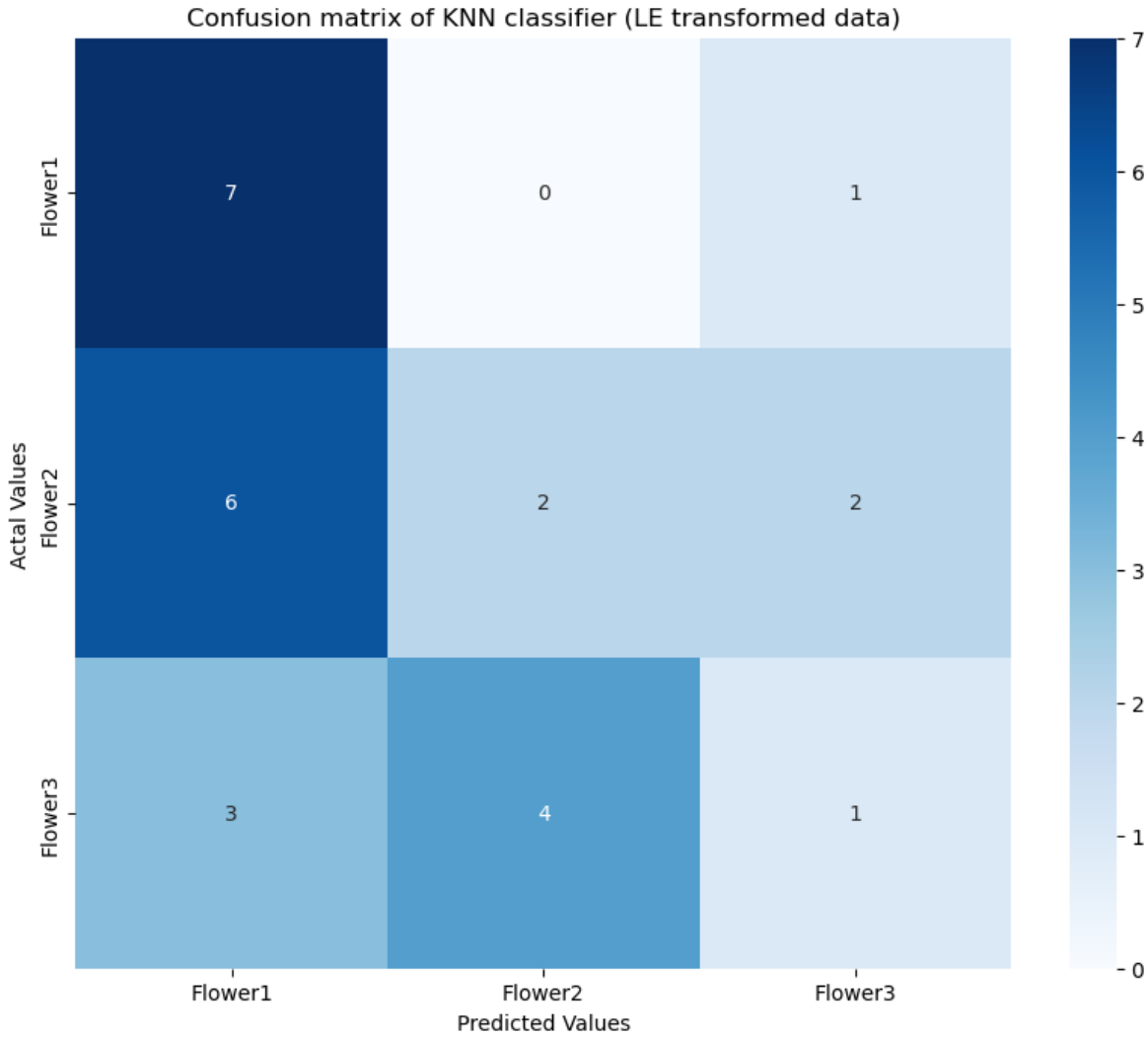


The plot generated revealed that the optimal k value for the KNN classifier was 2, but it failed to classify instances of Flower3.



As a result, I chose the next best k value, which was 5. This value resulted in an accuracy score of approximately 0.38, which was an improvement over the optimal k value of 2. More importantly, this k value was able to classify instances of Flower3 correctly, making it a more practical and suitable choice.

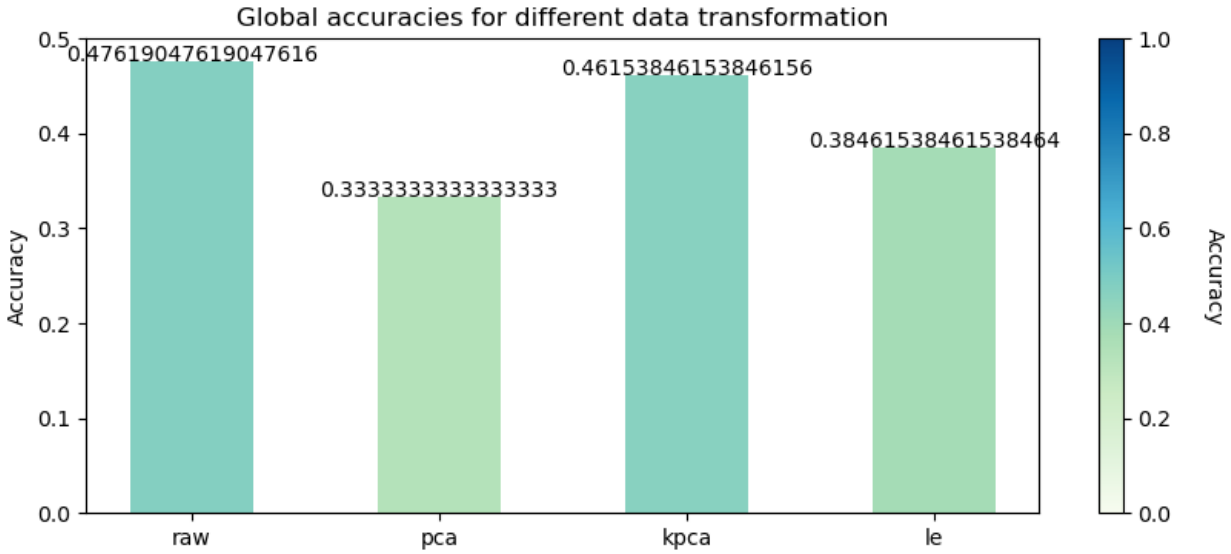




The results showed that the overall accuracy of the model was 38.46%, indicating that the LE transformation did not improve the baseline performance of the KNN classifier significantly. However, it is worth noting that the LE transformed data still performed better than PCA transformed data. Among the three flower classes, Flower1 had the highest accuracy of 0.875, indicating that the KNN classifier was effective in distinguishing Flower1 from the other two classes. The accuracy scores for Flower2 and Flower3 were lower, at 0.2 and 0.125, respectively. The F1 scores for these classes were 0.58333333, 0.25, and 0.16666667 for Flower1, Flower2, and Flower3, respectively. Overall, the LE transformation did not lead to a significant improvement in the KNN classifier's performance, but it still outperformed PCA transformation in terms of accuracy for Flower1.

Conclusion

In conclusion, this project aimed to explore the effectiveness of different representation methods, such as PCA, kPCA, and Laplacian Eigenmaps, on improving the performance of the KNN classifier in classifying flower species. The evaluation results showed that kPCA and Laplacian Eigenmaps outperformed PCA in terms of overall accuracy and accuracy by class.



Although neither representation methods improved the overall baseline performance of the KNN classifier, they both showed promising results in improving the accuracy for specific classes. These findings demonstrate the potential benefits of using representation methods for high-dimensional datasets in machine learning applications.