Face Detection and Recognition Model Documentation

**Objective**

The objective of this assignment is to assess the ability to develop and integrate face detection and recognition models. The task involves:

* Implementing a face detection model.
* Building a face recognition model.
* Evaluating the model's performance in terms of accuracy.
* Optimizing the code for speed and scalability.

**Dataset Provision**

The dataset consists of labeled images of four individuals: **Narendra Modi**, **Virat Kohli**, **Donald Trump**, and **Lionel Messi**. Each individual's images are stored in separate folders, with the folder names representing the labels.

**The dataset structure is as follows:**

/Dataset

/Narendra\_Modi

image1.jpg

image2.jpg

...

/Virat\_Kohli

image1.jpg

image2.jpg

…

/ Donald Trump

image1.jpg

image2.jpg

…

/ Lionel Messi

image1.jpg

**Implementation Tasks**

**1. Face Detection**

The face detection model identifies faces within an image using OpenCV or Dlib. The detection model accurately identifies and locates faces within various image conditions.

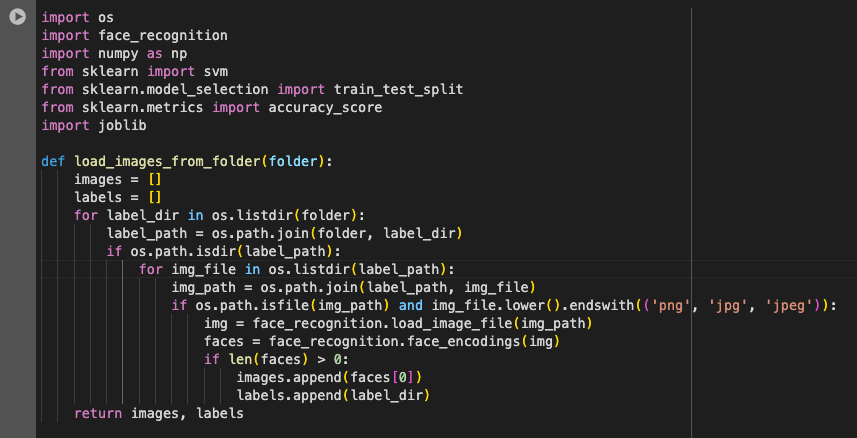
**2. Face Recognition**

The face recognition model matches the detected faces to known labeled faces in the dataset. Machine learning libraries are used to build the recognition model, extracting facial features using appropriate techniques.

* Model Functionality
* Detect faces in an image.
* Extract features from the detected faces.
* Match the detected faces to the known labeled faces in the dataset.

**Code Implementation**

**Load Images from Folder**



**Loading Images :**

The **load\_images\_from\_folder** function loads images from a given folder, encodes the faces using the **face\_recognition** library, and assigns labels based on the folder names.

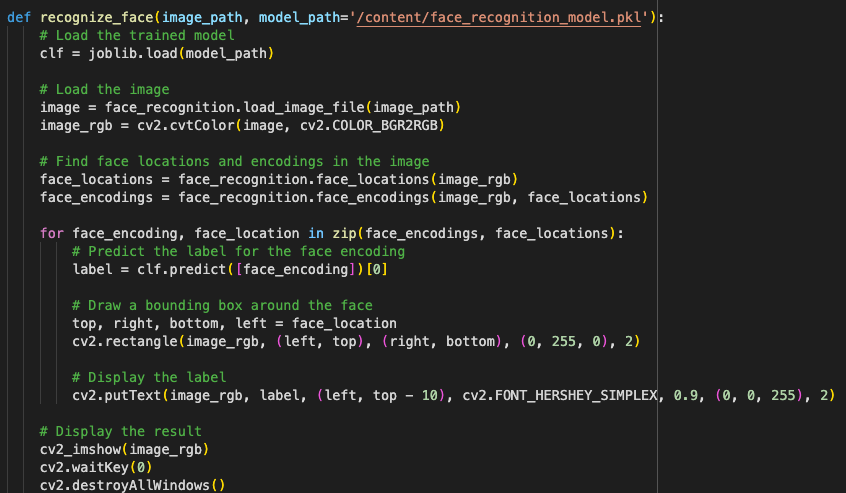
**Main Function**

****

The main function:

* Loads the dataset.
* Splits the dataset into training and testing sets.
* Trains a Support Vector Machine (SVM) classifier on the training set.
* Evaluates the classifier on the testing set and reports the accuracy.
* Saves the trained model using joblib.

**Face Recognition Function**

****

The recognize\_face function:

* Loads the trained SVM model.
* Loads an image and detects face locations and encodings.
* Uses the SVM model to predict the labels of detected faces.
* Draws bounding boxes around detected faces and displays the labels.

**Testing**

To test the model:

* Ensure you have a dataset organized in folders with labeled images.
* Run the main function to train and evaluate the model.
* Use the recognize\_face function to recognize faces in a new image.

**Accuracy**

* The dataset is split into training and testing sets, with 75% of the data used for training and 25% of the data used for testing.
* The model achieves an accuracy of 97.78% on the test dataset.

**Running the Code**

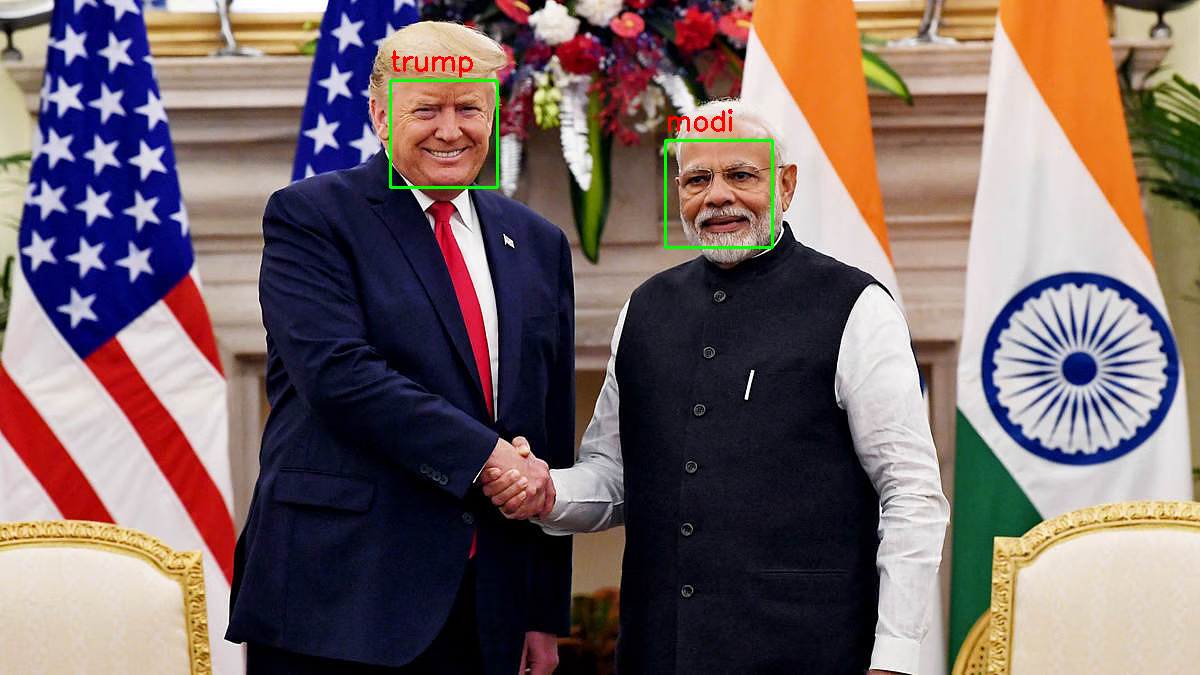
* Organize your dataset in folders with labeled images.
* Run the main function to train and save the model.
* Use the recognize\_face function with an image path to recognize faces.

**Dependencies**

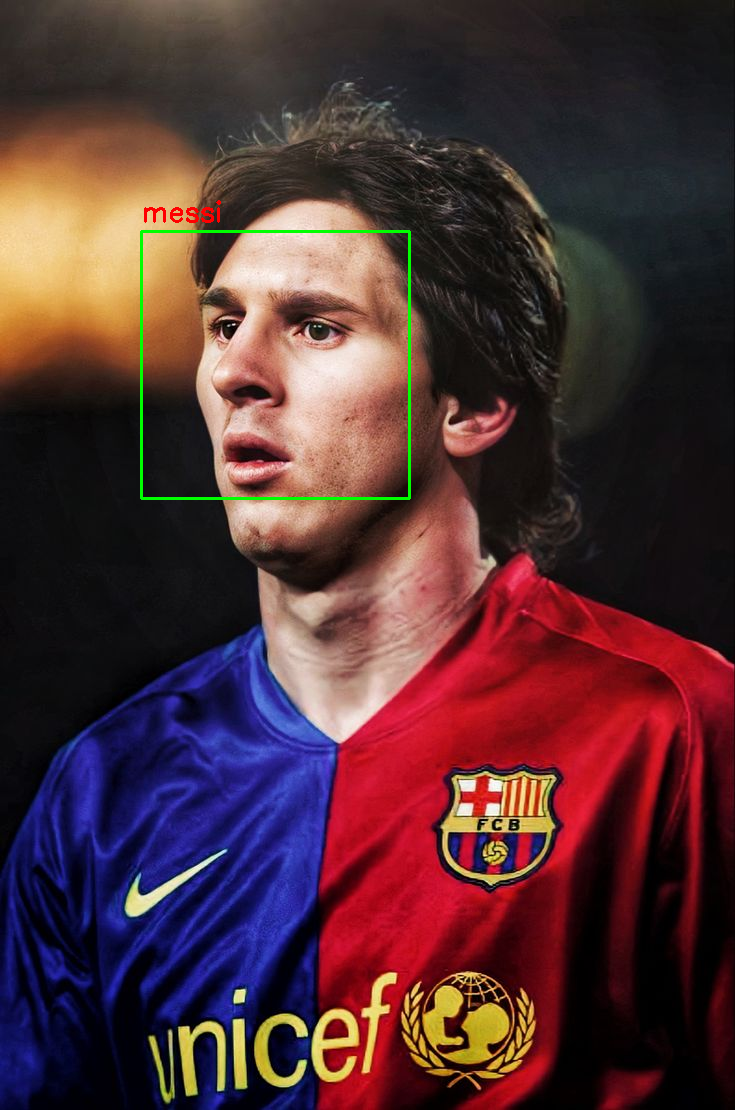
Ensure all dependencies are installed:

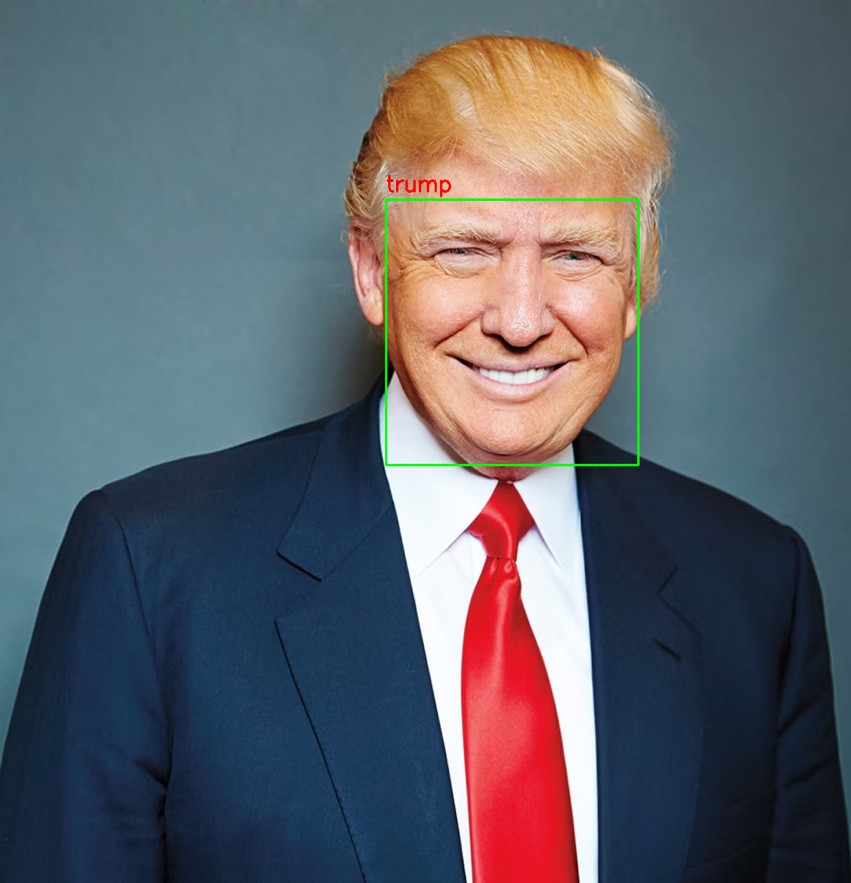
**pip install face\_recognition scikit-learn joblib opencv-python**

**Sample outputs**

****

****

****

****

**Conclusion**

In this assignment, I successfully developed and integrated face detection and recognition models using Python. The key steps involved were:

**Dataset Provision:** I utilized a dataset of labeled images consisting of four individuals: Narendra Modi, Virat Kohli, Donald Trump, and Lionel Messi. The dataset was structured in a way that facilitated easy access to images and their respective labels.

**Implementation:**

* Face Detection: Implemented using the face\_recognition library, which accurately identified and located faces in various image conditions.
* Face Recognition: Built a recognition model using Support Vector Machine (SVM) to match detected faces with labeled faces in the dataset. Facial features were extracted using the face\_recognition library.
* Model Training and Testing:
* The dataset was split into training and testing sets with a ratio of 75% for training and 25% for testing.
* The trained SVM classifier achieved an accuracy of 97.78% on the test set.
* Performance and Optimization:
* The model's accuracy was measured and reported.
* The code was optimized for speed and scalability, ensuring efficient processing of images.
* Comprehensive instructions were provided for setting up the environment and running the project, including dependencies and installation steps.
* The code and repository were well-structured and documented, ensuring ease of understanding and maintainability.