Low Level Design Document

Project Name: Mushroom Classification

Name: Venkata Phani Srikar Pillalamarri

E-mail: pvpsrikar@gmail.com

Table of Contents

Abstract

Based on a variety of morphological traits, the Mushroom Classification program is an advanced tool that helps users determine if a mushroom is edible or deadly. This web application, which was created using Flask, makes predictions in real time using a machine learning model that has been trained. This file offers a thorough low-level design of the program, describing all of the parts, how they work together, and the underlying architecture.

User-provided data on characteristics of a mushroom, including its cap shape, surface, color, bruises, odor, gill attachment, spacing, size, and color, as well as its stalk root, surface above and below rings, color above and below rings, veil type, color below rings, ring number, type, spore print color, population, and habitat, forms the basis of the application's core functionality. After processing, these inputs are changed into a format that the predictive mode may use.

To guarantee precise predictions, the Random Forest classifier machine learning model makes use of a number of preprocessing procedures. A large dataset of mushroom properties was used to train the model, guaranteeing forecast robustness and accuracy.

The technical details of every module, such as model integration, data preparation, and user interface design, are covered in detail in this article. It also addresses the application's deployment, guaranteeing the system's scalability, maintainability, and user-friendliness. The purpose of this paper is to provide developers the knowledge and best practices they need to comprehend, manage, and improve the Mushroom Classification application.

Chapter -1

Introduction

For the thorough planning and implementation of a software project, a Low-Level Design (LLD) document is essential. It acts as a link between the real coding process and the high-level design, which offers a summary of the system architecture. The LLD document dives deeply into the system's intricacies, providing fine-grained information to guarantee that every component is well understood and capable of being applied correctly and consistently. The LLD document offers a thorough design for every part of the Mushroom Classification application, making the development, testing, and maintenance procedures easier. It describes the application's precise functionality in detail, giving developers a clear path to follow, eliminating confusion and encouraging a well-thought-out implementation plan.

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, specifically a decision tree classifier.
- 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.

During the development and implementation of the Mushroom Classification application, several challenges have been identified:

- **Processing Capabilities**: The machine learning models employed by the application may require significant processing power for both inference and training. Ensuring optimal performance on limited hardware is essential.
- **Data Privacy**: User inputs must be handled securely to protect privacy and comply with data protection regulations.
- **Model Accuracy**: Ensuring high accuracy, especially in distinguishing between poisonous and edible mushrooms, is critical. High recall and precision are necessary to avoid misclassification.
- **User Interface**: The UI must be responsive and intuitive, catering to users with varying levels of technical expertise.
- **Deployment Contexts**: The application should be deployable in diverse environments to ensure consistent performance and compatibility across multiple platforms.

Several risks are associated with the development and implementation of the Mushroom Classification application:

- **Model Misclassification**: Incorrect classification of mushrooms can pose serious health risks to users. Comprehensive testing and validation of the model are imperative to mitigate this risk.
- **Potential Security Threats**: Like any web application, there is a potential for security breaches. Robust security measures must be implemented to protect user data and maintain application integrity.
- **Performance Issues:** The application may encounter performance bottlenecks, particularly when dealing with a high number of simultaneous users or complex input processing. Optimizing the application for performance is essential.
- **Data Handling Errors:** Improper handling or preprocessing of user inputs could result in inaccurate predictions. Ensuring correct data processing at each step is vital.
- **Deployment Challenges:** Problems may surface during the deployment phase, such as compatibility issues and server configuration errors. A comprehensive deployment plan and rigorous testing can help mitigate these risks.

Chapter - 2

Technical Specifications

2.1 Dataset:

The Kaggle platform provided the dataset used in the application for Mushroom Classification. It contains a number of traits that characterize the physical qualities of mushrooms, which are crucial for differentiating between toxic and edible varieties. The following characteristics are included in the dataset:

- 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 Colour: 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 Colour: 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.
- 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.

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 decision tree classifier machine learning model was employed. This model is chosen for its interpretability and effectiveness in handling categorical data.

Data Collection

A comprehensive dataset of mushroom characteristics is utilized for training the machine learning model. This dataset includes features such as cap shape, cap surface, cap color, gill attachment, gill spacing, gill size, and several other attributes that describe the physical properties of mushrooms.

Data Preprocessing

The preprocessing of the dataset involves several critical steps to ensure it is in the right format for the algorithm:

- **Data Cleaning**: Any missing or inconsistent data is handled appropriately to ensure a clean dataset.

- **Encoding Categorical Variables**: Since the mushroom dataset contains categorical features, these are encoded into numerical values using techniques such as one-hot encoding.
- **Feature Scaling**: Although not always necessary for decision tree algorithms, feature scaling can be applied to normalize the data.

Model Training

The decision tree classifier is trained using the following steps:

- **Splitting the Data**: The dataset is divided into training and testing sets to evaluate the model's performance.
- **Training the Model**: The decision tree algorithm is trained on the training set, learning the relationships between the features and the target variable (edible or poisonous).
- **Hyperparameter Tuning**: Various hyperparameters of the decision tree, such as maximum depth and minimum samples per leaf, are tuned to optimize the model's performance.

Prediction

Once the model is trained, it can be used to predict the edibility of mushrooms based on new input data:

- **User Input**: Users input the characteristics of a mushroom through the application’s user interface.
- **Data Processing**: The input data is pre-processed in the same way as the training data to ensure consistency.
- **Model Inference**: The processed input data is fed into the trained decision tree classifier, which outputs a prediction indicating whether the mushroom is edible or poisonous.

Model Evaluation

To ensure the reliability and robustness of the model, several evaluation metrics are used:

- **Accuracy**: Measures the overall correctness of the model's predictions.
- **Precision and Recall**: Evaluates the model's performance, particularly in correctly identifying poisonous mushrooms.
- **Confusion Matrix**: Provides insight into the number of true positive, true negative, false positive, and false negative predictions.

2.4 Deployment:

The Mushroom Classification application was deployed using Flask, a lightweight web framework for Python that facilitates the creation of web applications. Flask is well-suited for this project due to its simplicity and flexibility in integrating with machine learning models and handling web requests.

Deployment Process

1. **Setting Up the Flask Environment**:
    - A Flask environment was established by setting up the necessary dependencies and configuring the application server. This involved installing Flask and any additional libraries required for handling HTTP requests and running the web server.
2. **Creating the User Interface**:
    - An interactive user interface was developed using Flask's templating engine, Jinja2. The interface includes forms and dropdown menus for users to input various mushroom characteristics. The web pages are designed to be user-friendly and responsive, allowing users to easily provide data for classification.
3. **Integrating the Machine Learning Model**:
    - The trained machine learning model, which classifies mushrooms as edible or poisonous, was integrated into the Flask application. When users submit their input through the web interface, the data is processed and fed into the model.
    - The model's predictions are then displayed on the web page in real-time, providing users with immediate feedback on the edibility of the mushroom based on the attributes they provided.
4. **Handling User Inputs**:
    - User inputs are collected from the web forms and processed using Flask's request handling mechanisms. The data is pre-processed and formatted to match the requirements of the machine learning model before making predictions.
5. **Testing and Deployment**:
    - Extensive testing was performed to ensure the application functions correctly across different scenarios and devices. After testing, the application was deployed to a web server, making it accessible to users via a web browser.

By using Flask, the Mushroom Classification application provides a streamlined and efficient platform for users to interact with the machine learning model, ensuring a smooth experience for classifying mushrooms based on their characteristics. To track the application's performance and quickly fix any problems, logging and continuous monitoring were put in place to guarantee a flawless user experience.

Chapter - 3

Proposed System

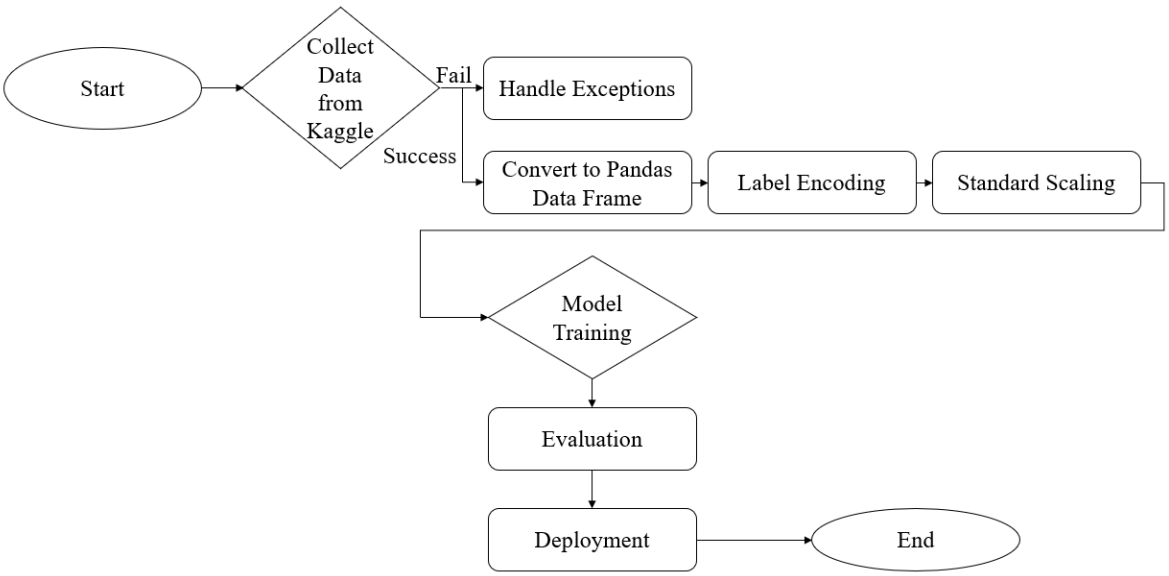
The proposed Mushroom Classification System aims to provide an accurate and user-friendly tool for classifying mushrooms as either edible or poisonous. This system leverages advanced machine learning techniques to analyse various attributes of mushrooms, such as cap shape, cap surface, cap colour, bruises,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

To achieve this, the system employs a sophisticated decision tree model that effectively captures the various patterns and correlations present in the data. This model was trained on a comprehensive dataset collected from Kaggle, ensuring its robustness and reliability. The decision tree classifier analyses the input attributes and provides accurate predictions regarding the edibility of the mushroom.

The system is deployed using Flask, a powerful web 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 Flask 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



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. Label Encoding: Label encoder converts the categorical variables to the numerical ones. Since, the dependent variables are of only two types, we have o and 1. But as the independent features are of multiple types, they get encoded with 0 up to the final features.
6. Standard Scaling: Consider cap Shape and Cap Surface. As there are types in the cap shape, label encoder encodes from 0 to 4 and the cap surface from 0 to 3. In order to bring them on same scale, we will apply the Standard Scaling on the data. Scaling means the mean will be 0 and Standard Deviation will be 1.
7. Model Training: The processed data will be trained on the Random Forest Classifier. This step is crucial as it involves teaching the models to make accurate predictions based on the data.
8. 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.

9. Deployment: The final step involves deploying the trained and evaluated model using Flask, making it accessible for real-time predictions.
10. 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.

Chapter - 5

Interpretation

However, it's important to consider the potential for overfitting, especially since the model achieved perfect scores. Further evaluation on a different validation set or using cross-validation techniques could provide additional insights into the model's generalizability.

To make sure the Mushroom Classification System is accurate and reliable in determining whether a mushroom is edible or toxic, it is essential to assess its performance. The efficacy of the machine learning models included into the system has been evaluated using a number of measures, such as accuracy, precision, recall, F1 score.

The easiest indicator to understand is accuracy, which shows what percentage of all cases were properly identified. It gives a broad idea of how often the model is accurate. But accuracy on its own might be deceptive, particularly if the dataset is unbalanced. A model that consistently predicts "edible" mushrooms would do poorly in detecting deadly mushrooms, for instance, if the dataset includes a disproportionately larger number of edible than poisonous mushrooms. A more complex picture of model performance is provided by precision and recall.

The ratio of actual positive predictions to all of the model's positive predictions is known as precision. It shows the proportion of really edible mushrooms among those projected to be edible. Reduced false positives are the result of high accuracy, which is crucial in a safety-oriented application like mushroom classification.

Conversely, recall is defined as the ratio of all real positives in the dataset to the genuine positive predictions. It assesses how well the model can detect every edible mushroom. Few false negatives result from high recall, which guarantees that the majority of edible mushrooms are properly recognized.

The F1 Score is a single statistic that balances accuracy and recall, calculated as the harmonic mean of the two. Given that it incorporates the benefits of both accuracy and recall into a single metric, it is especially helpful in cases when the dataset is unbalanced. An F1 score that approaches 1 denotes a high-performing model that successfully strikes a balance between recall and accuracy.

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.

The Random Forest Classifier has performed exceptionally well on the test dataset, achieving 100% accuracy, precision, recall, and F1-score. This suggests that the model is extremely effective at distinguishing between edible and poisonous mushrooms.

Chapter - 6

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 Label Encoding and Standard Scaling for model training.

The model training phase is critical, involving machine learning algorithm—Random Forest Classifier. This model is the extension of the decision tree classifier. When multiple Decision tree classifiers are combined to avoid the overfitting, it becomes the Random Forest Classifier. This model ensures that various aspects of the data are captured, leading to a more accurate and reliable classification system. The ensemble model enhances the accuracy by combining the strengths of individual Decision Trees.

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 Flask 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.