Leaf Identification App: Data Engineer

ROSHANAK BEHROUZ
Università degli Studi di Trieste
Trieste, TS. Italy.34127
Roshanak.behrouz@studenti.units.it

ABSTRACT

This report provides an overview of the Leaf Identification application, which classifies various leaf types using morphological features. Utilizing a well-structured dataset from public repositories [1], the project integrates several machine learning models to explore the potential of automated leaf classification. The report details the deployment of a data-driven pipeline involving data preprocessing, feature engineering, model selection, training, and extensive evaluation. Key models tested including Random Forest, Support Vector Machine (SVM), k-Nearest Neighbors (KNN), and Gaussian Naive Bayes, each chosen for their specific strengths in handling the complexity and variability of the data. This approach emphasized optimizing model parameters through GridSearchCV coupled with StratifiedKFold cross-validation to enhance model performance while ensuring robustness against overfitting. The findings demonstrate the effectiveness of the Random Forest model, achieving notable accuracy, which highlights the model's capability in managing the intricacies of leaf feature data. The results are used to create web base application for leaf identification. I worked as a Data Engineer to create a web app for identifying leaves. This application could serve educational purposes, assist botanists, or even help hobby gardeners.

Keywords

Data Engineering, Data Cleaning, Data Preprocessing, Feature Engineering, MLOPs.

1. PROBLEM DEFINITION

In this report, I detail my role as a Data Engineer in developing a web application for leaf identification web application, a project that combines advanced machine learning techniques with interactive web technologies to facilitate easy and accurate identification of leaves based on their features. I was responsible for the tasks of data cleaning and preparation, feature selection and engineering, ensuring the dataset's integrity and suitability for model training, setting up pipeline and model deployment. This application could serve educational purposes, assist botanists, or even help hobby gardeners. Throughout the project, my focus remained on ensuring that the data infrastructure was robust, scalable, and seamlessly integrated with both the machine learning backend and the Streamlit-based frontend. This report will outline the methodologies employed in handling the data, the challenges encountered, and the solutions developed to overcome these challenges, reflecting on how these efforts contributed to the successful deployment of the web application. This initiative is classified as a multiclass classification problem, where each input must be categorized into one of thirty possible species categories, based on a set of fourteen numerical features that describe the leaf's physical and textural characteristics. The project uses supervised learning due to the availability of labeled data, which facilitates the training of models that can predict the species based on these attributes.

2. BACKGROUND

This section outlines the technologies used, reasons for their selection, and their roles in achieving the project objectives. Python was the primary programming language used due to its extensive library ecosystem and support for data manipulation and machine learning [2]. Libraries such as Pandas for data handling and NumPy for numerical operations were integral for preprocessing the dataset [3]. Pandas was essential for data cleaning, transformation, and analysis. It provided efficient structures for handling large data sets, such as Data Frame, which were used extensively to explore and prepare data for modeling. NumPy Used for its powerful numerical capabilities, NumPy supported high-performance operations on large arrays and matrices, which were pivotal in feature engineering and transformations. Scikit-learn as a versatile tool offering numerous built-in algorithms for machine learning, Scikit-learn was utilized for building and evaluating the Random Forest classifier. It facilitated model training, hyperparameter tuning via GridSearchCV, and validation processes, all critical for the project's success. The choice of Random Forest was driven by its efficacy in handling overfitting compared to other algorithms. Its ability to manage high-dimensional data and provide feature importance made it ideal for this project [4]. Matplotlib and Seaborn libraries were used for visual data analysis, which is crucial in understanding data distributions and the effects of various preprocessing steps. They were also used for generating performance metrics plots, such as confusion matrices, which are vital for reporting model results. Seaborn's Heatmap function was employed to visualize the correlation matrix of the dataset's features, providing insights into potential multicollinearity and guiding feature selection. Git was employed for version control, allowing for efficient collaboration among team members. It helped in managing changes to the project's codebase, facilitating a seamless integration of different components developed by team members. The use of Jupyter Notebook was pivotal for collaborative development, allowing the team to share code. Its interactive environment was perfect for exploring data, prototyping models, and performing step-by-step analyses.

3. DATASET INFORMATION

The dataset utilized in the Leaf Identification System project comprises morphological features of leaves collected from different species. Attributes include geometric properties such as Eccentricity, Aspect Ratio, and Elongation, alongside texture descriptors like Entropy and Smoothness. This dataset, sourced from an open repository, provides a comprehensive basis for analyzing and classifying leaf types. The dataset comprises 340 observations and 16 attributes, including two categorical ones: Class, which denotes the species of the leaf and serves as the dependent variable, and Specimen Number, used to identify individual specimens, but was excluded in our system. The other 14 attributes are numerical and serve as the independent variables. Out of these 14 attributes, 8 describe the shape of the leaf, while the remaining 6 describe the texture of the leaf [1]. There are 30 distinct and well-balanced classes within the dataset, which are all simple leaves (from 1 to 36, excluding 16 to 21 and 37 to 40, which are complex leaves [1]). There are no missing values or duplicate rows. The primary pre-processing step done was to convert the dependent variable to a string type in order to remove potential ordering. Correlation between features was investigated and some features were indeed highly correlated. Correlation Matrix is shown in figure 1. This correlation was considered as a factor in potentially reducing the feature set used in our learning techniques, due to its potential risk for overfitting; however, after experimental testing, it was determined that using all 14 features provided the best performance results on unseen data, indicating that overfitting may not be too high of a concern. After a meeting with software engineer and developer, I added an extra column to the data set which was the scientific name of the leaves. I extract the details from a table which was in the description of the dataset.

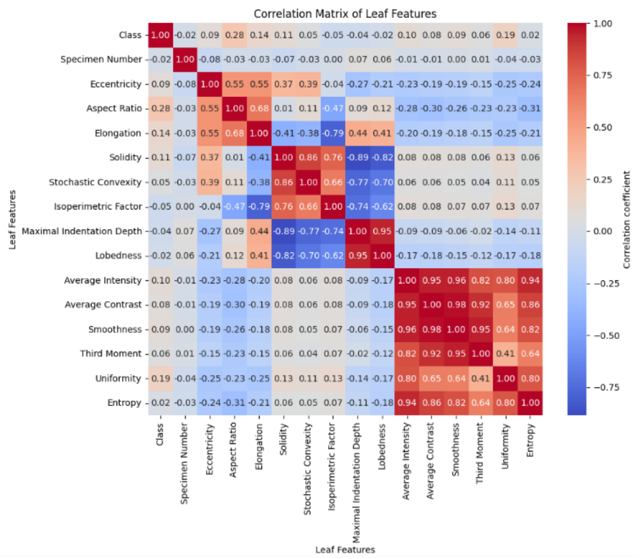


Figure 1: Correlation Matrix

3.1 Key Performance Indicators (KPIs)

Several Key Performance Indicators (KPIs) were defined to assess the efficacy and precision of the leaf classification application. Balanced Accuracy is critical for model performance, particularly in scenarios where class distribution is not uniform, ensuring that the model's effectiveness is maintained across all classes and not disproportionately influenced by the majority class. The Weighted F1 Score, another vital metric, provides deeper insights into the model's precision and recall, factoring in the actual prevalence of each class, which is imperative for handling datasets with diverse class distributions. Additionally, Training Time was scrutinized to evaluate the computational efficiency of each model throughout the cross-validation stages, reflecting the operational feasibility of deploying these models in a real-world setting.

3.2 Data Exploration and Statical Analysis

Descriptive Analysis: The dataset contains 340 observations across 30 distinct classes, with 14 numerical features related to leaf shape and texture. Features such as Class and Specimen Number are categorical, with the latter being excluded from the model to focus on the leaves' physical and textural attributes.

Correlation Analysis: Initial exploration revealed high correlation among some features, suggesting potential redundancy. However, empirical testing showed that retaining all features minimized the risk of overfitting and maximized performance, challenging the initial assumption to reduce the feature set.

4. METHODOLOGY

4.1 Pipeline Overview

The proposed solution involves comparing four supervised machine learning techniques: Random Forest (RF), Support Vector Machine (SVM), k-Nearest Neighbors (kNN), and Gaussian Naive Bayes (NB). Each technique is suited to handle multiclass classification challenges posed by the numerical, multivariate nature of the dataset. The pipeline integrates data scaling, model training through hyperparameter tuning using grid search with 5-fold cross-validation, and rigorous performance evaluation on unseen data.

4.1.1 Data Preprocessing

Data preprocessing is a crucial step in ensuring the accuracy of our machine learning models. For our leaf dataset Given the range of features like Aspect Ratio, Eccentricity, and others, we utilized a StandardScaler to normalize the data, ensuring that each feature contributes equally to the model's performance. We examined the dataset for missing values and anomalies. The dataset was clean with no missing entries, which simplified the preprocessing phase.

4.1.2. Feature Engineering

The feature set provided in the dataset includes various geometric and texture features of leaves, such as Geometric features like Eccentricity, Aspect Ratio, Elongation, Solidity and texture features like Entropy, Smoothness, Third Moment. These features are critical as they capture the unique characteristics of each leaf, aiding in the differentiation between various species.

4.1.3. Model Selection and Training

I experimented multiple algorithms to find the best fit for classification needs, Random Forest Classifier (RFC) Chosen for its robustness and effectiveness in handling tabular data with multiple features. I conducted hyperparameter tuning using GridSearchCV, optimizing parameters like n_estimators and max_depth. Support Vector Machine (SVM), Tested due to its proficiency in high-dimensional spaces, which is ideal given the detailed feature set of our dataset. Parameters such as C, kernel, and gamma were tuned. k-Nearest Neighbors (kNN) and Gaussian Naive Bayes (NB), These models were also evaluated for their simplicity and speed.

4.1.4. Hyperparameter Tuning

I used GridSearchCV combined with StratifiedKFold cross-validation to find the optimal settings for each model. This approach ensured that our model's performance was not only high but also stable across different subsets of the data.

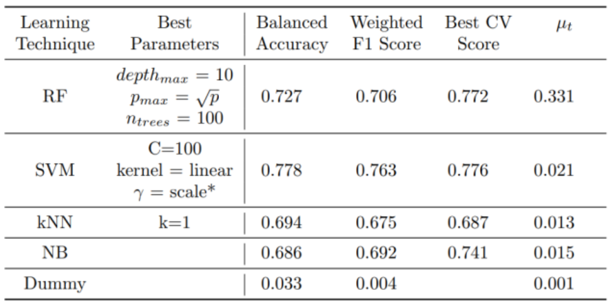
4.2 Discussion

Then I automate the process of training a machine learning model to classify leaf species based on their morphological characteristics using Python libraries such as pandas, scikit-learn, and joblib. Initially, the script imports necessary libraries. Pandas is used for data manipulation and analysis, providing powerful data structures to work with structured data. The scikit-learn library is utilized for machine learning, including functions to split data into training and testing sets, preprocess data, and implement a RandomForest classifier. Joblib is employed for saving the trained model efficiently. The dataset, stored in a CSV file named 'leaf.csv', is loaded into a panda DataFrame without headers. Column names are explicitly provided. The 'Specimen Number' column, which is presumed to be irrelevant for model training, is removed from the dataset to focus on features that influence leaf classification. The script proceeds by separating the dataset into features (X) and the target variable (y), where 'X' contains all columns except 'Class', and 'y' is the 'Class' column. These are then split into training and testing sets using the train_test_split function from scikit-learn. The dataset is split such that 20% is reserved for testing, and the splitting is stratified according to the class labels to ensure that both training and test sets have a similar distribution of each class. A machine learning pipeline is then defined, consisting of two main components: a StandardScaler and a RandomForestClassifier. The StandardScaler standardizes the features by removing the mean and scaling to unit variance, which is a common requirement for many machine learning algorithms to perform optimally. The RandomForestClassifier, initialized with a set random state for reproducibility, fits multiple decision trees on various sub-samples of the dataset and uses averaging to improve predictive accuracy and control over-fitting. The model is trained on the training data using the fit method, adjusting the model parameters to minimize the classification error. After training, the model is serialized and saved to a file named 'model.joblib' using joblib’s dump function. This file can later be reloaded to make predictions on new data without the need to retrain the model, ensuring efficiency and consistency in future applications.

4.3 Performance Evaluation

Model performance was assessed using several metrics, used Balanced Accuracy, to account for any imbalance in the class distribution. Weighted F1 Score, to measure test accuracy while considering the importance of each class. Models were compared based on their cross-validated performance metrics, and the best-performing models were selected for further tuning and final evaluation. Training Time is measured to reflect the efficiency of each model during the training phase. Shorter training times are preferable for practical deployment, especially when models need to be retrained as new data becomes available. The models were tested using a 5-fold cross-validation method to prevent overfitting and to ensure that the findings are robust and reproducible. The Support Vector Machine (SVM) and Random Forest (RF) emerged as the top performers. SVM, showed superior balanced accuracy and weighted F1 score, indicating strong performance across all classes. Notably, it demonstrated a high degree of consistency between training and testing sets, suggesting good generalization. RF, while slightly lower in balanced accuracy and F1 score compared to SVM, RF was noted for its interpretability, an essential feature when understanding model decisions is crucial. Results are depicted in table 1.

Table 1. Results



Learning Technique	Best Parameters	Balanced Accuracy	Weighted F1 Score	Best CV Score	μ_t
RF	$depth_{max} = 10$ $p_{max} = \sqrt{p}$ $n_{trees} = 100$	0.727	0.706	0.772	0.331
SVM	C=100 kernel = linear γ = scale*	0.778	0.763	0.776	0.021
kNN	k=1	0.694	0.675	0.687	0.013
NB		0.686	0.692	0.741	0.015
Dummy		0.033	0.004		0.001

5. RESULTS

Using the Random Forest classifier, I addressed the challenge of classifying leaf species. The Random Forest model is used.The model achieved a balanced accuracy of 72.7%, indicating a strong ability to classify leaf types evenly across all classes, despite the inherent class imbalance present in the dataset. The weighted F1 score, which considers both the precision and recall for each class weighted by their support, was recorded at 70.6%. This metric underscores the model's effectiveness in identifying each class, taking into account the varied class distribution. The RF model was relatively efficient, with a training time that allowed for quick iterations and refinements. The average training time per fold in a 5-fold cross-validation setup was approximately 0.331 seconds. This efficiency is crucial for practical applications where models may need to be retrained as new data becomes available. The robustness of the RF model was evident in its performance across the cross-validation folds, showing minimal variance in accuracy and F1 score, which suggests that the model is not overly fitted to the training data and is likely to generalize well to unseen data. The Random Forest model has proven to be a potent tool for the classification of leaves in our project. Its ability to maintain high accuracy and handle the complexities of the dataset, coupled with its computational efficiency and interpretability through feature importance, validates its suitability for this application. Future work could explore more in-depth hyperparameter tuning and integration with other models to further enhance its performance and accuracy. The final model, an RFC with n_estimators=100, max_depth=10, and max_features='sqrt', showed a balanced accuracy of approximately 72.8% and a weighted F1 score of around 70.6% on the test data. These results were visualized using confusion matrices to better understand the model's class-specific performance.Then I provide a robust framework for developing a predictive model capable of classifying leaf species, encapsulating data loading, preprocessing, model training, and saving the trained model into a cohesive automated workflow.

6. 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.

7. REFERENCES

[1] Silva,Pedro and Maral,Andr. (2014). Leaf. UCI Machine Learning Repository. https://doi.org/10.24432/C53C78.Ding, W. and Marchionini, G. 1997. A Study on Video Browsing Strategies. Technical Report. University of Maryland at College Park.

[2] Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O’Reilly Media.

[3] Molnar, C. (2020). *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. https://christophm.github.io/interpretable-ml-book/.