**Low Level Design**

**Flight Price Prediction**

|  |  |
| --- | --- |
| **Written by** | **Denistan B** |
| **Documented Version** | **1.0** |
| **Last Revised Date** | **13 December 2024** |

**Contents**

1. **Abstract**
2. **Introduction**2.1 Why this Low-Level Design Document?  
   2.2 Scope  
   2.3 Constraints  
   2.4 Risks
3. **Technical Specifications**3.1 Dataset  
   3.2 Predicting Sales  
   3.3 Logging  
   3.4 Database
4. **Technology Stack**
5. **Proposed Solution**
6. **Model Training/Validation Workflow**
7. **User I/O Workflow**
8. **Exceptional Scenarios**
9. **Key Performance Indicators (KPI)**

### **1. Abstract**

The **Flight Fare Prediction System** is a machine learning solution designed to forecast airline ticket prices based on key factors such as flight timing, destination, duration, and seasonal variations. Air travel has become an essential part of modern life, with dynamic pricing driven by demand fluctuations, special occasions, and other external factors. This system aims to equip users with an estimate of flight fares, enabling them to make informed decisions, save money, and plan their trips effectively.

By leveraging historical flight fare data, advanced machine learning techniques, and features like date of travel, airline, and stopovers, the system delivers accurate predictions. The solution ensures scalability, real-time usability, and reliability, making it a valuable tool for travelers seeking cost-effective options and efficient planning.

### **2. Introduction**

### 2.1 Why this Low-Level Design Document?

The purpose of this Low-Level Design Document (LLD) is to provide a comprehensive and detailed blueprint for implementing the **Flight Fare Prediction System**. This document serves as a crucial reference for developers, testers, and stakeholders. It deconstructs the system into manageable components, such as data preprocessing, model training, user workflows, and storage design, ensuring alignment with project objectives and efficient development processes.

### 2.2 Scope

This LLD addresses the following components:

* **Data Collection**: Gathering historical flight fare data and related factors (e.g., date, airline, destination, duration).
* **Machine Learning Models**: Using algorithms like Gradient Boosting and Random Forest for fare prediction.
* **Feature Engineering**: Creating or transforming input features to enhance predictive accuracy.
* **User Input and Prediction**: Processing real-time user inputs to generate fare predictions.
* **Deployment**: Deploying the system on a cloud platform (e.g., AWS or GCP) for accessibility and scalability.
* **Storage**: Storing prediction results and user data securely in a database for future analysis.

### 2.3 Constraints

* **Data Availability**: Reliance on high-quality and comprehensive historical flight fare datasets.
* **Resource Constraints**: Limited computational resources may impact the speed of model training and real-time prediction generation.
* **Latency**: Potential delays in processing and delivering predictions for large-scale user inputs.
* **Data Format**: Variations in input formats or missing/inconsistent values could affect model performance.

### 2.4 Risks

* **Model Overfitting**: Complex models may overfit, especially with imbalanced or noisy data.
* **Market Dynamics**: Unexpected changes in airline pricing trends, holidays, or unforeseen events may reduce model accuracy.
* **Cloud Downtime**: Cloud platform disruptions can affect system availability and reliability.
* **Security Risks**: User data and fare predictions must be safeguarded against unauthorized access or breaches.

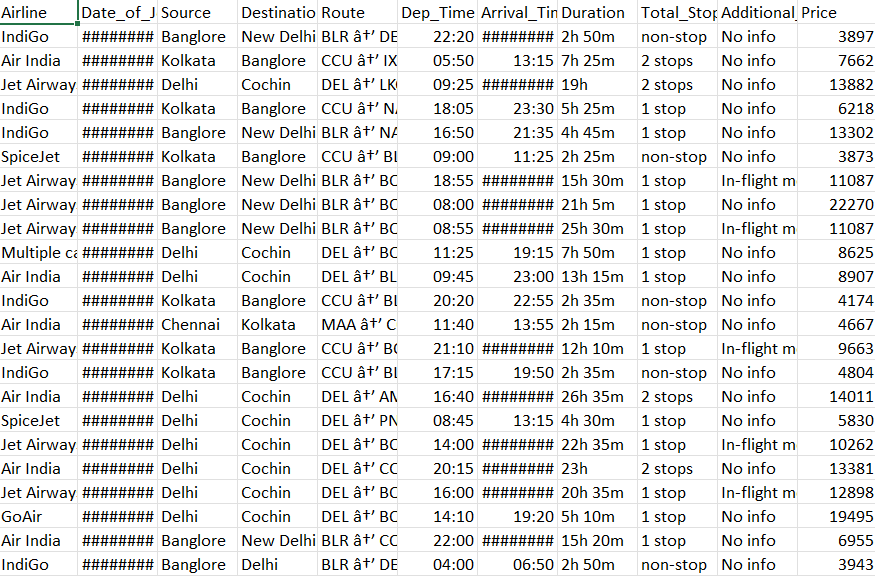
### 3. Introduction

#### 3.1 Dataset

The **Flight Fare Prediction System** leverages structured datasets containing historical flight details, airline-specific attributes, and other influential factors to predict fares accurately. The key features include:

#### **Key Attributes**

* **Historical Flight Data**:
  + **Airline**: Name of the airline operating the flight.
  + **Source**: Departure location for the flight.
  + **Destination**: Arrival location for the flight.
  + **Date of Journey**: The date the flight is scheduled.
  + **Departure Time**: Time of departure.
  + **Arrival Time**: Time of arrival.
  + **Duration**: Total duration of the flight journey.
* **Flight-Specific Attributes**:
  + **Stops**: Number of layovers during the flight (e.g., non-stop, 1-stop).
  + **Additional Info**: Details such as meal options or inflight amenities.
* **External Factors**:
  + **Seasonality**: Whether the journey coincides with festive seasons, holidays, or peak travel periods.
  + **Day of the Week**: Influence of weekdays versus weekends on ticket prices.
* **Target Variable**:
  + **Flight Fare**: The price of the flight ticket to be predicted.



### Preprocessing Steps

#### Dimensionality Reduction

* **Initial Features**: The dataset originally included 11 features.
* **Final Features**: After statistical analysis and correlation evaluation, the data was reduced to 6 critical features:
  1. **Airline**
  2. **Source**
  3. **Destination**
  4. **Duration**
  5. **Stops**
  6. **Flight Fare** (Target Variable)

This reduction improved training efficiency and ensured model accuracy.

#### Encoding

* **Categorical Variables**: Features like Airline, Source, and Destination were numerically encoded to ensure compatibility with machine learning models.
  + **Example Transformations**:
    - Airline categories were converted into numerical values using one-hot encoding (e.g., "IndiGo" = 1, others = 0).
    - Similarly, Source and Destination were transformed into binary columns.

These transformations allowed seamless integration with models such as Random Forest Regressor and Linear Regressor.

#### Data Transformation

* **Logarithmic Transformations**: Applied to continuous variables to stabilize variance and address skewness.
  + Variables transformed include:
    - **Flight Fare**
    - **Duration**

**Purpose**:

* Stabilize variance across the dataset.
* Smooth distributions to improve model performance.

#### Missing Value Handling

* **Imputation Methods**: Missing values in key columns were addressed as follows:
  + **Mean**: For continuous variables like Duration.
  + **Mode**: For categorical attributes such as Airline or Stops.
  + **Median**: Applied to handle outliers or extreme values in numerical data.

#### Feature Importance

* Post-preprocessing, feature importance was determined using:
  1. **Correlation Analysis**: Ensuring features with significant relationships to Flight Fare were retained.
  2. **Impact on Target Variable**: Evaluated to identify features with the strongest influence on fare predictions.

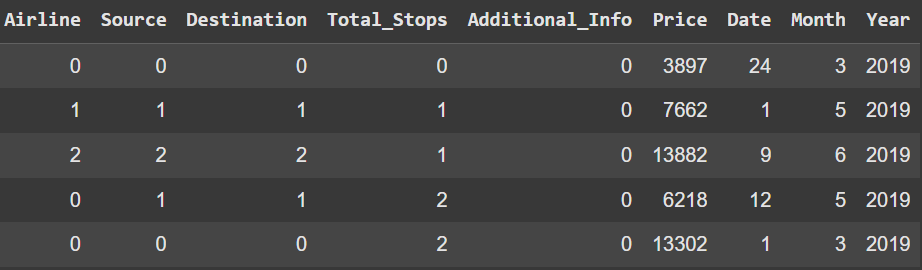
Key features like Airline, Stops, and Duration were found to have the highest impact on predicting flight fares.

#### Final Structure

The processed dataset includes:

1. **Numerical Columns**:
   * Continuous features such as Duration and Flight Fare.
2. **Encoded Categorical Columns**:
   * Transformed attributes such as Airline, Source, and Destination.

This structured dataset is optimized for machine learning algorithms like **Random Forest Regressor** and **Linear Regressor**, ensuring robust and accurate flight fare predictions.



### 3.2 Input Schema

The input data for the flight fare prediction system is structured in a **CSV file** or **database format**. After preprocessing, the required columns for the model are as follows:

#### **Input Columns**

1. **Airline**: Encoded categorical variable representing the airline of the flight.
2. **Source**: Encoded categorical variable indicating the flight's origin.
3. **Destination**: Encoded categorical variable indicating the flight's destination.
4. **Duration**: Continuous variable representing the total flight time.
5. **Stops**: Encoded categorical variable showing the number of stops (e.g., direct, 1-stop, etc.).

#### **Output Column**

* **Flight Fare**: The target variable representing the predicted fare for the flight.

### 3.3 Prediction Workflow

#### **3.3.1 Pretrained Model Setup**

* The system utilizes a pretrained machine learning model trained on a structured flight fare dataset named train\_data. Initially, the dataset included 11 features comprising both numerical and categorical variables.
* After preprocessing and feature engineering, the dataset was refined and reduced to critical features relevant to flight fare prediction.
* Categorical variables were **label encoded** or **one-hot encoded** to ensure compatibility with machine learning models. For example:
  + A feature with 3 unique categories was converted into 3 binary columns.
  + Another feature with 5 unique categories was expanded into 5 binary columns.
* These transformations were essential for ensuring seamless integration with algorithms like **Random Forest Regressor** and **Linear Regressor**.

#### **3.3.2 Model Training**

* The dataset was split into **training** and **testing sets** in an **80/20 ratio** to ensure proper validation and generalization.
* Given the target variable, **Flight Fare**, exhibited skewness, **logarithmic transformations** were applied to stabilize variance and improve model performance by reducing the impact of outliers.
* The system employed two machine learning models:
  1. **Random Forest Regressor**: Leveraging ensemble learning for its ability to handle non-linear relationships and capture intricate patterns in the data.
  2. **Linear Regressor**: Providing a baseline prediction model with simplicity and interpretability.

#### **3.3.3 User Input**

* The system allows users to provide input through a **Streamlit-based interface**, ensuring simplicity and usability.
* The input data undergoes the **same preprocessing steps** as during model development, including:
  + **Handling Missing Values**: Imputation techniques are applied if missing values are detected.
  + **Encoding Categorical Variables**: Variables like Airline, Source, and Destination are encoded to match the format of the pretrained models.
  + **Feature Transformation**: Continuous variables such as Duration may undergo scaling or normalization for consistency with the model.

#### **3.3.4 Prediction Workflow**

The prediction process is designed for efficiency and accuracy:

1. **Input Validation and Processing**: User inputs are validated to ensure consistency and processed to align with the trained models' requirements.
2. **Model Prediction**:
   * The cleaned and preprocessed input is fed into the **Random Forest Regressor** or **Linear Regressor** to generate predictions.
   * Predictions are initially generated on a log-transformed scale.
3. **Reversing Transformations**: The predictions are reverted to their original scale using an **exponential function**, ensuring that the output is interpretable and realistic.

#### **3.3.5 Output Delivery**

* Predictions are presented to users in a **clear and concise format** on the application interface.
* Alongside predictions, the system provides key performance metrics to ensure transparency:
  + **R² Score**: Measures the model’s accuracy and goodness-of-fit.
  + **Mean Absolute Error (MAE)**: Indicates the average magnitude of prediction errors.
  + **Mean Squared Error (MSE)**: Highlights larger errors by squaring them.
* For additional clarity, users can view side-by-side comparisons of **predicted vs. actual values** for selected samples, demonstrating the model's performance and reliability.

### 4. Technology Stack

1. **Programming Language**
   * Python: For data preprocessing, model training, evaluation, and deployment.
2. **Libraries/Frameworks**
   * **Pandas**: For data manipulation and cleaning.
   * **NumPy**: For numerical computations and handling arrays.
   * **Scikit-learn**: For building, training, and evaluating machine learning models.
   * **Streamlit**: To create an interactive and user-friendly interface for real-time predictions.
   * **Pickle**: For saving and loading trained machine learning models during deployment.

### 5. Proposed Solution

#### 1. **Data Collection**

* Historical flight fare data and associated attributes such as airline, source, destination, and travel duration are collected.
* Additional factors, like departure time and stopovers, are included to build a structured dataset.

#### 2. **Data Preprocessing**

* **Handling Missing Values**: Missing data is imputed using appropriate statistical methods:
  + **Mean**: For continuous variables.
  + **Mode**: For categorical attributes.
* **Encoding Data**: Categorical variables like Airline, Source, and Destination are transformed using **label encoding** or **one-hot encoding** for compatibility with machine learning models.
* **Scaling**: Continuous variables, such as Duration and Fare, are log-transformed to handle skewness and stabilize variance.

#### 3. **Model Training**

* The preprocessed data is used to train advanced machine learning models:
  + **Random Forest Regressor**: An ensemble-based model that builds multiple decision trees to improve prediction accuracy and robustness.
  + **Linear Regressor**: A simple and interpretable model for establishing a baseline.
* Both models are evaluated using metrics such as **R² Score**, **MAE**, and **MSE** to ensure reliable performance.

#### 4. **User Input**

* Real-time inputs, such as departure date, airline, source, destination, duration, and stopovers, are collected through a **Streamlit-based interface**.
* The system validates the inputs and applies preprocessing steps, including encoding and scaling, to align with the trained model’s requirements.

#### 5. **Prediction Output**

* The prediction workflow includes:
  1. **Input Transformation**: User inputs are processed and normalized as per the model’s training setup.
  2. **Prediction Generation**: Predictions for flight fares are generated using the trained models.
  3. **Reversing Transformations**: Log-transformed predictions are converted back to their original scale using an exponential function for interpretability.
* Results are displayed to the user through the Streamlit application, along with evaluation metrics to convey the reliability of predictions.

#### 6. **Deployment**

* The application, built with **Streamlit**, is deployed for real-time usage, enabling users to access fare predictions interactively.
* **Pickle** is used to load pretrained models efficiently during deployment, ensuring a seamless user experience.

###### ChatGPT said:

ChatGPT

### 6. Model Training/Validation Workflow

1. **Data Collection**:
   * Gather historical flight fare data, including factors like source, destination, airline, travel dates, duration, and external variables such as promotions or holidays.
2. **Data Preprocessing**:
   * Handle missing values (using imputation techniques like mean, median, or mode).
   * Normalize continuous features such as Flight Duration and Fare to bring them into a consistent scale.
   * Encode categorical variables such as Airline, Source, and Destination using one-hot encoding or label encoding.
3. **Model Training**:
   * Train **Gradient Boosting Regressor** and **Random Forest Regressor** using the preprocessed dataset to predict the flight fare based on the collected attributes.
4. **Model Optimization**:
   * Use **GridSearchCV** to perform hyperparameter tuning and find the best-performing parameters for both models, ensuring higher prediction accuracy.
5. **Model Validation**:
   * Evaluate the models using performance metrics such as:
     + **R² Score**: Measures the proportion of variance explained by the model.
     + **Mean Squared Error (MSE)**: Measures the average squared differences between predicted and actual values.
     + **Mean Absolute Error (MAE)**: Measures the average absolute differences between predicted and actual values.
6. **Model Storage**:
   * Save the best performing models (using **Pickle**) for future use in the deployment phase. This ensures that the models can be loaded and reused without retraining.
7. **Deployment**:
   * Deploy the model via a **Streamlit** interface to allow real-time prediction of flight fares based on user input.

### 7. User I/O Workflow

1. **Input Collection**:
   * Users will input details such as:
     + **Weight**: Weight of the item in question (for the sales prediction context) or a related feature.
     + **Outlet Type**: Type of outlet/store (e.g., supermarket, grocery store).
     + **MRP**: Maximum Retail Price of the product.
     + **Location Type**: The geographical location of the store (e.g., urban, semi-urban, rural).
     + **Size**: The size of the outlet/store (e.g., small, medium, large).
2. **Validation**:
   * The input data will be validated to ensure all required fields are filled and the data format is correct.
   * Any missing or invalid input will trigger a prompt asking the user to correct or complete the data.
3. **Prediction**:
   * The system will load the appropriate pre-trained machine learning model.
   * It will then generate predictions for the given inputs using the trained model, providing an estimate of flight fares or item sales.
4. **Output Display**:
   * The system will display the predicted value (e.g., predicted fare or sales) to the user in a clear, user-friendly format, along with the prediction confidence if available.
5. **Data Storage**:
   * User inputs and the resulting predictions will be stored in a database for future reference and analysis. This helps track system usage and provides a historical record of predictions.
6. **Visualizations**:
   * **MRP vs Item Outlet Sales**: A graphical representation that shows how the MRP (Maximum Retail Price) correlates with the predicted Item\_Outlet\_Sales for a better understanding of sales trends.
   * **Size vs Item Outlet Sales**: A visualization showing how the size of the outlet impacts the predicted sales, helping businesses optimize their store layouts and product placements.

### 8. Exceptional Scenarios

1. **Missing Inputs**:
   * If any required input is missing, the user will be prompted to complete the missing fields before proceeding. This ensures the system operates with complete and accurate data.
2. **Database Failure**:
   * If the database connection fails, the system will log the error and attempt to retry the connection. An error message will be shown to the user if the issue persists.
3. **Model Not Found**:
   * If the model cannot be loaded (e.g., due to a missing file), the system will return a default prediction along with a warning message to the user indicating that the model is unavailable at the moment.
4. **Invalid Input Format**:
   * If the input data format is invalid (e.g., incorrect data type), specific error messages will be displayed to guide the user in correcting the input.

### 9. Key Performance Indicators (KPIs)

1. **Prediction Accuracy**:
   * Evaluate the models using the following metrics:
     + **R² Score**: Measures how well the predictions fit the actual data.
     + **Mean Squared Error (MSE)**: Measures how far off the predictions are from the actual values, with emphasis on larger errors.
     + **Mean Absolute Error (MAE)**: Measures the average magnitude of prediction errors, making it easier to interpret.
2. **Latency**:
   * The time taken to generate and display predictions in real-time, ensuring that the system is responsive and user-friendly.
3. **Scalability**:
   * The system's performance when handling multiple users and large volumes of input data, ensuring that it can efficiently handle increased load.
4. **User Satisfaction**:
   * Measured through user feedback, including ease of use and satisfaction with the accuracy and speed of predictions.
5. **Database Efficiency**:
   * The speed and efficiency with which the system stores and retrieves user inputs and predictions, ensuring minimal delays and efficient data handling.