LOW LEVEL DOCUMENT

Flight Fare Prediction

Written by	**Sourav Dey**
Document Version	**1.0**
Last Revised Data	**08 December 2022**

1 Document Version Control:

Version	Date	Author	Comments
1.0	**08 December 2022**	**Sourav Dey**	

Table of Contents

1.Introduction

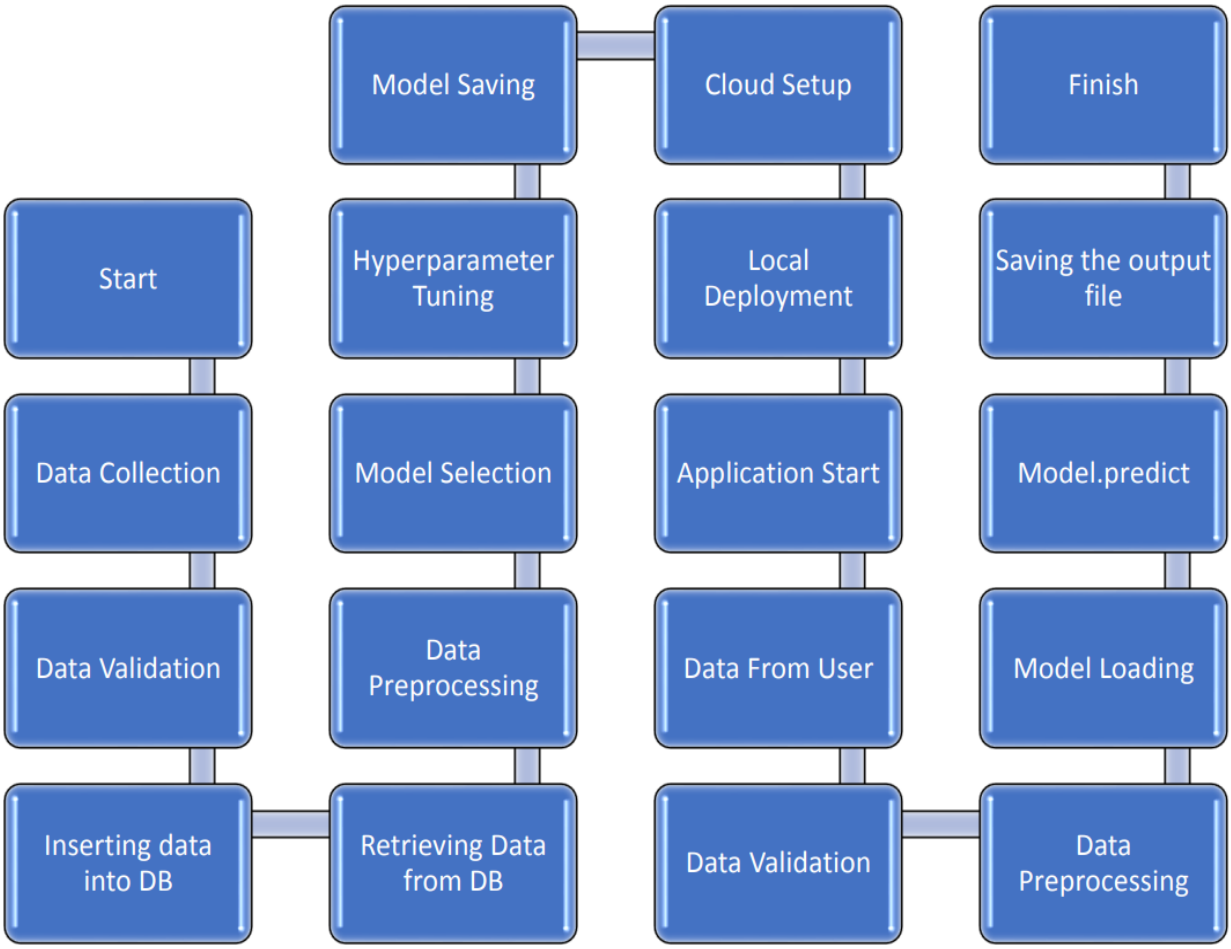
1.1 What is Low-Level design document?

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 estimation System. 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

We have 10k Dataset row columnar data includes the flight service, flight fare, number of stops, total number of duration, departure-arrival date and all. These is given in the Excel. These data is collected from the Kaggle which contains both the test data and train data.

3.2 Data Validation

In Data Validation part we basically check whether files inside the folder provide are having same columns or not with the data type of each column each by each file. If the file is found to matching the name of columns and dtype of the columns then we are supposed to copy the file from the folder.

3.3 Inserting into DB

We are going to insert the data into Database that is CassandraDB .For this we have used Cassandra driver Our Database name is ineuron and keyspace is Flights.

3.4 Retrieving Data From DB

Retrieving the data and storing into a file called as file.csv

3.5 Data Preprocessing

Data Preprocessing step is very necessary for model creation, We have mainly categorical data into our csv file so we need to convert that categorical data into numerical format.

3.6 Model Selection For model

Training we are just choosing XGBRegressor because it is performing best than other models When calculating mean_absolute_error XGB always had a very low mae.

3.7 Hyperparameter Tuning

We are doing hyperparameter tuning only for Random Forest Reggresor and because of tuning our model has a low mean_absolute_error.

3.8 Model Saving

Model Saving is the final step in the training part. We are saving the model in folder as model.pickle and I am using module pickle for saving the ML model.

3.9 Data from User

On Application Starting user will be interacting with a UI which is designed using HTML/CSS. Where in user user will input data and will get the predicton.

3.10 Data Validation

In Data Validation part we basically check whether files inside the folder provide are having same columns or not with the data type of each column each by each file.

3.11 Data Preprocessing

In data preprocessing I will be handling categorical columns plus some other columns as well as like Duration,Arrival_Time column.

3.12 Model Loading

Model loading is actually very simple we will be calling the model which we saved as modle. For loading the model I am using pickle library

3.13 Model.predict

After loading the model everything is very simple you just have to predict the preprocessed data.

10. Test Cases

Test cases are given below

Test Case Description	Pre-Requisite	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 Response time of url from backend model.	1. Application is accessible	The latency and accessibility of application is very faster we got in Heroku service.
Verify whether user is giving standard input.	1. Handeled test cases at backends.	User should be able to see successfully valid results.
Verify whether user is able to edit all input fields	1. Application is accessible	User should be able to edit all input fields
Verify whether user gets Custom File Predict, Default File Predict button to submit the inputs	1. Application is accessible	User should get both buttons to submit the inputs
Verify whether user is presented with recommended results on clicking submit	1. Application is accessible	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	The recommended results should be in accordance to the selections user made