Low-Level Design (LLD) Flight Fare Prediction

Revision Number – 1.2
Last Date of Revision: 11/08/2023
Author : Aditya Singh Jadon

Document Version Control Date Version Description Author: 30/07/2023

1.0
Abstract, Introduction Architecture : 28/07/2023

1.1
Architectural Design

28/07/2023

1.2
Deployment, Unit Test Cases

Contents

Abstract

The recent changes in the international market had a large impact on the Aviation sector because of several reasons. These impact the two class folks, the first is the Business perspective and the second is Customer perspective. The major reason for such impact is the governments around the world amended totally different rules to their various Airline firms. Taking these factors into consideration, the value of the flight tickets has varied from one place to another. Booking a flight ticket has its price tag split into two, one is online bookings and other is offline bookings. Each of these have their various criteria for the value of the price, one such example is the server load and therefore the range of booking requests. During this machine learning implementation, we are going to see numerous factors that impact the price of the flight ticket and predict the acceptable price of the ticket.

1. Introduction

1.1 Why is the LLD Document?

The main goal of the LLD document is to give the internal logic design of actual code implementation and supply the outline of the machine learning model and its implementation. Additionally, it provides the description of how our project will be designed end-to-end.

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 Design

This project is designed to make an interface for the User to predict its approximate flight ticket price.

3.1 Data Collection

The data for these project is collected from the Kaggle Dataset, the URL for the dataset is kaggle.com/datasets/nikhilmittal/flight-fare-prediction-mh

3.2 Data Description

Flight Fare Prediction is a 10K+ dataset publicly available on the Kaggle. The information in the dataset is present in two separate excel files named as train.xlsx and test.xlsx. The dataset contains 10683 rows which shows the information such as Date of Journey, Source, Destination, Arrival Time, Departure Time, Total stops, Airlines, Additional Info, and Price. A glance of the Dataset is :

3.3 Importing 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 info with the name FlightInfo.
- clash command is written for making the info table with needed parameters.
- And finally, a clash command is written for uploading the Knowledge Set into the data table by bulk insertion.

- Exporting Data from Database
- In the above-created API, the download URL is also being created, which downloads the data into a CSV file format.

- Data Preprocessing
- Checked for info on the Dataset, to verify the correct datatype of the Columns.
- Checked for Null values, because the null values can affect the accuracy of the model.
- Converted all the desired columns into Date time format.
- Performed One – Hot encoding on the desired columns.
- Checking the distribution of the columns to interpret their importance.
- Now, the info is prepared to train a Machine Learning Model.

- Model Creation

The Preprocessed info is now envisioned and drawn insights help us to select the feature that improves the accuracy of the model. The info is randomly used for modeling with different machine learning algorithms to create a model to predict the Flight ticket price. After performing on different algorithms, we use Random Forest Regression to create a model and then also perform Hyperparameter Tuning to improve the accuracy of the model.

- Data from User

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

- Data Validation

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

- Rendering the Results

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

3. Deployment

The tested model is then deployed to Heroku. So, users can access the project from any internet device.

3.1 Unit Test Cases

Test Case Description

Pre - Requisites

Expected Results

Verify whether the
Web Page is accessible to the User or not.

Webpage URL should be defined.

Webpage should be accessible to the User.

Verify whether the web page is completely loaded for the User or not.

- Webpage URL is accessible.
- Webpage is deployed.

The Webpage should be completely loaded for the User when it is accessed.

Verify whether the user is able to enter data in input fields or not.

- Webpage URL is accessible.
- Webpage is deployed.
- Webpage input fields are editable.

The User is able to enter data in input fields.

Verify whether the user is able to submit details or not.

- Webpage URL is accessible.
- Webpage is deployed.
- Webpage input fields are editable.

The User is able to submit details to the process.

Verify whether the user gets recommended
results on submitting the details or not.

- Webpage URL is accessible.
- Webpage is deployed.
- Webpage input fields are editable.

The User gets recommended results on submitting the details.