

PRML ASSIGNMENT - 3

Name : Aditya Sahani

Roll No : B22CS003

Google Colab :

https://colab.research.google.com/drive/1wyEKX9_An5IEcUw4DGJzJMUKyZHd0EVe#scrollTo=wrXyz3tOsBwW

PERCEPTRON

Task 0: Generating Synthetic Dataset

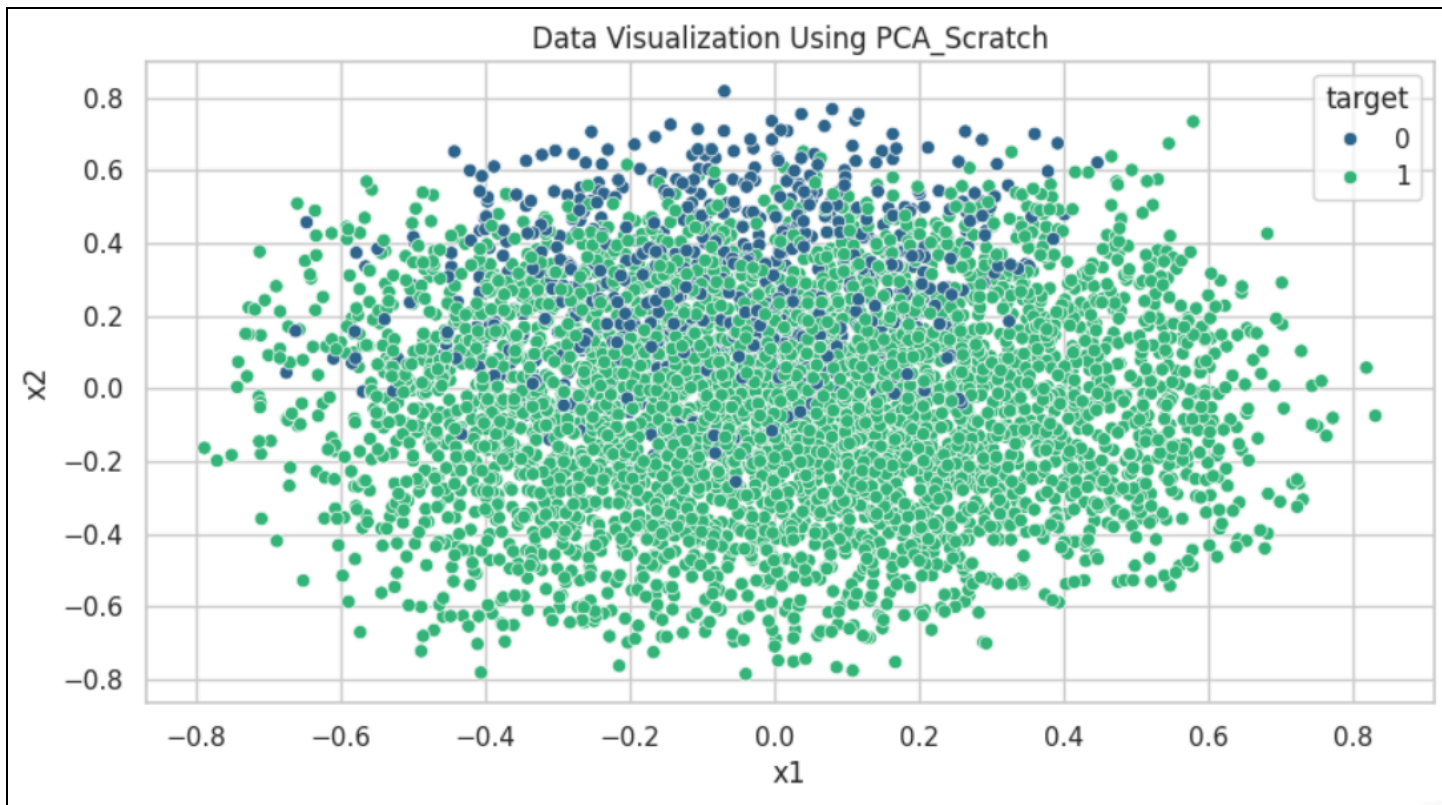
A synthetic four-dimensional dataset was generated by randomly initializing weights and using them to compute a function. This function was then used to classify data points into two classes. Each data point consists of four features (x1, x2, x3, x4) and a binary label (label).

The dataset was saved as both a CSV file (data.csv) and a text file (data.txt).

Task 1: Training the Model

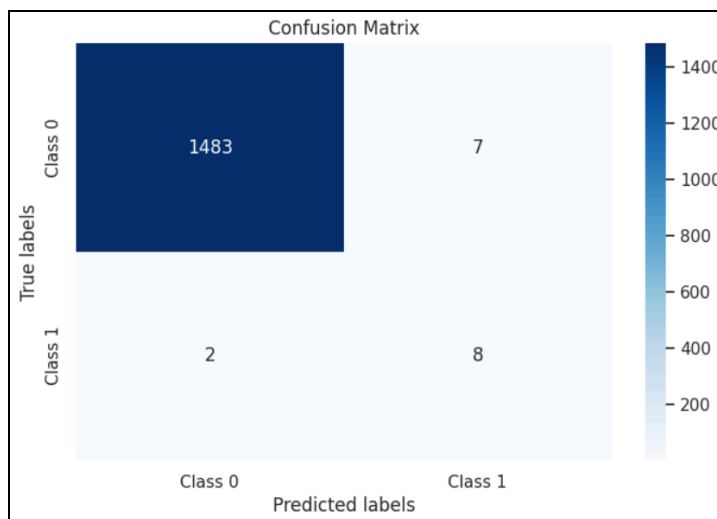
The Perceptron model was trained using the synthetic dataset. The dataset was preprocessed using **MinMaxScaler** to scale the features **between 0 and 1**. The model achieved an accuracy of approximately **0.9954** on the test set.

A visualization of the dataset was performed using PCA (Principal Component Analysis) to reduce the dimensionality to two components for plotting.



Task 2: Testing the Model

The trained Perceptron model was tested on the test data, and its accuracy was calculated. Additionally, a confusion matrix was plotted to visualize the model's performance in terms of true positive, false positive, true negative, and false negative predictions.



Task 3: Checking Accuracy for Different Test Sizes

The accuracy of the Perceptron model was evaluated for different train sizes (20%, 50%, and 70%). For each test size, the dataset was split into training and testing sets, and the model was trained and tested accordingly. The accuracies achieved were approximately 0.9932, 0.9924, and 0.994 for train sizes of 20%, 50%, and 70%, respectively.

Percentage Split (Train_Size)	Accuracy
0.2	99.32 %
0.5	99.24 %
0.7	99.4 %

Reason for HIGH ACCURACY:

- As we are creating our own synthetic dataset, the data points are linearly separable so we can separate the data points effectively by a line.
- Data also contains a large number of points(around 5000), it provides ample information for the Perceptron model to learn and generalize well to unseen data.
- An appropriate learning rate of **0.01** is chosen, a well-tuned learning rate ensures that the model converges efficiently to the optimal solution without overshooting or diverging.

Reason for not 100% ACCURACY:

- Initialization: The initial weights of the Perceptron model can influence its convergence and final performance. If the initial weights are not set properly, it may take longer for the model to converge or converge to a suboptimal solution.

- Data Distribution: Even in linearly separable datasets, the distribution of data points may not be uniform across classes. Class imbalance or clustering of data points can lead to misclassifications, reducing the overall accuracy.
 - Convergence Criterion: The stopping criterion used for training the Perceptron model can affect its final accuracy. If the convergence criterion is not strict enough, the model may stop training before reaching a perfect solution.
 - Number of Iterations: The number of iterations or epochs used for training the Perceptron model can impact its performance. If the model is trained for a limited number of iterations, it may not have sufficient time to converge to the optimal solution.
-

PRINCIPAL COMPONENT ANALYSIS(PCA)

Task 1: Data Preprocessing

In this task, the face dataset from the Labeled Faces in the Wild (LFW) database was preprocessed to prepare it for further analysis. The following steps were performed:

Dataset Loading: The dataset was loaded using the `fetch_lfw_people` function from `sklearn.datasets`. This function retrieves face images from the LFW dataset, ensuring a minimum threshold of 70 images per person and resizing them to 40% of their original size.

Data Exploration: The dimensions of the dataset were explored to understand its structure. The dataset consisted of a total of 1288 images, each with a height of 50 pixels and a width of 37 pixels, resulting in a total of 1850 features.

Data Splitting: The dataset was split into training and testing sets in an 80:20 ratio using the `train_test_split` function from `sklearn.model_selection`. This step ensured that a portion of the data was reserved for evaluating the performance of the trained models.

Data Scaling: To ensure that all features have the same scale and to improve the performance of machine learning algorithms, the data was standardized using `StandardScaler` from `sklearn.preprocessing`. This step involved subtracting the mean and dividing by the standard deviation for each feature.

Task 2: EigenFaces Implementation

In this task, Principal Component Analysis (PCA) was implemented from scratch to perform dimensionality reduction on the face dataset. The following steps were carried out:

PCA Implementation: The `PCA_Scratch` class was defined to encapsulate the PCA implementation using eigenvectors and eigenvalues computation. This class included methods to fit the data by calculating the mean, standard deviation, covariance matrix, and extracting principal components based on sorted eigenvalues.

Projection and Reconstruction: The dataset was projected onto the principal components obtained from PCA. The projected data was then reconstructed to visualize the effect of dimensionality reduction on the original data.

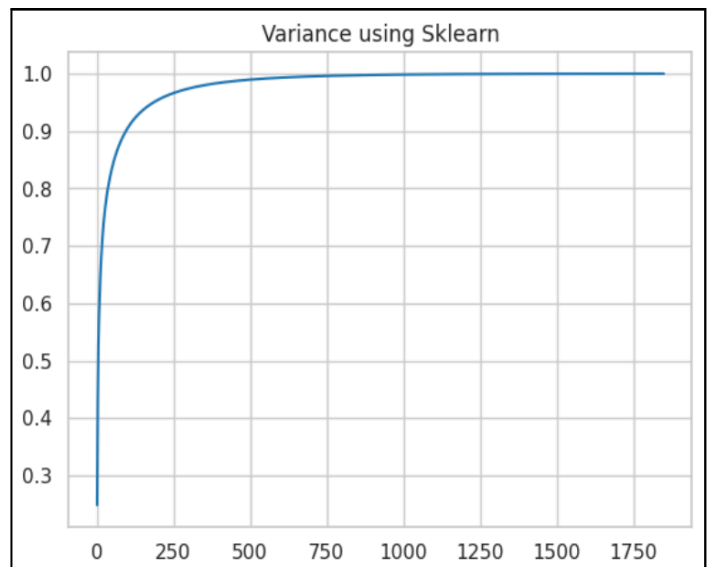
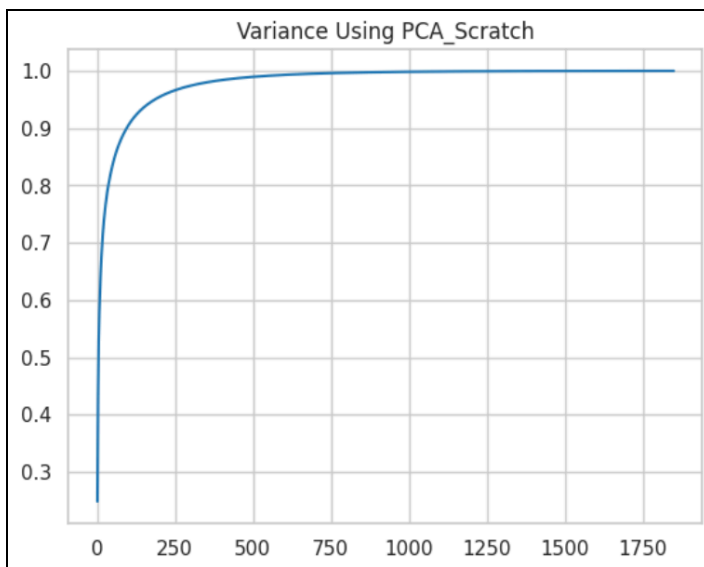
Explained Variance Ratio:

- It represents the amount of variance explained by each of the selected components.

- After performing PCA on your data, you get a set of principal components.
- Each principal component captures a certain amount of variation in the original data.
- The 'explained_variance_ratio_' attribute gives the ratio of variance explained by each of these components to the total variance in the data.
-

The explained variance ratio was calculated to determine the proportion of variance explained by each principal component. This information provided insights into the importance of each component in capturing the variability of the data.

Using Explained Variance Ratio, we determined the optimum value of `n_components` that is capturing **more than 90% variance of the data**.



From the above graph, we can see that beginning from `n_components` = 100, we are getting a variance of more than 90%.

Appropriate value of `n_components` would be around 100 because for `n_components` = 100, we are able to capture more than 90% of variance of the data. Accuracy of the model for `n_components` = 100 is **63.54%** for neighbors = 5.

Task 3 and Task 4: Model Training and Evaluation

In this task, the dataset was projected and reconstructed using both the PCA implementation from scratch and the PCA implementation from sklearn. The reconstructed data was visualized, and Eigenfaces were extracted to represent facial features. Additionally, the following steps were performed:

Model Training: Various classifiers including K-Nearest Neighbors (KNN), Decision Tree, and Random Forest were trained on the transformed data to recognize faces.

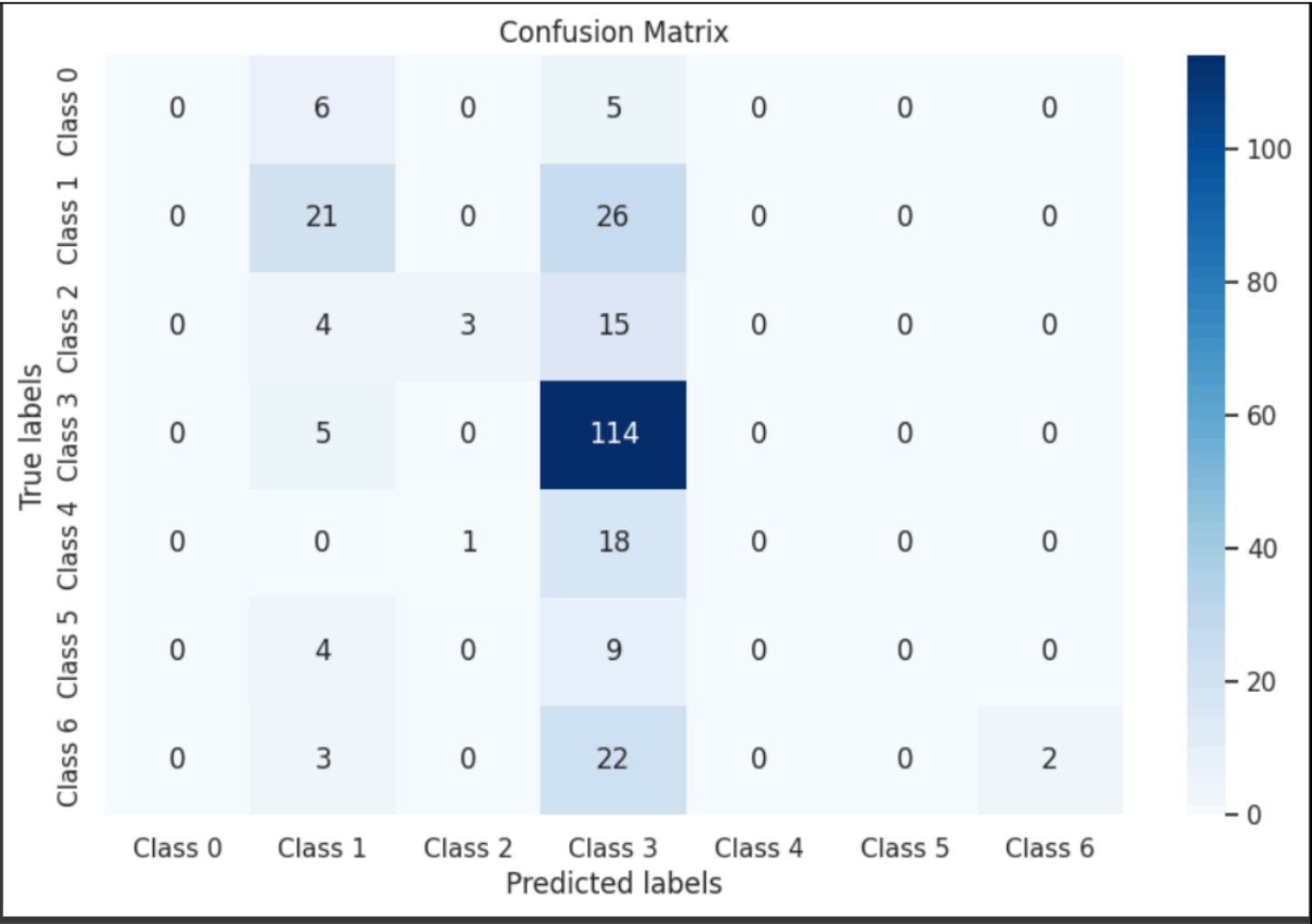
Model Evaluation: The trained classifiers were evaluated using the testing data to assess their performance in face recognition. Accuracy scores were computed to measure the effectiveness of each classifier.

ACCURACY:











	n_components	KNN_train	KNN_test	DTC_train	DTC_test	RFC_train	RFC_test
0	2	0.400971	0.457364	0.564078	0.406977	0.645631	0.403101
1	12	0.480583	0.511628	0.860194	0.422481	0.979612	0.608527
2	22	0.519417	0.546512	0.930097	0.484496	0.998058	0.705426
3	32	0.520388	0.538760	0.932039	0.449612	0.998058	0.705426
4	42	0.519417	0.546512	0.937864	0.472868	0.997087	0.689922

CLASSIFICATION REPORT:

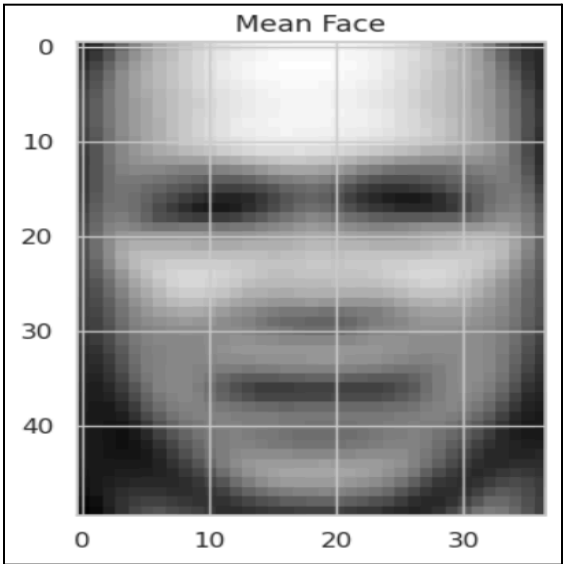
	precision	recall	f1-score	support
Class 0	0.00	0.00	0.00	11
Class 1	0.49	0.45	0.47	47
Class 2	0.75	0.14	0.23	22
Class 3	0.55	0.96	0.70	119
Class 4	0.00	0.00	0.00	19
Class 5	0.00	0.00	0.00	13
Class 6	1.00	0.07	0.14	27
accuracy			0.54	258
macro avg	0.40	0.23	0.22	258
weighted avg	0.51	0.54	0.44	258



PREDICTION :

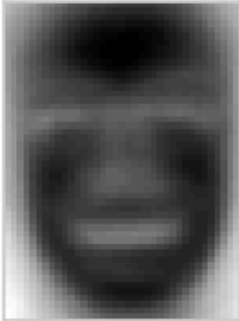
<div>Predicted: Donald Rumsfeld Actual: Donald Rumsfeld</div> 	<div>Predicted: Colin Powell Actual: Donald Rumsfeld</div> 	<div>Predicted: George W Bush Actual: Colin Powell</div> 	<div>Predicted: George W Bush Actual: George W Bush</div> 	<div>Predicted: George W Bush Actual: Tony Blair</div> 
<div>Predicted: George W Bush Actual: George W Bush</div> 	<div>Predicted: Colin Powell Actual: Colin Powell</div> 	<div>Predicted: George W Bush Actual: George W Bush</div> 	<div>Predicted: George W Bush Actual: George W Bush</div> 	<div>Predicted: George W Bush Actual: George W Bush</div> 

MEAN FACE



EIGENFACES USING PCA FROM SCRATCH

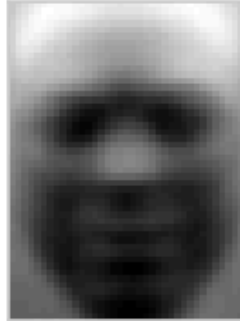
Eigenface 1



Eigenface 2



Eigenface 3



Eigenface 4



Eigenface 5



Eigenface 6



Eigenface 7



Eigenface 8



Eigenface 9



Eigenface 10



Original Face



Reconstructed Face using 1850 Eigenfaces



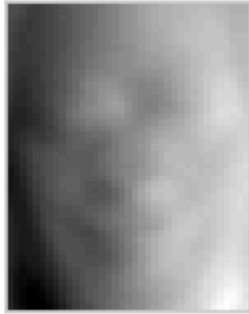
EIGENFACES USING PCA FROM SKLEARN

Top-10 Eigenfaces Using Sklearn

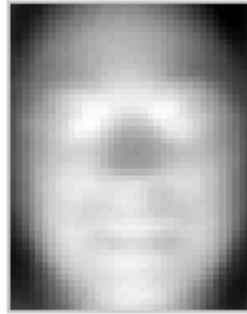
Eigenface 1



Eigenface 2



Eigenface 3



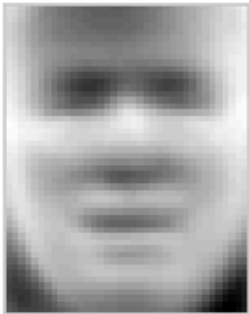
Eigenface 4



Eigenface 5



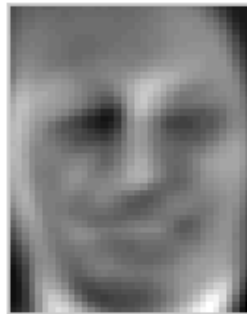
Eigenface 6



Eigenface 7



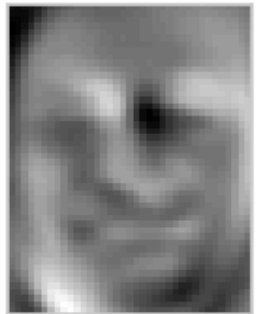
Eigenface 8



Eigenface 9



Eigenface 10



Original Face



Reconstructed Face using 1850 Eigenfaces



TYPES OF IMAGES ON WHICH MODEL IS FAILING :

Class Imbalance: The dataset may have a class imbalance since it primarily focuses on one individual. This imbalance could lead to biases in the model's predictions, favoring images of George W. Bush over others.

Generalization to Other Faces: Since the dataset is heavily skewed towards one person, the model may not generalize well to faces of other individuals. This could result in lower accuracy when classifying faces other than George W. Bush's.

Translational Distortion : When we convert a 2D image to 1D, information regarding the relative position of the pixels is lost, hence the nature of the image does not remain the same, leading to poor training.

Outliers: Images which are outlier to the normal distribution of the images, will be classified wrongly.

Poor Image Quality: Images with low resolution, noise, or poor lighting conditions may be challenging for the model to correctly classify.

METHODS TO IMPROVE THE MODEL :

Data Augmentation: Increase the diversity of the training set by applying transformations such as rotation, scaling, and flipping to images. This can help the model generalize better to variations in pose, expression, and lighting.

Model Architecture: Consider using more complex architectures such as deeper convolutional neural networks (CNNs) or pretrained models like VGG, ResNet, or EfficientNet. These models are better equipped to learn intricate patterns in images.

Data Cleaning: Remove noisy or irrelevant images from the training set to reduce confusion for the model.

Hyperparameter Tuning: Experiment with different hyperparameters such as learning rate, batch size, and optimizer to find the optimal configuration for your model.

Ensemble Learning: Train multiple models and combine their predictions to improve accuracy. This can help mitigate errors caused by similarity between faces.

Task 5: Experimentation with different n_components

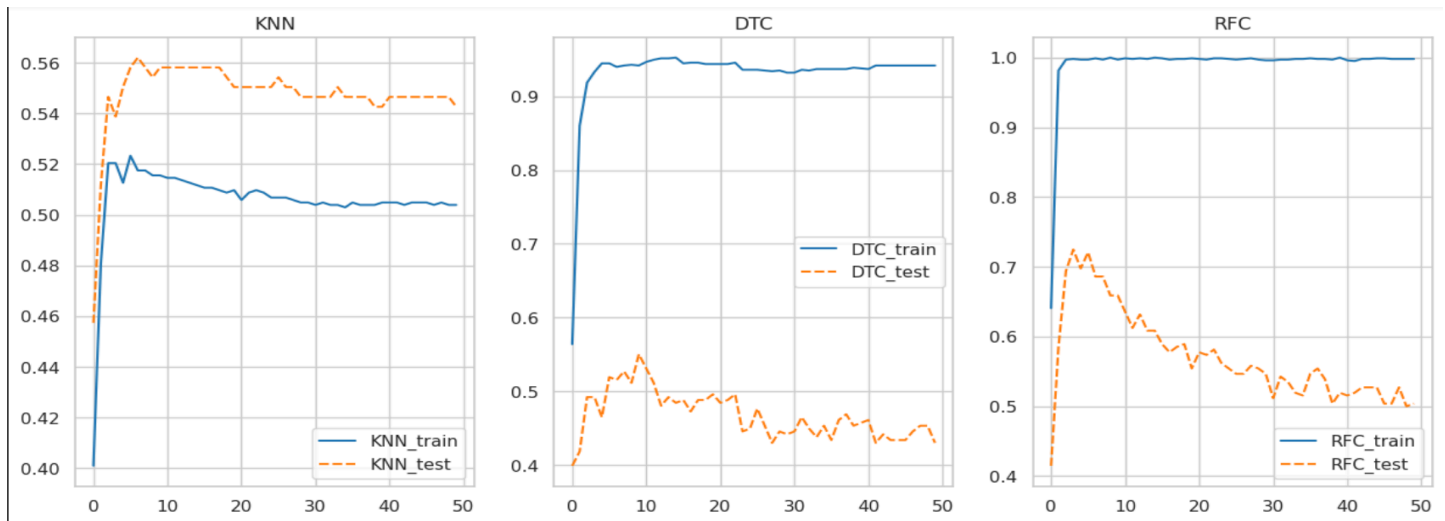
In this task, the impact of different numbers of components (n_components) on model training and evaluation was investigated. The dataset was transformed using PCA with varying numbers of components, and classifiers were trained and evaluated accordingly. The following steps were conducted:

Dimensionality Reduction: The dataset was transformed using PCA with different numbers of components to observe the effect on model performance.

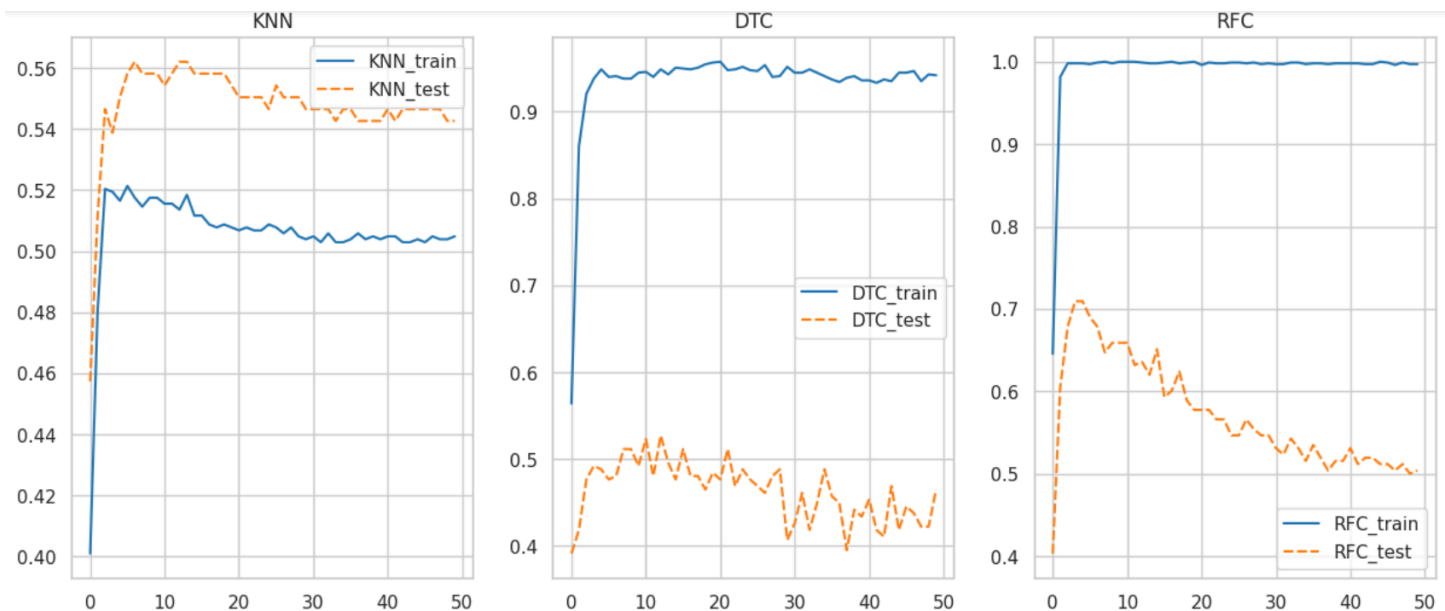
Classifier Training and Evaluation: For each number of components, classifiers such as KNN, Decision Tree, and Random Forest were trained and evaluated. The training and testing accuracies were recorded and plotted against the number of components to visualize the impact of dimensionality reduction on model performance.

Models were trained for both PCA_Scratch class and PCA using Sklearn to note the difference between accuracies. Accuracies were almost the same for both the classes, suggesting that our implementation is working fine.

PCA FROM SCRATCH

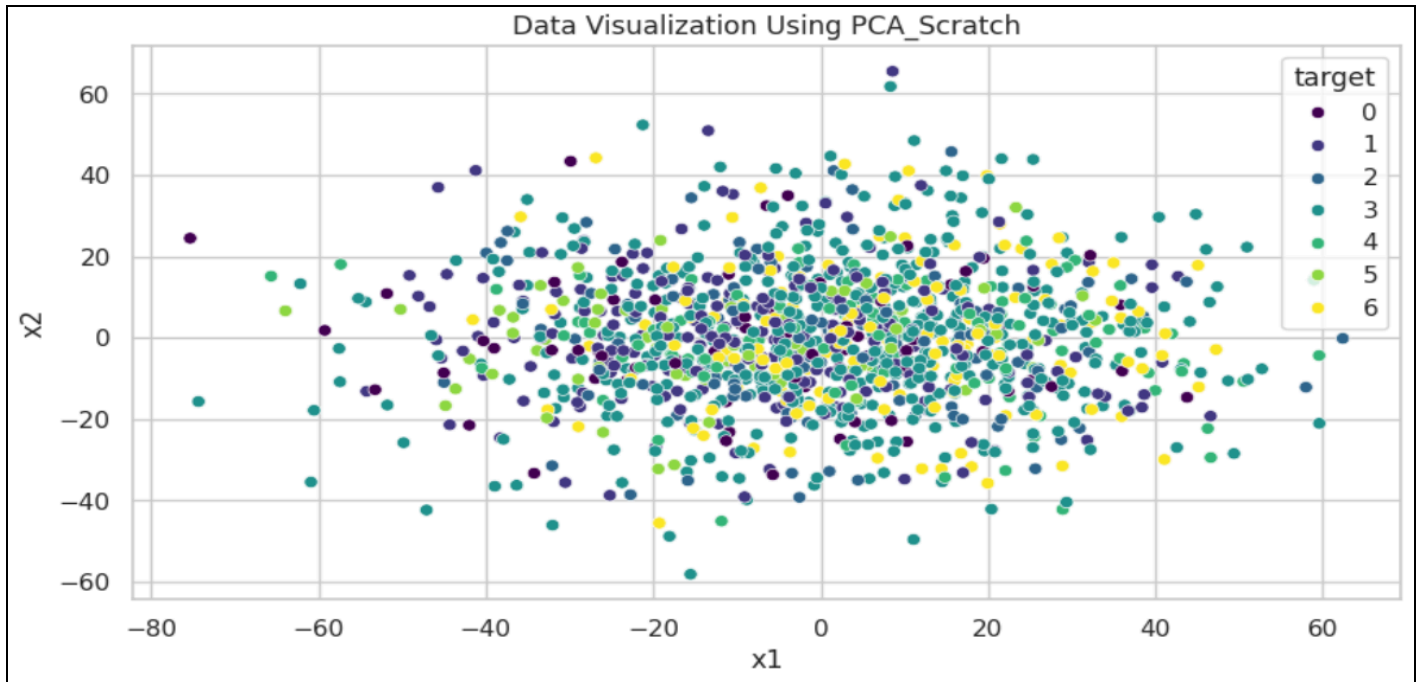


PCA FROM SKLEARN



From the graphs, it is clear that KNN is performing better than other 2 models. Decision Tree and Random Forest Classifier is overfitting over the data.

Reason for good performance of KNN :



- Above scatterplot shows the projection of the dataset over 2 axes. Data is too complex and points belonging to one class are clustered together. KNN assigns each data point to the cluster represented by the majority of its k nearest neighbors, based on some distance metric (e.g., Euclidean distance).
- KNN is robust to outliers in the data because it considers the local neighborhood of each data point rather than the global structure of the data. Outliers are less likely to significantly influence the clustering results, especially when using larger values of k .

Reason for bad performance of DTC and RFC:

- As the number of dimensions (features) increases, the space in which the data resides becomes increasingly sparse. This phenomenon, known as the curse of dimensionality, can lead to overfitting as decision trees try to partition the data space into smaller and smaller regions.
- With higher dimensional data, decision trees tend to become more complex, resulting in deeper trees with many nodes. This complexity can lead to overfitting, where the model captures noise in the data rather than the underlying patterns.

- Clustering algorithms typically optimize some measure of similarity or dissimilarity between data points, which may not align with the impurity measures used in decision trees.