Low Level Design (LLD)

MUSHROOM CLASSIFICATION

CHHOTTU DA MODAK

Low Level Design (LLD)

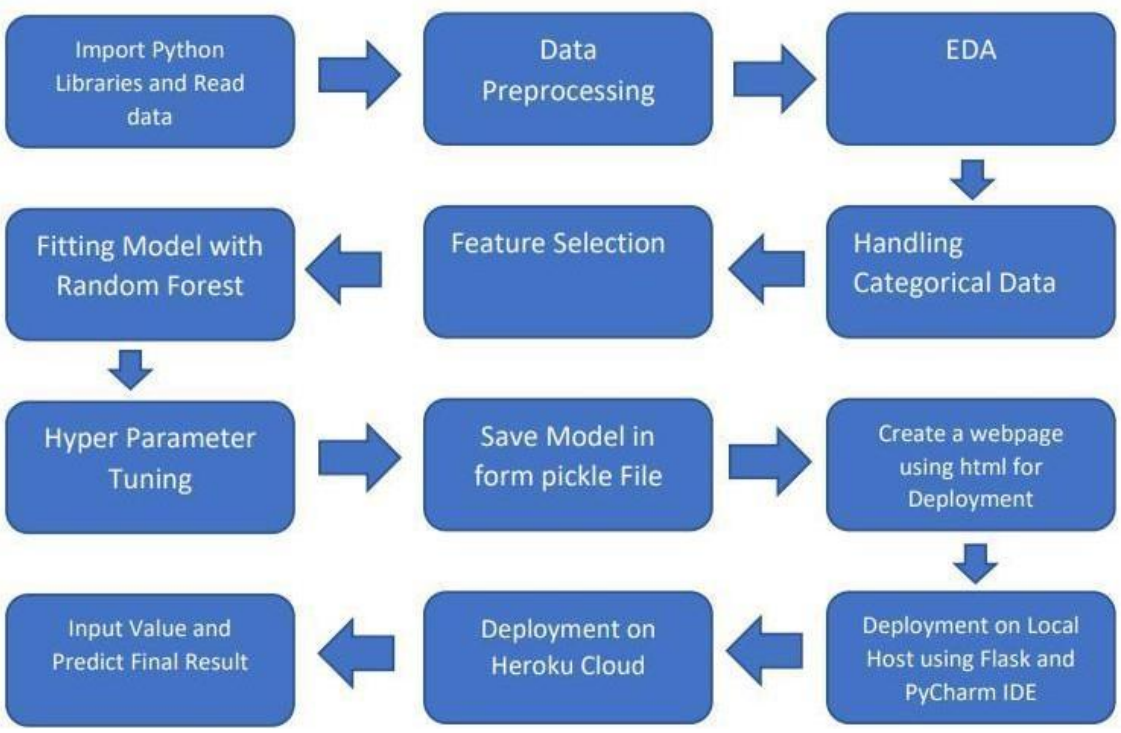
Low Level Design (LLD)

❖ Introduction

- **Why this Low-Level Design Document?**

The primary objective of this Low-Level Design (LLD) documentation is to present the necessary project details and provide a comprehensive overview of the machine learning model, including the associated code. It also offers an in-depth explanation of the end-to-end project design.

1 Architecture



2. Architecture Design

This project involves developing a user interface that allows users to estimate the approximate cost of flight tickets. Additionally, to gain real-time project experience, we are importing the collected data into our own database and initiating the project from the ground up.

2.1. Data Gathering

The data for the current project is being gathered from Kaggle dataset, the link to the data is:
https://www.kaggle.com/datasets/uciml/mushroom-classification

2.2. Tool Used

- **Python 3.7** is utilized for model development, leveraging frameworks such as **NumPy, Pandas, Scikit-learn**, and other relevant modules.
- **PyCharm** serves as the Integrated Development Environment (IDE) for coding.
- Data visualizations are implemented using **Seaborn** and components from **Matplotlib**.
- **Prophetess** database is employed for data collection, integrated with version control.
- **Netlify** is used for deployment purposes.

2.3 Data Description

Mushroom dataset is the biggest publicly available dataset. This dataset contains 8314 rows and 23 columns.

2.4 Import Data into Database

- Created associate api for the transfer of the info into the Cassandra info, steps performed are:
- Connection is created with the info.
- Created a info with name Mushroom.
- Cqlsh command is written for making the info table with needed parameters.
- And finally, a cqlsh command is written for uploading the knowledge set into data table by bulk insertion.

2.5 Export data from database

In the above created api, the download url is also being created, which downloads the data into a csv file format.

2.6 Data pre-processing

Steps performed in pre-processing are:

- First the data types are being checked and found only the price column is of type integer.
- Checked for null values as there are few null values, those rows are dropped.
- Converted all the required column into the date time format.
- Performed one-hot encoding for the required columns.
- Scaling is performed for required data.

- And, the data is ready for passing to the machine learning algorithm.

2.7 Modelling

The pre-processed information is then envisioned and everywhere the specified insights are being drawn. Though from the drawn insights, the info

is at random unfold however still modelling is performed with completely different machine learning algorithms to form positive we tend to cowl all the chances and eventually, for sure **Gradient Boosting classifier** performed well and any hyperparameter calibration is finished to extend the model's accuracy.

2.8 UI Integration

Both CSS and HTML files are being created and are being integrated with the created machine learning model. All the required files are then integrated to the app.py file and tested locally.Note CSS and HTML is not done by me.

2.3 Data from User

The data from the user is retrieved from the created HTML web page.

2.4 Data Validation

The data provided by the user is then being processed by application.py file and validated. The validated data is then sent for the prediction.

2.11 Rendering Result

The data sent for the prediction is then rendered to the web page.

3. Deployment

The tested model is then deployed on AWS beanstalk . So, users can access the project from any internet device

4 Unit Test Case:

Test Case Description	Pre-Requisites	Expected Results
Verify whether the User Interface URL is accessible to the user.	1. User Interface URL should be defined.	User Interface URL should be accessible to the user.
Verify whether the User Interface loads completely for the user when the URL is accessed.	1. User Interface URL is accessible. 2. User Interface is deployed.	The User Interface should load completely for the user when the URL is accessed.
Verify whether user is able to edit all input fields.	1. User Interface is accessible.	User should be able to edit all input fields.