**LOW LEVEL DESIGN**

**OF MUSHROOM CLASSIFIER PROJECT**

Revision Date: 19/7/2024

Contents:

1. Introduction

1.1: Goal of this Low-Level Design Document

2. Architecture

2.1: Architecture Diagram

2.2: [Data Description](https://docs.google.com/document/d/11ZsH8kpfHudKxEpP882RrH1Ay4Ax44Sf/edit#heading=h.tyjcwt)

2.3: Data Preprocessing and Data Cleaning

3. Model Building

3.1: OneHotEncoding

3.2: Machine learning Model

4. Building the UI

4.1: Homepage

4.2: Page for ML model and its prediction

5. Deployment

5.1: Deployment using Cloud Services

6. Testing

6.1: Testing

**1)Introduction**

1.1: Goal of this Low-Level Design Document

The main goal of this LLDD is to show how the internal logic of the Mushroom Classifier. It also shows class diagrams to illustrate the methods of classes within a system. These diagrams, along with program specifications, provide a detailed blueprint for each module. This blueprint enables programmers to translate the design directly into code.

**2)** **Architecture**

2.1: Architecture Diagram

A diagram of data flow

Description automatically generated

2.2: Data Description

This Dataset had been donated to UCI ML on the 27th of April 1987.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.

2.3: Data Preprocessing and Data Cleaning

For this step we used pandas library.

Firstly, we checked for any empty columns or any empty cells. It was found out that the dataset had no empty cells. Furthermore, we also found out that there were no duplicate rows too.

Next, we performed EDA (Exploratory Data Analysis) using matplotlib. We plotted a simple bar graph so we can see how much of categories we are dealing with.

**3)** **Model Building**

3.1: OneHotEncoding

In order to train our model, we cant just use categorical data letters to train the model as the algorithm cant numerically process strings. So we have to convert every category to a number. This can easily be done using the OneHotEncoding module in the scikit-learn library.

3.2: Machine learning Model

After OneHotEncoding, we can now train our data. We used Logistic regression since its best suited for Binary Classification (in this case p and e). We used 80% of the dataset to train the model. After training and testing, we got an accuracy of almost 99.8%.

**4)** **Building the UI**

4.1: Homepage

To improve user experience, it would become necessary to add a homepage. The goal of this homepage is to give instructions to the user to utilise the application to its full potential. Hence, we used Flask to create one. The homepage contains a brief introduction of the application and some important abbreviations. It also shows detailed instructions to use the application.

4.2: Page for ML model and its prediction

This page was also developed from Flask and contains a number of input fields to type in the respective description. It also contains a button to predict our output once all fields have filled up. We can then see the predicted output below the page with our input we provided.

**5) Deployment**

5.1) Deployment of project

After all of the steps above, we can successfully host our project in a cloud application like Azure or AWS.

**6) Testing**

|  |  |  |
| --- | --- | --- |
| **Test Case Description** | **Pre-Requisite** | **Expected Result** |
| Verify if URL works or not | Application URL should be defined | URL should be accessible to user |
| Verify if Applications loads up completely when URL is accessed | Deploy Application and accessible URL | The website should load up completely. |
| Verify if application works as expected with high accuracy and without any errors | Having description of mushroom with high accuracy. | The website should be able to predict if a mushroom is poisonous or not. |