Flight Fare Predication

Low Level Design

|  |  |
| --- | --- |
| Written By | **Shripad Garat** |
| Document | **0.1** |
| Last Revised | **04/01/2023** |

[Date:04/01/2023]

# Document Control:

Change Record:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 0.1 | 04/01/2023 | Shripad Garta | First Review |
|  |  |  |  |

Reviews:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Reviewer | Comments |
|  |  |  |  |
|  |  |  |  |

Approval Status:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Date | Reviewer | Approver | Comment |
|  |  |  |  |  |
|  |  |  |  |  |

**Contents:**

**1.Introduction............................................................................................................3**

1.1.What\_is\_Low\_Level\_design\_document? ..........................................................3

1.2.Scope...................................................................................................................3

**2.Architecture..................................................................................…......................4**

**3.Architecture\_Description..........................….........................................................5**

# 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 Predication Application. 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.

# Architecture

Data-Base

Data Transformation

Data Validation

Data Ingestion

User Input Data

Model Evaluation

Model Training

Prediction

Predicted Result

Model Pusher

Base Dataset

Retraining Dataset

Training Pipeline

Prediction Pipeline

API

# Architecture Description

* 1. ***Base Data-set***

The base data-set is data-set we used for to determine the validation rules and also to train the base model of our project. The features in this data set will going to be same entire time with same rules and format.

* 1. ***Retraining Data-set***

In future if the model don’t do well we can retrain the model using this retraining data-set which we collect from the application.

* 1. ***Data Base***

The Data Base we are using in the project is MongoDB. Here the Training and base data set are stored.

* 1. ***Data Ingestion***

In Data Ingestion component it is used for the extracting the data from database and stored it in the artifact for the further use. Here we are going to extract, split and store data set in artifact with training and testing data set format.

* 1. ***Data Validation***

In data validation component it is used for validating the data which we have extracted from the data base with our base dataset and the data description we receive from the data base. If the validation pass then data will push to store in the good data artifact else it will pushed to bad data artifact.

* 1. ***Data Transformation***

In data transformation component it is used to transform the data set in to trainable dataset as we data come from the data base will going to be not in trainable format. Here we will do some EDA and feature engineering on the dataset and transfer it to required dataset format to train our model. Once the dataset get transfer the dataset will get save along with transformer object in the data transformation artifact.

* 1. ***Model Training***

In Model training component it is used to train the model on the transferred dataset and it will give the score of the model on both train and test dataset. It will check on the overfiting and expected score. Once the check passes the model will get save into model training artifact.

* 1. ***Model evaluation***

In model evaluation component it is used to evaluate the train model with current present model and if the model do well on testing dataset than previous one then the train model will push to the model pusher component.

* 1. ***Model Pusher***

In model pusher component it is used to push the train model in to the saved model artifact for further used to predict the fair of the flight.

* 1. ***API***

In the API, we are going to build the API for the entire project and then we will extracted the dataset from the user and then we will perform the prediction component to predict the fair of the flight.