Low Level Design (LLD)

FLIGHT FARE PREDICTION

Revision Number: 2.0

Last Date of Revision: 01/06/2023

Document Version Control

Date Issued	Version	Description	Author
31/05/2023	1.0	Introduction, Architecture	Debasish Bhagawati
1/06/2023	2.0	Architecture Description, Unit Test Cases	Debasish Bhagawati

Low Level Design (LLD)

Contents

Low Level Design (LLD)

1. Introduction

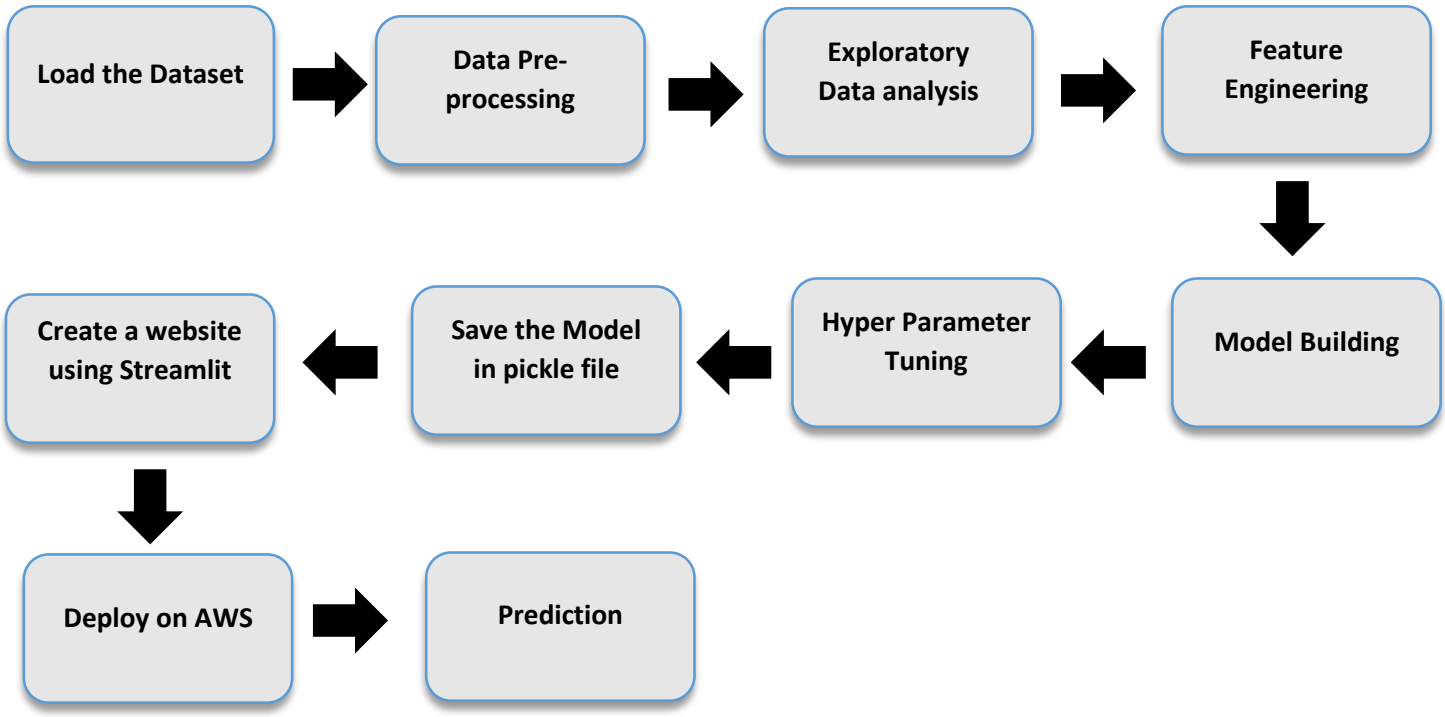
1.1 What is Low Level Design Document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Flight Fare Prediction. 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 Collection

The data is collected from the kaggle website. link : Dataset

3.2. Data Description

We have two dataset one for training and other one is for testing. Training dataset consist of 10683 rows and 11 columns. 11 columns are –

- **airline** – Airline companies name.
- **Date_of_Journey** – This column will inform us of the date the passenger's travel will begin.
- **Source** - This column indicates the names of the location from which the passenger's journey will begin.
- **Destination** – Destination city of the passenger.
- **Route** - names of the location from where the customer's journey would begin.
- **Dep_Time** - The time when the plane leaves the gate of the airport (departure time)
- **Arrival_Time** - the time the airplane arrives at its gate
- **Duration** - The flight's endurance in hours..
- **Total_Stops** - The total number of breaks in the voyage.
- **Additional_Info** - It will indicate whether a meal is included with the journey or not.
- **Price** – Fare of that journey.

The test dataset consist of 2671 rows with 10 columns exactly same as the training data excluding the Price column.

3.3 Data Pre-processing and Feature Engineering

steps :

- Univariate Analysis of each columns.
- Handling the missing value.
- Convert all the desired column into the date-time format.
- Handling the categorical columns.

3.4 Model Building

We then split the training dataset into two parts using sklearn's train_test_split function and build different regression model using the training data and test it on the test data. Whichever model gives higher accuracy we will do hyperparameter tuning to enhance the performance of the model. In our case, Random forest is the model which gives the better accuracy.

3.5 Data from User

Here we collect user data from an HTML web page that has been created.

3.6. Data Validation

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

3.7 Prediction

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

3.8 Deployment

We will be deploying the model to AWS so that anyone can access.

4. Unit Test Cases

Test Case Description	Pre-Requisite	Expected Result
Verify whether the User Interface URL is accessible to the user.	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. 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 edit all input fields.	User Interface URL is accessible.	User should be able to edit an input field.
Verify whether user gets Predict button to make the prediction.	1. Application URL is accessible. 2. Application is deployed. 3. Input fields are fill up.	User should get the predicted price.