# LOW LEVEL DESIGN DOCUMENT(LLD)

MUSHROOM CLASSIFICATION

MUSHROOM CLASSIFICATION

Revision No – 1.3

Last Date of Revision – 14/01/2020

Author: Akshay Pandurang Paunikar

## Document Version Control:

| Date Issued | Version | Description |
|---|---|---|
| 11/01/2023 | 1 | Basic Formatting |
| 13/01/2023 | 1.1 | Introduction |
| 14/01/2020 | 1.2 | Architecture |
| 14/01/2020 | 1.3 | Unit Test Cases |



## CONTENTS

## Abstract

Mushrooms have been consumed since earliest history. The word Mushroom is derived from the French word for Fungi and Mold. Now-a-days, Mushroom are popular valuable food because they are low in calories, carbohydrate, Fat, sodium and also cholesterol free. Besides this, Mushroom provides important nutrients, including selenium, potassium, riboflavin, niacin, Vitamin D, proteins and fiber. All together with a long history as food source. Mushroom are important for their healing capacity and properties in traditional medicine. It has reported beneficial effects for health and treatment of some disease. Many nutraceutical properties are described in Mushroom like cancer and antitumor attributes. Mushroom act as antibacterial, immune system enhancer and cholesterol lowering Agent. Additionally, they are important source of bio-active compounds. This work is a machine learning model that classifies mushrooms into 2 classes: Poisonous and Edible depending on the features of the mushroom. During this machine learning implementation, we are going to see which features are important to predict whether a mushroom is poisonous or edible.

LOW LEVEL DESIGN(LLD) iNeuron

# 1. Introduction

## 1.1 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 Food Recommendation System. 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.

## 1.2 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 then refined during data design work.

# 2. Architecture

# 3. Architecture Description

This project is designed to make an interface for the user to predict whether a mushroom is poisonous or not.

## 3.1 Data Collection

The data for this project is collected from the Kaggle Dataset, the URL for the dataset is given below:

Dataset: https://www.kaggle.com/datasets/uciml/mushroom-classification.

## 3.2 Data Description

This dataset includes descriptions of hypothetical samples corresponding to 23 species of gilled

mushrooms in the Agaricus and Lepiota Family Mushroom drawn from The Audubon Society Field Guide to North American Mushrooms (1981). Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one.

### 3.3 Exploratory Data Analysis

The dataset is a csv file. There are 8124 rows and 23 columns in this data. All the columns are of categorical type. There are two classes present in our target column which are 'p' - poisonous and 'e' - edible. Also, we have nearly equal counts for poisonous and edible classes in our data. Hence, we can say that our data is balanced.

### 3.4 Handling Missing Data

At first, we observed that there no missing/null values in the dataset. However, if you go through the data description (check the link) you will find that the missing values in one column is replaced with "?". There are 2480 missing values in 'stalk-root' column. First, we will replace these values with np.nan so that we can handle missing data. we will impute the missing values in 'stalk-root' column using sklearn SimpleImputer with strategy='most_frequent'.

### 3.5 Data Visualization

For data visualization, we have used only those columns which we found most relevant to the target column in the feature selection stage. We analyzed various count plots and gathered as much insights as we can about the target column.



### 3.6 Data Preprocessing

In this step, first we have dropped the column 'veil-type' as it has only one value throughout the data. So, it won't give us much information regarding the class of the mushroom. Next, we mapped our target column to 0 (poisonous) & 1 (edible) values. We used Label Encoder to convert categorical values to numerical then we scaled our data to bring them to same class.

### 3.7 Feature Selection

After splitting the data into train and test set, we used SelectKBest method with score_func=chi2 to find out which features are most relevant to target column and we found that there are 12 columns out of 21 which we needed for training our model.

### 3.8 Model Training & Evaluation

We used XGBClassifier as a model for model training it was very fast compared to the other models and it produced 100% accuracy on train data as well as on test data which is a very good for our project.

### 3.9 Model Deployment

We created a webpage using HTML and CSS. We created a Flask web app and first tested in on our local machine. Then we deployed our model using Heroku. We used different combination of input and predicted the output and the results were accurate. The app was working fine and there were no issues found.



## 4. Unit Test Cases

| Test Case Description | Pre - Requisites | Expected Results |
|---|---|---|
| Verify whether the Webpage is accessible to the User or not. | Webpage URL should be defined. | Webpage should be accessible to the User. |

| | | |
|---|---|---|
| Verify whether the webpage loads completely for the User or not. | 1. Webpage URL is accessible.<br>2. Webpage is deployed. | The Webpage should be able to load completely for the User when it is accessed. |
| Verify whether the user is able to select data in input fields or not. | 1. Webpage URL is accessible.<br>2. Webpage is deployed. 3. Webpage input fields are editable. | The User is able to select data in input fields. |
| Verify whether the user is able to submit details or not. | 1. Webpage URL is accessible.<br>2. Webpage is deployed. 3. Webpage input fields are editable. | The User is able to submit details to process. |
| Verify whether the user gets recommended results on submitting the details or not. | 1. Webpage URL is accessible.<br>2. Webpage is deployed. 3. Webpage input fields are editable. | The User gets recommended results on submitting the details |