Leaf Identification App: Software Developer

ROSHANAK BEHROUZ
Università degli Studi di Trieste
Piazzale Europa, 1
Trieste, TS. Italy.34127
ROSHANAK.BEHROUZ@studenti.units.it

ABSTRACT

This report presents the development of a web application designed for the identification of leaves based on their features. Utilizing a machine learning approach, the application enables users to predict by inputting specific morphological features through an interface. Built with Streamlit, the app provides a rapid prototyping environment that allows for direct interaction with the predictive model.

The core functionality of the application is underpinned by a robust machine learning model, primarily using the RandomForest classifier, which has demonstrated effectiveness in handling multiple features and producing reliable predictions. Users can manually adjust feature values via sliders or enter values manually in embedded boxes. After that user can check the scientific name of the leaf by entering the class number.

Despite its functional completeness and user-friendly design, initial testing has revealed some limitations in model performance under certain conditions. improvements aim to refine the user experience and prepare the platform for a broader deployment in educational and research contexts where plant species identification is required.

Keywords

Web Application Development, User Interface Design, Streamlit, Frontend Development, Joblib.

1. INTRODUCTION

This application could serve educational purposes, assist botanists, or even help hobby gardeners. This web app Allows users to predict the species of a plant based on specific morphological features of its leaves. Users input quantitative data about leaf characteristics, and the machine learning model generates a prediction of the plant species. Users interact with the web application through a series of sliders that adjust the values of various leaf features such as Eccentricity, Aspect Ratio, Solidity, and more, reflecting the physical attributes of the leaf they are examining. User can also enter values manually which uses the same machine learning model to predict. The app which is made using Streamlit provides a responsive and interactive user interface that updates the predictions in real-time as the input sliders are adjusted.

1.1 Functional Requirements

The WebApp should be able to take in user input for the following features required by the machine learning model:

- "Eccentricity": (Float <= 1.0),
- "Aspect Ratio": (Float ≥ 1.0 and <=20.0),
- "Elongation": (Float <= 1.0),
- "Solidity": (Float ≥ 0.4 and 1.0),
- "Stochastic Convexity": (Float ≥ 0.3 and ≤ 1.0),
- "Isoperimetric Factor": (Float ≤1.0),
- "Maximal Indentation Depth": (Float ≤0.2),
- "Lobedness": (Float ≤8.0),
- "Average Intensity": (Float ≤0.2),
- "Average Contrast": (Float ≤0.3),
- "Smoothness": (Float ≤0.08),
- "Third Moment": (Float ≤0.03),
- "Uniformity": (Float ≤0.003),
- "Entropy": (Float ≤3.0)

- The system shall validate that all user-provided input values fall within predefined acceptable ranges before processing.
- The application shall provide a class prediction upon the user's submission of the form.
- The application shall provide scientific name of the species looking up by class number.
- User can choose between two choices of data insertion which are slider or manual insert.

1.2 Nonfunctional Requirements

- The WebApp should be user-friendly and intuitive (even if the WebApp is a prototype).
- The application's codebase should be maintainable and scalable, allowing for straightforward updates or additions to the input fields, reflecting potential enhancements to the machine learning model.

2. BACKGROUND

2.1 Technologies used

I used Streamlit which is a Python framework is particularly well-suited for rapidly developing WebApp prototypes in machine learning contexts. Streamlit's simplicity and maintainability make it an ideal tool for this project, allowing for quick implementations and easy adaptation to future improvements [1].

2.2 Unified Modeling Language Diagram

Unified Modeling Language (UML) diagrams are employed in this project to provide a visual representation of the system's architecture and interactions.
• Use Case Diagram: It provides a high-level view of the system's functionality and the actors involved.
• Sequence Diagram: It provides a high-level view of the system's behavior and the sequence of interactions between
the actors and the system [2].

3. METHODOLOGY/APPROACH

3.1 UML Usecase Diagram

This use case allows the user to choose between slider or manual insertion option. Two modes are not distinct models but rather

- Slider: allow users to input morphological features of leaves by adjusting the slider handles within predefined ranges for each feature, providing a visual and intuitive method for data entry.
- Manual Insertion: on the other hand, this option involves users directly typing numerical values into input fields for each feature, offering precise control over the data provided to the model for making species predictions.
- Lead Name Retrieve: This option consists of entering class number and receiving scientific name of the leaf which use Leaf.csv file and scientific name column.

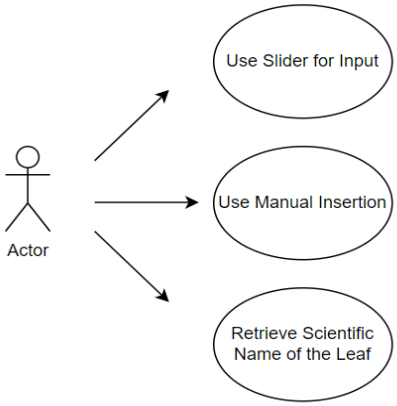

Figure 1: use case diagram

3.2 UML sequence diagram

In this report, I demonstrated all three use-case in sequence diagram.

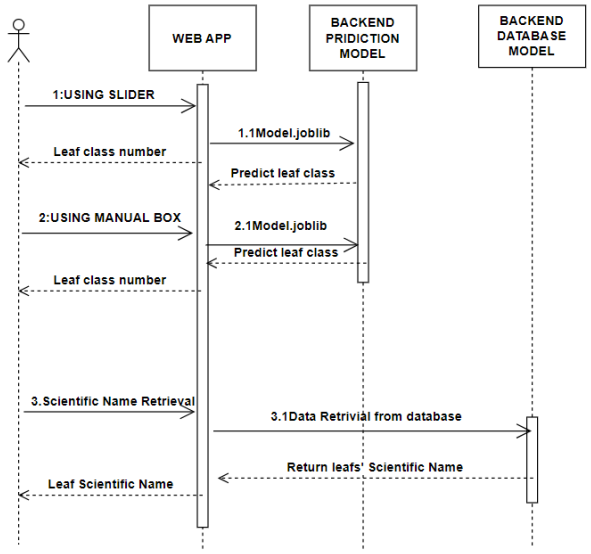

Figure 2: Sequence Diagram

3.3 FRONTEND IMPLEMETATION

The objective was to create a data-driven web application that identifies leaf using 14 specific features. I utilized Streamlit to develop a user-friendly interface with interactive sliders for inputting leaf characteristics like Eccentricity, Aspect Ratio, and Solidity, which are essential for the machine learning model to accurately predict the class. Streamlit's capability for rapid prototyping made it an ideal choice for the frontend.

The web application uses sliders for feature input, allowing users to adjust values between predefined min and max ranges specific to each feature. Once the user sets the values and hits the "Predict" button, the preloaded, serialized machine learning model processes the inputs and displays the predicted class. Additionally, the application can display the scientific name of the leaf by referencing a mapping from class numbers to leaf names stored in a CSV file, enhancing the educational value of the application.

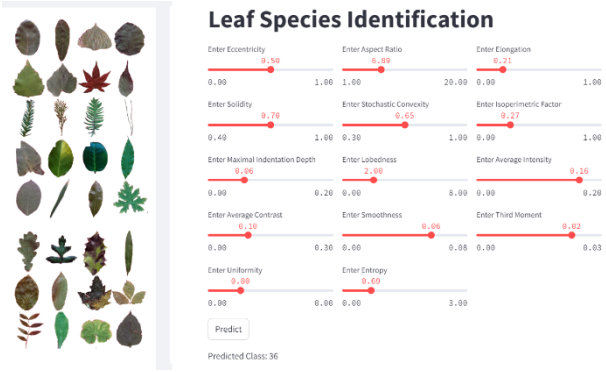

Figure 3: Slider Input

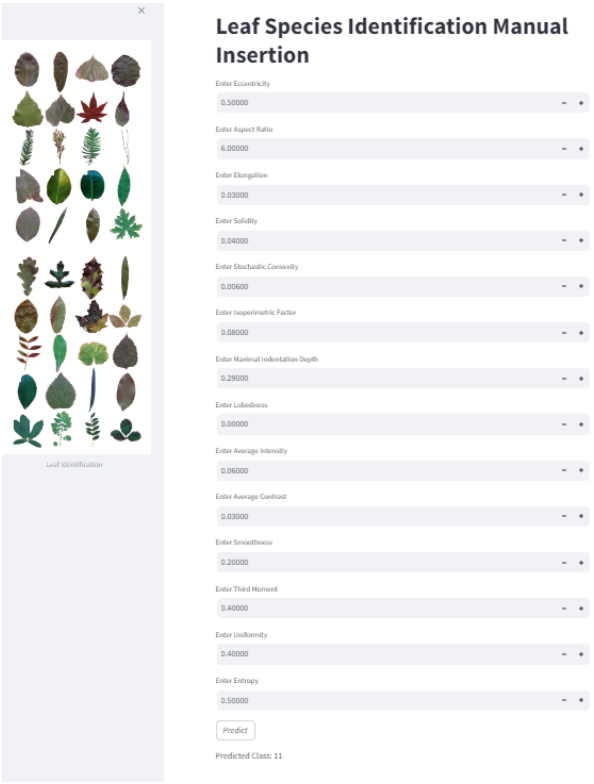

Figure 4: Box Manual Input

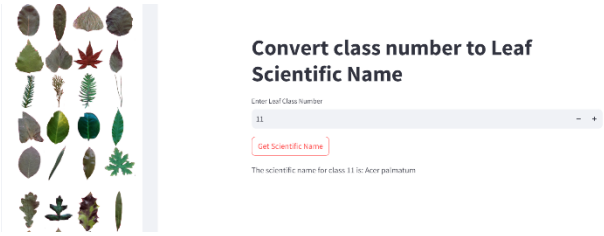

Figure 5: Conversion of class number to Leaf Scientific Name

4. DISCUSSION AND CONCLUSION

Configuring the Streamlit app for deployment involved ensuring that the environment for running the app had all the necessary dependencies and was stable enough to handle potential user interactions without crashing or freezing.

Initially, the plan was to modify slider colors for better visual feedback, which Streamlit does not directly support. This required exploring CSS hacks, which were eventually deemed unnecessary for the project's goals.

5. TEAM STRUCTURE

In the ideal setup for this project on plant species identification using a web application, the workload was intended to be distributed among three key roles: a Data Engineer, a Software Engineer, and a Software Developer. Each role was designed to leverage specific skill sets to contribute to various aspects of the project, ensuring efficiency and specialization in handling complex tasks from data management to user interface design. Due to the unavailability of group mates for this project, I undertook all the responsibilities associated with each role. Managing these roles solo provided a unique challenge and learning opportunity, pushing me to acquire and apply a broad range of skills.

6. REFERENCES

[1] Streamlit. (2022). Streamlit Documentation. Retrieved from https://docs.Streamlit.io

[2] Derek Banas. (2017). UML Diagrams Full Course (Unified Modeling Language). YouTube. Available at: https://www.youtube.com/watch?v=UI6lqHOVHic[Accessed 11 June 2024].