High Level Design (HLD)

Flight Fair Prediction

Revision Number: 1.0
Last date of revision: 26/07/2024

Document Version Control

Date Issued	Version	Description	Author
20/07/2024	1	Initial HLD - V1.0	Biswajit Jena
26/07/2024	2	Updated KPI - V1.1	Biswajit Jena

Contents

Contents

Abstract

Recent trends are to build tall buildings in big cities as a way out of the current housing overpopulation problem. These new structures unveil problems that if not addressed in time could cause catastrophes of unimaginable impact. Some of those problems is the incidence of
a fire threat happening upstairs in one of those buildings, medical emergencies due to any road accidents or mob that may cause threat to the human kind. This work discusses the implementation of the unmanned ground vehicles to spot the real location of the medical emergencies due to road mishap, mob or illegal activities such as hooliganism, snatching, robbery and the fire emergency and accordingly channelize or route them to the concerned helpline for quick mitigation and avoid disaster.

1 Introduction

1. Why this High-Level Design Document?

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level.

The HLD will:

- Present all of the design aspects and define them in detail
- Describe the user interface being implemented
- Describe the hardware and software interfaces
- Describe the performance requirements
- Include design features and the architecture of the project
- List and describe the non-functional attributes like: o Security
  oReliability
  oMaintainability o Portability
  oReusability
  oApplication compatibility o Resource utilization
  oServiceability

2. Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

3. Definitions

Term	*Description*
Database	Collection of all the information monitored by this system
IDE	Integrated Development Environment
AWS	Amazon Web Services

2 General Description

2.1 Product Perspective

The Flight Fare Prediction System is designed to provide users with accurate fare predictions for various flights based on their input details. Utilizing machine learning algorithms and a comprehensive dataset, this system aims to help users find the best fares for their travel plans efficiently.

2.2 Problem statement

To create an AI-based solution for predicting flight fares, addressing the following use cases:.

- Predict flight fares based on user-provided details such as departure city, arrival city, travel dates, and airline preferences.
- Compare fares across multiple airlines and booking platforms to ensure the user gets the best deal..
- Analyze historical fare data to provide insights into price trends and suggest the best time to book flights.
- Send alerts to users when there is a significant drop in flight prices for their desired routes.

2.3 PROPOSED SOLUTION

The The solution involves several key components:

1. **Data Collection**: Collect data related to flight fares from various sources.
2. **Web Scraping:** Use web scraping techniques to gather flight fare data from websites.
3. **Data Transformation**: Transform the scraped data into a suitable format for analysis.
4. **Data Insertion into Database**: Insert the transformed data into a database for storage.
5. **Data Preprocessing**: Clean and preprocess the data for further analysis.
6. **Model Building**: Build predictive models using machine learning techniques to predict flight fares.
7. **Cloud Setup**: Set up a cloud environment to deploy the model.
8. **Pushing App to Cloud**: Deploy the application to the cloud for scalability.
9. **Application Start**: Start the application to begin accepting user inputs and processing data.
10. **Data from User**: Collect flight details and preferences from the user.
11. **Data Validation**: Validate the user input data.
12. **Model Call for Prediction**: Use the predictive model to estimate flight fares based on the user inputs.
13. **Saving Output in Database**: Save the predicted fare and related information in the database.
14. **Fare Recommendation**: Provide fare recommendations to the user.

2.4 FURTHER IMPROVEMENTS

Continuously update fare predictions based on real-time data.

Improve model accuracy by incorporating user feedback on fare predictions.

Integrate more data sources to improve prediction accuracy.

Develop a mobile application for easier access to fare predictions on the go.

5. Technical Requirements

This document addresses the requirements for predicting flight fares and providing fare recommendations to users based on their input. The system should efficiently collect, process, and analyze flight fare data to offer accurate predictions and suggestions. The key components and requirements are:

- **Data Collection Tools**: Tools for web scraping to gather flight fare data from multiple sources.
- **Database**: A robust database system to store and manage the collected flight fare data.
- **Machine Learning Libraries**: Libraries such as scikit-learn, TensorFlow, or similar, for building and training predictive models.
- **Web Framework**: A framework like Flask or Django to develop the user interface and handle user interactions.
- **Cloud Services**: Cloud infrastructure (e.g., AWS, Azure) for deploying the machine learning models and web application to ensure scalability and high availability.
- **Data Preprocessing Tools**: Tools for cleaning, transforming, and normalizing the data to prepare it for model training.
- **User Interface**: A web-based interface for users to input their flight details and receive fare predictions and recommendations.

6. Data Requirements

The data requirements for a flight fare prediction system are crucial to ensure accurate and reliable predictions. Here are the detailed requirements:

Data Collection

Sources: Collect data from multiple sources such as airline websites, travel booking platforms, and third-party APIs.

Data Points: The data should include the following fields:

Flight Details: Airline, flight number, departure and arrival cities, departure and arrival times, duration, layovers.

Fare Information: Base fare, taxes, fees, total fare.

Booking Class: Economy, premium economy, business, first class.

Date Information: Booking date, travel date, day of the week, seasonality.

Data Volume

Historical Data: A large volume of historical fare data is essential. At least 1,000 records per route should be targeted to ensure robustness in model training.

Class Balance: Ensure a balanced dataset across different fare classes and travel seasons.

Data Format

CSV/JSON: Data should be stored in a structured format such as CSV or JSON to facilitate easy parsing and processing.

Data Preprocessing

Cleaning: Handle missing values, remove duplicates, and correct inconsistent data entries.

Normalization: Normalize fare amounts and other numerical fields to ensure consistency.

Feature Engineering: Create new features such as fare per mile, day of the week, and month to capture trends and seasonality.

Data Storage

Database: Use a relational database like PostgreSQL or MySQL to store the collected data. Ensure the database is optimized for fast read and write operations.

Data Types

Numeric Data: Fare amounts, distance between cities, duration of flights.

Categorical Data: Airline names, booking class, city names.

Date/Time Data: Departure and arrival times, booking date, travel date.

Data Quality

Accuracy: Ensure that the fare data collected is accurate and up-to-date.

Completeness: Ensure that all relevant fields are populated and no critical information is missing.

Consistency: Ensure that data formats are consistent across different sources.

Data Privacy and Compliance

User Data: If collecting user-specific data (e.g., booking preferences), ensure compliance with data privacy regulations like GDPR or CCPA.

Security: Implement secure data storage and transmission practices to protect sensitive information.

2.7 Tools used

Python programming language and frameworks such as NumPy, Pandas, Scikit-learn, HTML, CSS,Matplotlib, VsCode, jupyter, AWS and Flask are used to build the whole model.

- **NumPy**: Numerical operations on arrays
- **Pandas**: Data manipulation and analysis
- **Scikit-learn**: Machine learning in Python
- **HTML**: Web page structure definition
- **CSS**: Styling web page elements
- **Matplotlib**: Data visualization in Python
- **VSCode**: Code editor for development
- **Jupyter:** Interactive notebooks for coding
- **AWS**: Cloud computing services provider
- **Flask**: Web framework for Python

8. Constraints

The flight fare prediction system must be user-friendly, as automated as possible, and users should not be required to know any of the technical workings.

9. Assumptions

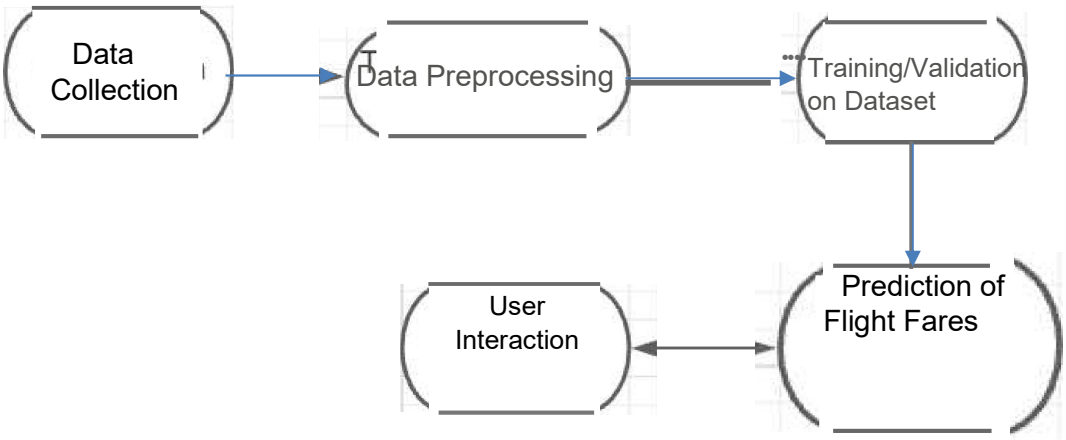
The main objective of the project is to predict flight fares based on historical and real-time data. Machine learning models will be used to provide fare predictions based on the input data. It is also assumed that all components of this project will work seamlessly together as expected.

3 Design Details

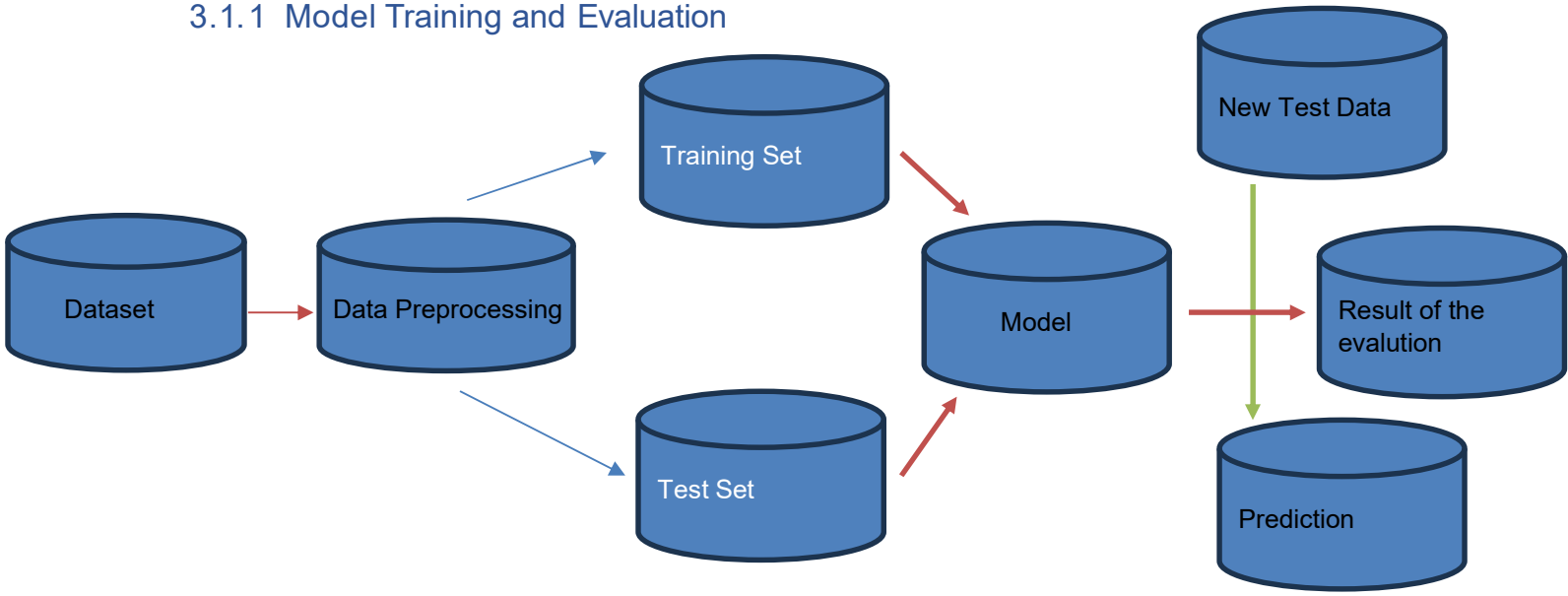
3.1 Process Flow

For predicting flight fares, we will use a machine learning-based model. Below is the process flow diagram as shown below.

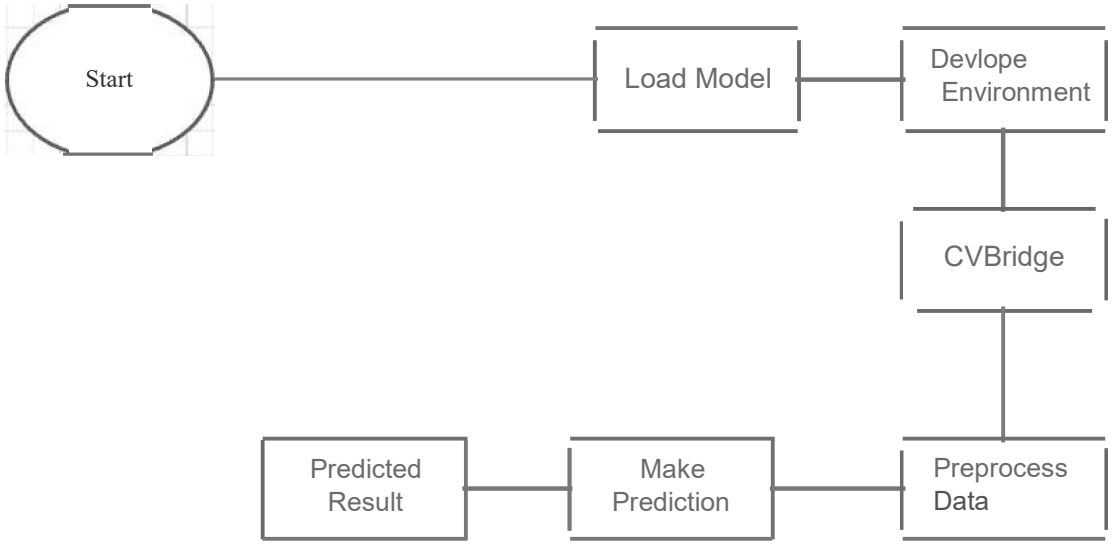
Proposed methodology



3.1.1 Model Training and Evaluation



3.1.2 Deployment Process



2. Event log

The system should log every event so that the user will know what process is running internally.

Initial Step-By-Step Description:

1. The System identifies at what step logging required
2. The System should be able to log each and every system flow.
3. Developer can choose logging method. You can choose database logging/ File logging as well.
4. System should not hang even after using so many loggings. Logging just because we can easily debug issues so logging is mandatory to do.

3. Error Handling

Should errors be encountered, an explanation will be displayed as to what went wrong? An error will be defined as anything that falls outside the normal and intended usage.

4 Performance

The flight fare prediction solution is used for predicting accurate flight fares for users. Whenever the system predicts flight fares, it should be as accurate as possible to provide reliable information to users. This ensures that users can make informed decisions about their travel plans. Additionally, model retraining is very important to improve the performance and accuracy of the predictions over time.

1. Reusability

The code written and the components used should have the ability to be reused with no problems.

2. Application Compatibility

The different components for this project will be using Python as an interface between them. Each component will have its own task to perform, and it is the job of the Python to ensure proper transfer of information.

3. Resource Utilization

When any task is performed, it will likely use all the processing power available until that function is finished.

4. Deployment

Microsoft Azure

Google Cloud

amazon web services

5 Dashboards

Dashboards will be implemented to display and indicate certain KPIs and relevant indicators for the unveiled problems that if not addressed in time could cause catastrophes of unimaginable impact.

As and when, the system starts to capture the historical/periodic data for a user, the dashboards will be included to display charts over time with progress on various indicators or factors.

1. KPIs (Key Performance Indicators)

Key indicators displaying a summary of the flight fare prediction performance:

1. Prediction Accuracy
- Measure the accuracy of the fare predictions compared to actual fares.

2. User Satisfaction
-Collect and analyze user feedback on the fare prediction service.

3. Response Time
- Time taken to process user inputs and provide fare predictions.

4. Number of Users
- Track the number of users accessing the fare prediction service.

5. Model Update Frequency
- Frequency of updates to the prediction model with new data.

6. Error Rate
- Rate of incorrect fare predictions provided to users.

7. Scalability
- Ability to handle increasing numbers of users and data without performance degradation.

8. System Uptime
-Measure the availability and reliability of the fare prediction service.

9. Data Coverage
- Extent of flight routes and airlines covered by the prediction model.

10. User Engagement
- Monitor user interaction and repeat usage of the service.

6 Conclusion

The designed Flight Fare Prediction system will predict flight fares based on various historical and real-time data used to train our algorithm. This allows us to provide users with accurate fare predictions and recommendations, helping them make informed decisions and potentially save money on their travel plans.

7 References

1. https://github.com/BiswajitJena2002/Flight_Booking
2. Google.com for images of Software.