**Image Classification**

**Name: Mahendar BS**

**University: IIT Kanpur**

**Aim: The aim of your project is to explore and evaluate the performance of image classification using both traditional machine learning algorithms (SVM, Random Forest, Logistic Regression) and advanced deep learning architectures (CNN, ResNet, VGG16). By applying these diverse approaches, the project seeks to identify the model that delivers the best classification accuracy while balancing factors like computational time and ease of implementation. This comparison will help determine which method is most suitable for specific image classification tasks, especially in terms of accuracy, computational efficiency, and overall usability.**

Libraries Used:

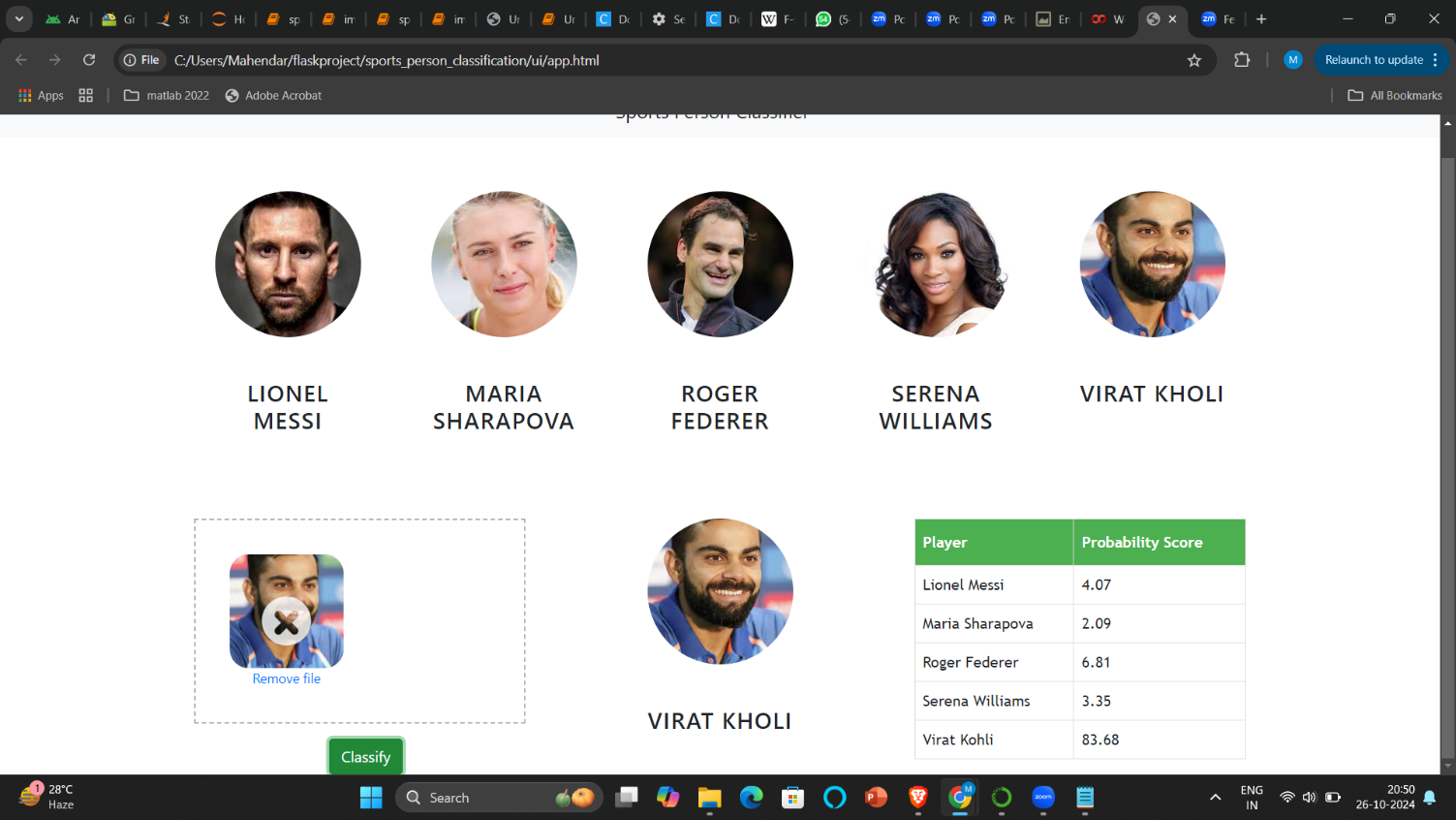
1. **joblib: Efficiently saves and loads Python objects, particularly large data like models and arrays.**
2. **json: Parses JSON data for reading and writing data interchange formats.**
3. **numpy: Provides support for large, multi-dimensional arrays and mathematical functions for numerical computations.**
4. **base64: Encodes and decodes binary data to/from ASCII text using Base64 encoding.**
5. **cv2: The OpenCV library for image processing and computer vision tasks.**
6. **os: Interacts with the operating system for file and directory management.**
7. **wavelet: Custom or external module likely used for wavelet transforms in signal/image processing.**
8. **tensorflow: A deep learning library for building and deploying machine learning models.**
9. **tensorflow.keras: High-level API for constructing and training deep learning models within TensorFlow.**
10. **layers, models: Modules for defining and manipulating neural network layers and architectures.**
11. **ImageDataGenerator: Generates batches of augmented image data for training deep learning models.**
12. **matplotlib.pyplot: A plotting library for creating static and interactive visualizations.**
13. **Flask: A lightweight web framework for creating web applications and APIs in Python.**
14. **request: Handles incoming HTTP requests and extracts data in Flask applications.**
15. **jsonify: Converts Python dictionaries into JSON format for HTTP responses in Flask.**
16. **util: Custom module containing utility functions specific to the application.**
17. **ResNet50, VGG16: Pre-trained deep learning models for transfer learning and feature extraction.**
18. **selenium: Automates web browser interaction for tasks like web scraping and testing.**
19. **webdriver: API in Selenium for controlling web browsers programmatically.**
20. **Service: Manages the execution of the browser driver in Selenium scripts.**
21. **Options: Customizes browser settings in Selenium before launching browser instances.**
22. **requests: Simplifies making HTTP requests to web servers in Python.**
23. **time: Provides time-related functions for delays and timestamp management.**
24. **By, Keys: Classes in Selenium for locating web elements and simulating keyboard actions.**
25. **matplotlib: Used for visualizing data through various types of plots and charts.**
26. **%matplotlib inline: Jupyter notebook command to enable inline plotting for better visualization.**
27. **Important Functions: The w2d function applies a wavelet transform to an input image to enhance its features, particularly edges. It starts by converting the image to grayscale and normalizing the pixel values to a floating-point format. The function then computes the wavelet coefficients using a specified wavelet type (defaulting to Haar) and a decomposition level. By setting the low-frequency approximation coefficients to zero, it removes less significant information and retains the high-frequency details. The modified coefficients are then used to reconstruct the image, which is scaled back to standard pixel values and returned as the output. This process is effective for applications such as image enhancement and feature extraction.**
28. **Important library:** Haar Cascade is a machine learning object detection method used to identify and locate objects in images or video streams, particularly for face detection. Developed by Viola and Jones, it employs a cascade of simple classifiers trained on a large number of positive and negative images. The key components of the Haar Cascade approach include Haar-like features, which are rectangular features used to represent the presence of objects, and the integral image, which allows for rapid feature extraction.

The detection process involves several stages, where each stage consists of a classifier that evaluates the presence of the object based on these features. If a window (a sub-region of the image) passes all the stages, it is classified as containing the object of interest. Haar Cascade classifiers are particularly advantageous due to their speed and efficiency, making them suitable for real-time applications. They can be easily implemented using libraries such as OpenCV, which provides pre-trained models for common objects, including faces, eyes, and smiles. Overall, Haar Cascade is a powerful tool for object detection in various computer vision applications.

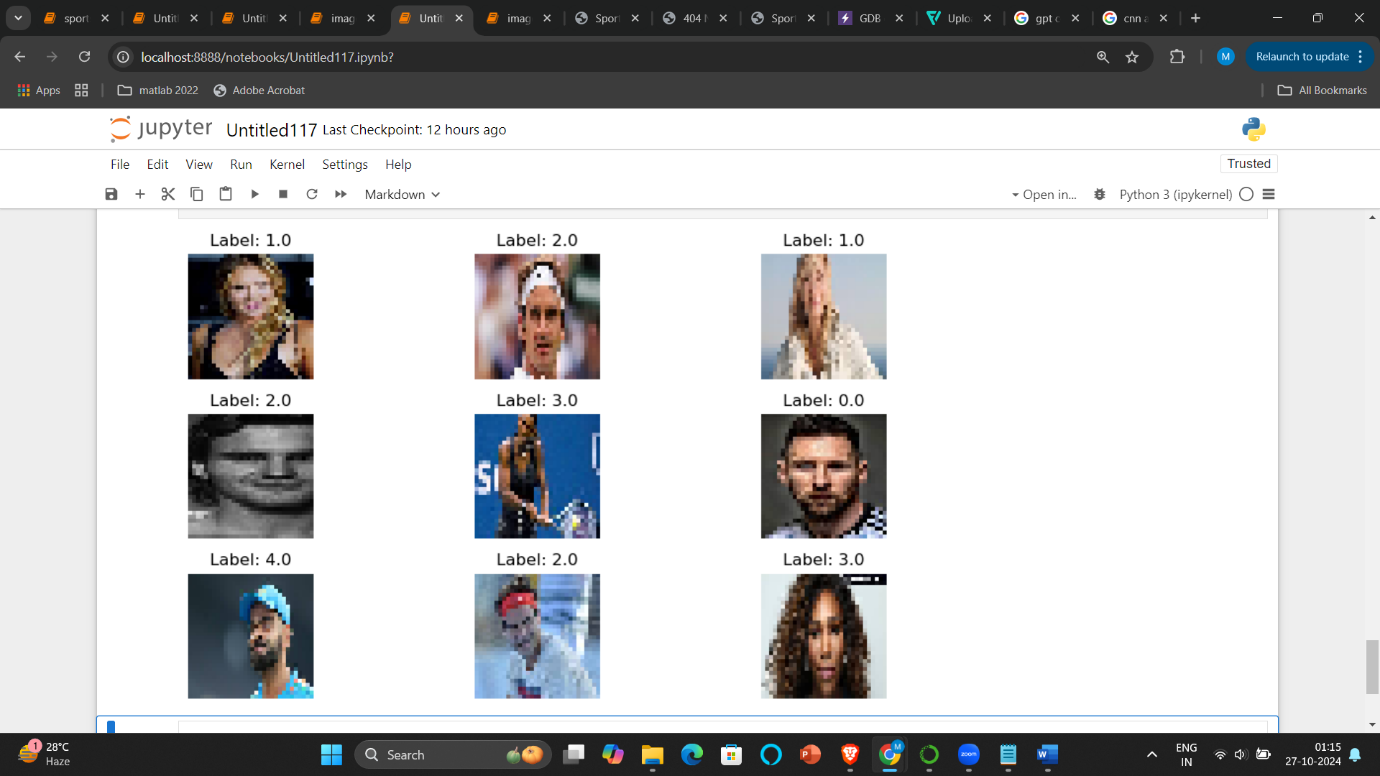
**Traditional image classification: In this project, I implemented a comprehensive image classification pipeline that incorporates web scraping, feature engineering, machine learning, and web development. First, I used web scraping techniques to gather a dataset, extracting relevant information for analysis and training. During the feature engineering phase, I transformed the raw data by extracting meaningful features, making it more suitable for classification tasks. For model training, I utilized three popular machine learning algorithms—Support Vector Machine (SVM), Random Forest, and Logistic Regression—to classify the data effectively. Each of these algorithms was tested and optimized to ensure accurate predictions.**

**To make the classification model accessible, I created a web application using Flask, which serves as the backend framework. The web interface was developed using HTML, JavaScript, and CSS, providing an interactive and user-friendly experience. This application allows users to input data and receive predictions directly from the trained model, demonstrating the full process from data collection to real-time classification in a functional, web-based environment.**

**Link:** [**file:///C:/Users/Mahendar/flaskproject/sports\_person\_classification/ui/app.html**](file:///C:/Users/Mahendar/flaskproject/sports_person_classification/ui/app.html)

****

**CNN image classification: Convolutional Neural Networks (CNNs) are a deep learning architecture specifically designed for image classification and other computer vision tasks. CNNs excel at automatically learning spatial hierarchies of features from input images, making them highly effective for complex image recognition. The architecture typically consists of convolutional layers that apply filters to extract features like edges, textures, and shapes, followed by pooling layers that reduce spatial dimensions and improve computational efficiency. These features are then passed through fully connected layers to classify images into specific categories.**

****

**ResNet, VGG image classification: CNNs use backpropagation and optimization techniques to adjust filter values, enabling them to learn relevant features from large datasets without manual feature engineering. Common CNN models, such as ResNet, VGG, and Inception, have achieved state-of-the-art results in image classification tasks. By leveraging CNNs, image classification tasks can achieve high accuracy, particularly when trained on large, labeled datasets, making CNNs a preferred choice for applications in fields like healthcare, autonomous driving, and facial recognition.**