

# Machine Learning

## ASSIGNMENT 3

---

### Members:

Mark Magdy : 2205040

Arsany Osama : 2205122

Rana Ashraf : 2205019

### Dataset Understanding

We worked on ORL (Olivetti Research Laboratory) dataset that consists of 40 subjects each consists of 10 images with different facial expressions, different times, and different lighting. Images are all grayscale meaning they contain values ranging from 0 (black) to 255 (white), with dimensions (92 x 112) that represent 92 pixels in width and 112 pixels in height.

```
# Define the image size (92 x 112)
image_size = (92, 112)

# Define the number of subjects and images per subject
num_subjects = 40
images_per_subject = 10
```

## Data Preprocessing

### 1) Data Matrix and Label vector :

Each image is converted into a vector of 10304 value (92x112). Then all 400 vectors (image) are converted to data matrix with dimensions (400x10304) indicating 400 images and 10304 pixels of each image.

Then generate the label vector y range from 1 to 40, corresponding to the subject IDs.

```
# Total images = 40x10 = 400 image
# Total pixels in 1 image = 92x112 = 10,304
D = np.zeros((num_subjects * images_per_subject, image_size[0] * image_size[1])) # row=400 , col=10304
y = np.zeros(num_subjects * images_per_subject) # labels 400
```

Label vector shape: 1-dim (40) to identify each subject.

Data Matrix shape: 2-dim (400, 10304) means that there are :

400 rows and 10304 columns in the Data Matrix.

Each row represents an image, and each column represents a pixel in the image.

So, there are a total of 400 images, and each image has 10304 pixels which means image size (92 x 112).

```
Label vector shape: (400,)
Data Matrix shape: (400, 10304)
```

### 2) Split Dataset:

The Data set was separated into two sets testing and training, select odd rows for training and even rows for testing. Each set includes 200 images, five images for each subject (200/40). Their dimensions are (200x10304).

## Principal Component Analysis (PCA)

PCA reduces dimensions of the dataset and finds the directions along which the data varies the most. These directions are determined by the eigenvectors of the covariance matrix, and their eigenvalues represent the amount of variance explained by each principal component.

It evaluates the accuracy using a simple K-Nearest Neighbors for different values of alpha and k.

### I. Compute Mean

Calculate the mean of each feature in training data matrix

```
mean = np.mean(training_DataMatrix, axis=0)
```

### II. Center Data

Subtract the mean (10304) of each feature from each data point to center the data around the origin. It makes dataset have a mean of '0' (origin)

Centering makes that dataset is normalized, unbiased and ensure that each variable contributes equally to the PCA

```
training_data_centralized = training_DataMatrix - mean
```

### III. Compute Covariance Matrix

Covariance matrix (200,200) measures how much two variables change together.

Each element of the covariance matrix represents the covariance between two variables.

The diagonal elements represent the variances of individual features.

A positive covariance means that the variables are directly, while a negative covariance means that the variables are inversely

```
cov_matrix = training_data_centralized @ training_data_centralized.T
```

@ for matrix multiplication, and T represents the transpose operation.

## IV. Compute Eigenvectors and Eigenvalues (CovMatrix.lamda = lamda.V)

### 1) Eigenvalues (lamda)

represent the amount of variance explained by each principal component. The larger the eigenvalue, the more variance is explained by the corresponding principal component.

### 2) Eigenvectors (V)

Eigenvectors are directions in which data varies the most

The first principal component (eigenvector) corresponds to the eigenvector with the largest eigenvalue and explains the most variance in the data.

```
eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)
```

## V. Sort Eigenvalues and their Eigenvectors

The largest eigenvalue corresponds to the first principal component, the second largest to the second principal component.

The eigenvalues are sorted from largest to smallest because the first few principal components explain the majority of the variance

By sorting the eigenvalues and eigenvectors, we ensure that the principal components are aligned with the directions of maximum variance in the data

```
idx = np.argsort(eigenvalues)[::-1]
eigenvalues = eigenvalues[idx]
eigenvectors = eigenvectors[:, idx]
```

## VI. Cumulative Explained Variance Ratio

The explained variance ratio for each principal component is calculated by dividing its eigenvalue (variance along that principal component) by the total sum of eigenvalues. This ratio represents amount of variance explained by each principal component

```
explained_variance_ratio = np.cumsum(eigenvalues) / np.sum(eigenvalues)
```

## VII. Determine Number of Components

Determines the number of principal components to keep based on the cumulative explained variance ratio and the specified alpha.

We want to retain at least (alpha) of the total variance in the dataset.

```
no_components = np.argmax(explained_variance_ratio >= alpha) + 1
```

## VIII. Reduce the basis

Transforming the data into the space spanned by the eigenvectors (principal components)

We normalize the transformed data to ensure that each principal component (eigenface) has unit length or magnitude.

```
eigenvectors_converted = training_data_centeralized.T @ eigenvectors  
eigenfaces = eigenvectors_converted / np.linalg.norm(eigenvectors_converted, axis=0)
```

## IX. Reduce the basis

Reduce the dimensionality of the data by selecting only the first no\_components columns from the normalized eigenvectors matrix (eigenfaces). These selected columns represent the most significant principal components that capture the majority of the variance in the original data. The projected data is obtained by multiplying the centralized training data by this reduced set of principal components.

## Principal Component Analysis x Accuracy

- **PCA Computation:**

For each value of alpha in the alpha\_values list, PCA is performed on the training data (X\_train). The PCA function returns the mean face (mean\_face), the matrix of eigenfaces (eigenfaces), and the projected data (projected\_data) obtained by transforming the training data into the lower-dimensional subspace spanned by the eigenfaces.

```
for alpha in alpha_values:
    mean_face, eigenfaces, projected_data = PCA(X_train, alpha)
    projected_train = (X_train - mean_face) @ eigenfaces #compare train
    projected_test = (X_test - mean_face) @ eigenfaces
```

- **Classification with KNN:**

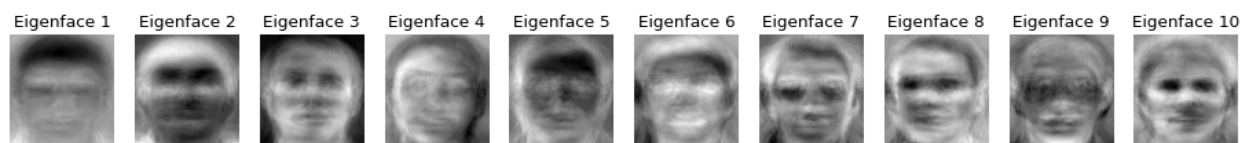
For each value of k in the k\_values list, a KNN classifier is trained on the projected training data (projected\_train) and predict the projected test data (projected\_test). The accuracy of the classifier is calculated and stored in list.

```
for k in k_values:
    classifier = KNeighborsClassifier(n_neighbors=k)
    classifier.fit(projected_train, y_train)
    y_pred = classifier.predict(projected_test)
    accuracy = np.mean(y_pred == y_test) * 100
    accuracies.append(accuracy)
```

- **Eigenface Visualization:**

Additionally, for each combination of alpha and k, the code plots the first 10 eigenfaces obtained from PCA. These eigenfaces represent the principal components capturing the most variance in the face images.

Eigenfaces for Alpha = 0.8 and k-value =1



- **Accuracy Array:**

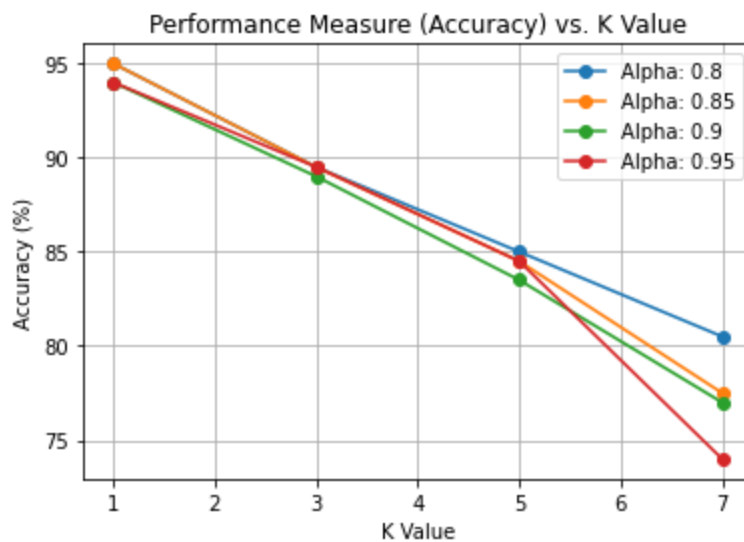
Finally, the accuracy results are reshaped into a 2D array, where each row represent alpha, each column represent K. This reshaped array facilitates the visualization and analysis of how accuracy varies with different combinations of alpha and k.

```
[[95.  89.5 85.  80.5]
 [95.  89.5 84.5 77.5]
 [94.  89.  83.5 77. ]
 [94.  89.5 84.5 74. ]
 ...]
accuracies = np.array(accuracies).reshape(len(alpha_values), len(k_values))
```

- **Plot Accuracy x Alpha:**

As alpha increases, the accuracy tends to decrease for most values of k.

Higher values of k generally result in lower accuracy compared to lower values of k.



## Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a dimensionality reduction technique that is used to reduce the number of features in a dataset while maintaining the class separability. LDA is a supervised technique, meaning that it uses the class labels to perform the dimensionality reduction.

### Compute Class Means and Sizes:

- For each class in the dataset, compute the mean vector ( $\text{class\_means}$ ) and the number of samples in each class ( $\text{class\_sizes}$ ).

### Compute Overall Mean:

- Compute the mean vector of the entire dataset ( $\text{overall\_mean}$ ). This is used for centering the data during the computation of the within-class scatter matrix.

### Compute Within-Class Scatter Matrix ( $S_W$ ):

- For each class:
  - i. Select the samples belonging to that class.
  - ii. Center the data by subtracting the class mean from each sample.
  - iii. Compute the covariance matrix for the centered data and add it to  $S_W$ .
- Regularize  $S_W$  to avoid singularity issues. Singularity refers to the condition where a matrix is not invertible, which can occur if the matrix is singular or close to being singular. It prevents it from becoming singular.

### Compute Between-Class Scatter Matrix ( $S_B$ ):

- For each class:
  - i. Compute the difference between the class mean and the overall mean.
  - ii. Compute the outer product of this difference vector with itself, scaled by the number of samples in the class, and add it to  $S_B$ .



## Sort Eigenvectors:

- Sort the eigenvectors based on the sorted eigenvalues in descending order.

## Projection Matrix:

- Take the first 39 dominant eigenvectors, which correspond to the largest eigenvalues, to form the projection matrix.
- The projection matrix obtained from LDA is used to transform the original data into a lower-dimensional subspace where the separation between classes is maximized.

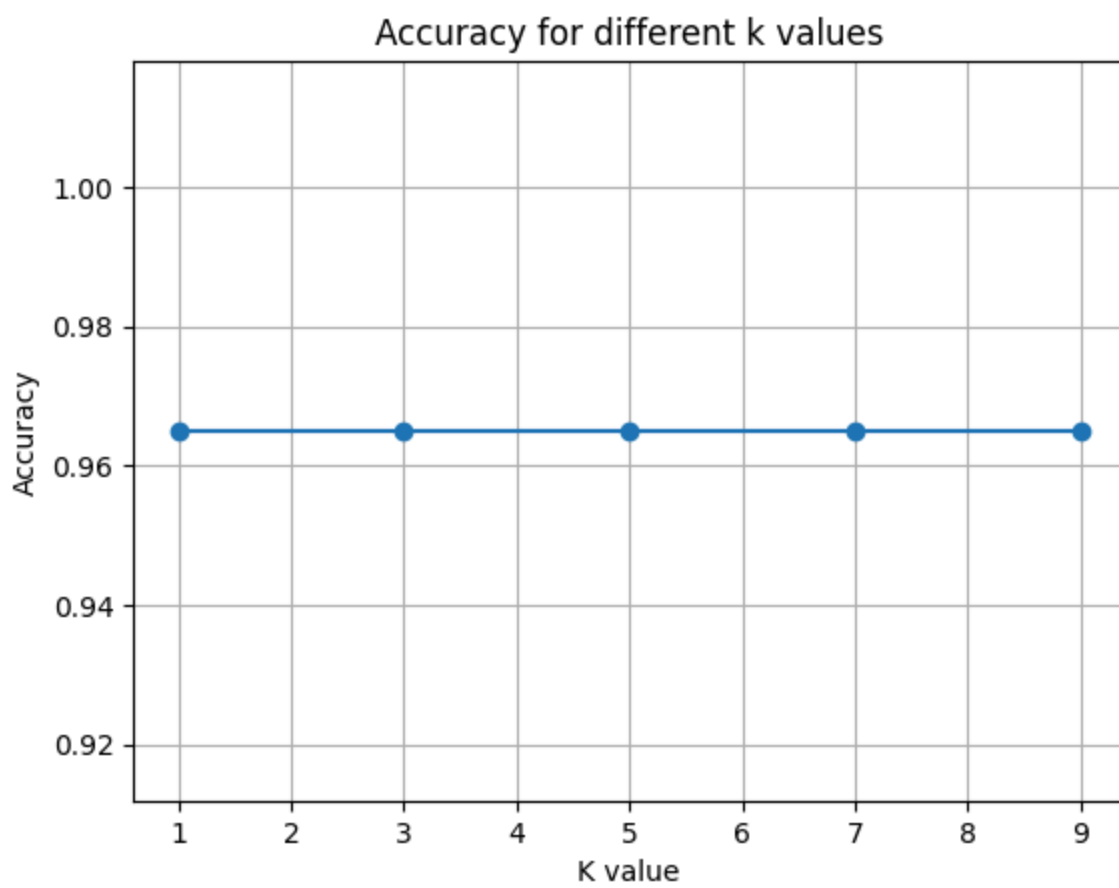
## Test LDA Accuracy:

- Test the accuracy of LDA using a **k-NN classifier** with a specified number of neighbors (k). It first projects the training and testing data onto the lower-dimensional subspace using the LDA\_projected\_data function.
- Then, it initializes a k-NN classifier with the specified number of neighbors, fits it to the projected training data, predicts the labels for the projected testing data, and calculates the accuracy of the predictions.

```
def Test_LDA(k , LDA_projection_matrix):  
    projected_X_train, projected_X_test = LDA_projected_data(X_train,X_test,LDA_projection_matrix)  
    knn = KNeighborsClassifier(n_neighbors=k)  
    knn.fit(projected_X_train, y_train.ravel())  
    y_pred = knn.predict(projected_X_test)  
    accuracy = accuracy_score(y_test, y_pred.ravel())  
    return accuracy
```

## Tuning LDA:

- Perform tuning for LDA by testing different values of k for the k-NN classifier. It loops over the specified k values, fits the k-NN classifier to the projected training data, predicts the labels for the projected testing data, calculates the accuracy for each k, and stores the results in a DataFrame.
- Observations: We observe that tuning is not important in this case, as the accuracy remains consistent across different values of k.



## Faces vs. NonFaces

### I. Data Loading and Preprocessing:

- Loaded face and non-face images, converted them to grayscale, and resized them to 92x112 pixels.
- Faces were labeled as 1 and non-faces as 0.
- Shuffled the data to prevent any bias due to the ordering of samples.

### II. Exploratory Data Analysis:

- Visualized a subset of face and non-face images with their labels

Faces:



NonFaces:



### III. Data Splitting:

- A. Split the data into training and testing sets using a 50-50 split and shuffled them.

Training Data:

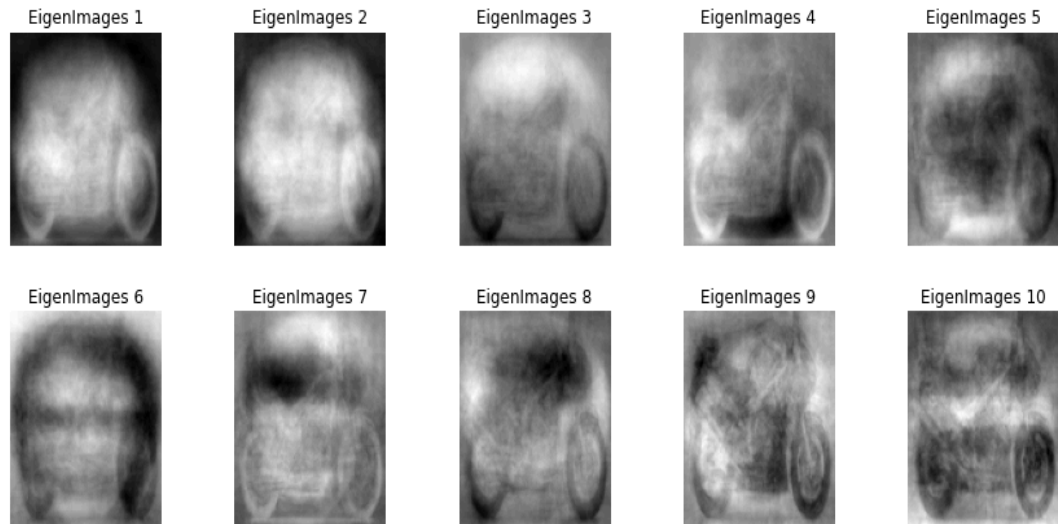


Testing Data:



#### IV. Principal Component Analysis (PCA):

- A. Explored the variance explained by different components.
- B. Transformed the training and testing data using the selected number of components.



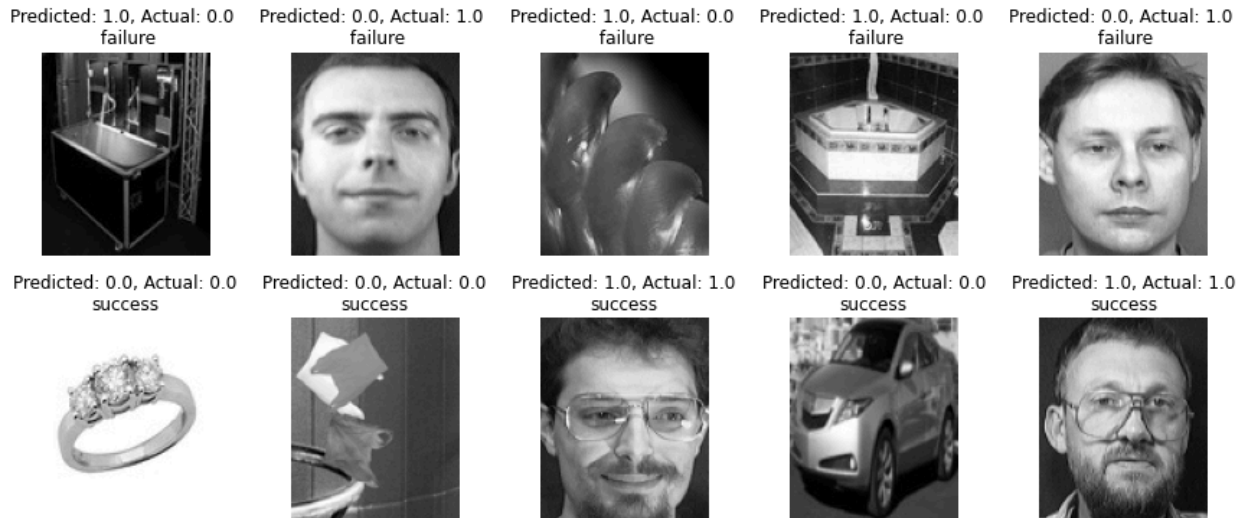
#### V. K-Nearest Neighbors (KNN) Classifier:

- A. Implemented a KNN classifier with  $k=1$ .
- B. Evaluated the accuracy of the KNN classifier before and after applying PCA.

#### VI. Linear Discriminant Analysis (LDA):

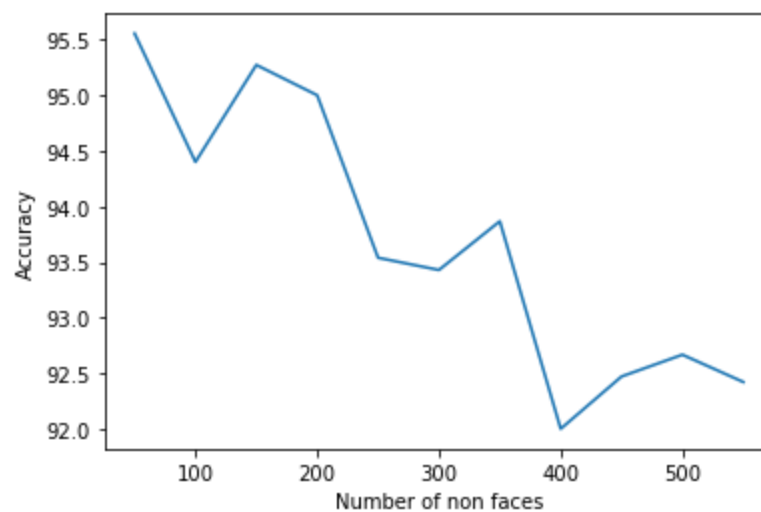
- A. Computed the number of dominant eigenvectors used for LDA.
- B. Projected the training and testing data using LDA.
- C. Evaluated the accuracy of the KNN classifier after applying LDA.

## VII. Display success and failure cases of the classifier.



## VIII. Accuracy vs. Number of Non-Faces Images:

- Plotted the accuracy of the KNN classifier against the number of non-face images in the training data.
- Analyzed how the accuracy changes with the increasing number of non-face images.





## IX. Accuracy Comparison with Different Data Splits:

- A. Compared the accuracy of the KNN classifier with PCA for different data splits (70-30 vs. 50-50).
- B. Evaluated the impact of the data split ratio on classification accuracy.

## Comparison Between

### Principal Component Analysis (PCA) & Linear Discriminant Analysis (LDA)

They are both techniques used in the field of pattern recognition and machine learning, particularly for dimensionality reduction and feature extraction. While they are often used for different purposes, they can sometimes be confused due to their similarities. Let's break down their differences and purposes:

#### I. PCA

primarily used for dimensionality reduction. It transforms the data into a new coordinate system such that the greatest variance by any projection of the data lies on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.

It's an unsupervised technique, meaning it doesn't consider class labels during its operation.

Aims to maximize the variance of the data in the transformed space. It seeks to find the directions (principal components) along which the data varies the most.

The output of PCA is a set of orthogonal axes (principal components) along which the data can be projected.

It's commonly used for noise reduction, data compression, and visualization. It is often used as a preprocessing step before applying other machine learning algorithms to reduce the dimensionality of the feature space.

The number of principal components to retain is usually determined based on the explained variance ratio or by cross-validation.

less affected by class separability. It focuses solely on the variance of the data and may not be optimal for classification tasks.



## II. LDA

It's a supervised dimensionality reduction technique. It considers both the input features and the associated output labels to find the feature subspace that optimally discriminates between different classes.

Aims to maximize the between-class scatter while minimizing the within-class scatter. It seeks to find a projection that maximizes the distance between the means of different classes while minimizing the variance within each class.

The output of LDA is also a set of axes, but these axes are chosen to maximize class separability.

It's commonly used for classification tasks where the goal is to maximize the separation between classes. It is particularly useful when the classes are well-defined and separable.

The number of discriminant components is at most  $C-1$  where  $C$  is the number of classes in the dataset.

LDA explicitly considers class separability and is more suitable for classification tasks, especially when classes are well-separated.

## III. Accuracies Comparison

K-values Alpha values	1	3	5	7
0.8	95%	89.5%	85%	80.5%
0.85	95%	89.5%	84.5%	77.5%
0.9	94%	89%	83.5%	77.5%
0.95	94%	89.5%	84.5%	74%
LDA	96.5%	96.5%	96.5%	96.5%

## Questions

### PCA

#### 1. What is Principal Component Analysis (PCA)?

*PCA is a dimensionality reduction technique used in statistics and machine learning to transform high-dimensional data into a lower-dimensional representation, preserving the most important information.*

#### 2. What is Principal Component?

Principal components represent the directions of the data that explain a **maximal amount of variance**. The lines that capture most information of the data. The relationship between variance and information here, is that, the larger the variance carried by a line, the larger the dispersion of the data points along it, and the larger the dispersion along a line, the more information it has.


#### 3. What is Covariance Matrix?

The covariance matrix is a  $p \times p$  symmetric matrix (where  $p$  is the number of dimensions). It's actually the sign of the covariance that matters:

If positive then: the two variables increase or decrease together (correlated)

If negative then: one increases when the other decreases (Inversely correlated)

#### 4. What are Eigenvectors and Eigenvalues?



The eigenvectors of the covariance matrix are actually *the directions of the axes where there is the most variance* (most information) and that we call Principal Components. And eigenvalues are the coefficients attached to eigenvectors, which give the *amount of variance carried in each Principal Component*. Their number is equal to the number of dimensions of the data. For example, for a 3-dimensional data set, there are 3 variables, therefore there are 3 eigenvectors with 3 corresponding eigenvalues.

## 5. Disadvantages of Principal Component Analysis

1. **Interpretation of Principal Components:** The principal components created by Principal Component Analysis are linear combinations of the original variables, and it is often difficult to interpret them in terms of the original variables. This can make it difficult to explain the results of PCA to others.
2. **Data Scaling:** Principal Component Analysis is sensitive to the scale of the data. If the data is not properly scaled, then PCA may not work well. Therefore, it is important to scale the data before applying Principal Component Analysis.
3. **Information Loss:** Principal Component Analysis can result in information loss. While Principal Component Analysis reduces the number of variables, it can also lead to loss of information. The degree of information loss depends on the number of principal components selected. Therefore, it is important to carefully select the number of principal components to retain.
4. **Non-linear Relationships:** Principal Component Analysis assumes that the relationships between variables are linear.

However, if there are non-linear relationships between variables, Principal Component Analysis may not work well.

5. **Overfitting:** Principal Component Analysis can sometimes result in overfitting, which is when the model fits the training data too well and performs poorly on new data. This can happen if too many principal components are used or if the model is trained on a small dataset.

## LDA

### Why is the number of dominant eigenvectors used for LDA = 1?

Number of Classes: In a binary classification problem (faces vs. non-faces), there are two classes: faces and non-faces. Number of Dominant Eigenvectors for LDA: In LDA, the number of dominant eigenvectors used is often limited by the number of unique classes minus one. This is because LDA seeks to find discriminant directions in the data that maximize class separability. In a binary classification problem, using one dominant eigenvector is common since it's the maximum number allowed by the formula. (unique classes - 1)