Mushroom Classification

Low Level Design Document

Revision Number: 0.0

Last date of revision: 26/06/2023

Contents

Document Version Control

Date Issued	Version	Description	Author
19/03/2023	1.0	Initial HLD – V1.0	Md Imran

1. Introduction

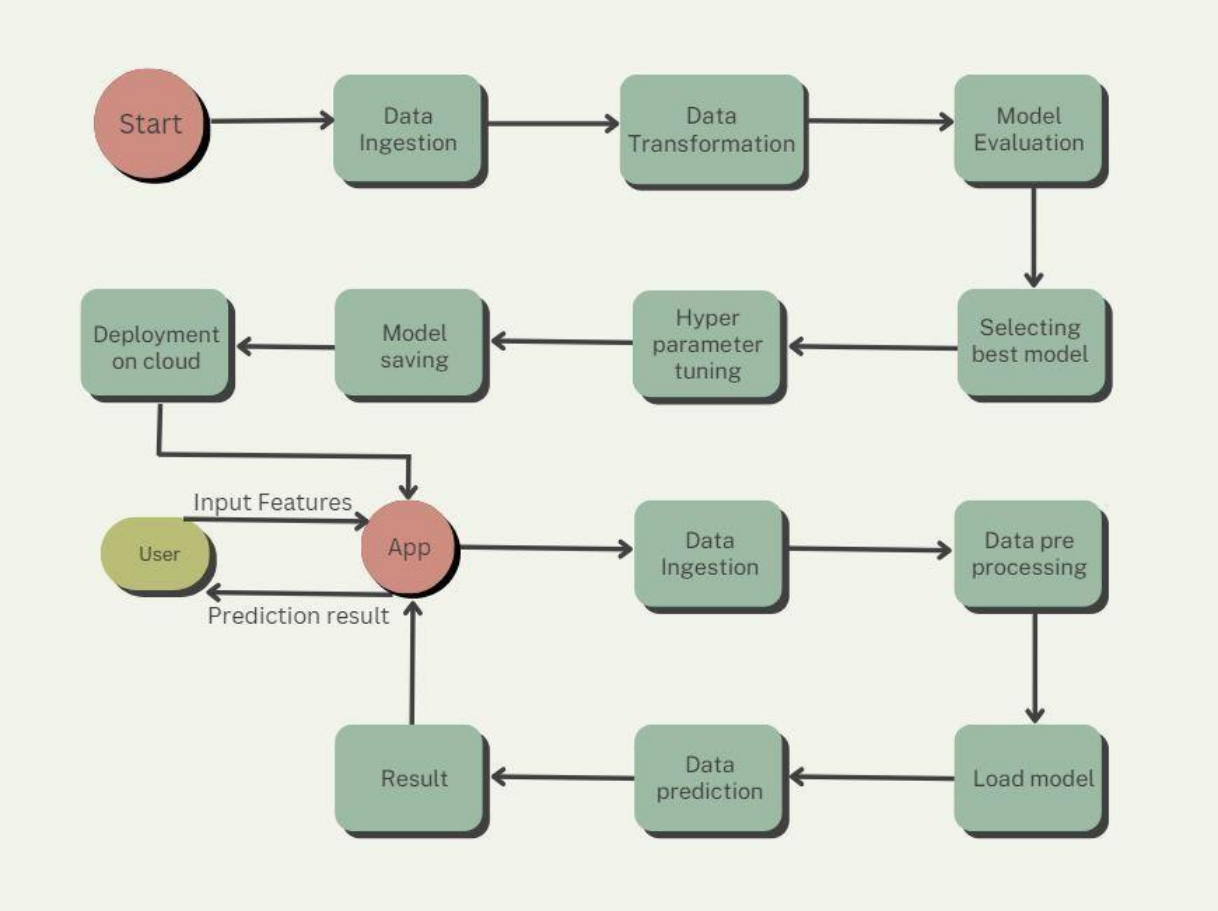
1.1 What is Low Level Design Document

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for 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 organization 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, or of unknown edibility and not recommended. This latter class was combined with the poisonous one.

3.2 Data Ingestion

Data is downloaded in csv format from Kaggle and is split in training and testing set and again stored in different csv.

3.3 Data Transformation

Data type of columns – All features of this data is in object format. They will be converted to numerical format through one hot encoding.

Null values in columns – This data does not contain null values but there are some missing values which are to be filled through imputation.

Name of columns - The name of the columns should be as per the columns name of the dataset.

Target column will be separated for training and testing data and preprocessing will be done on it.

Preprocessing object will be stored in pickle format for further use.

3.4 Model Evaluation

These models will be tested in the data using

1. XGBoostClassifier
2. GradientBoostClassifier
3. Random Forest

Best model will be selected according to accuracy and will be used train and predict data.

3.5 Deployment

Model will be deployed on AWS server.

3.6 Data from User

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

Input features 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

3.7 Prediction

Data will be entered by user through a HTML/CSS front end application and Flask at back end.

Data will then be pre-processed through transformation pickle and it will then be used to make predictions using pickle file for model.

The prediction results will then be showed to user on user interface.

Unit Test Cases

Test Case Description	Expectation	Outcome
Verify whether the Application URL is accessible to the user	1. Application URL should be defined	Application URL is 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 is correctly deployed.
Verify whether user is able to see input fields on Web Page	1. Application is accessible 2. Application is deployed	User is 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 is 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 gets 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 is presented with recommended results on clicking submit button.
Verify whether the recommended results are in accordance to the selections user made	1. Application is accessible 2. Application is deployed	The recommended results are 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 modifies as per the user inputs for the different mushroom data