High Level Design (HLD)

Mushroom Classification Application

By

ANNAMALAI SUBRAMANI

Revision Number: 1.0
Last date of revision: 29/01/2024

Document Version Control

Date Issued	Version	Description Author
28/01/2024	2	Initial HLD - V1.0 Annamalai

Contents

Abstract

The Mushroom Classification project focuses on utilizing machine learning algorithms for accurately categorizing mushrooms into different classes based on their characteristics. This High-Level Design (HLD) document provides an in-depth overview of the project's architecture, design strategies, and key components.

The project initiates with the collection of mushroom data from diverse sources, encompassing features like cap color, odor, habitat, and more. Thorough data cleaning and preprocessing procedures are implemented to handle any discrepancies, missing values, or outliers, ensuring the quality of the dataset. The chosen classification algorithms include Logistic Regression, Decision Trees, Random Forest, Naive Bayes, and k-Nearest Neighbors (KNN), all of which contribute to achieving a commendable accuracy rate.

Model selection involves a systematic evaluation of each algorithm's performance, considering factors like precision, recall, and F1-score. Feature selection techniques and cross-validation are applied to enhance the models' overall performance and ensure robustness across different datasets.

The deployment phase encompasses hosting the trained models on a scalable platform, with exposed APIs for seamless integration into various systems. To ensure ongoing reliability, a monitoring system with predefined metrics and alerting mechanisms is implemented. A re-training strategy is employed to adapt the models to evolving data patterns, maintaining their effectiveness over time.

Security measures are implemented to safeguard sensitive information, and model explainability techniques are utilized for transparency in the classification process. Optionally, a user interface or dashboard may be designed to provide end-users with a user-friendly interaction for exploring mushroom classifications.

The scalability aspect is addressed by planning for potential increases in data volume or user requests, ensuring the system can handle growing demands. This HLD document serves as a comprehensive guide for the development, deployment, and maintenance of the Mushroom Classification ML Project, emphasizing accuracy, interpretability, and scalability in its design.

1 Introduction

1.1 Why this High-Level Design Document?

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradiction prior to coding and can be used as a reference manual for how the modules interact at a high level.

The HLD will:

- Present all of the design expects and define them in detail
- Describe the user interface being implemented
- Describe the hardware and software interfaces
- Describe the performance requirements
- Include designs which are the architecture of the project
- List and describe the non functional attributes like:
    - Security
    - Reliability
    - Maintainability
    - Portability
    - Reusability

1.2 Scope

The HLD documentation presents structure of the system, such as database architecture, application architecture (layers), application flow (Navigation), and technology architecture.

1.3 Definitions

HTML - HTML is the standard markup language for creating Web pages **CSS** - CSS is a style sheet language used for describing the presentation of HTML **Flask** - Flask is a micro web framework written in Python

Model - Machine Learning model

2 Project Overview

2.1 Project Objective

The goal of the project is to predict Mushroom is Poisonous or Edible based on historical data

2.2 Key Features

Accurate Classification: The project utilizes machine learning algorithms to precisely categorize mushrooms based on their features.

Model Transparency: Emphasis on interpretability provides clear insights into the decision-making process, aiding better understanding of classification results.

User-Friendly Integration: Seamless incorporation with a user interface ensures an intuitive and accessible experience for interacting with the mushroom classification system.

3 Data Collection and Preprocessing

3.1 Data Source

Mushroom Features: The data source for mushroom classification includes features such as cap color, odor, habitat, and other relevant characteristics.

Diverse Mushroom Data: The dataset encompasses information on various mushroom types, providing a comprehensive source for training and evaluating the classification models.

3.2 Data Cleaning and Preprocessing

Quality Enhancement: Rigorous data cleaning is applied to address missing values, outliers, and imbalances in the mushroom dataset, ensuring high data quality.

Variable Transformation: The preprocessing involves feature scaling and encoding of categorical variables, optimizing the dataset for effective training and evaluation of mushroom classification models.

4 Machine Learning Model

4.1 Algorithm Selection

The classification algorithms chosen for mushroom classification include Random Forest, Logistic Regression, Decision Tree, K-Nearest Neighbours, and other suitable models.

4.2 Feature Selection

Relevant features are identified through techniques such as feature importance, ensuring that the most informative attributes contribute to the mushroom classification models.

4.2 Model Training

Multiple models are trained using the selected algorithms, and their performance is evaluated comprehensively. Cross-validation and hyperparameter tuning are employed to enhance model robustness and effectiveness in mushroom classification.

5 Model Evaluation

5.1 Evaluation Metrics

The evaluation metrics for mushroom classification encompass accuracy, precision, recall, and F1-score.

5.2 Cross Validation

Employ k-fold cross-validation to assess model stability.

6 Model Deployment

6.1 Deployment Environment

Deploy the model on a cloud platform (e.g., AWS, Azure) for scalability.

6.2 API Integration

Expose model predictions through APIs for integration with other systems.

7 Monitoring and Maintenance

7.1 Monitoring Metrics

- Define metrics for model performance monitoring.
- Implement an alerting system for potential issues.

7.2 Re-Training Strategy

Plan periodic retraining to adapt to changing data patterns.

8 Security and Compliance

8.1 Data Privacy

Implement measures to ensure the privacy of sensitive financial data.

8.2 Model Explainability

Utilize techniques for model explainability and interpretability.

9 User Interface (Optional)

9.1 Dashboard

Design a user interface or dashboard for end-users to interact with predictions.

10 Scalability

10.1 Scalability Plan

Plan for scaling the system to handle increased data volume or user requests.

11 Documentation

11.1 Code Documentation

Document the machine learning codebase thoroughly.

11.2 User Guide

Create a user guide for end-users or developers interacting with the system.

12 Dependencies

12.1 Software Dependencies

List external libraries, frameworks, and tools used in the project.

13 Testing

13.1 Unit Testing

Specify unit tests for individual components.

13.2 Integration Testing

Plan for integration tests to ensure seamless collaboration among different modules.

14 Cost Estimates

14.1 Infrastructure Costs

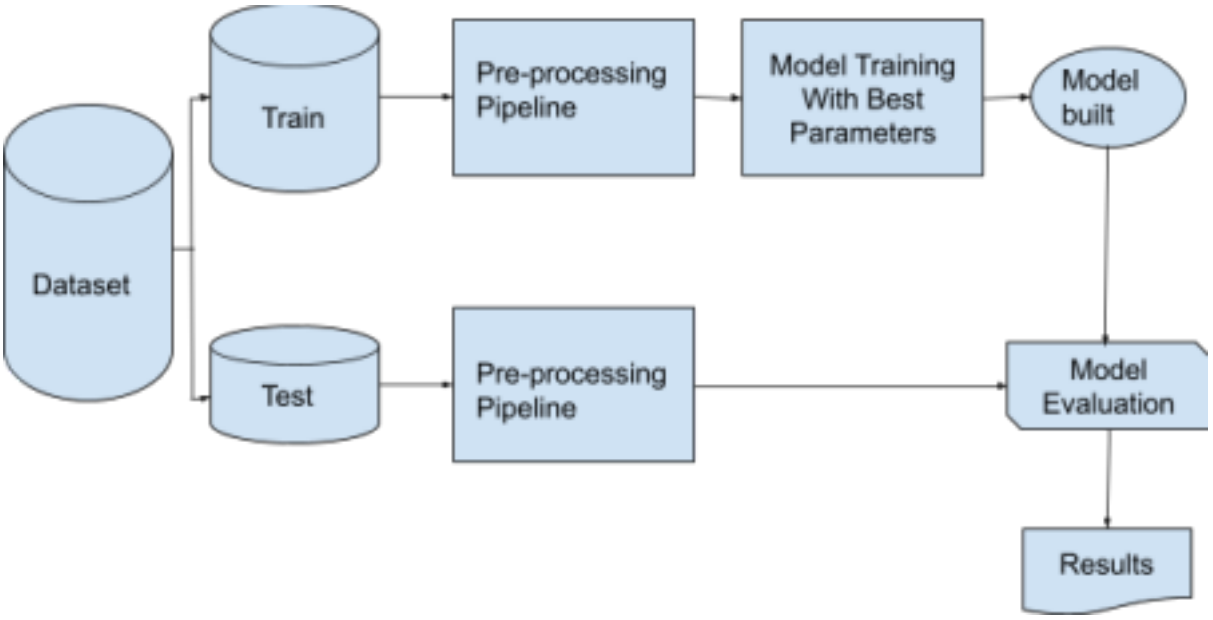
Estimate costs associated with infrastructure, cloud services, and third-party tools.

15 Tools and Technologies

- Visual Studio Code is used as an IDE.
- Machine Learning models are built using **scikit-learn**
- **API** is developed using Flask.
- UserInterface dashboard is developed using HTML and CSS
- Data is stored in the vector database ‘MongoDB’.
- GitHub is used as a version control system.
- ML-Flow is used to log experiments

16 Design Details

16.1 Process Flow



16.2 Event log

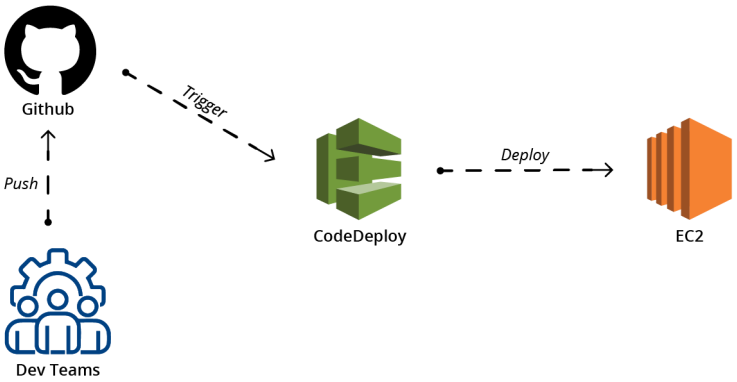
Should log each and every action and outcome throughout the project.

16.3 Error Handling

Should errors be encountered, an explanation will be displayed as to what went wrong.
An error will be defined as anything that falls outside the normal and intended usage.

17 Deployment

17.1 Deployment



18 Conclusion

In summary, the Mushroom Classification project adopts a systematic approach to accurately categorize mushrooms. Through thorough data cleaning, preprocessing, and feature selection, the project ensures the quality and relevance of the dataset. The inclusion of various classification algorithms, such as Random Forest, Logistic Regression, Decision Tree, and K-Nearest Neighbours, contributes to a comprehensive model training process. Emphasizing interpretability, the system allows users to understand the factors influencing classification decisions. The integration with a user-friendly interface enhances accessibility, while continuous monitoring and scalability planning ensure sustained performance. This High-Level Design document serves as a roadmap for the project's development, deployment, and maintenance, highlighting its commitment to accuracy, interpretability, and user-centric design.