Easy Question

Write a Python program to **compute and display the confusion matrix** by comparing two input files containing actual and predicted class labels.

Input Files:

1. `predictions.csv` – contains predicted labels (each line is either 0 or 1)
2. `actual.csv` – contains true labels (each line is either 0 or 1)

Notes:

- Both files will have **the same number of lines** (e.g. 200).
- The label on line *N* in `predictions.csv` corresponds to line *N* in `actual.csv`.

Confusion Matrix Definition:
Your output should be a **2x2 matrix** structured as: `[[TN, FP], [FN, TP]]`

where:

- **TP (True Positives):** actual = 1, predicted = 1
- **FP (False Positives):** actual = 0, predicted = 1
- **FN (False Negatives):** actual = 1, predicted = 0
- **TN (True Negatives):** actual = 0, predicted = 0

Example:

“predictions.csv”:	1 0 1 0 0
“actual.csv”:	1 1 0 0 1
Output:	`[[1, 1], [2, 1]]`

Submission:

- Upload the code and the predicted labels for the test dataset. The format should follow the guidance of the real problems.

Medium Question

Train a machine learning classifier to **predict the species of Iris flowers** based on their morphological features.

Dataset Description: The Iris dataset (https://archive.ics.uci.edu/static/public/53/iris.zip), originally collected by Edgar Anderson, contains measurements of three Iris species – *Setosa*, *Versicolor*, and *Virginica*. Each species is represented by 50 samples, with four features measured for each flower (in centimeters):

1. Sepal length
2. Sepal width
3. Petal length
4. Petal width

Input Files:

1. **`train.csv`:** Contains five columns – the first four are numerical features, and the fifth column is the corresponding species label (*Setosa*, *Versicolor*, or *Virginica*).
2. **`test.csv`:** Contains the same four numerical feature columns but **does not include labels**.

Your Task:

- Use `train.csv` to train a suitable classifier to predict flower species.
- Apply the trained classifier to `test.csv` to **assign species labels** to each test sample.
- **Submission:** Upload the code and the predicted labels for the test dataset. The format should follow the guidance of the real problems.

Difficult Question

Masked–Unmasked Face Verification Challenge

Overview

Facial recognition has become a cornerstone biometric modality due to its contactless nature, ease of acquisition, and high discriminative power. It is widely adopted in critical applications ranging from immigration control at airports to smartphone authentication in daily life. These systems function by comparing a known *enrollment image* (e.g., a passport photo or a previously registered image) with a *verification image* captured during the authentication attempt, determining whether both images belong to the same individual.

However, the COVID-19 global pandemic introduced a significant challenge to traditional face recognition systems: widespread and prolonged use of face masks. These occlusions drastically affect the visible facial features and thus reduce the performance of recognition systems that rely heavily on unoccluded facial cues.

Task Description

In this challenge, participants are required to address the *masked–unmasked face verification problem*. Specifically, you will be provided with:

- **Training Dataset**: A curated subset of the Labeled Faces in the Wild *(LFW)* dataset, which contains multiple facial images of individuals under varied conditions. These images are unmasked and serve as training data to build your face representation models.

- **Testing Dataset**: This contains a series of *image pairs*. Each pair includes:

    - An **unmasked enrollment image** (i.e., a reference image of an individual)

    - A **masked verification image** (i.e., a probe image of a potentially different or same individual wearing a mask)

For each image pair, your task is to predict whether both images belong to the **same person (label: 1)** or to **different people (label: 0)**.

Objective

Develop a robust face verification model that can accurately match masked and unmasked face pairs. Your solution should handle the variations introduced by facial occlusion while maintaining high verification accuracy.

Evaluation Metric

Submissions will be evaluated using the **Accuracy** of the prediction. Accuracy is calculated by dividing the number of correct predictions by the total number of predictions.

Deliverables

Participants must submit a list of binary predictions (`0` or `1`) for each image pair in the test set. Each prediction should indicate whether the pair contains the same individual (`1`) or different individuals (`0`). The code should also be submitted. The submission format should follow the guidance of the real problems.

To assist in your model development, the following resources are provided:

- **Facial Keypoint Detector**:
  This module detects key landmarks on a face such as the eyes, nose, mouth, and facial contours.
- **Starter Code**:
  A baseline Python script is provided to help you, Load the training and testing datasets, Apply facial keypoint detection, Preprocess images (e.g., alignment, cropping, resizing)