Low Level Design

Mushroom Classifier

Written By	Dhinagaran s
Document Version	0.1 (Alpha)
Last Revised Date	26-10-2022

Document Control
Change Record:

Version	Date	Author	Comments
0.1	25-10-2022	Dhinagaran s	Nothing

Reviews:

Version	Date	Reviewer	Comments

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

Contents

1. Introduction

1.1. What is a 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 the Mushroom Classifier. 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 organisation may be defined during requirement analysis and then refined during data design work

2. Architecture:

3. Architecture Description

3.1. Data Description

Data is taken from the kaggle Competition - Mushroom Classification.

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, jor of unknown edibility and not recommended. This latter class was combined with the poisonous one.

3.2. Data Preprocessing

The categorical data is encoded with OneHotEncoder and LabelEncoder

3.3. Data Selection

Data Selection is done by executing hypothesis test on the DataFrame of the dataset

3.4. Model Building

Four algorithms are selected to find the best model, they are,

- RandomForestClassifier
- SVC
- LogisticRegression
- XGBClassifier

The models are evaluated using the accuracy score and cross validation score, to find the best model.

GridSearchCV is used for hyper-parameter tuning.

A ModelCreation object is created to train and store the evaluation values in an instance.

3.5. Data from User

Inputs are collected from users using dropdown functionality in the Web Application Interface, which contains the hypothetical data.

The hypothetical data are,

- Cap-Shape
- Cap-Surface
- Cap-color
- Bruises
- Odor
- Gill-Attachment
- Gill-Spacing
- Gill-Size
- Gill-Color
- Stalk-Shape
- Stalk-root

- Stalk-Surface-Above-Ring
- Stalk-Surface-Below-Ring
- Stalk-Color-Above-Ring
- Stalk-Color-Below-Ring
- Veil-Type
- Veil-Color
- Ring-Number
- Ring-Type
- Spore-Print-Color
- Population
- Habit

These data are preprocessed and then given as input to the model which was created and saved earlier.

3.6. Deployment

The model is deployed in the Heroku, which is an Cloud providing the Platform As A Service (PAAS)

4. Unit Test Cases

Test Case Description	Prerequisite	Expected Result
Verify whether the Application URL is accessible to the user	1. Application URL should be defined	Application URL should be accessible to the user
Verify whether the Application loads completely for the user when the URL is accessed	1. Application URL is accessible 2. Application is deployed	The Application should load completely for the user when the URL is accessed
Verify whether user is able to see input fields on Web Page	1. Application is accessible 2. Application is deployed	User should be able to see input fields on Web Page
Verify whether user is able to edit all input fields	1. Application is accessible 2. Application is deployed	User should be able to edit all input fields
Verify whether user gets Submit button to submit the inputs	1. Application is accessible 2. Application is deployed	User should get Submit button to submit the inputs
Verify whether user is presented with recommended results on clicking submit	1. Application is accessible 2. Application is deployed	User should be presented with recommended results on clicking submit
Verify whether the recommended results are in accordance to the selections user made	1. Application is accessible 2. Application is deployed	The recommended results should be in accordance to the selections user made
Verify whether Predicted Output modify as per the user inputs for the different mushroom data	1. Application is accessible 2. Application is deployed	Predicted Output should modify as per the user inputs for the different mushroom data
Verify whether the Web Page indicate details of the suggested inputs	1. Application is accessible 2. Application is deployed	The Web Page should indicate details of the suggested inputs