LOW LEVEL DOCUMENT

FLIGHT FARE ESTIMATOR

Written By	Suraj Joshi
Document Version	1.0
Last Revised Data	25 Nov 2021

LOW LEVEL

1 Document Control:

Version	Date	Author	Comments
1.0	25/11/2021	Suraj Joshi	

2 Reviews:

Version	Date	Reviewer	Comments
1.0	25/11/2021	Suraj Joshi	

3 Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments
1.0	25/11/2021	Suraj Joshi	Suraj Joshi	

Table of Contents

LOW LEVEL

1.Introduction

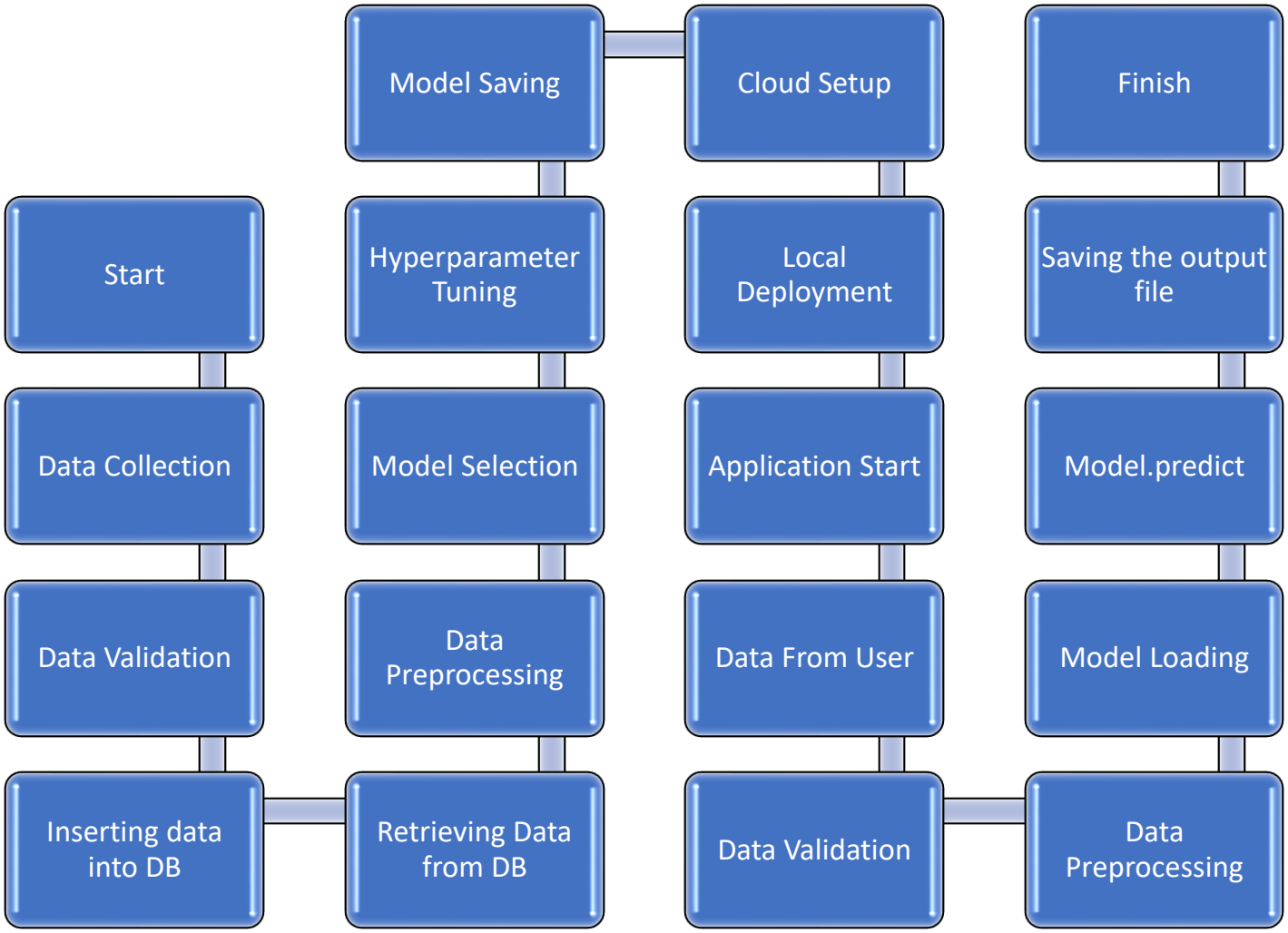
1.1 What is Low-Level design document?

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-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 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 separated value format (.csv). These data is collected from the Kaggle which contains both the test data and train data.

3.2 Data Validation

In Data Validation part we basically check whether files inside the folder provide are having same columns or not with the data type of each column each by each file. If the file is found to matching the name of columns and dtype of the columns then we are supposed to copy the file from the folder to Training_Batch_Files/Good_Data or else to Training_Batch_Files/Bad_Data

3.3 Inserting into DB

We are going to insert the data of Good_Data into Database that is CassandraDB .For this we have used Cassandra driver Our Database name is Flight_Fare and keyspace is training.

3.4 Retrieving Data From DB

Retrieving the data and storing into a file called as Testing_file.csv

3.5 Data Preprocessing

Data Preprocessing step is very necessary for model creation, We have mainly categorical data into our csv file so we need to convert that categorical data into numerical format. Data Preprocessing is getting done by the file data_preprocessing.py

3.6 Model Selection

Model Selection is performed by file model_training.py which is inside Model_Training. For model Training we are just choosing XGBRegressor because it is performing best than other models When calculating mean_absolute_error XGB always had a very low mae.

3.7 Hyperparameter Tuning

We are doing hyperparameter tuning only for XGB and because of tuning our model has a low mean_absolute_error.

3.8 Model Saving

Model Saving is the final step in the training part. We are saving the model in folder Model_for_prediction as model.pickle and I am using module pickle for saving the ML model.

3.9 Data from User

On Application Starting user will be interacting with a UI which is designed using HTML/CSS. There are two buttons,

First button is Custom File Predict which is for predicting files which you have in your local machine for this u just copy the path of that folder and paste it into the button above custom file predict and

Second button is Default File Predict used in that case when the user don't have any files for prediction but want to check my application in that case you can simply click on the button. So these were the two ways by which user can provide data for prediction

3.10 Data Validation

It is not necessary that the csv files provided by the user will always be in required format so to check this I have included this part which is data validation.

In Data Validation part we basically check whether files inside the folder provide are having same columns or not with the data type of each column each by each file. If the file is found to matching the name of columns and dtype of the columns then we are supposed to

copy the file from the folder to Training_Batch_Files/Good_Data or else to Training_Batch_Files/Bad_Data.

3.11 Data Preprocessing

In data preprocessing I will be handling categorical columns plus some other columns as well as like Duration,Arrival_Time column.

3.12 Model Loading

Model loading is actually very simple we will be calling the model which we saved as modle.pickle inside folder Model_for_prediction. For loading the model I am using pickle library

3.13 Model.predict

After loading the model everything is very simple you just have to predict the preprocessed data.

3.14 Saving the output file

Saving the output file inside the folder Output_File as Predicted_data.csv. I am using library Pandas to save the csv file code is df.to_csv(filename).

4 Test Cases

Test cases are given below

LOW LEVEL

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 heroku 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 Custom File Predict, Default File Predict button to submit the inputs	1. Application is accessible 2. User is logged in to the application	User should get both buttons 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

LOW LEVEL

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