**Low Level Design (LLD)**

[Flight Fare Prediction](https://github.com/ganeshss0/flight-fare-prediction)

Revision Number: 1.0

Last Date of Revision: 20/07/2024

[**Arshdeep Singh**](https://github.com/arshdeep-singh-0/flight-fare-prediction)

Contents

|  |  |
| --- | --- |
| * Abstract | 3 |
| 1. Introduction  * Why this LLD documentation? | 4  4 |
| 1. Architecture  * Architecture Design * Data Gathering * Tools Used * Data Description * Data pre-processing * Modelling * UI Integration * Data from User * Data Validation * Rendering the results * Deployment | 5  5  5  5  5  6  6  6  6  6  6  6  6 |
| 1. Unit Test Cases | 7 |

* **Abstract**

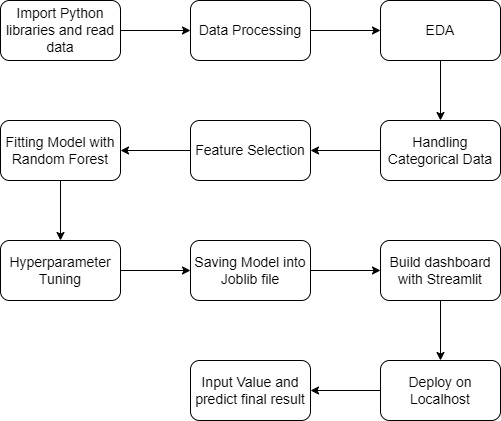
The recent global situations had a huge impact on the aviation sector due to many reasons. This impact has two category people, the first is business perspective and the second is the customers perspective. As safety is the major reason for such impact on the aviation sector, the governments around the world amended different rules to their respective airlines companies. These restrictions had made the availability of the flights and their attendee capacity less. Taking all these factors in consideration the cost of the flight tickets has increased and vary from one place to the other. Booking a flight ticket has split into two, one is the online and the other is the offline bookings. Both these have their respective criteria for cost of the ticket, one such example is the server load and the number of booking requests. In this machine learning implementation, we will see various factors that impact the cost of the flight ticket and predict the appropriate fare of the ticket.

1. **Introduction**

* **Why this Architecture Design documentation?**

The main objective of the Architecture design documentation is to provide the internal logic understanding of the flight fare prediction code. The Architecture design documentation is designed in such a way that the programmer can directly code after reading each module description in the documentation.

1. **Architecture**



**2.1. Architecture design**

This project is to create an interface for the user to know their approximate flight ticket price, in addition to this, in need of getting the real time project experience we are importing the gathered data into our own database and then start the project from the scratch.

**2.2. Data Gathering from main source**

The data for the current project is being gathered from Kaggle dataset, the link to the data is: [Indian Flight Fare Dataset (kaggle.com)](https://www.kaggle.com/datasets/dopodix/indian-flight-fare-dataset)

**2.3. Tool Used**

* Python 3.7 is employed because the programming language and frame works like NumPy, Pandas, Scikit-Learn and XGBoost alternative modules for building the model.
* Visual Studio Code is employed as IDE.
* For visualizations Seaborn and components of Matplotlib are getting used.
* Streamlit.io is employed for deployment

**2.4. Data description**

There are about 10k+ records of flight information such as “Airlines”, “Date of Journey”, “Source”, “Destination”, “Departure Time”, “Arrival Time”, “Duration”, “Total Stops”, “Additional Information” and “Price”.

**2.5. 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.
* The data is ready for passing to the machine learning algorithm.

**2.6. Modelling**

The pre-processed data is then visualized, and all the required insights are being drawn. Although from the drawn insights, the data is randomly spread but still modelling is performed with different machine learning algorithms to make sure we cover all the possibilities. And finally, as expected random forest regression performed well and further hyperparameter tuning is done to increase the model’s accuracy.

**2.7. UI integration**

Streamlit library is used to create the webpage of project. The webapp is created inside `app.py` file.

**2.8. Data from user**

The data from the user is retrieved from the created Streamlit webpage.

**2.9. 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.

**2.10. Rendering the results**

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

**2.11. Deployment**

The tested model is then deployed to Streamlit.io. So, users can access the project from any internet devices.

1. **Unit Test Cases**

|  |  |  |
| --- | --- | --- |
| Test Cases Description | Pre-Requisites | Expected Results |
| Verify whether the user interface URL is accessible to 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. |