Ex No. 1

Jupyter Notebook and Numpy

Date: 31/08/2024

Question 1

i) Array Creation and Basic Operations

a) Create a 1D array of numbers from 0 to 19 using `np.arange()`.

b) Reshape this array into a 4x5 2D array.

c) Create a 3x3 array of all zeros, and another 3x3 array of all ones.

d) Generate a 1D array of 10 evenly spaced numbers between 1 and 100 using `np.linspace()`.

ii) Indexing, Slicing, and Iterating

a) From the 4x5 array created in 1b, extract the following:

- The element at position (2,3)
- The second row
- The last column

b) Reverse the order of elements in the second row.

c) Create a 3x3 array with random integers between 1 and 100, then find all elements greater than 50.

iii) Conditional Statements and Boolean Operations a) Create a 5x5 array of random integers between 0 and 100. b) Create a boolean mask for elements between 30 and 70 (inclusive). c) Use this mask to replace all elements outside this range with the value -1.

iv) Basic Mathematical Operations

a) Create two 3x3 arrays with random integers between 1 and 10.

b) Perform element-wise addition, subtraction, multiplication, and division on these arrays.

c) Calculate the dot product of these two arrays.

v) Universal Functions a) Using the array from task 3a, calculate:

- The mean of the entire array
- The standard deviation of each row
- The variance of each column
- The median of the entire array
- The minimum and maximum values of each row

vi) Shape Manipulation

a) Create a 1D array of 24 random integers between 0 and 100.

b) Reshape this array into a 4x6 2D array.

c) Flatten this 2D array back to 1D.

d) Transpose the 2D array.

e) Stack two 3x3 arrays vertically, then stack the result with another 3x3 array horizontally.

vii) Copy Methods

a) Create a 3x3 array of random integers.

b) Create three copies of this array using direct assignment, `view()`, and `copy()`. c) Modify an element in each copy and explain the differences in how the original array is affected.

viii) Handling NULL Values

a) Create a 4x4 array of random floats between 0 and 1.

b) Replace some values with NaN (Not a Number).

c) Use `np.isnan()` to create a boolean mask identifying the NaN values.

d) Calculate the mean of the array, ignoring the NaN values.

ix) Advanced Challenge

a) Create a 10x10 array of random integers between 1 and 100.

b) Find the indices of the 5 largest values in the entire array.

c) Calculate the running average along each row (hint: look into `np.cumsum()`).

d) Normalize each column of the array so that it sums to 1.

Question 2

Write a python program to perform matrix operations on an M x N matrix and solve a system of linear equations. Use inbuilt functions to implement the operations. Get two matrices from the user. The program should support the following Menus,

1. Matrix Addition
2. Matrix Subtraction
3. Scalar Matrix Multiplication
4. Elementwise Matrix Multiplication
5. Matrix Multiplication
6. Matrix Transpose
7. Trace of a Matrix
8. Solve System of Linear Equations
9. Determinant
10. Inverse
11. Eigen Value and Eigen Vector

12. Exit

Ex No. 2 **Pandas and Matplotlib**

Date: 31/08/2024

Question 1

Dataset

For this assignment, we'll be using the "Heart Failure Prediction" dataset from Kaggle. You can find it at:
https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction

Download the CSV file and name it 'heart_failure_prediction.csv'.

Instructions

Complete all the following tasks using Pandas. Make sure to import Pandas at the beginning of your script.

1. **Series Creation and Access**

   a) Create a Pandas Series using a list of the first 10 prime numbers.

   b) Create a Pandas Series using a NumPy array of 5 random numbers between 0 and 1.

c) Create a Pandas Series using a dictionary where the keys are 'A', 'B', 'C', 'D', 'E' and the values are random integers between 1 and 100.

d) For the Series created in (c), access and print the value associated with key 'C'.

2. **DataFrame Creation**

   a) Create a DataFrame using a dictionary where each key is a column name and each value is a list of 5 random integers between 1 and 100. Use the column names 'A', 'B', and 'C'.

   b) Create a DataFrame with 3 columns and 4 rows of random numbers between 0 and 1. Set the index to ['W', 'X', 'Y', 'Z'] and the column names to ['Col1', 'Col2', 'Col3'].

3. **Loading and Writing Data**

   a) Load the 'heart_failure_prediction.csv' file into a DataFrame.

   b) Write the first 100 rows of this DataFrame to a new CSV file named 'heart_failure_sample.csv'.

   c) Write the entire DataFrame to an Excel file named 'heart_failure_data.xlsx'.

   d) Write the DataFrame to a JSON file named 'heart_failure_data.json'.

4. **DataFrame Information**

   a) Display the info of the heart failure DataFrame.

   b) Show the first 10 rows of the DataFrame.

c) Show the last 5 rows of the DataFrame.

5. **Column Operations**

   a) Extract and display the 'Age' and 'Sex' columns.

   b) Add a new column named 'AgeGroup' where patients under 50 are labeled 'Young', those between 50 and 70 are 'Middle-aged', and those over 70 are 'Senior'.

   c) Delete the 'RestingECG' column.

6. **Accessing Data**

   a) Access the value in the 5th row and 3rd column using direct indexing.

   b) Use `loc` to select all rows where 'Sex' is 'M' and display their 'Age' and 'ChestPainType'.

   c) Use `iloc` to select the first 5 rows and the 2nd to 4th columns.

7. **Conditional Statements**

   a) Create a new DataFrame containing only patients over 60 years old with heart disease.

   b) Find the average age of patients for each chest pain type.

8. **Handling Missing Values**

   a) Check if there are any missing values in the DataFrame.

   b) If there are missing values, fill numeric columns with the mean of that column and categorical columns with the mode (most frequent value) of that column.

9. **Concatenating Data**

a) Split the DataFrame into two parts: one with males and another with females.

b) Add a new column 'Group' to each part, with value 'Male' for the male DataFrame and 'Female' for the female DataFrame.

c) Concatenate these two DataFrames back together.

10. **Applying Functions**

a) Create a custom function that calculates BMI (Body Mass Index) using the formula: BMI = weight(kg) / (height(m))^2. Apply this function to create a new 'BMI' column. (Assume 'Weight' is in kg and 'Height' is in meters)

b) Use a lambda function to create a new column 'AgeTimes100' that multiplies the 'Age' by 100.

Question 2

Instructions

Using matplotlib, create a single figure with multiple subplots that demonstrate your understanding of various line plotting techniques and customizations. Follow the steps below and provide the code for each task.

```
import numpy as np

import matplotlib.pyplot as plt


# Use this array for x-values in all plots

x = np.linspace(0, 10, 100)
```

1. Create a figure with 2 rows and 2 columns of subplots. Set the figure size to 12 inches wide by 10 inches tall.

2. In the first subplot (top-left):

   a) Plot three lines: y = x, y = x^2, and y = x^3

   b) Use different colors for each line

   c) Set different line widths for each line (0.5, 1, and 2)

   d) Add a legend

   e) Set the title to "Basic Line Plots"

3. In the second subplot (top-right):

   a) Plot y = sin(x) four times

   b) Use the same color for all lines

   c) Set different line styles for each: solid, dashed, dash-dot, and dotted

   d) Add a legend that describes each line style

   e) Set the title to "Line Styles"

4. In the third subplot (bottom-left):

   a) Plot y = cos(x) four times

   b) Use the same line style and color for all lines

   c) Add markers to each line: '+', 'o', 's', and '^'

   d) Vary the size of markers: 5, 8, 11, and 14

e) Add a legend that describes each marker

f) Set the title to "Markers"

5. In the fourth subplot (bottom-right):

   a) Plot y = tan(x) once

   b) Use a custom dashed line style (hint: use set_dashes())

   c) Add circular markers

   d) Set the face color of markers to yellow and edge color to red e) Set the title to "Custom Line and Marker Styles"

6. For all subplots:

   a) Set x-axis and y-axis labels

   b) Add a grid

7. Adjust the layout to prevent overlapping and add a main title to the entire figure: "Matplotlib Line Plot Customization"
8. Save the figure as a PNG file named "matplotlib_assignment.png"

Provide your code and the resulting image. Make sure to comment your code to explain key steps.

Lab - 03 **Euclidean Distance and Correlation Analysis**

Date: 06/09/2024

1. Given tabular data is collected at a small ice cream store. To determine if there is a correlation between temperature and the number of customers, calculate the Pearson correlation coefficient, **without using any built-in functions or libraries**. Compare your results with calculation using the inbuilt function.

Temperature (°F)	Number of Customers
98	15
87	12
90	10
85	10
95	16
75	7
92	14
80	9
88	11
93	13

2. You are given the famous Iris dataset (iris.csv), which contains samples of iris flowers, each described by four features: sepal length, sepal width, petal length, and petal width. The dataset is labeled with three classes: Setosa, Versicolor, and Virginica. The iris.csv is present in the Submission link.

    a. **First take 2 dimensions/features, that is, Sepal Length and Sepal Width.**

    Your task is to implement a classification algorithm using the Euclidean distance metric to predict the class of a given iris sample.

        i. Load the Iris dataset and split it into a training set and a testing set. The testing set should contain the first 3 samples of each class. **Do NOT use the inbuilt function for splitting the data.**
        ii. Plot all the training samples using scatter plot and write the inferences.
        iii. For each sample in the testing set, compute its Euclidean distance to all samples in the training set.
        iv. Identify the K-nearest neighbors with the smallest Euclidean distances for each sample in the testing set, where K is a hyperparameter.
        v. Determine the majority class among the K-nearest neighbors and assign it as the predicted class for the sample.
        vi. Evaluate the accuracy of your classifier on the testing set and report the results.
        vii. Classify using sklearn library knn-classifier and compare your result.

    b. **Repeat the same for all the four dimensions and skip plotting the data.**

    Your Python code should include functions to calculate Euclidean distance, perform classification, and compute accuracy. Also, experiment with different values of K to find the optimal performance.

Submit your Python code along with a brief explanation of your implementation and the accuracy achieved on the testing set.

3. For the same IRIS dataset, your next task is to **perform correlation analysis** to understand the relationships between these attributes.

   a. Load the Iris dataset into Python and explore its structure, ensuring it contains relevant numerical attributes for correlation analysis.
   b. Calculate the correlation coefficient between each pair of attributes in the Iris dataset.
   c. Visualize the correlation matrix using a heatmap to better understand the strength and direction of the correlations among the iris flower attributes.
   d. Identify highly correlated attribute pairs, both positively and negatively correlated, and discuss their implications in the context of iris flower characteristics.
   e. Based on the correlation analysis, draw insights into which attributes might have a strong influence on the characteristics of iris flowers.

   Your Python code should include functions to calculate correlation coefficients, create the correlation matrix, and generate the heatmap. Ensure to provide clear explanations and insights based on your correlation analysis.

4. Given $(hq - ht)^T = (0.5\ 0.5\ -0.5\ -0.25\ -0.25)$ and

$$A = \begin{pmatrix} 1 & 0.135 & 0.195 & 0.137 & 0.157 \\ 0.135 & 1 & 0.2 & 0.309 & 0.143 \\ 0.195 & 0.2 & 1 & 0.157 & 0.122 \\ 0.137 & 0.309 & 0.157 & 1 & 0.195 \\ 0.157 & 0.143 & 0.122 & 0.195 & 1 \end{pmatrix}$$

Find the **quadratic form distance**.

Lab - 04 Mahalanobis Distance and Correlation Analysis, Bhattacharyya Distance, Cosine Distance and KL distance

Date: 13/09/2024

Part A

NOTE :- Do not use inbuilt function to find the *Mahalanobis distance and Correlation Calculation.*

1. In this question, you will explore the concept of Mahalanobis distance and its application to classifying samples from the Iris dataset. The Iris dataset is a commonly used dataset in machine learning and consists of three classes of iris plants: Setosa, Versicolor, and Virginica. You will compute the Mahalanobis distance for one sample from each class and classify the samples based on their Mahalanobis distance.

   Tasks:

   - Load the Iris dataset (csv file present in the classroom).
   - Choose one random sample from each class (Setosa, Versicolor, and Virginica) which will act as the test data.
   - Compute the mean vector and covariance matrix for each class (without the sample picked in the previous part, now it will act as the test data).

- Calculate the Mahalanobis distance for each of the selected samples with each of the class using the formula:

  Mahalanobis distance = $\text{sqrt}((x - \mu)^T * \Sigma^{-1} * (x - \mu))$

  Where:

  x is the feature vector of the sample.
  μ is the mean vector for each class.
  Σ^{-1} is the inverse of the covariance matrix for each class.

- Compare the Mahalanobis distances for the three samples and classify each sample to the class with the smallest Mahalanobis distance.
- Print the original class and the predicted class for each sample, along with their Mahalanobis distances.

2. You are conducting a study to explore the relationships between various factors and their potential correlation with the test scores of a group of individuals. You have gathered data from a group of individuals, and the table below represents the hours spent on different activities, along with their corresponding test scores. Your task is to analyze the data and determine the type of correlation, if any, between each variable and the test scores, without using built-in functions for calculating correlation.

Hours Studied	Hours Watching TV	Outdoor Activity Time	Hours Listening to Music	Water Consumed	Test Score
2	4	2	2	5	65
3	3	4	3	6	70
4	2	6	4	5	75
5	1	8	1	6	80

Hours Studied	Hours Watching TV	Outdoor Activity Time	Hours Listening to Music	Water Consumed	Test Score
6	0	10	5	4	85
7	0	12	0	5	90

- Based on the provided data, calculate and interpret the correlation coefficient for each variable in relation to the test scores. Identify the variables that show a positive correlation, a negative correlation, and no significant correlation with the test scores.
- Explain why there might be a positive correlation between the "Hours Studied" variable and the test scores. Provide a brief discussion on how this information could be valuable for improving academic performance.
- Calculate and interpret the correlation coefficient between "Hours Watching TV" and test scores. Explain the implications of this correlation in terms of academic achievement and time management.
- Calculate and interpret the correlation coefficient between "Hours Listening to Music" and test scores. Discuss the potential impact of this correlation on concentration and study habits.
- Calculate and interpret the correlation coefficient between "Water Consumed" and test scores. Provide a potential explanation for the observed correlation and its relevance to cognitive function.
- Calculate and interpret the correlation coefficient between "Outdoor Activity Time" and test scores. Discuss how physical activity might influence academic performance.

You are **not** allowed to use built-in functions to calculate correlation. Instead, you should perform the calculations manually, showing all steps clearly. Analyze the correlations between each variable and the test scores to draw meaningful insights from the data.

3. Consider the following images. Obtain the histograms for each of the images. Using a suitable distance measure (Bhattacharyya Distance) , find the distance between the query image and reference images. You can use the inbuilt function to calculate the Bhattacharyya Distance between two histograms, check classroom post for more information.

Figure 1: Query Image

Figure 2: Reference Image 1

Figure 3: Reference Image 2

Part B

NOTE :- **Do NOT use the inbuilt function** for calculating KL Distance, Bhattacharyya Distance and Cosine Distance directly, you can use functions required to build the distances from scratch like inverse of a matrix, etc.

1. Calculate the distance between the two normalized histograms H1 and H2 using each of the following methods:

   (a) KL Distance

   (b) Bhattacharyya Distance

H1 = [0.24, 0.2, 0.16, 0.12, 0.08, 0.04, 0.12, 0.04]

H2 = [0.22, 0.23, 0.16, 0.13, 0.11, 0.08, 0.05, 0.02]

2. Compare two text files doc1.txt and doc2.txt using cosine distance.

   doc1.txt

   " MATLAB is a program for solving engineering and mathematical problems. The basic MATLAB objects are vectors anuse of this program."

   doc2.txtd matrices, so you must be familiar with these before making extensive

   "MATLAB works with essentially one kind of object, a rectangular numerical matrix. Here is some basic information on using MATLAB matrix commands."

Ex No. 5 MST

Date: 20/09/2024

1. In this question, you will implement the Minimum Spanning Tree (MST) approach to cluster the Iris dataset using only the sepal length and sepal width features. Your task is to write a Python program that performs the following steps:
   a. Load the Iris dataset.
   b. Extract the sepal length and sepal width columns.
   c. Calculate the pairwise Euclidean distances between data points using the selected features.
   d. Build the Minimum Spanning Tree using Prim's/Kruskal's algorithm.

e. Identify the two longest edges in the MST and remove them to create three clusters.
f. Visualize the Minimum Spanning Tree with identified clusters.

NOTE: For help you can refer to presentation no 04.

Lab No. 6 Bayes Classifier

Date: 05/10/2024

1. Find and plot the decision boundary between class ω1 and ω2. Assume P(ω1) = P(ω2).
   ω1 = [1,6; 3,4; 3,8; 5,6]
   ω2 = [3,0; 1,-2;3,-4;5,-2]

2. Find and plot the decision boundary between class ω1 and ω2. Assume P(ω1) =0.3; P(ω2)=0.7
   ω1 = [1,-1; 2,-5; 3,-6; 4,-10; 5,-12; 6,-15]
   ω2 = [-1,1; -2,5; -3,6; -4,10, -5,12; -6, 15]

3. Find and plot the decision boundary between class ω1 and ω2. Assume P(ω1) = P(ω2).
   ω1 = [2,6; 3,4; 3,8; 4,6]
   ω2 = [3,0; 1,-2; 3,-4; 5,-2]

4. Implement Bayes Classifier for Iris Dataset.

   Dataset Specifications:

Total number of samples = 150
Number of classes = 3 (Iris setosa, Iris virginica, and Iris versicolor)
Number of samples in each class = 50

Use the following information to design classifier:

Number of training feature vectors (first 40 in each class) = 40
Number of test feature vectors (remaining 10 in each class) = 10
Number of dimensions = 4
Feature vector = <sepal length, sepal width, petal length, petal width>

Use only two features: Petal Length and Petal Width, for 3 class classification and draw the decision boundary between them (2 dimension, 3 regions also called as multi-class problem)

5. Consider the 128- dimensional feature vectors given in the “face feature vectors.csv” file.

Dataset Specifications:

Total number of samples = 800
Number of classes = 2 (labeled as “male” and “female”)
Samples from “1 to 400” belongs to class “male”
Samples from “401 to 800” belongs to class “female”
Number of samples per class = 400

Use the following information to design classifier:

Number of test samples (last 5 in each class) = 5
Number of training samples (remaining 395 in each class) = 395
Number of dimensions = 128

Lab No. 7 Perceptron Algorithm

Date: 18/10/2024

NOTE :-

- Do not use built-in functions.
- Use necessary comments to explain.

1. Train a single perceptron to learn two classes with two inputs x_1 and x_2. Assume that all the weights of the perceptron are initialized as 0, learning rate as 1. Show the calculation for each step. Plot all the samples and final decision boundary.

x1	x2	ω
0.5	3.0	2
1	3.0	2
0.5	2.5	2
1	2.5	2
1.5	2.5	2
4.5	1	1
5	1	1
4.5	0.5	1
5.5	0.5	1

2. Train a single perceptron to learn an AND gate with two inputs x_1 and x_2. Assume that all the weights of the perceptron are initialized as 0. Show the calculation for each step and also draw the decision boundary for each update.

3. Train a single perceptron to learn the two classes in the following table.

x_1	x_2	ω
2	2	1
-1	-3	0
-1	2	1
0	-1	0
1	3	1
-1	-2	0
1	-2	0
-1	-1	1

where x_1 and x_2 are the inputs and ω is the target class. Assume that all the weights of the perceptron are initialized as 0 with learning rates 0.001, 0.01, 0.5 separately. Plot the samples and decision boundary. Also, tabulate the number of iterations required to converge the perception algorithm with these learning rates.

4. From the iris dataset, choose the 'petal length', 'sepal width' for "setosa" and "virginica" flowers. Learn a decision boundary for the two features using a single perceptron. Assume that all the weights of the perceptron are initialized as 0 with the learning rate of 0.01. Output the final weight vector and plot the samples and the decision boundary.
[*Note: Use iris.csv file in the attachments.*]

Lab No - 08 SVM

1. Train a SVM to learn an AND gate with two inputs x_1 and x_2. Assume that all the weights of the perceptron are initialized as 0. Show the calculation for each step and also draw the decision boundary for each update.

2. Train a single perceptron to learn the two classes in the following table.

x_1	x_2	ω
2	2	1
-1	-3	0
-1	2	1
0	-1	0
1	3	1
-1	-2	0
1	-2	0
-1	-1	1

where x_1 and x_2 are the inputs and ω is the target class. Assume that all the weights are initialized as 0 with learning rates 0.001, 0.01, 0.5 separately. Plot the samples and decision boundary. Also, tabulate the number of iterations required to converge the algorithm with these learning rates.

3. From the iris dataset, choose the 'petal length', 'sepal width' for setosa, versicolor and virginica flowers. Learn a decision boundary for the two features using a SVM. Assume that all the weights are initialized as 0 and learning rate as 0.01. Plot the samples and draw the decision boundary. *[Note: => Use iris.csv file in the attachments.*

=> 3 class classification problem.]

4. Use the attached classification dataset.

Files in dataset:

- cv-train.txt - Training instances + labels (200 rows x 58 cols)
- cv-test.txt - Test instances + labels (50 rows x 58 cols)
- Each line of these text files is a separate training example; the first 57 columns correspond to features, while the last column is the label (+1/ − 1).

Train a SVM model to classify the features. Calculate the accuracy of the trained SVM model.

Lab No - 09 PCA

Date : 02-11-2024

1. Consider the 128- dimensional feature vectors (d=128) given in the "gender.csv" file. (2 classes, male and female)
   a) Use PCA to reduce the dimension from d to d‟. (Here d=128).
   b) Display the eigenvalue based on increasing order, select the d‟ of the corresponding eigenvector which is the appropriate dimension d‟ (select d‟ S.T first 95% of λ values of the covariance matrix are considered).
   c) Use d‟ features to classify the test cases (use any classification algorithm taught in class like Bayes classifier, minimum distance classifier, and so on).

   Dataset Specifications:
   Total number of samples = 800

Number of classes = 2 (labeled as “male” and “female”)
Samples from “1 to 400” belongs to class “male”
Samples from “401 to 800” belongs to class “female”
Number of samples per class = 400
Number of dimensions = 128
Use the following information to design classifier:
Number of test cases (first 10 in each class) = 20
Number of training feature vectors (remaining 390 in each class) = 390
Number of reduced dimensions = d‟ (map 128 to d‟ features vector)

2. **Eigenfaces**-Face classification using PCA (40 classes)
   a) Use the following “face.csv” file to classify the faces of 40 different people using PCA.
   b) Do not use the in-built function for implementing PCA.
   c) Use appropriate classifier taught in class (use any classification algorithm taught in class like Bayes classifier, minimum distance classifier, and so on)
   d) Refer to the following link for a description of the dataset: https://towardsdatascience.com/eigenfaces-face-classification-in-python-7b8d2af3d3ea

Question No 1

Step for PCA

1. calculate mean vector
2. calculate covariance matrix W of [dxd]
3. calculate eigen values
4. calculate eigen vectors
5. choose top 'k' eigen vectors corresponding to top 'k' eigen values
6. Project the features d to k (d') orthogonally to get reduced feature vector
7. Let X is the given data which consists of 'N' samples of 'd' dimension
8. Here X is [800x128] and W is [128x128]; Let k is 20 then new W is [20x128] and W^T is [128x20] ;
9. $X_{[800x128]}.W^T_{[128x20]}$ It can be reduce to

   $[X. W^T]_{[800x20]}$

In general,

$d' = X. W^T$

$d' = X_{[Nxd]}. W_{[kxd]}$

$d' = X_{[Nxd]}. W^T_{[dxk]}$

$d' = [X. W^T]_{[Nxk]}$

Lab No - 10 LDA

Date : 08-11-2024

1. For the given data points, plot the lines of best fit for dimensionality reduction after applying LDA and PCA.

x	y	Class Label
1	2	0

3	5	0
4	3	0
5	6	0
7	5	0
6	2	1
9	4	1
10	1	1
12	3	1
13	6	1

2. Consider the 128- dimensional feature vectors (d=128) given in the "gender.csv" file. (2 classes, male and female)
   a) Use LDA to reduce the dimension from d to d'. (Here d=128)
   b) Choose the direction "W" to reduce the dimension d' (select appropriate d').
   c) Use d' features to classify the test cases (use any classification algorithm taught in class, Bayes classifier, minimum distance classifier, and so on).

Dataset Specifications:

Total number of samples = 800
Number of classes = 2 (labeled as "male" and "female")
Samples from "1 to 400" belongs to class "male"
Samples from "401 to 800" belongs to class "female"
Number of samples per class = 400
Number of dimensions = 128
Use the following information to design classifier:
Number of test cases (first 10 in each class) = 20
Number of training feature vectors (remaining 390 in each class) = 390

Number of reduced dimensions = d' (map 128 to d' features vector)

3. **Fisherfaces**- Face classification using LDA (40 classes)

   a) Use the following "face.csv" file to classify the faces of 40 different people using LDA.

   b) Do not use the in-built function for implementing LDA.

   c) Use appropriate classifier taught in class (use any classification algorithm taught in class like Bayes classifier, minimum distance classifier, and so on)

   d) Refer to the following link for a description of the dataset: https://towardsdatascience.com/eigenfaces-face-classification-in-python-7b8d2af3d3ea