**PROJECT DOCUMENTATION**

(AI&ML)

**Pollen's Profiling: Automated Classification of Pollen Grains**

**Introduction:**

* **Project title :** Pollen’s Profiling
* **Team members:** 1. Tangella Rohith Lakshman Kumar

2.Sunkara Manasa Sasi Devi

* **Team ID :** LTVIP2025TMID35515

**Project Overview:**

* **Purpose:** The goal of Pollen's Profiling is to develop an intelligent automated system for the identification of pollen grains through the use of an automated image processing and deep learning framework. Pollen identification has traditionally been a manual process, and a slow one, mainly owing to requiring specific expertise. This project aims to automate and modernize the pollen identification process by utilizing artificial intelligence to analyze pollen and classify it using its main morphological parameters (such as shape, size and features of its surface). The aim is to increase the speed of identification, consistency, and accuracy of an identification, while also decreasing human effort and error. The system will provide abilities to apply to practices such as environmental monitoring, allergy analysis, and Agri-science research, which will allow faster, data-driven decisions to be made - alongside improving the overall out of ecological studies, health-related studies and agriculture yield using the framework to collect information.
* **Features**: Pollen's Profiling is equipped with a variety of features that can automate and improve the pollen grain classification process. It utilizes automatic image processing to clean, segment, and enhance the microscopic images of pollen samples. Ultimately, it uses deep-learning protocols, more specifically convolutional neural networks (CNN), to classify pollen grains according to key morphological attributes, including size, shape, and surface texture. It uses multi-class classification, allowing it to classify many types of pollen, including closely related types that may look similar at first glance. Pollen's Profiling can run analysis in real-time, making it ideal for time-sensitive applications like environmental monitoring and clinical diagnostics.

Once analyzed and classified, a simple and easy to use interface allows users to upload images, receive classification results report, and generate a report. The platform can also be trained on user specific datasets, allowing it to classify various pollen types or specific use cases. With data logging, saving results, and automated report generation built in, Pollen's Profiling can provide traceability and documentation for every analysis. Finally, Pollen's Profiling is scalable, all the way from laboratory or field unit processing to working with cloud based services. It reflects high accuracy and consistency while being adaptable.

**Architecture:**

1.Front-end (HTML Interface):

* A basic, web-based user interface that allows users to upload images of pollen grains.
* Built with HTML, served by the Flask application.
* Provides the option for users to select images and view classification results in their browser.

2. Back-end (Flask Application):

* The Flask application acts as the web server that services the HTTP requests from the front-end.
* Receives the uploaded images and processes them from the HTML form for the analysis.
* Interfaces with the trained CNN model in order to classify the pollen.
* Sends the prediction (pollen type, confidence) to the front-end for display.
* Manages routing from the upload form to the display of the results.

3. Machine Learning Module (CNN Model):

* The functionality of a convolutional neural network trained on pre-processed pollen images.
* What this model does is performs feature extraction, and classification (pollen grain) based on morphological traits.
* The module can run inference on the images uploaded to the Flask back-end.
* Returns in a data structure a class for the predicted pollen and a score in confidence to the Flask application.

**Methodology:**

**Data Collection**

The first step involves collecting a comprehensive dataset of pollen grain images. These images can be sourced from public datasets, research institutions, or manually captured through a microscope-camera setup. The dataset must include a variety of pollen types representing different species, ensuring diversity and robustness in the training process.

**Data Preprocessing**

Raw images undergo preprocessing to enhance model performance:

**Resizing:** All images are resized to a uniform dimension (e.g., 128x128 or 224x224 pixels) to standardize input size for the CNN model.

**Normalization:** Pixel values are normalized to a range between 0 and 1 to improve convergence during training.

**Data Augmentation:** Techniques such as rotation, flipping, zooming, and shifting are applied using tools like ImageDataGenerator to artificially increase dataset size and help the model generalize better.

**Train-Test Split:** The dataset is divided into training and testing sets, typically in an 80:20 ratio, to evaluate model performance on unseen data.

**Model Design and Development**

A Convolutional Neural Network (CNN) architecture is chosen for its proven effectiveness in image classification tasks.

**Input Layer:** Accepts preprocessed pollen images of fixed size.

**Convolutional Layers:** Multiple convolutional layers apply filters to extract hierarchical features such as edges, textures, and shapes.

**Pooling Layers:** Max-pooling layers reduce spatial dimensions while retaining important features, helping to reduce computational load.

**Fully Connected Layers:** Flattened feature maps are passed through dense layers to learn complex combinations of features.

**Output Layer:** A softmax activation layer outputs probability scores for each pollen class.

The model is compiled using an optimizer like Adam, with categorical cross-entropy as the loss function and accuracy as the evaluation metric.

**Model Training**

The CNN is trained on the augmented training set using batch processing and multiple epochs. The ImageDataGenerator class facilitates real-time data augmentation during training. Validation accuracy is monitored after each epoch to detect and mitigate overfitting, employing callbacks such as early stopping if necessary.

**Model Evaluation**

After training, the model’s performance is evaluated on the test dataset using metrics including:

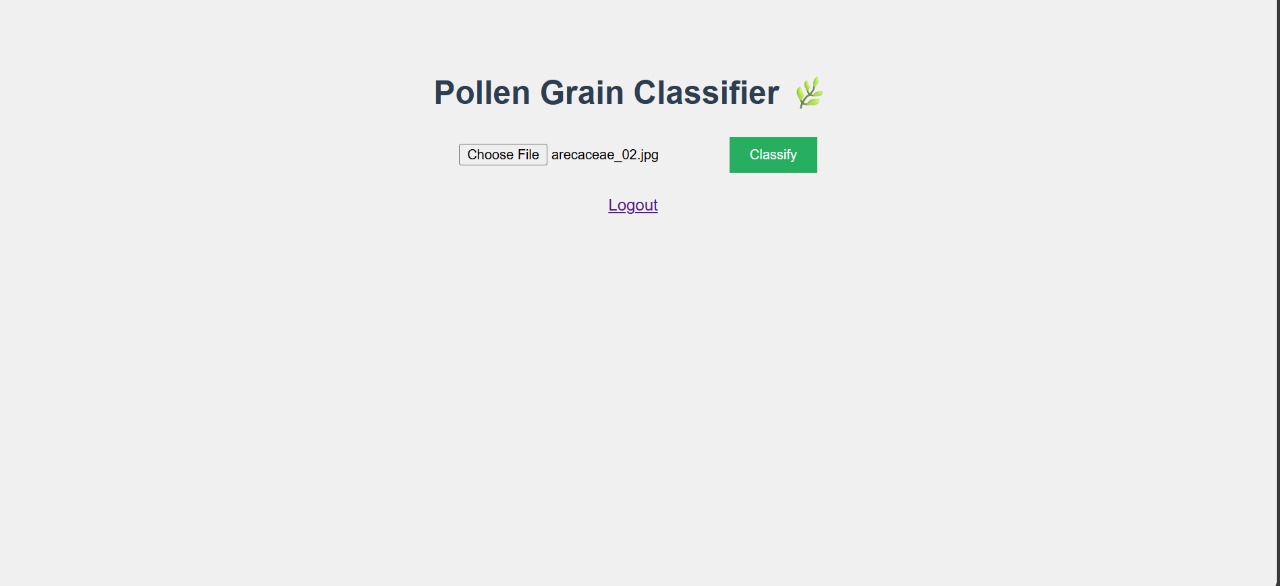
**Accuracy:** Overall correctness of predictions.

**Model Deployment**

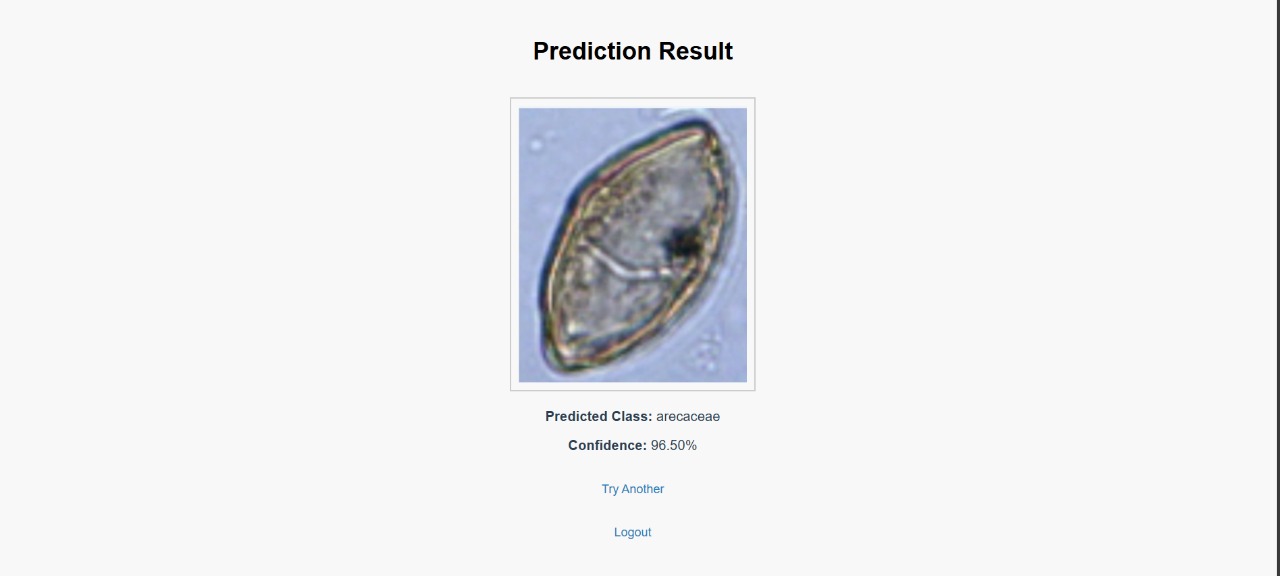
The trained CNN model is saved and integrated within a Flask web application. The Flask backend accepts images uploaded via the frontend, processes the image, feeds it to the model for prediction, and returns the results to the user interface for display. This setup enables real-time pollen classification through a user-friendly web interface.

**Implementation:**

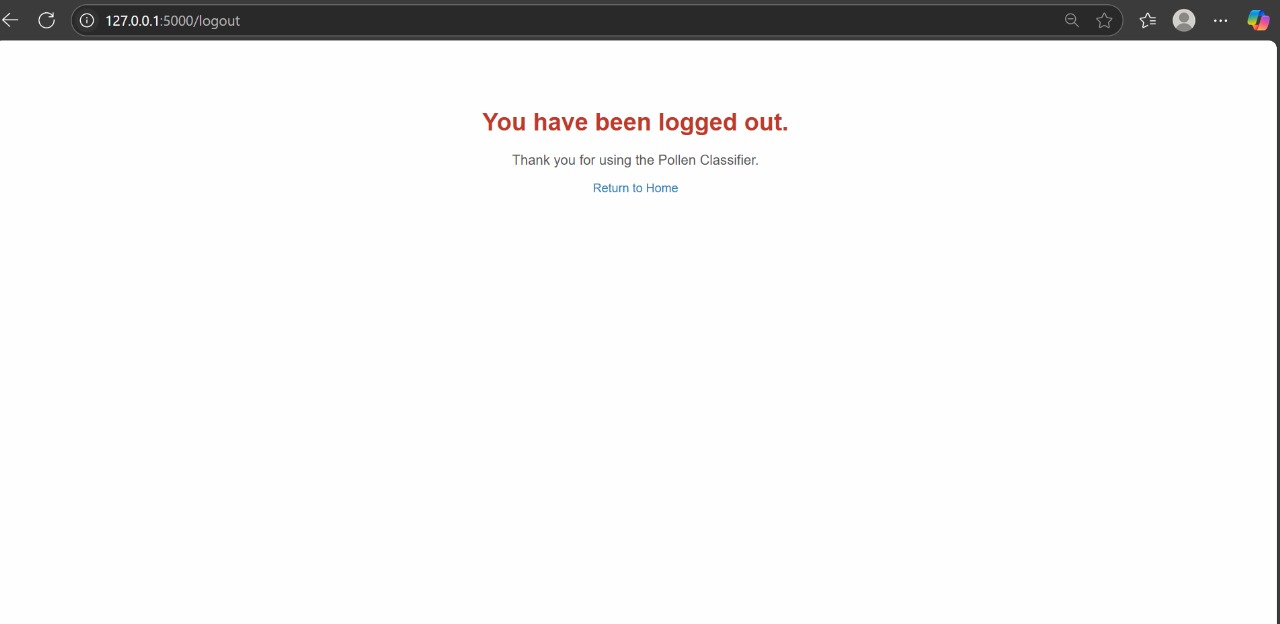
The implementation of *Pollen’s Profiling* involves a simple web interface built with HTML, CSS, and JavaScript that allows users to upload pollen images..



These images are sent to a Flask backend, which processes the input by resizing and normalizing it before passing it to a pre-trained CNN model built with TensorFlow. The model predicts the pollen type and returns the result to the backend, which then displays the classification and confidence score on the frontend. The system integrates image preprocessing using OpenCV and provides a smooth user experience by handling uploads, running inference, and showing results all within a lightweight, local web application.



The system also includes a simple user session management feature with login and logout functionality. After logging in, users can upload images and view results, and when they choose to logout, their session is securely terminated, redirecting them to the login page. This ensures that user data and access remain protected during usage, enhancing overall security and user experience within the web application.



**Conclusion:**

*Pollen’s Profiling* successfully demonstrates the automation of pollen grain classification using deep learning and image processing techniques. By integrating a convolutional neural network with a user-friendly Flask web interface, the system enables accurate, fast, and accessible pollen identification. This automation reduces manual effort, enhances consistency, and supports diverse applications in environmental monitoring, healthcare, and agriculture. The project lays a strong foundation for further improvements such as expanding the dataset, incorporating more pollen types, and adding user management features to make the system more robust and scalable.