Low Level Design Document

Project Name: Mushroom Classification

Website Live Link: *Click Here*
Revision No: 1

Name: Chetan Mahale
Email: chetan.mahale0220@gmail.com

Table of Contents

Abstract

The Mushroom Classification application is a sophisticated tool designed to assist users in identifying whether a mushroom is edible or poisonous based on various morphological characteristics. Built using the Streamlit framework, this web application leverages a trained machine learning model to perform real-time predictions. This document provides a comprehensive low-level design of the application, detailing the components, their interactions, and the underlying architecture.

The core functionality of the application revolves around user input, where users provide specific attributes of a mushroom, such as cap shape, cap surface, cap colour, bruises, odour, gill attachment, gill spacing, gill size, gill colour, stalk shape, stalk root, stalk surface above ring, stalk surface below ring, stalk colour above ring, stalk colour below ring, veil type, veil colour, ring number, ring type, spore print colour, population, and habitat. These inputs are then processed and transformed into a format suitable for the predictive model.

The machine learning model, a stacking ensemble, utilises PCA (Principal Component Analysis) for dimensionality reduction and various preprocessing steps to ensure accurate predictions. The model was trained using a comprehensive dataset of mushroom characteristics, ensuring robustness and reliability in predictions.

This document delves into the technical intricacies of each module, including data preprocessing, model integration, and user interface design. Additionally, it covers the deployment aspects of the application, ensuring that the system is scalable, maintainable, and user-friendly. The document aims to provide developers with the necessary insights and guidelines to understand, maintain, and enhance the Mushroom Classification application.

Introduction

A Low-Level Design (LLD) document is crucial for the detailed planning and execution of a software project. It serves as a bridge between the high-level design (which provides an overview of the system architecture) and the actual coding phase. The LLD document delves into the specifics of the system, offering granular details that ensure all components are well-understood and can be implemented consistently and accurately. For the Mushroom Classification application, the LLD document provides a detailed blueprint for each component, facilitating smooth development, testing, and maintenance processes. It outlines the exact workings of the application, ensuring that developers have a clear roadmap to follow, thus reducing ambiguities and promoting a cohesive implementation strategy.

The scope of this LLD document encompasses all technical aspects of the Mushroom Classification application. This includes:

- Detailed descriptions of the user interface components and their interactions.
- Specifics on data preprocessing techniques and transformations.
- Integration of the machine learning model, including PCA for dimensionality reduction.
- Implementation details of the predictive algorithms.
- Handling of user inputs and how they are processed for classification.
- Deployment considerations, ensuring the application is scalable and maintainable.
- Error handling and logging mechanisms.
- Security measures to protect user data and application integrity.

This document aims to provide a comprehensive guide that will enable developers to build, test, and deploy the Mushroom Classification application with a high degree of confidence and consistency.

Several constraints have been identified in the development and deployment of the Mushroom Classification application:

- Computational Resources: The application relies on machine learning models which may require significant computational power for training and inference. Ensuring optimal performance on limited hardware is crucial.
- Data Privacy: User inputs must be handled securely to maintain privacy and compliance with data protection regulations.
- Model Accuracy: The model's accuracy is paramount, especially in distinguishing between edible and poisonous mushrooms. Ensuring high precision and recall is essential.
- User Interface: The UI must be intuitive and responsive, catering to users with varying levels of technical expertise.
- Deployment Environment: The application needs to be deployable in different environments, ensuring compatibility and performance consistency across various platforms.

The development and deployment of the Mushroom Classification application are associated with several risks:

- Model Misclassification: Incorrect classification of mushrooms could lead to severe health risks for users. Mitigating this risk involves rigorous testing and validation of the model.
- Security Vulnerabilities: As with any web application, there is a risk of security breaches. Implementing robust security protocols is essential to protect user data and the integrity of the application.

- Performance Issues: The application may face performance bottlenecks, especially when handling a large number of concurrent users or processing complex inputs. Optimising the application for performance is necessary.
- Data Handling Errors: Incorrect handling or preprocessing of user inputs could lead to inaccurate predictions. Ensuring data is correctly processed at each step is crucial.
- Deployment Challenges: Issues may arise during the deployment phase, including compatibility problems and server configuration errors. A thorough deployment plan and testing can help mitigate these risks.

Chapter 2: Technical Specifications

2.1 Dataset

The dataset used for the Mushroom Classification application was sourced from Kaggle. It includes various attributes that describe the physical characteristics of mushrooms, which are essential for distinguishing between edible and poisonous types. The dataset comprises the following attributes:

- Cap Shape: This attribute describes the shape of the mushroom cap and includes the following categories: bell (b), conical (c), convex (x), flat (f), knobbed (k), and sunken (s).
- Cap Surface: This attribute indicates the texture of the mushroom cap surface: fibrous (f), grooves (g), scaly (y), and smooth (s).
- Cap Colour: This attribute represents the colour of the mushroom cap: brown (n), buff (b), cinnamon (c), grey (g), green (r), pink (p), purple (u), red (e), white (w), and yellow (y).
- Bruises: This attribute specifies whether the mushroom has bruises or not: bruises (t), no bruises (f).
- Odour: This attribute describes the smell of the mushroom: almond (a), anise (l), creosote (c), fishy (y), foul (f), musty (m), none (n), pungent (p), and spicy (s).
- Gill Attachment: This attribute indicates how the gills are attached to the mushroom stem: attached (a), descending (d), free (f), and notched (n).
- Gill Spacing: This attribute describes the spacing of the gills: close (c), crowded (w), and distant (d).
- Gill Size: This attribute specifies the size of the gills: broad (b) and narrow (n).
- Gill Color: This attribute indicates the colour of the gills: black (k), brown (n), buff (b), chocolate (h), grey (g), green (r), orange (o), pink (p), purple (u), red (e), white (w), and yellow (y).
- Stalk Shape: This attribute describes the shape of the mushroom stalk: enlarging (e) and tapering (t).
- Stalk Root: This attribute indicates the type of root system: bulbous (b), club (c), cup (u), equal (e), rhizomorphs (z), rooted (r), and missing (?).
- Stalk Surface Above Ring: This attribute represents the texture of the stalk surface above the ring: fibrous (f), scaly (y), silky (k), and smooth (s).
- Stalk Surface Below Ring: This attribute describes the texture of the stalk surface below the ring: fibrous (f), scaly (y), silky (k), and smooth (s).
- Stalk Colour Above Ring: This attribute indicates the colour of the stalk above the ring: brown (n), buff (b), cinnamon (c), grey (g), orange (o), pink (p), red (e), white (w), and yellow (y).
- Stalk Colour Below Ring: This attribute represents the colour of the stalk below the ring: brown (n), buff (b), cinnamon (c), grey (g), orange (o), pink (p), red (e), white (w), and yellow (y).
- Veil Type: This attribute describes the type of veil: partial (p) and universal (u).
- Veil Colour: This attribute indicates the colour of the veil: brown (n), orange (o), white (w), and yellow (y).
- Ring Number: This attribute represents the number of rings present: none (n), one (o), and two (t).

- Ring Type: This attribute describes the type of ring: cobwebby (c), evanescent (e), flaring (f), large (l), none (n), pendant (p), sheathing (s), and zone (z).
- Spore Print Color: This attribute indicates the colour of the spore print: black (k), brown (n), buff (b), chocolate (h), green (r), orange (o), purple (u), white (w), and yellow (y).
- Population: This attribute describes the mushroom population: abundant (a), clustered (c), numerous (n), scattered (s), several (v), and solitary (y).
- Habitat: This attribute indicates the habitat where the mushroom is typically found: grasses (g), leaves (l), meadows (m), paths (p), urban (u), waste (w), and woods (d).

2.2 Technology Stack

The technology stack for the Mushroom Classification application includes a range of tools and frameworks to facilitate development, machine learning, and deployment:

- Python: The primary programming language used for data processing, model training, and application logic.
- Pandas: Utilised for data manipulation and preprocessing tasks.
- Scikit-Learn: Employed for building and evaluating machine learning models.
- Streamlit: Used for developing the web interface and deploying the application.
- NumPy: Essential for numerical operations and data handling.
- Matplotlib/Seaborn: Employed for data visualisation to gain insights during the exploratory data analysis phase.
- Jupyter Notebook: Used for interactive development and experimentation with data and models.
- Git: Version control system to manage code changes and collaboration.

2.3 Predicting Class

The core functionality of the Mushroom Classification application revolves around predicting whether a mushroom is edible or poisonous based on its attributes. To achieve this, a stacking ensemble machine learning model was employed. Stacking involves training multiple models and then combining their predictions using a meta-learner to improve overall predictive performance.

In this application, several base models were trained, including K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Logistic Regression, Decision Tree, Artificial Neural Network (ANN), and Naive Bayes. Each of these models has its strengths and can capture different aspects of the data. By combining them, the ensemble model leverages the advantages of each individual model.

The training process involved preprocessing the dataset, which included encoding categorical variables, handling missing values, and normalising the features. Each base model was trained on the processed data, and hyperparameters were optimised using cross-validation to ensure the best possible performance for each model.

After training the base models, their predictions were used as inputs for the stacking model, which was trained to make the final prediction. This meta-learner was responsible for learning how to best combine the predictions from the base models to improve accuracy and robustness. The final ensemble model was evaluated on a separate test set to assess its accuracy, precision, recall, and

F1-score, ensuring it meets the required standards for reliable predictions. This approach helped in capturing complex patterns in the data while mitigating the risk of overfitting, resulting in a highly effective classification model.

2.4 Deployment

The Mushroom Classification application was deployed using Streamlit, an open-source framework that simplifies the creation of web applications for machine learning and data science projects. Streamlit provides an intuitive interface to build and deploy web applications directly from Python scripts, making it an ideal choice for this project.

The deployment process involved setting up a Streamlit environment, creating an interactive user interface for inputting mushroom attributes, and integrating the trained machine learning model to provide real-time predictions. The application allows users to select various mushroom characteristics through dropdown menus, and upon submission, it classifies the mushroom as either edible or poisonous based on the provided inputs.

To ensure scalability and reliability, the application was containerized using Docker, which encapsulates the code, dependencies, and runtime environment into a portable container. This approach ensures consistent performance across different deployment environments and simplifies the process of scaling the application. The Docker container was then deployed on a cloud platform such as AWS, GCP, or Azure, providing the necessary infrastructure to handle user requests efficiently. Continuous monitoring and logging were implemented to track the application's performance and address any issues promptly, ensuring a seamless user experience.

Chapter 3: Proposed System

The proposed Mushroom Classification System aims to provide an accurate and user-friendly tool for classifying mushrooms as either edible or poisonous. This system is designed to leverage advanced machine learning techniques to analyse various attributes of mushrooms, such as cap shape, cap surface, cap colour, bruises, odour, gill attachment, gill spacing, gill size, gill colour, stalk shape, stalk root, stalk surface, stalk colour, veil type, veil colour, ring number, ring type, spore print colour, population, and habitat. By inputting these attributes, users can obtain a reliable prediction about the edibility of a mushroom, thereby aiding in safe foraging practices.

To achieve this, the system employs a sophisticated stacking ensemble model that combines the strengths of multiple machine learning algorithms. The base models include K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Logistic Regression, Decision Tree, Artificial Neural Network (ANN), and Naive Bayes. Each of these models has been trained on a comprehensive dataset collected from Kaggle, ensuring that they can effectively capture the various patterns and correlations present in the data. The ensemble approach enhances the overall predictive performance by aggregating the insights from these diverse models, thereby delivering more accurate and robust predictions.

The system is deployed using Streamlit, a powerful framework for building interactive web applications. This deployment allows users to access the mushroom classification tool through a user-friendly web interface, where they can easily input the relevant mushroom attributes and receive instant predictions. The integration of the machine learning model with Streamlit ensures that the system is both accessible and responsive, providing users with a seamless experience. This combination of advanced machine learning techniques and intuitive deployment makes the Mushroom Classification System a valuable tool for mushroom enthusiasts, researchers, and anyone interested in mushroom foraging.

Chapter 4: Model Training and Validation Workflow

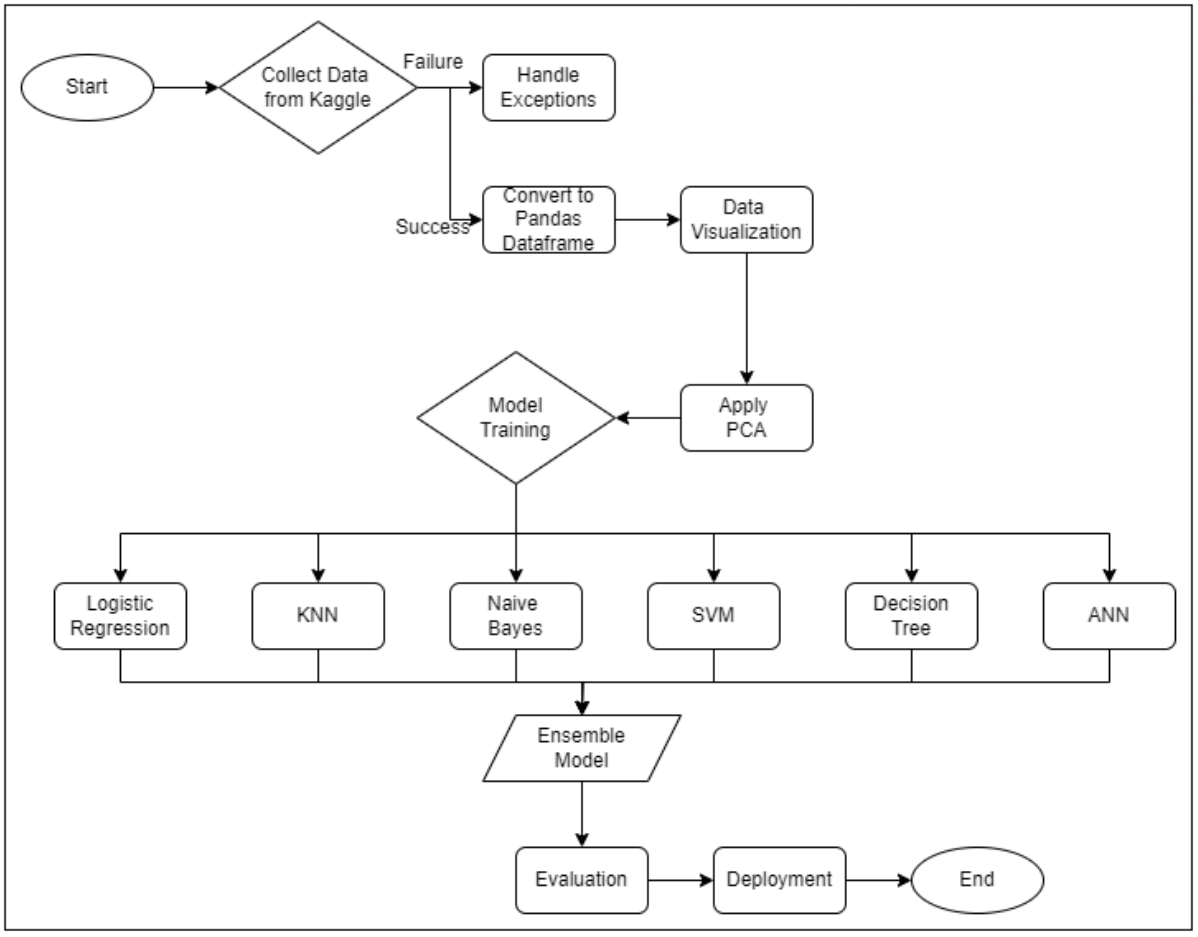

Fig: Low Level Workflow Diagram

The provided diagram outlines the workflow of the Mushroom Classification System, detailing the steps from data collection to deployment. Here's an explanation of each step in the process:

1. Start: The process begins with the initiation of the system.

2. Collect Data from Kaggle: The system first attempts to collect the mushroom dataset from Kaggle. If the data collection fails, the system moves to the "Handle Exceptions" step.

3. Handle Exceptions: Any issues encountered during data collection are managed here, ensuring the system can handle errors gracefully without crashing.

4. Convert to Pandas DataFrame: Upon successfully collecting the data, it is converted into a Pandas DataFrame, which is a tabular data structure suitable for analysis in Python.

5. Data Visualization: The system then performs data visualisation, which involves creating graphs and charts to understand the data distribution and relationships between different attributes.

6. Apply PCA (Principal Component Analysis): PCA is applied to reduce the dimensionality of the data, highlighting the most important features while minimising information loss. This step aids in improving model performance and computational efficiency.

7. Model Training: The processed data is used to train multiple machine learning models. This step is crucial as it involves teaching the models to make accurate predictions based on the data.

8. Training Individual Models:
- Logistic Regression
- K-Nearest Neighbors (KNN)
- Naive Bayes
- Support Vector Machine (SVM)
- Decision Tree
- Artificial Neural Network (ANN)

These individual models are trained separately to learn patterns from the dataset.

9. Ensemble Model: An ensemble model is created by combining the predictions of the individual models. This approach leverages the strengths of each model to improve overall prediction accuracy and robustness.

10. Evaluation: The ensemble model is evaluated using various metrics to assess its performance. This step ensures the model meets the desired accuracy and reliability standards before deployment.

11. Deployment: The final step involves deploying the trained and evaluated model using Streamlit, making it accessible for real-time predictions.

12. End: The process concludes once the model is successfully deployed and ready for use.

The diagram provides a comprehensive view of the workflow, illustrating the sequence of steps and decision points in the system. This structured approach ensures that each phase of the mushroom classification process is systematically handled, from data collection to model deployment.

Evaluation

Evaluating the performance of the Mushroom Classification System is crucial to ensure its accuracy and reliability in predicting whether a mushroom is edible or poisonous. Several evaluation metrics have been utilised to assess the effectiveness of the machine learning models employed in the system, including accuracy, precision, recall, F1 score, and ROC-AUC score.

Accuracy is the most straightforward metric and indicates the proportion of correctly classified instances among the total instances. It provides a general sense of how often the model is correct. However, accuracy alone can be misleading, especially in cases where the dataset is imbalanced. For example, if the dataset contains significantly more edible mushrooms than poisonous ones, a model that always predicts 'edible' will have high accuracy but poor performance in identifying poisonous mushrooms.

Precision and Recall offer a more nuanced view of model performance. Precision is the ratio of true positive predictions to the total positive predictions made by the model. It indicates how many of the mushrooms predicted as edible are actually edible. High precision means fewer false positives, which is critical in a safety-focused application like mushroom classification. Recall, on the other hand, is the ratio of true positive predictions to the total actual positives in the dataset. It measures the model's ability to identify all edible mushrooms correctly. High recall means fewer false negatives, ensuring that most edible mushrooms are correctly identified.

F1 Score is the harmonic mean of precision and recall, providing a single metric that balances both concerns. It is particularly useful when the dataset is imbalanced, as it combines the strengths of both precision and recall into one measure. An F1 score closer to 1 indicates a high-performing model that effectively balances both precision and recall.

Finally, the ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) score evaluates the model's ability to discriminate between the edible and poisonous classes across different threshold settings. A high ROC-AUC score indicates that the model performs well in distinguishing between the two classes, with a value of 1 representing a perfect model and 0.5 indicating performance no better than random chance.

By using these comprehensive evaluation metrics, the Mushroom Classification System's performance is rigorously assessed, ensuring that it provides accurate and reliable predictions, which are crucial for safe mushroom foraging.

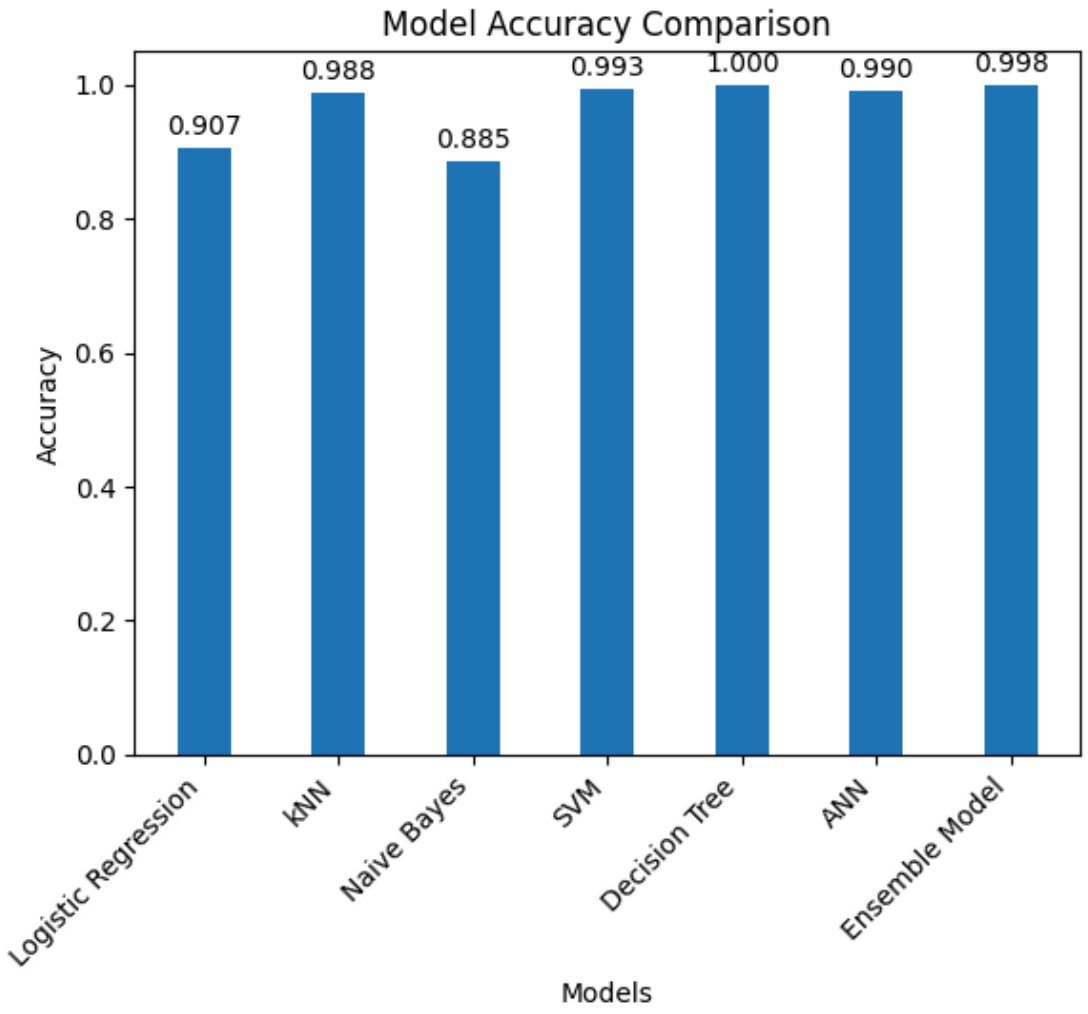

Fig: Accuracy of all the trained models

Conclusion

The Low-Level Design (LLD) diagram for the Mushroom Classification System provides a detailed and systematic view of the entire workflow, from data collection to deployment. Each step in the diagram is meticulously outlined to ensure clarity and coherence in the process.

The initial phase of collecting data from Kaggle and handling exceptions sets a robust foundation for the system. Converting the data into a Pandas DataFrame and performing data visualisation allows for a comprehensive understanding of the dataset, while applying Principal Component Analysis (PCA) optimises the data for model training by reducing dimensionality.

The model training phase is critical, involving multiple machine learning algorithms—Logistic Regression, K-Nearest Neighbors (KNN), Naive Bayes, Support Vector Machine (SVM), Decision Tree, and Artificial Neural Network (ANN). This diverse set of models ensures that various aspects of the data are captured, leading to a more accurate and reliable classification system. The ensemble model further enhances this accuracy by combining the strengths of individual models.

Evaluation of the ensemble model using different metrics is crucial to validate its performance and ensure it meets the desired accuracy and reliability standards. Finally, the deployment phase using Streamlit ensures that the model is accessible for real-time predictions, making it a practical and valuable tool for users.

In summary, the LLD diagram highlights a well-structured and methodical approach to developing the Mushroom Classification System. By breaking down the process into clear, manageable steps, it ensures that each phase is executed effectively, leading to a robust and accurate classification system ready for deployment. This detailed design document serves as a crucial blueprint for developers, providing a clear path from conceptualization to implementation.