Low Level Document

FLIGHT FARE PREDICTION APPLICATION

Written By	Sumit singh pal
Document Version	0.1
Last Revised Date	21-nov-2021

Document Control

Change Record:

Version	Date	Author	Comments
0.1	21-nov-2021	sumit	Introduction & Architecture defined

Reviews:

Approval Status:

Contents

1.Introduction

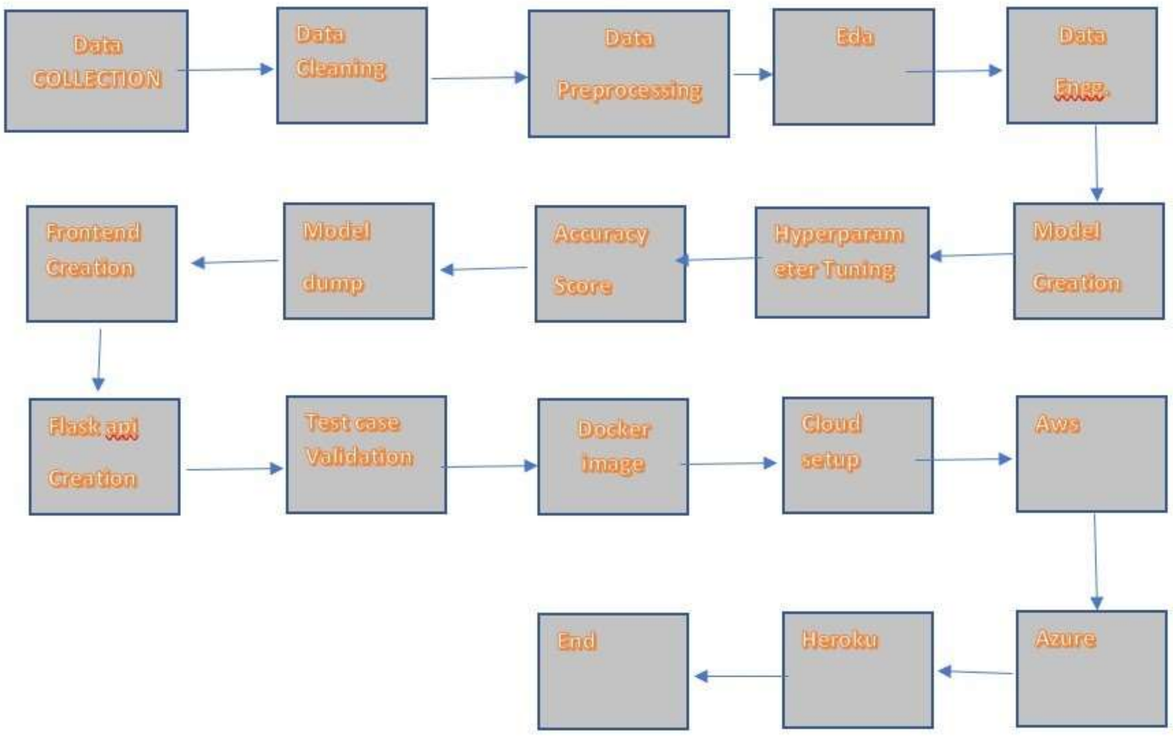
1.1. What is Low-Level design document?

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-bystep 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 14k 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 comma seprated value format (.csv).These data is collected from the Kaggle which contains both the test data and train data.

3.2. Data Cleaning

In the Cleaning process, We have cleaned up all the data because data is present in very bad format which was cannot reconigzed by machine.So data engineering is done very first.

3.3. Data Pre-processing

Data Pre-processing steps we could use are Null value handling, One hot encoding, columns in a integer format by implementing proper techniques to manage the columnar data.

3.4. Exploratory Data Analysis

In eda we have seen various insights from the data so we have selected which column is most important and dropped some of the columns by observing their sperman rank co-relation and plotting their heatmap from seaborn library also we done outlier removal and null value managed in a efficient manner and also implemented one hot encoding their.

Model Creation:

After cleaning the data and completing the feature engineering. we have done splitted data in the train data and test data and implemented various regression algorithm like Linear, DecisionTree, SVR, RandomForest, K-NN, AdaBoost, GradientBoost Regression and also calculated their accuracies on test data and train data.

Hyperparameter Tuning:

In hyperparameter tuning we have implemented various ensemble techniques like random forest regressor, bagging and boosting we also done randomized search cv or grid search cv and from that we also implemented cross validation techniques for that. From that we have choosen best parameters according to hyperparameter tunning and best score from their accuracies so we got 85% accuracy in our random forest regression after hyper parameter tuning.

Model Dump:

After comparing all accuracies and checked all roc, auc curve we have choosen hyperparameterized random forest regression as our best model by their results so we have dumped these model in a pickle file format with the help of joblib python module.

User Interface :

In Frontend creation we have made a user interactive page where user can enter their input values to our application. In these frontend page we have made a form which has beautiful styling with css and bootstrap. These html user input data is transferred in json format to backend. Made these html fully in a decoupled format.

3.10. Data from User

Here we will collect users requirement to catch their flight like a number of stops, departure date, destination , Source, Number of hours , Day etc.

3.11. Data Validation

Here Data Validation will be done, given by the user

3.12. User Data Inserting into Database

Collecting the data from the user and storing it into the database. The database can be Cassandra cloud Version.

3.13. Model Call/.pkl file loaded

Based on the User input will be throwing to the backend in the dictionary format so our we are loading our pickle file in the backend and predicting some total number of hours as a output and sending to our html page.

3.14. Deployment

We will be deploying the model to Heroku cloud platforms This is a workflow diagram for the Flight Fare production.

1. Unit Test Cases

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 aws ec2 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 see input fields on logging in	1. Application is accessible 2. User is logged in to the application	User should be able to see input fields on logging in
Verify whether user is able to edit all input fields	1. Application is accessible 2. User is logged in to the application	User should be able to edit all input fields
Verify whether user gets Predict Flight Price button to submit the inputs	1. Application is accessible 2. User is logged in to the application	User should get Submit button to submit the inputs
Verify whether user is presented with recommended results on clicking submit	1. Application is accessible 2. User is logged in to the application	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 2. User is logged in to the application and database	The recommended results should be in accordance to the selections user made
Verify whether user has options to	1. Application is accessible	User should have options to filter

5

filter the recommended results as well		the recommended results as well
	3. User is logged in to the application	
Verify whether the KPIs indicate details Of the correct inputs	1. Application is accessible 2. User is logged in to the application	The KPIs should indicate details of the suggested inputs to users.

6