Low Level Design Document(LLD)

Flight Fare Prediction

By

Vishal Patil

Abstract

Travelling through flights has become an integral part of today's lifestyle as more and more people are opting for faster travelling options. The flight ticket prices increase or decrease every now and then depending on various factors like timing of the flights, destination, duration of flights. various occasions such as vacations or festive seasons. Therefore, having some basic idea of the flight fares before planning the trip will surely help many people save money and time.

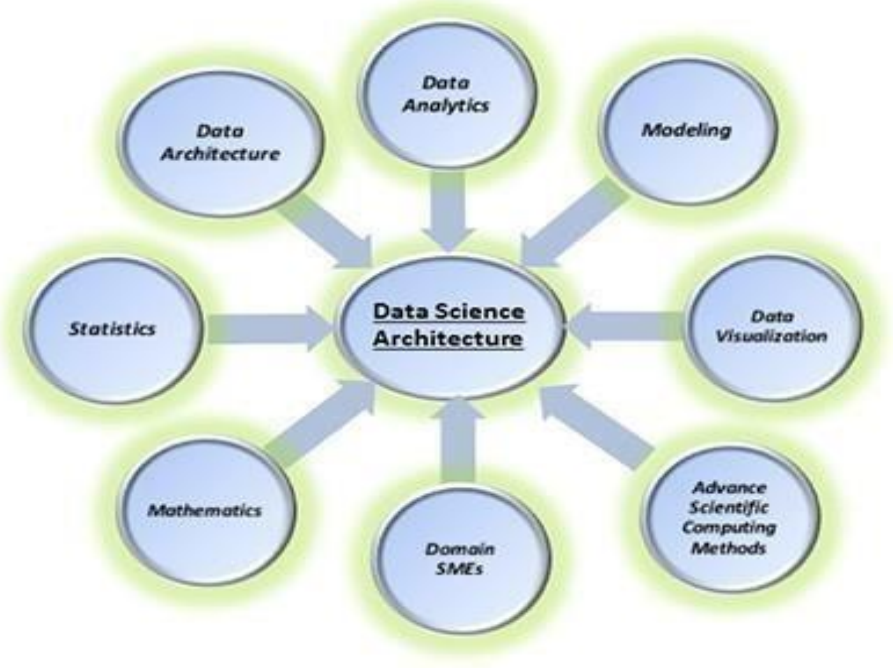
In the proposed system a predictive model will be created by applying machine learning algorithms to the collected historical data of flights. This system will give people an idea about the trends that prices follow and provide a predicted price value which they can refer to before booking their flight tickets to save money. This kind of system or service can be provided to the customers by flight booking companies which will help the customers to book their tickets accordingly.

1 Introduction

1.1 Why this Low-Level Design Document?

The main purpose of this LLD documentation is to feature the required details of the project and supply the outline of the machine learning model and the written code. This additionally provides a careful description However the complete project has been designed end-to-end.

1.2 Architecture



2. Architecture Design

For this project, we have implemented the machine learning life cycle to create a basic web application which will predict the flight prices by applying machine learning algorithms to historical flight data using python libraries like Pandas, NumPy, Matplotlib, seaborn and sklearn.

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/nikhilmittal/flight-fare-prediction-mh

2.2 Tool Used

- Python 3.10 is employed because of the programming language and frameworks like numpy, pandas, sklearn and alternative modules for building the model.
- VsCode is employed as an IDE.
- For visualizations seaborn and components of matplotlib are getting used
- For information assortment prophetess info is getting used version management.
- Local Machine is employed for deployment.

2.3 Data Description

We have 2 datasets here: training set and test set. The training set contains the features, along with the prices of the flights. It contains value. Following is the description of features available in the dataset 10683 records, 10 input features and 1 output column 'Price'. The test set contains 2671 records and 10 input features. The output 'Price' column needs to be predicted in this set. We will use Regression techniques here since the predicted output will be continuous.

- Following is the description of features available in the dataset :-

  1. Airline: Name of the Airline from which the Ticket is Booked.
  2. Date_of_Journey: Date of Journey of the Traveler.
  3. Source: Source from which the Airline Would Departure.
  4. Destination: Destination to Which Airline Would Arrive.
  5. Route: Route of the Airline from Source to Destination.
  6. Dep_Time: Time at which Flight Would Departure from the Source.
  7. Arrival_Time: Time at which Flight Would Arrive at the Destination.
  8. Duration: Duration that Airline Takes to fly from Source to Destination.
  9. Total_Stops: Total No of Stops that Airline takes Between Source and Destination
  10. Additional_Info: Any Additional Info about the Airline.
  11. Price: Fare of the Ticket to fly from Source to Destination.

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 an info with name flight fare.
- Cqlsh command is written for making the info table with needed parameters.
- And finally, a cqlsh command is written for uploading the knowledge set data into the table by bulk insertion.

2.5 Export Data into 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 Preprocessing

Steps performed in pre-processing are:

- Checked for null values as their square measure few null values, those rows square measure born.
- Convert the string data type into the desired data type.
- Extract the Date Column separately Day, Month, Year likewise.
- hours and minutes are extracted from departure time.

2.7 Modelling

After selecting the features which are more correlated to price the next step involves applying machine algorithms and creating a model. As our dataset consists of labelled data, we will be using supervised machine learning algorithms. Also, in supervised we will be using regression algorithms as our dataset contains continuous values in the features. Regression models are used to describe relationships between dependent and independent variables. The machine learning algorithms that we will be using in our project are:

1) Linear Regression
2) Decision Tree
3) Random Forest
4) Performance Metrics

2.8 UI Integration

The front of the application will be created using the HTML, CSS , Bootstrap framework where users will have the functionality of entering their flight data. This data will be sent to the back-end service where the model will predict the output according to the provided data. The predicted value is sent to the front-end and displayed.

2.9 Data from User

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

2.10 Data Validation

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

2.11 Rendering Result

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

3. Deployment

After getting the model with the best accuracy we store that model in a file using the pickle module. The back end of the application will be created using Flask Framework where API end-points such as GET and POST will be created to perform operations related to fetching and displaying data on the front-end of the application.

3.1 Unit Test

Data	Expected Result	Actual Result	Status
User Interface	Verify whether a user can give input to all fields.	As Expected,	Pass
User Interface URL	verify whether hosted URL is properly accessible or not	As Expected,	Pass
User Interface URL	Verify whether the user Interface loads completely for the user when the URL is accessed	As Expected,	Pass