LOW LEVEL DESIGN

FLIGHT FARE PREDICTION

Written By	**Panuganti Arun Kumar**
Document Version	**0.1**
Last Revised Date	**27-11-2022**

Contents

Document Control

Change Record:

Version	Date	Author	Comments

Reviews:

Version			

Approval Status:

1. Introduction

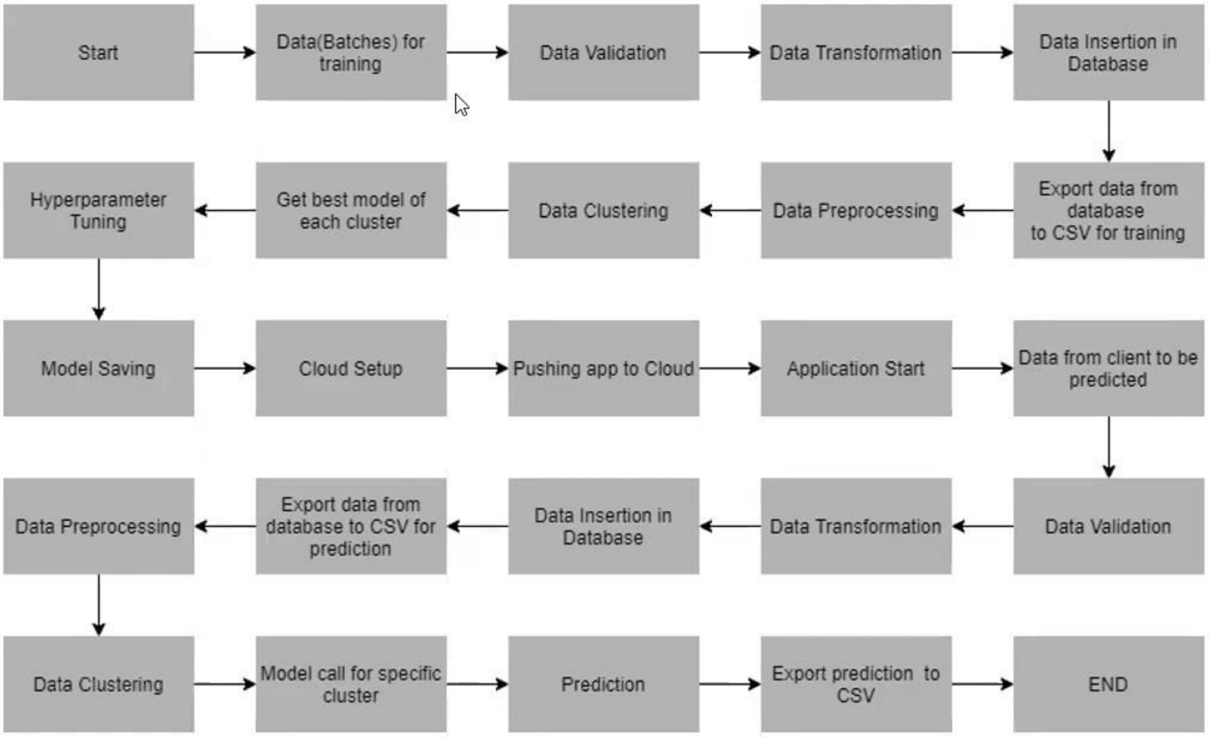
1.1 What is Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Food Recommendation 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 requirements analysis and then refined during data design work

2. Architecture


Start
Data(Batches) for training
Data Validation
Data Transformation
Data Insertion in Database
Export data from database to CSV for training
Data Preprocessing
Data Clustering
Get best model of each cluster
Hyperparameter Tuning
Model Saving
Cloud Setup
Pushing app to Cloud
Application Start
Data from client to be predicted
Data Validation
Data Transformation
Data Insertion in Database
Export data from database to CSV for prediction
Data Preprocessing
Data Clustering
Model call for specific cluster
Prediction
Export prediction to CSV
END

3. Architecture Description

3.1 Data Description

The dataset contains is flight related data such with columns namely

1. Airline
2. Source
3. Destination
4. Total Stops
5. Journey Day
6. Journey Month
7. Departure Hour
8. Departure Minute
9. Arrival Hour
10. Arrival Minute
11. Duration Hour
12. Duration Minute
13. Price

And the dataset contains both numerical and categorical data with 10684 records.

3.2 Splitting the data

The data is split happens to divide the data for training and validation, further this data will be stored in artifact folder.

3.3 Data Validation

Here the Data will be fetched from the artifact folder and the Schema of the data will be checked like column names and their data type. If every constraint is satisfied then the data will be allowed into the data transformation stage.

3.4 Data Transformation

In the Transformation Process, we will convert our original dataset which is in CSV format stored in artifact folder into required format which will help our model to train. In this stage data pre-processing will be executed and finally the data again stored in the artifact folder and also the pickle object of pre-processing will be stored.

3.4 Data Pre-processing

Data Pre-processing steps we could use are Null value handling, Deleting duplicates, Imbalanced data set handling, Standardization, etc.,

3.5 Model Building

The pre-processed data will be fetched from the artifact folder and then Model training process will be initiated. The model can be trained using several machine learning algorithms using grid search cv. Then the model will be pushed into artifact folder along with the model pickle file.

3.6 Model Evaluation

In this stage the model’s performance will be checked with the previously trained model by comparing accuracy. If the present model is giving good accuracy than previous model then this one will be replaced with previous one otherwise this stage will reject the current model.

3.7 Model Pusher

If the model evaluation gives green signal, then this stage will save the present model into ‘saved models’ folder.

3.8 Deployment

Deployment will be happened in the form of Continues Integration and Continues Deployment (CI/CD). So here we use the Heroku platform to deploy the project by using Git Actions.

4 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 whether user is able to see the options in the web page	1. Application is accessible	The user should be able to see the web page will all required options
Verify whether user gets train button to train the model	1. Application is accessible	User should able to click the train button to train model
Verify whether user is able to see input fields after click in estimate button	1. Application is accessible	User should be able to edit all input fields
Verify whether user gets submit button to submit the inputs	1. Application is accessible	User should get submit button to submit inputs

Verify whether user is presented with recommended results on clicking submit	1. Application is accessible	User should be presented with recommended results on 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