LOW LEVEL DESIGN

LOW LEVEL DOCUMENT

FLIGHT FARE PREDICTION

Written By	Karan Doke
Document Version	1.0
Last Revised Data	11 Feb 2022

LOW LEVEL DESIGN

1 Document Control:

Version	Date	Author	Comments
1.0	11/02/2022	Karan Doke	

2 Reviews:

Version	Date	Reviewer	Comments
1.0	11/02/2022	Karan Doke	

3 Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments
1.0	11/02/2022	Karan Doke	Karan Doke	

LOW LEVEL DESIGN

Table of Contents

LOW LEVEL DESIGN

1.Introduction

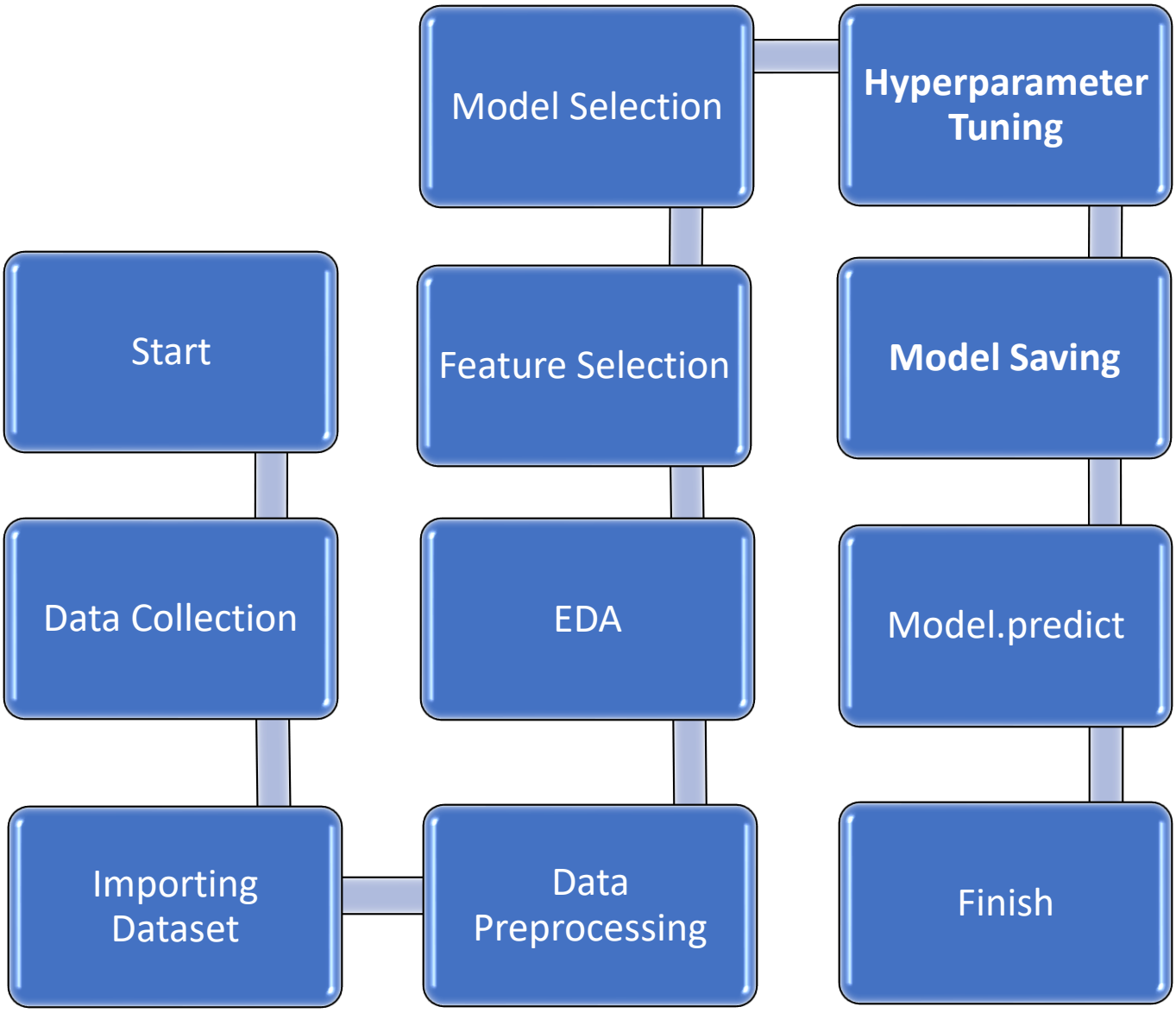
1.1 What is Low-Level design document?

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 Importing Dataset

- Since data is in form of excel file we have to use pandas read_excel to load the data.

3.3 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 flight_price.ipynb

1. First I checked whether is there any null values present inside the data and I found that there is a single row only that's why I directly removed it.

2. Date_of_Journey This column contains the date of the journey all the data was of the year 2019 so I created two columns from there first is Journey_ day and the second is Journey_month.

3. Column Dep_Time This column was containing time only I created the data of Dep_Time into only hours, For example, if time is 08:30 it will be 8.5.

4. Arrival_Time I dropped this column as there was already a column called Duration Hours , So having two columns representing the same information or data is not good, So I kept only the Duration column.

5. Handling Duration column it was in the format of 8h 45m, I created a new columnuration_Hours and converted the data into 8.75 for 8h 45m and so on for every data.

6. Now, My main task was to handle categorical data and the data was mostly categorical. One can find many ways to handle categorical data. Some of them categorical data are,
    - Nominal data → data are not in any order → OneHotEncoder is used in this case.
    - Ordinal data → data are in order → LabelEncoder is used in this case.

7. Now after handling the categorical column there was a column called Route which was actually of no use in model creation so I removed the column.

3.4 Model Selection

- In model training I tried a few algorithms like XGBRegressor, and Random Forest Regressor because I know that they give the best score than others.

- Random Forest Regressor was giving me less **Coefficient Of Determination (R^2) – 0.8122** as compare to XGBRegressor **Coefficient Of Determination (R^2) – 0.8463**. I finally decided to keep only XGB as my model as it takes less time for training than rf and I had to do hyperparameter tuning also which will take a lot of time.

3.5 Hyperparameter Tuning

- After choosing model I did parameter tuning of my model using RandomizedSearchCV and it helped me to Increase my **Coefficient Of Determination (R^2)** to **0.8514** from **0.8463** which was previous R^2 without any hyperparameter tuning.

3.6 Model Saving

- Model Saving is the final step in the training part. We are saving the model in file flight_rf.pkl and I am using module pickle for saving the ML model.

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

3.7 Model.predict

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

3.8 Creating API

- Now I created an API for my model using Flask.

3.9 Deployment

- Finally My project is created and I am going to deploy it to the Hereko Platform.

LOW LEVEL DESIGN

4 Test Cases → Test cases are given below

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 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 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