Mathematics and Applications of Eigenfaces

Brian K. Too
[1] Mathematics of Machine Learning,
Howard University, Washington, D.C.

May 3rd 2024

1 Abstract

This paper explores the application of machine learning techniques to the National Institute of Standards and Technology (NIST) Mugshot dataset, aiming to classify individuals based on racial characteristics using an eigenface model coupled with a Support Vector Machine (SVM). The study primarily focuses on distinguishing between Black and non-Black individuals, addressing the challenges and implications of using facial recognition technologies in sensitive contexts. Our methodology involves the preprocessing of 3,248 segmented 8-bit grayscale mugshot images, followed by the application of Principal Component Analysis (PCA) to extract eigenfaces, and the use of SVM for classification. The model was evaluated on its ability to generalize across varied datasets, with particular attention to biases introduced during data preparation and model construction. Results indicate that while the model achieves a balance between precision and recall with an overall accuracy of 81.60%, it also exhibits potential biases in racial classification. This raises significant ethical, legal, and social concerns, particularly regarding racial profiling. The paper concludes with recommendations for improving model fairness through diversified data and advanced bias-mitigation techniques and stresses the importance of responsible deployment in real-world applications.

2 Introduction

The utilization of Machine Learning (ML) in automated decision-making processes has become increasingly prevalent across various fields such as education, employment, and law enforcement [1]. However, the deployment of these technologies often raises significant ethical concerns, partially regarding bias and fairness in algorithmic decisions [1]. This paper explores the application of Machine Learning techniques to the National Institute of Standards and Technology (NIST) Mugshot dataset with the objective of predicting racial background, specifically classifying convicts as Black or non-Black. With these results, our model will show if there is a higher percentage of Black convicts or convicts. With clean data and a model constructed on Jupiter Notebook, observing the accuracy of this model will allow us to determine the reliability of such techniques in place of human decision-making. The study aims to shed light on potential biases embedded within facial recognition algorithms and to assess the implications of using such models in real-world applications.

Facial recognition technology, despite its widespread adaptation, has been criticized for perpetuating and even amplifying existing racial biases[1]. In this project, we employ an eigenface model, a foundational approach in facial recognition that utilizes principal component analysis (PCA) to reduce dimensionality and extract significant features from facial images. Then, we apply a Support Vector Machine (SVM) classifier to categorize individuals based on the derived eigenfaces, providing a clear framework for studying the model's behavior across different racial backgrounds.

The selection of the NIST Mugshot dataset, which comprises 3,248 segmented 8-bit grayscale mugshot images of 1,573 individuals, allows for a controlled study of racial classification in a law enforcement context[2]. By focusing on this dataset, we aim to critically analyze the accuracy and fairness of the eigenface and SVM-based model when applied to racial classification. A central question guiding our research is the ethicality behind using such ML models for determining demographic composition. Especially in high-impact scenarios such as assessing institutional diversity for applicants or current bodies based on individuals' images. This involves not only technical evaluations of model performance but also a broader discussion on the ethical, legal, and social implications of deploying facial recognition technologies in sensi-

tive and impactful contexts.

This paper is structured as follows: the first section is a background that provides a detailed description of the methodology, including model training and evaluation metrics. The subsequent sections present the results and discuss the findings in light of current ethical standards and practices in AI. Finally, we conclude with recommendations for minimizing bias in ML models and considerations for their responsible use in society.

3 Background

3.1 Principal Component Analysis

Principal Component Analysis (PCA) is a method used to simplify a complex data set by transforming it into a new coordinate system. In this new system, the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on[3]. PCA is typically performed by first, standardizing the data set. Consider a dataset with $d+1$ dimensions, where d represents features and the additional dimension could be the label or target variable. For PCA, we focus only on the feature set, reducing the dimensionality to d[3]. For a simplified example, assume $d = 3$ (three features across our dataset). Then we compute the mean of each illustration[3]. To illustrate this, consider a simple dataset where each row represents a student, and each column represents their scores in different subjects:

$$\text{Math:} \quad \frac{60+90+70}{3} = 73.33$$

$$\text{English:} \quad \frac{70+60+90}{3} = 73.33$$

$$\text{Art:} \quad \frac{80+70+60}{3} = 70.00$$

Now we compute the covariance matrix. The covariance matrix expresses the covariance of each pair of features in the dataset. This matrix is pivotal in PCA because it helps identify the directions in which the data varies the most[3]. With our covariance matrix, we will compute eigenvectors and corresponding eigenvalues. Our eigenvectors and eigenvalues help to understand the directions of maximum variance in the data. The mathematical details on how to calculate these are critical for correctly applying PCA[3]. The next PCA step involves sorting eigenvectors by decreasing eigenvalues and constructing matrix W. After calculating eigenvectors and eigenvalues, sort them by the magnitude of the eigenvalues. The top k eigenvectors are selected to form a new matrix W, which has the dimensions $d \times k$. For a reduction to two dimensions, you select the top two eigenvectors[3]. The final step involves using the matrix W to transform the original dataset into a new subspace[3]. This transformation is achieved by projecting the original data onto the space defined by the selected eigenvectors:

$$Y = W^T X \tag{1}$$

This transformation results in a new dataset where the most significant variances are captured in the first few dimensions[3]. PCA helps in reducing the dimensionality by identifying the directions along which the variance of the data is maximized. It's particularly useful in preprocessing steps for machine learning algorithms that perform poorly with high-dimensional data due to the curse of dimensionality.

3.2 Eigenfaces

Eigenfaces are essentially the principal components derived when PCA is applied to a collection of face images represented as high-dimensional vectors[4]. The process begins by vectorizing each image in the training set. If each image is $m \times n$, it can be converted into a vector of dimension mn, where each pixel's intensity becomes an element of the vector[4].

3.2.1 Average Face

First, an average face vector $\bar{f}$ is computed from all the image vectors in the training set[4]:

$$\bar{f} = \frac{1}{N} \sum_{i=1}^{N} f_i \tag{2}$$

where f_i is the vector representation of the i-th face image and N is the total number of images[4].

3.2.2 Covariance Matrix

The difference between each face vector and the average face vector is calculated, leading to a set of deviation vectors[4]:

$$d_i = f_i - \bar{f} \tag{3}$$

These deviation vectors are used to form a covariance matrix C, representing the variations between the face vectors:

$$C = \frac{1}{N-1} \sum_{i=1}^{N} d_i d_i^T \tag{4}$$

Performing PCA on this covariance matrix involves calculating its eigenvectors, which yield the eigenfaces[4]. These eigenvectors (termed eigenfaces) are the principal components that capture the directions in which the face vectors most significantly vary from the average face[4].

3.2.3 Weights and Feature Vectors

The importance of eigenfaces becomes evident when considering the concept of weights. If we have a collection of basic facial patterns—these are our eigenfaces. To describe a specific person's face using these patterns, it is necessary to determine the coefficients (weights) that combine the eigenfaces to reconstruct the original face vector[4]. For an image vector f_i, the weights are given by:

$$w_{ik} = d_i \cdot u_k \tag{5}$$

where u_k is the k-th eigenface and w_{ik} is the weight or coefficient corresponding to the k-th eigenface for the i-th image. Each weight corresponds to one of the eigenfaces, and collectively, they form a feature vector[4]:

$$\text{Feature vector for } f_i = [w_{i1}, w_{i2}, \ldots, w_{ik}]$$

These feature vectors are pivotal for facial recognition and classification tasks[4]. For instance, when distinguishing between Black and non-Black individuals in our project, the feature vectors—comprised of these weights—are used as input for classification algorithms. The ability of these feature vectors to encapsulate key facial features allows classifiers to efficiently and effectively differentiate between categories based on the patterns distilled by PCA[4].

3.3 Linear Support Vector Machine (SVM)

A Support Vector Machine (SVM) is a powerful type of supervised learning model used primarily for classification tasks[5]. In our project, SVM is employed to distinguish between Black and non-Black individuals-binary classification problems. The strength of SVM lies in its ability to find the optimal boundary between different classes of data. SVM operates by constructing a hyperplane in a high-dimensional space[5]. While many potential hyperplanes might classify the data, the optimal hyperplane is the one that provides the greatest separation margin between the data points of two classes[5]. Below is an illustration of a hyperplane as a two-dimensional line or a flat surface (in multi-dimensional planes) that neatly divides the data points into different classes.

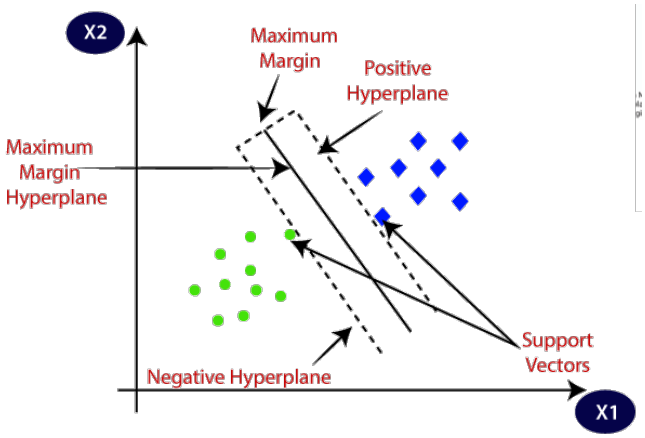

Figure 1: Illustration of a hyperplane separating two classes of data points.
[6]

Support Vectors: These are the data points that are closest to the hyperplane. The optimal separating hyperplane will be defined with the help of these points[6].

Margin: This refers to the distance between the hyperplane and the nearest data points (support vectors). A large margin is considered beneficial in SVM, offering robustness. Margins are categorized into hard margins (no data points between the margins) and soft margins (some data points are allowed between the margins)[6].

For our project, each face's representation using eigenfaces' weights becomes a point in this high-dimensional space. The goal of the SVM is to find the best possible hyperplane that separates these points into two groups: Black and non-Black individuals. The best hyperplane is the one that leaves the maximum margin, or distance, between the nearest data points of each class[5]. The integration of SVM with the reduced dimensionality data from PCA (via eigenfaces) is crucial. By reducing the number of dimensions (features) that the SVM needs to consider (thanks to PCA), the classification process becomes much more efficient and effective[5].

3.3.1 Classification Report

The classification report provides a comprehensive evaluation of a Linear Support Vector Machine (SVM) through several statistical measures[7]. It details the model's accuracy and reliability via key metrics such as Precision, Recall, F-1 Score, and Support. Precision reflects the accuracy of positive predictions

for each class, while Recall (or sensitivity) measures the model's ability to identify all relevant cases (true positives) within a dataset[7]. The F1-score combines precision and recall into a single metric, providing a balance between accuracy and completeness in predictions[7]. Support counts the number of instances for each class in the dataset, helping gauge class distribution.

Additionally, the report includes overall assessment scores like accuracy, macro average, and weighted average. Accuracy describes the model's success rate across all predictions[7]. The Macro Average evaluates performance across classes equally, while the Weighted Average considers the proportion of each class in the dataset, ensuring metrics reflect the impact of class size on model evaluation[7].

4 Methodology

4.1 Data Preproccesing

Our data cleaning process was designed to ensure the integrity and usability of the NIST Mugshot dataset[2] for subsequent machine learning tasks. The initial dataset comprised 3,248 segmented 8-bit grayscale mugshot images, each associated with metadata that includes gender, age, and standing position in pictures of subjects. The dataset was categorized based on the number of offenses recorded, segregating individuals with single offenses from those with multiple offenses.

4.1.1 Standardization

Standardization of input data is a crucial step in preparing datasets for machine learning models. In the context of image processing, resizing images to a uniform dimension across the dataset is one of the primary standardization techniques employed. This subsection explores the benefits of this practice. We removed all metadata files so that when our models process the data, it only gets PNG images without textual descriptions. Non-image files would cause our image-orientated models to output incorrect results or crash. The photos also underwent a standardization process by resizing them to a uniform scale. This normalization is vital for PCA to effectively reduce dimensionality without distortion from varied image sizes. Additionally, we performed image enhancements where necessary to correct lighting variations and alignment, ensuring that each image was presented in the best possible quality and pose for feature extraction.

Figure 2: (a) Before Standardization (b) After Standardization

As seen in the difference in these two samples, we have lost quality in images. It's important to note that resizing images can also lead to a loss of critical detail and resolution, potentially obscuring fine features that are essential for accurate model predictions, particularly in tasks requiring high levels of visual fidelity.

4.1.2 Training Set

We put all Non-Black and Black individuals into folders, *"nonblack_convicts"* and *"black_convicts"* respectively. We used these predetermined cases of individuals from this dataset being Black and Non-Black to train our model. We categorized if a person is Black or Non-Black based on three key Afro-eccentric features:

- **Skin Tone:** Evaluated the skin tone visible in the mugshot images, focusing on shades that typically correlate with Black or Non-Black ethnicities based on a predetermined color scale.
- **Hair Texture:** Identified typical hair textures associated with Black individuals, such as curly, coily, or kinky hair, versus straight or wavy textures more commonly found in Non-Black individuals.
- **Facial Structure:** Considered common facial structure traits such as nose width, lip fullness, and cheekbone prominence that are often cited in anthropological research as distinguishing features among different ethnic groups.

We also had to cut down on the amount of Non-Black convict front and rear-facing images we included in our training model from 1237 images to 624 images so that we could have an equal amount of images to represent both cases. Balancing the number of Non-Black and Black convict images in our training dataset was crucial to avoid introducing a class imbalance,

which could skew the model's performance by making it biased towards the majority class.

(a) Sample of individuals from Black Convicts folder

(b) Sample of individuals from Non-Black Convicts folder

Figure 3: Samples of individuals from different categories

This ensures that our machine learning model learns to recognize patterns and make predictions with equal accuracy for both Black and Non-Black individuals, thus improving its generalizability and fairness. However, this process introduces bias, as my group and I determining if a person is Black or Not leaves room for human error. The reliance on visual assessments and cultural interpretations can introduce personal biases and assumptions that might not accurately reflect an individual's ethnic background.

4.1.3 Testing Set

We put all single offenses and all multiple offenses convict images into folders *"single_and_multiple"* which will be used to test our model. There is no room for bias in this step as we already know if a convict has a single or multiple offense. This will allow us to observe our model's ability to accurately predict whether Black individuals have a higher population in single or multiple-offense convicts. These manual steps allow us to manipulate these images using code without having to sift through the various files they were placed in before.

Cleaning the dataset was an indispensable step. Without this, the PCA might extract features from metadata or noise -such as variations in image size and quality- rather than from the facial characters of interest. Similarly, the SVM classifier relies on the distinction of features to determine the separating hyperplane between classes accurately. Therefore, the preprocessing of images directly influences the efficacy of the eigenfaces and SVM in the subsequent stages of facial recognition.

4.2 Calculating Eigenfaces and Weights of Images

In this next section, we're going to look at how we figure out eigenfaces and the weights of our images, which help us tell faces apart in our project. Think of eigenfaces as the main features that make each face unique, how you might recognize someone by their eyes or smile. We'll start with a simple example using two small 2×2 matrices, sort of looking at the very basic building blocks of a much bigger picture. Along the way, we'll showcase eigenfaces we've found from the NIST mugshots. These pictures are key to how we teach our machine-learning model to see the difference between one person's face and another's. First we obtain face images $I(x, y)$ as training faces:

$$I(x,y) = \left\{ \begin{bmatrix} 1 & 1 \\ -2 & -3 \end{bmatrix}, \begin{bmatrix} 1 & -1 \\ -3 & 2 \end{bmatrix} \right\}$$

Consider the number of images(m) = 2 and the size of each image $N \times N$, $N = 2$

$$I_1 = \begin{bmatrix} 1 & 1 \\ -2 & -3 \end{bmatrix}_{2\times 2} \quad I_2 = \begin{bmatrix} 1 & -1 \\ 3 & 2 \end{bmatrix}_{2\times 2}$$

Mathematically we represent every image I_i as a vector Γ_i. The image vector Γ_i is used in all calculation steps. Each face image in the dataset is converted into a grayscale image and resized to the same dimensions, ensuring uniformity. These images are then converted into a one-dimensional array. For instance, a 64×64 image becomes a single vector of 4,096 pixel values. All these vectors (one per image) are compiled into a matrix where each row represents one face.

Figure 4: Γ_1 = Image one in our dataset

The vectorization of images is foundational to the eigenface approach, which applies principal component analysis (PCA) to a large dataset of faces to identify the vectors (eigenfaces) that best account for the variation among faces.

$$\Gamma_1 = \begin{bmatrix} 1 \\ -2 \\ 1 \\ -3 \end{bmatrix}_{N^2 \times 1} \qquad \Gamma_2 = \begin{bmatrix} 1 \\ 3 \\ -1 \\ 2 \end{bmatrix}_{N^2 \times 1}$$

$N = 2, M = 2$ (Representing the training set of M images of size $(N \times N)$). Three major takeaways from vectorization:

1. Converting images to vectors simplifies algebraic calculations, making it easier to apply principal component analysis across the dataset.

2. Vectorization boosts storage and computation efficiency because modern computing systems and software libraries are highly optimized for vector and matrix operations.

3. By converting images from matrices to vectors, the data is stored in a contiguous block of memory, which can be accessed and processed more efficiently by computer systems.

Then we calculate the mean of face images. Mean face vector:

$$\psi = \frac{1}{M} \sum_{i=1}^{m} \mathbf{\Gamma_i} \tag{6}$$

$M = 2, m = 2$, therefore:

$$\psi = \frac{1}{2} \sum_{i=1}^{2} (\Gamma_i) = \frac{1}{2} (\Gamma_1 + \Gamma_2)$$

$$\psi = \frac{1}{2} \begin{bmatrix} 1+1 \\ -2+3 \\ 1-1 \\ -3+2 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 2 \\ 1 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{1}{2} \\ 0 \\ -\frac{1}{2} \end{bmatrix}$$

The mean face represents the "average" features of the set of faces, and subtracting it from each face image highlights the unique features of each face by removing the commonalities. For example, the "mean face" has a highlight on its forehead, if we subtract the" mean face" from a face image we will be able to observe the features of this image's forehead without the interruption of highlights.

Figure 5: ψ = average features of 1248 facial images

This process ensures that variations among faces are not due to such things as lighting, orientation, or scale, but rather due to the unique features of each facial features of each individual. This step is crucial for subsequent steps, such as PCA, which identifies the principal components (eigenfaces) that capture the most significant variations among the face images. These variations are what the algorithm uses to differentiate between different faces in the dataset.

$$\phi_\mathbf{i} = \Gamma_i - \psi \tag{7}$$

$$\phi_1 = \begin{bmatrix} 1 \\ -2 \\ 1 \\ -3 \end{bmatrix} - \begin{bmatrix} 1 \\ \frac{1}{2} \\ 0 \\ -\frac{1}{2} \end{bmatrix} = \begin{bmatrix} 0 \\ -\frac{5}{2} \\ 1 \\ -\frac{5}{2} \end{bmatrix}$$

$$\phi_2 = \begin{bmatrix} 1 \\ 3 \\ -1 \\ 2 \end{bmatrix} - \begin{bmatrix} 1 \\ \frac{1}{2} \\ 0 \\ -\frac{1}{2} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{5}{2} \\ -1 \\ \frac{5}{2} \end{bmatrix}$$

Figure 6: This is what a sample image looks like after being subtracted by the mean.

In Figure 6 we see the difference between each image and the average. this step highlights the unique features of each face that are crucial for distinguishing between different individuals in the subsequent computation and facial recognition process. The difference highlights Finding the difference between each image and the average face is necessary to center the data around the mean, which effectively normalizes the face images by removing the common features shared across the dataset.

We then find the covariance matrix (C) by dividing the number of image sets (M) by the product of the matrix of the difference between images and the average (A) and its transpose (A^T). A covariance matrix is a mathematical construct that quantifies the linear relationship between each pair of variables in a dataset. In the context of eigenfaces, it is a big table that helps us understand if and how different pieces of information, the brightness of pixels in an image, move or change together. With face recognition, it helps to pick out the important patterns that make each face unique.

$$C = \frac{1}{M} A \times A^T \tag{8}$$

$$A = \begin{bmatrix} 0 & 0 \\ -\frac{5}{2} & \frac{5}{2} \\ 1 & -1 \\ -\frac{5}{2} & \frac{5}{2} \end{bmatrix}_{N^2 \times M}$$

$$A^T = \begin{bmatrix} 0 & -\frac{5}{2} & 1 & -\frac{5}{2} \\ 0 & \frac{5}{2} & -1 & \frac{5}{2} \end{bmatrix}_{M \times N^2}$$

When you take a matrix of difference (where each row represents a face with its unique differences from the average face) and multiply it by its transpose, you're essentially comparing each face's differences with every other face's differences, across all pixels.

$$A \times A^T = \begin{bmatrix} 0 & 0 \\ -\frac{5}{2} & \frac{5}{2} \\ 1 & -1 \\ -\frac{5}{2} & \frac{5}{2} \end{bmatrix} \times \begin{bmatrix} 0 & -\frac{5}{2} & 1 & -\frac{5}{2} \\ 0 & \frac{5}{2} & -1 & \frac{5}{2} \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{25}{2} & -5 & \frac{25}{2} \\ 0 & -5 & 2 & -5 \\ 0 & \frac{25}{2} & -5 & \frac{25}{2} \end{bmatrix}$$

$$C = \frac{1}{2}(A \times A^T) = \frac{1}{2} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{25}{2} & -5 & \frac{25}{2} \\ 0 & -5 & 2 & -5 \\ 0 & \frac{25}{2} & -5 & \frac{25}{2} \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{25}{4} & -\frac{5}{2} & \frac{25}{4} \\ 0 & -\frac{5}{2} & 1 & -\frac{5}{2} \\ 0 & \frac{25}{4} & -\frac{5}{2} & \frac{25}{4} \end{bmatrix}$$

The covariance matrix is a new matrix that tells us how much the pixel values of the images vary together. This is a crucial step for finding patterns in the data, which is what allows us to identify which features are most important for distinguishing faces.

Next, we compute eigenvalues and eigenvectors of the coverage matrix (C). This step includes linear algebra computations to determine the eigenvalues and eigenvectors of the covariance matrix. It tells us the principal axes of variance which are the directions in which the data varies the most. Our Eigenvalues: $\lambda_1 = \frac{27}{2}$, $\lambda_2 = 0$, $\lambda_3 = 0$, $\lambda_4 = 0$. Eigenvalues measure the amount of variance in the direction of their corresponding eigenvectors. These results suggest that most of the variance in the dataset can be captured along a single dimension since there's only one non-zero eigenvalue. The zero eigenvalues indicate that those axes do not add additional information/variance. Our Eigenvectors: $\mathbf{V_1}$ = (0,1,$\frac{-5}{2}$,1), $\mathbf{V_2}$ = (0,-1,0,1), $\mathbf{V_3}$ = (0,$\frac{2}{5}$,1,0), $\mathbf{V_4}$= (1,0,0,0). Eigenvectors are the directions or components in the feature space that maximize the variance of the data. In facial recognition, these directions correspond to the features that most significantly differentiate the faces in the dataset. V_1 corresponds to the direction with the most variance. The rest show other perpendicular directions of variance but in this case, since their corresponding eigenvalues are 0, they do not contribute to differentiating data.

Figure 7: Calculation of eigenvectors, also known as eigenfaces, of our images. In this figure, we sample only four images from our dataset.

- **Normalization and Centering:** The images are scaled to have values between 0 and 1, and the mean face is subtracted to center the data around zero.
- **PCA:** Sklearn's PCA is used to compute the

eigenfaces, which are essentially the eigenvectors of the data's covariance matrix.

- **Eigenfaces:** The principal components (eigenfaces) are reshaped back into the original image dimensions and displayed.
- **Projection:** The script projects four sample images onto the eigenfaces to reduce their dimensionality, showing how much variance of each sample image is captured by each eigenface.

Now we must normalize our eigenvectors. To normalize a vector $\mathbf{V_i}$, we divide the vector by its magnitude as shown in the following equation:

$$\mathbf{v}_{\text{normalized}} = \frac{\mathbf{v}}{\|\mathbf{v}\|} \tag{9}$$

$$\|\mathbf{v}\| = \sqrt{\sum_{i=1}^{n} v_i^2} \tag{10}$$

Here, $\|\mathbf{v}\|$ denotes the Euclidean norm (or the L2 norm) of the vector $\mathbf{V_i}$. The Euclidean norm, also known as the L2 norm, is a measure of the "length" of a vector in Euclidean space. It's defined as the square root of the sum of the squares of the individual elements of the vector. This is essentially the Pythagorean theorem. In the context of machine learning and specifically in eigenfaces, the Euclidean norm is often used to measure the similarity between different faces by comparing the "distance" between their feature vectors in the feature space. The smaller the norm of the difference between two vectors, the more similar the corresponding faces are considered to be. Normalizing eigenvectors ensures that they have a unit length, providing consistency and stability in mathematical computations and data analysis. This standardization is crucial in applications of the eigenface method, where it allows for meaningful comparison and combination of facial features. Our normalized eigenvectors $\mathbf{U_1} = \frac{1}{\sqrt{33}}(0, 2, -5, 2)$, $\mathbf{U_2} = \frac{1}{\sqrt{2}}(0, -1, 0, 1)$, $\mathbf{U_3} = \frac{1}{\sqrt{29}}(0, 2, 5, 0)$, $\mathbf{U_4} = \frac{1}{\sqrt{1}}(1, 0, 0, 0)$. Our normalized eigenvector matrix $[\mathbf{U_1}, \mathbf{U_2}, \mathbf{U_3}, \mathbf{U_4}]$ is considered our eigenfaces because each eigenvector corresponds to a principal component that captures a significant variance direction in the facial data. These eigenfaces form a basis set for representing faces as a combination of these key features, which are essential for facial recognition tasks.

Finally, we calculate the weight combination of each eigenface:

$$\mathbf{W_{k,i}} = U_k^T \times \phi_i \tag{11}$$

Here, $\mathbf{W_{k,i}}$ is the weight of the k-th eigenface for the i-th image, $\mathbf{U_k^T}$ is the transpose of the k-th normalized eigenvector, and $\phi_\mathbf{i}$ s the vector representing the difference between the i-th image and the mean image. This dot product gives you a scalar value representing how much of the k-th eigenface is present in the i-th image's unique features. With this, we can calculate the weights ($\mathbf{\Omega_1}$) of image I_1. As seen in equation (11), the weight $\mathbf{W_1}$ of I_1 is calculated:

$$U_1^T \times \phi_1 = \frac{1}{\sqrt{33}} \begin{bmatrix} 0 \\ 2 \\ -5 \\ 2 \end{bmatrix} \times \begin{bmatrix} 0 \\ -\frac{5}{2} \\ 1 \\ -\frac{5}{2} \end{bmatrix} = \frac{-15}{\sqrt{33}}$$

Therefore, $\mathbf{W_{2,1}} = \frac{-5\sqrt{2}}{4}$, $\mathbf{W_{3,1}} = \frac{-5\sqrt{29}}{29}$, and $\mathbf{W_{4,1}} = 0$. We can repeat this process to calculate the weights for other images in training set I_2 to obtain $\mathbf{\Omega_2} = [\mathbf{W_{1,2}}, \mathbf{W_{2,2}}, \mathbf{W_{3,2}}, \mathbf{W_{4,2}}]$, and in context of our dataset, to get $\mathbf{\Omega_{image}}$.

A **heatmap** of image weights is a graphical representation that shows the weights (coefficients) of images when they are projected onto a set of features, such as eigenfaces in the context of facial recognition. Each weight describes how much a particular feature (eigenface) contributes to reconstructing the original image in a reduced-dimensional space. The heatmap uses color coding to represent the magnitude of these weights, making it easy to visualize complex data at a glance.

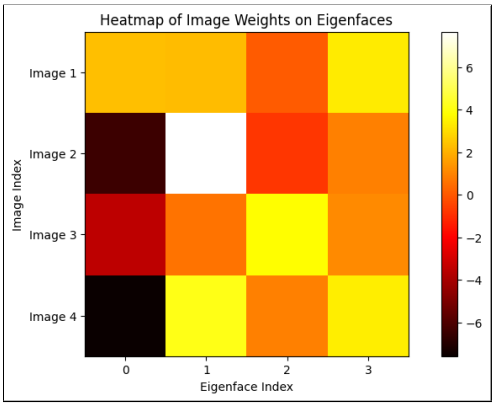

Figure 8: This is a heatmap of image weights on eigenfaces from figure 7

Each row in the heatmap typically represents an individual image from your dataset. Each column represents a feature, such as an eigenface, onto which the images have been projected. The intensity or color in each cell of the heatmap indicates the magnitude of the weight. Warmer colors (red or yellow) often indicate higher values, while cooler colors (blue or green) indicate lower values. Heatmaps can help visualize how images relate to the eigenfaces. For example, if certain images have high weights for certain eigenfaces, it implies those eigenfaces capture significant features of those images. Heatmaps are particularly useful in facial recognition to see how similarly different faces

are represented in terms of eigenfaces. In the context of eigenfaces, a heatmap can show you at a glance which eigenfaces are most significant across a set of images. For example, if the first column (Eigenface 1) is predominantly warm-colored, it means that the first eigenface is a significant component in representing many images in your dataset.

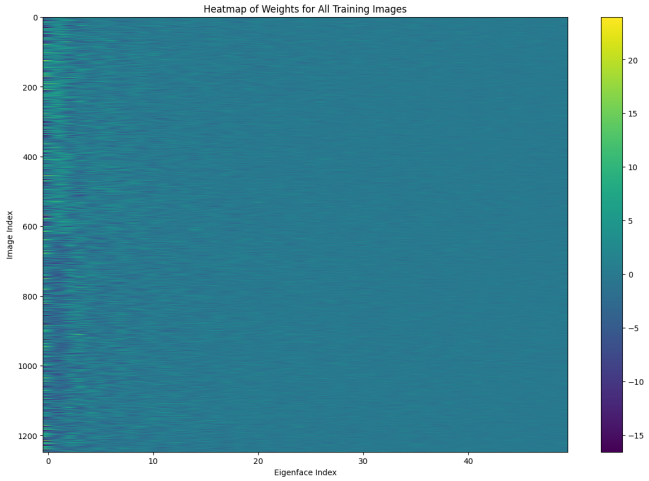

Figure 9: Here is a heatmap of the weights of all of our 1,237 training set images

As mentioned before, images are normalized to have pixel values between 0 and 1, and the mean image is subtracted from each image. This is performed on the entire dataset to extract a specified number of principal components (eigenfaces). The number of components is set to 50 in this example, but you can adjust this based on the desired level of detail or variance you wish to capture. All centered images are projected onto the space spanned by these eigenfaces, resulting in a set of weights for each image. These weights describe how much each eigenface contributes to reconstructing the original image. The heatmap visualizes these weights. Each row corresponds to an image and each column to an eigenface. The color intensity in the heatmap shows the magnitude of the weight, providing a perceptually uniform colormap that ranges from yellow (low values) to dark blue (high values). This heatmap provides a comprehensive view of how all images in your training set relate to the eigenfaces, highlighting patterns and possibly identifying outliers or anomalies in how images are represented. This can be particularly useful for understanding the behavior of your dataset in applications of facial recognition or other types of image-based machine learning tasks.

4.3 Employing a Support Vector Machine

In this project, the Support Vector Machine (SVM) is utilized as the primary classification tool, distinguishing between facial images categorized as either Black or non-Black. This binary classification employs the eigenface method for advanced feature extraction, where facial characteristics are reduced into a set of principal components—eigenfaces. These components are transformed into numerical weights, which the SVM uses to effectively differentiate between the two racial categories. The derivation of eigenfaces and their associated weights, as detailed previously, serves as the foundation for the SVM's input features. By using Principal Component Analysis (PCA), the high-dimensional data is condensed into a compact form, preserving essential information while reducing redundancy. This dimensional reduction facilitates a more efficient and accurate classification by the SVM.

The SVM algorithm aims to find a hyperplane that best separates the classes with maximum margin. In the case of non-linear classification boundaries, the RBF kernel is used to transform the data into a higher-dimensional space. The decision function for the SVM with the RBF kernel is expressed as:

$$f(x) = \text{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b\right) \tag{12}$$

where:

- α_i are the Lagrange multipliers,
- y_i are the class labels,
- x_i are the support vectors,
- b is the bias term,
- $K(x_i, x) = \exp\left(-\gamma \|x_i - x\|^2\right)$ is the RBF kernel,
- γ is a parameter that defines the spread of the kernel.

Data Preparation and Model Training: The dataset is split into training and testing sets to ensure the SVM model's reliability and generalizability. The `train_images` and their corresponding labels, initially processed through a Label Encoder, are used here. A pipeline, integrating `StandardScaler` for data normalization, PCA for dimensionality reduction, and `SVC` for the SVM classifier, is configured and applied. The PCA is set to retain 250 principal components, which capture the most significant features of the face images.

Training Set: This subset comprises a balanced number of eigenface weight vectors from both classifications (Black or non-Black) to train the SVM. The model adjusts its parameters to minimize classification errors, learning the discernible boundary between the two groups.

Testing Set: Comprising data unseen during the training phase, this set evaluates the SVM's effectiveness, essential for verifying real-world applicability and ensuring that the model does not merely fit the training data's nuances.

The SVM pipeline is trained using the `fit` method, and its performance is assessed on the test set. Predictions are made using the `predict` method, and the model's accuracy and classification metrics are detailed in a classification report generated through `classification_report`, offering insights into precision, recall, and F1-scores for each category.

Here is an example of the pipeline configuration and model evaluation process in Python:

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import
    StandardScaler, LabelEncoder
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.metrics import
    classification_report
from sklearn.model_selection import
    train_test_split

# Assume 'train_images' and 'train_labels'
    are your preprocessed datasets
label_encoder = LabelEncoder()
train_labels_encoded = label_encoder.
    fit_transform(train_labels)

# Create a pipeline with PCA and SVM
svm_pipeline = make_pipeline(
    StandardScaler(),
    PCA(n_components=250),  # Adjust the
    number of components as needed
    SVC(kernel='rbf', gamma='scale')
)

# Split data into training and test sets
X_train, X_test, y_train, y_test =
    train_test_split(
    train_images.reshape(len(train_images),
    -1),
    train_labels_encoded,
    test_size=0.2,
    random_state=42
)

# Train the SVM model with PCA
svm_pipeline.fit(X_train, y_train)

# Predict using the trained model
y_pred = svm_pipeline.predict(X_test)
```

Listing 1: Python code for SVM and PCA pipeline

This script encapsulates the complete process of training and evaluating an SVM model integrated with PCA, demonstrating practical application in facial classification. The `StandardScaler` component of the pipeline normalizes each feature by subtracting the mean and scaling to unit variance. The transformation applied to each feature x can be described mathematically as:

$$z = \frac{(x - \mu)}{\sigma} \tag{13}$$

where μ is the mean of the feature, and σ is the standard deviation of the feature. PCA is utilized for dimensionality reduction while preserving as much variance as possible. It transforms the original variables into a new set of variables (principal components), which are linear combinations of the original ones. These new variables are orthogonal and ordered by the variance of the data along them. The transformation is given by:

$$\mathbf{Y} = \mathbf{X}\mathbf{W} \tag{14}$$

where $\mathbf{X}$ is the data matrix, $\mathbf{W}$ is the matrix of eigenvectors, and $\mathbf{Y}$ is the transformed data matrix. The eigenvectors in $\mathbf{W}$ are selected to maximize the variance of $\mathbf{Y}$ and are computed as the eigenvectors of $\mathbf{X}^\top\mathbf{X}$.

The SVM model is trained using the dataset transformed by PCA, learning the parameters α_i and b, and utilizing the support vectors x_i. For prediction, new data points are first projected into the PCA-transformed space and then classified using the trained SVM model. This approach allows the handling of high-dimensional data more efficiently, reducing overfitting and potentially enhancing the classifier's performance by focusing on the most informative features.

5 Results

The performance of the SVM model, trained using eigenfaces as features, is quantitatively evaluated in the classification report provided below. The report details precision, recall, F1-score for each class, and the overall model accuracy, macro average, and weighted average scores.

Table 1: SVM Classification Report

Category	Precision	Recall	F1-score	Support
Black	0.80	0.84	0.82	122
Nonblack	0.84	0.80	0.82	128
Overall Accuracy: 0.82				

- **Precision:** This metric indicates the accuracy of positive predictions. The model achieved a precision of 0.80 for 'Black' and 0.84 for 'Nonblack', meaning 80% and 84% of these predictions respectively were correct.
- **Recall:** Also known as sensitivity, this metric measures the model's ability to correctly identify all relevant instances. The recall was 0.84 for 'Black' and 0.80 for 'Nonblack'.

- **F1-Score:** The F1-score, the harmonic mean of precision and recall, was 0.82 for both classes, indicating a balanced performance between precision and recall.
- **Support:** Indicates the actual number of cases for each class in the dataset, with 122 'Black' and 128 'Nonblack'.
- **Macro Average F1-Score:** Calculated as the average of the F1-scores for each class, not considering class imbalance, was 0.82.
- **Weighted Average F1-Score:** Reflects the F1-scores weighted by the support of each class, also 0.82.

The results demonstrate that the SVM model is capable of classifying facial images into 'black' and 'nonblack' categories effectively, with a good balance between precision and recall. The high accuracy and balanced F1-scores across both classes confirm the effectiveness of PCA for dimensionality reduction and SVM for classification in extracting and utilizing salient features for racial categorization in facial images. This performance underscores the model's potential in real-world applications involving automated racial classification.

5.1 Model Evaluations

The SVM model's performance was evaluated on a test dataset of 250 images, achieving an overall classification accuracy of 81.60%. This performance indicates that the model correctly predicted the racial category ('black' or 'nonblack') for 204 out of the 250 images.

Figure 10: Here are 50 predictions made by our model. Showcasing only 6 instances of incorrect predictions.

The model predicted 128 individuals as 'black', compared to the actual number of 122 'black' individuals in the test dataset. This result highlights the model's sensitivity and its tendency to slightly overestimate the presence of the 'black' category. The model demonstrates high precision and recall, particularly noted by its ability to identify and predict the 'black' category with a minor tendency towards over-prediction. This slight bias suggests potential areas for adjusting the classification threshold or revisiting feature weighting in the PCA process.

Table 1: Summary of Predictive Performance

Metric	Value
Total Predictions	250
Correct Predictions	204
Accuracy	81.60%
Predicted 'Black' Individuals	128
Actual 'Black' Individuals	122

The quantitative and visual results combined provide a robust framework for understanding the strengths and limitations of the SVM model in racial categorization tasks. While the model performs effectively in general, the slight over-prediction of the 'black' category suggests that further tuning may be beneficial.

6 Discussion

The model's reproducibility hinges on its ability to generalize across different datasets beyond the controlled environment of the NIST Mugshot dataset. Testing the model on a broader range of datasets with variations in image quality, lighting, and demographic distributions would provide a better understanding of its generalizability. Consistent application of preprocessing steps such as image standardization and eigenfaces extraction is crucial to maintain performance across varied datasets. **Bias in Model Construction:** The categorization of individuals into Black and non-Black based on features such as skin tone, hair texture, and facial structure introduces significant subjective bias. This is further complicated by manual adjustments in the dataset balance, potentially skewing the SVM's learning process. The inherent biases of the SVM algorithm need to be carefully managed through regularization techniques and hyperparameter tuning to avoid overfitting or underfitting.

The model could be employed in various sectors requiring rapid and automated facial image analysis, such as law enforcement for identifying individuals from surveillance footage or in border control to speed up identity verification processes. However, applications should be guided by stringent rules to ensure these technologies support rather than replace human judgment. This is crucial to address ethical, legal, and social problems. **Ethical Considerations:** Privacy and surveillance concerns are paramount with facial recognition technologies. The use of racial classification

systems raises significant ethical questions, especially if employed in contexts where they could reinforce racial profiling or contribute to discriminatory practices. **Legal Implications:** The deployment of facial recognition must comply with data protection laws, such as the GDPR or various state laws in the United States, which regulate the collection and use of biometric data. Transparency and accountability mechanisms are essential. **Social Impact:** The public acceptance of facial recognition technologies varies. Misidentifications, particularly of individuals from minority groups, could exacerbate social tensions and erode trust in institutions utilizing such technology.

While the eigenface and SVM-based model shows potential for classifying individuals based on racial characteristics with reasonable accuracy, careful management of its application is crucial, considering the broader ethical, legal, and social implications. Future work should aim to improve the model's fairness by incorporating a more diverse dataset and employing bias-minimization techniques. Engaging with stakeholders including policymakers, civil rights groups, and the public is essential to responsibly navigate the challenges posed by facial recognition technologies.

References

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] "Nist special database 18 – mugshot identification database," https://www.nist.gov/srd/nist-special-database-18, 2014, accessed: 2023-04-17.

[3] A. Jain, "The mathematics behind principal component analysis," https://towardsdatascience.com/the-mathematics-behind-principal-component-analysis-fff2d7f4b643, Aug 2020.

[4] M. A. Turk and A. P. Pentland, *Face Recognition Using Eigenfaces*. MIT Press, 1991, vol. 3, no. 1.

[5] V. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer, 1995.

[6] A. Vidhya. (2021) Support vector machines(svm): A complete guide for beginners. https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/.

[7] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R*. New York: Springer, 2013.