1. LOW LEVEL DESIGN (LLD )

Low Level Design

# Food Recommendation System

|  |  |
| --- | --- |
| Written By | Yash Patle, Sayali Zambre |
| Document Version | 0.3 |
| Last Revised Date | 23 – May -2023 |

ii LOW LEVEL DESIGN (LLD)

**Document Control**

### Change Record:

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| 0.1 | 19 – May -  2021 | Sayali Zambre | Introduction & Architecture defined |
| 0.2 | 20 – May -  2021 | Yash Patle | Architecture & Architecture Description appended and updated |
| 0.3 | 21 – May -  2021 | Yash Patle | Unit Test Cases defined and appended |
| 0.4 | 21 – May -  2021 | Sayali Zambre | Document Content , Version Control and Unit Test Cases to be added |

**Reviews:**

1. LOW LEVEL DESIGN (LLD )

### Approval Status: Version Comments

|  |  |  |
| --- | --- | --- |
| **Review Date**  **21-04-2023** | **Reviewed By** | **Approved By** |

FOOD RECOMMENDATION LLD ii iii LOW LEVEL DESIGN (LLD)

# Contents

* 1. [Introduction 1](#_TOC_250004)
     1. [What is Low-Level design document? 1](#_TOC_250003)
     2. [Scope 1](#_TOC_250002)
  2. [Architecture 2](#_TOC_250001)
  3. [Architecture Description 3](#_TOC_250000)

### 3.1. Data Description 3 3.2.

**Web Scrapping** 3 **3.3.**

**Data Transformation**..............................................................................................................3 **3.4. Data**

**Insertion into Database**.................................................................................................3 **3.5. Export**

**Data from Database**....................................................................................................3 **3.6. Data**

**Pre-processing** ...............................................................................................................3 **3.7. Data**

### Logistic Regression 3 3.10.

### Model Building and training 4

### Data from User 4

### Data Validation 4

**3.13. Logistic Regression**.................................................................................................4 **3.15. Model**

### Call for Classification Result 4 3.16.

### Prediction 4 4. Unit

### Test Cases 5

1. LOW LEVEL DESIGN (LLD )

# Introduction

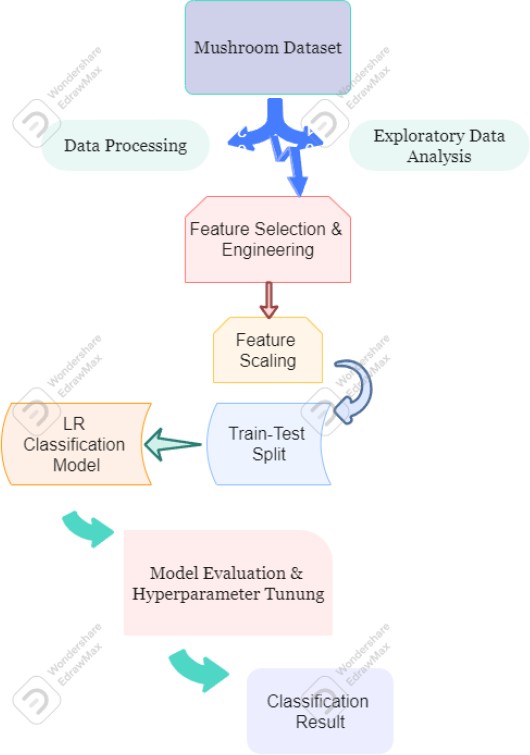
## What is Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Mushroom differentiation. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

## Scope

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code, and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and refined during data design work

# Architecture



1. LOW LEVEL DESIGN (LLD )

# Architecture Description

## Data Description

The Mushroom Data Set includes descriptions of hypothetical samples from 23 species of gilled mushrooms, labeled as definitely edible, definitely poisonous, or maybe edible but not recommended. The dataset has 8124 observations, 22 features, and a target variable indicating the mushroom's edibility. The data set provides information on the mushroom's cap color, odor, gill color, stalk shape, and other features to distinguish between different species. The dataset is commonly known as the UCI Mushroom Data Set.

## Web Scrapping

Web scraping can be used to extract additional information on the 23 species of gilled mushrooms, such as their geographic distribution, habitat preferences, and ecological roles, which can be used to develop more accurate models for predicting a mushroom's edibility based on its physical characteristics.

## Data Transformation

To transform the data for machine learning, we need to encode categorical variables using one-hot encoding, split the data into training and testing sets, scale numerical variables using techniques like standardization or normalization, select the most important features using

techniques like correlation analysis, feature importance scores, or principal component analysis, handle any missing values in the dataset by dropping rows or imputing missing values, and balance the dataset by oversampling the minority class or undersampling the majority class..

## Data Pre-processing

Data pre-processing is a necessary step for preparing the Mushroom Data Set for machine learning analysis. It includes cleaning, integrating, transforming, and reducing data to improve model performance and accuracy. This involves handling missing values, correcting inconsistent or erroneous data, combining data from different sources, modifying data to be suitable for analysis, and reducing the number of features or instances. Proper data pre-processing can lead to improved model accuracy and efficiency.

## Data Classifiacation

Data clustering groups similar data points together into clusters, and can be used to identify patterns and similarities among different mushroom species based on their features. Clustering algorithms group data points based on their distance or similarity to other points in the dataset, and the output can be analyzed to gain insights into the data, such as identifying which mushrooms are more likely to be poisonous or edible. Clustering is a useful tool for exploring and analyzing complex datasets, and can help identify patterns and insights that may not be apparent through manual inspection.

1. LOW LEVEL DESIGN (LLD )

## Model Building

Model building is the process of creating a machine learning model to predict whether a mushroom is poisonous or edible based on its features. This involves selecting an appropriate algorithm, splitting the data into training and testing sets, training the model on the training data, and evaluating its performance on the testing data. Hyperparameters may be tuned to optimize performance. The goal is to build an accurate model that predicts the target variable based on the features of the mushroom.

## Data from User

To build a solution to predict whether a mushroom is poisonous or edible, the data needs to be explored and cleaned, features must be engineered, different machine learning models need to be trained and evaluated, and the best performing model is selected and tested on new data.

## Data Validation

Data validation is an important step in the machine learning pipeline that involves checking the quality and consistency of the data. It is crucial to ensure that the data is accurate, complete, and representative of the problem being solved. This can involve techniques such as cross-validation, where the data is split into multiple training and testing sets to evaluate the performance of the models. Additionally, data validation can involve techniques such as outlier detection and removal, feature selection, and data augmentation. By performing data validation, we can ensure that the machine learning models are trained on high-quality data and are more likely to generalize well to new, unseen data

## 3.15. Model Call for Specific Cluster

Assign labels to the clusters: Inspect the resulting clusters and assign labels to each cluster based on their predominant mushroom type (poisonous or edible).

Split the data: Split the preprocessed mushroom data into training and testing sets. The training set will be used to train the model, and the testing set will be used to evaluate its performance.

Select a classification algorithm: Choose a classification algorithm that is appropriate for the size and complexity of the dataset, such as logistic regression, decision tree, or random forest.

Train the model: Train the classification algorithm on the training data, using the selected features and the labels assigned to the clusters.

Evaluate the model: Test the performance of the trained model on the testing data, using metrics such as accuracy, precision, recall, and F1 score.

Tune the model: Adjust the parameters of the classification algorithm or the feature selection process to optimize the performance of the model.

Make predictions: Use the trained model to make predictions on new, unseen mushroom data.

Monitor and update the model: Monitor the performance of the model over time,

and update it as needed to reflect changes in the mushroom data or to accuracy.

1. LOW LEVEL DESIGN (LLD ) 

## 3.17. Deployment

Choose a deployment platform: Choose a platform for deploying the model, such as a cloud-based service or a local server.

Set up the deployment environment: Set up the environment for deploying the model, including installing the

necessary libraries, configuring the server, and setting up any necessary databases or storage systems.

Export the model: Export the trained model to a format that can be used by the deployment platform, such as a serialized object or a container image.

Test the deployed model: Test the deployed model to ensure that it is functioning correctly and providing accurate predictions.

Integrate the model with other systems: Integrate the model with other systems as needed, such as web applications or mobile apps.

Monitor and maintain the deployed model: Continuously monitor the performance of the deployed model and make updates as needed to ensure that it remains accurate and effective.

1. LOW LEVEL DESIGN (LLD )

# Unit Test Cases

|  |  |  |
| --- | --- | --- |
| **Test Case Description** | **Pre-Requisite** | **Expected Result** |
| Test the model's prediction for a | Obtain a | The model should correctly predict |
| definitely edible mushroom. | sample of a | the mushroom as edible. |
|  | mushroom |  |
|  | known to be |  |
|  | definitely |  |
|  | edible. |  |
| Test the model's prediction for a | Obtain a sample | The model should predict the mushroom as poisonous since this category was merged with the toxic category. |
| maybe edible but not recommended | of a mushroom |
| mushroom. | known to be in |
|  | the maybe edible |
|  | but not |
|  | recommended |
|  | category. |
| Test the model's prediction for an unknown mushroom. | Obtain a sample of a mushroom that has not been identified or classified. | The model should not be able to provide a prediction since it has not been trained on this specific mushroom. |
| Test the model's prediction for a mushroom with similar features to a known edible mushroom. | Obtain a sample of a mushroom that has similar features to a known edible mushroom. | The model should predict the mushroom as edible since it has similar features to a known edible mushroom. |

|  |  |  |
| --- | --- | --- |
| **Test Case Description** | **Pre-Requisite** | **Expected Result** |
| Test the model's prediction for a | Obtain a | The model should correctly predict |
| definitely edible mushroom. | sample of a | the mushroom as edible. |
|  | mushroom |  |
|  | known to be |  |
|  | definitely |  |
|  | edible. |  |
| Test the model's prediction for a | Obtain a sample | The model should predict the |
| mushroom with similar features to | of a mushroom | mushroom as poisonous since |
| a known poisonous mushroom. | that has similar | it has similar features to a |
|  | features to a | known poisonous mushroom. |
|  | known |  |
|  | poisonous |  |
|  | mushroom. |  |

Page | 5