**Low-Level Design Document (LLD)**

Flight Fare Prediction Project

Project Overview

Project Name:

Flight Fare Prediction

Objective:

Develop a model that accurately predicts flight fares using Random Forest Regression, detailing data handling processes, feature engineering, model training, evaluation, and deployment.

Detailed Design

1. Data Handling

Data Sources:

* APIs from online travel agencies (e.g., Skyscanner, Kayak).
* Historical flight fare datasets from airline databases.
* Additional datasets (weather, holiday schedules).

Data Ingestion:

* APIs: Use Python libraries like requests or specialized API clients.
* Databases: Use SQL queries for data extraction from relational databases.

Data Storage:

* Store raw data in CSV files or databases (e.g., SQLite, PostgreSQL).
* Use pandas DataFrames for in-memory data manipulation.

Data Preprocessing Steps:

Data Cleaning:

* Remove duplicates.
* Handle missing values (imputation or removal).
* Correct data inconsistencies (e.g., standardize date formats).

Feature Engineering:

* Extract temporal features (day of the week, month, season).
* Calculate flight duration in minutes.
* Encode categorical variables (e.g., airlines, source, destination) using one-hot encoding.
* Create additional features such as holiday indicators and weather conditions.

Data Transformation:

* Normalize or standardize numerical features.
* Split data into training and testing sets (e.g., 80/20 split).

2. Model Training

Model Selection:

* Algorithm: Random Forest Regression.
* Libraries: scikit-learn for model implementation and training.

Model Training Steps:

Data Splitting:

* Split the preprocessed data into training and testing sets.
* Ensure that the split maintains the distribution of fare prices.

Training the Model:

* Train the Random Forest model using the training dataset.
* Utilize cross-validation to ensure model generalization and prevent overfitting.

Model Evaluation:

* Evaluate the model using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared.
* Validate the model on the testing dataset to assess performance.

3. Model Evaluation and Validation

Evaluation Metrics:

* Mean Absolute Error (MAE): Measures average magnitude of errors.
* Mean Squared Error (MSE): Measures average of the squares of the errors.
* R-squared: Indicates the proportion of variance explained by the model.

Validation Techniques:

* Cross-Validation: Use k-fold cross-validation to evaluate the model performance.
* Holdout Validation: Use a separate validation dataset if available.

4. Model Deployment

* Deployment Environment:
* Web Framework: Flask for creating a web API.

Deployment Steps:

API Development:

* Develop RESTful API endpoints using Flask.
* Create endpoints for model prediction (e.g., /predict).

Monitoring and Logging:

* Implement logging to track API requests and responses.
* Monitor model performance and resource usage.

5. Tools and Resources

Development Tools:

* IDE: Visual Studio Code.
* Version Control: Git and GitHub for code management and collaboration.

Data Handling Libraries:

* pandas: Data manipulation and analysis.
* NumPy: Numerical computing and array operations.

Visualization Libraries:

* Seaborn: Statistical data visualization.
* Matplotlib: Comprehensive plotting library.

Machine Learning Libraries:

* scikit-learn: Machine learning algorithms and utilities.

Web Development:

* Flask: Web framework for building APIs.
* HTML: Markup language for creating web pages and interfaces.

Documentation and Notebooks:

* Jupyter: Interactive notebooks for data exploration and documentation.

Conclusion

This Low-Level Design document outlines the technical details of the Flight Fare Prediction project, focusing on data handling, feature engineering, model training, evaluation, and deployment. By following these detailed steps and utilizing the specified tools and libraries, the development team can implement an effective and efficient flight fare prediction system.