Architecture Design Document (AD)

Flight Fare Prediction Project

Project Overview

Project Name:

Flight Fare Prediction

Objective:

Design the internal structure of the Flight Fare Prediction system, providing detailed class diagrams, method specifications, and relationships to guide the development process.

System Architecture

1. High-Level Overview

The Flight Fare Prediction system comprises the following components:

Data Ingestion

Data Preprocessing

Feature Engineering

Model Training

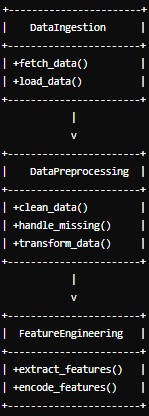
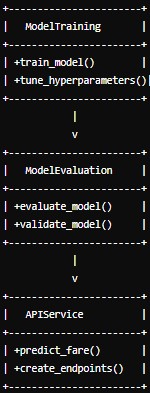
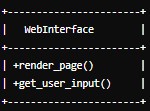
Model Evaluation

API Development

Web Interface

2. Class Diagram

Below is a high-level class diagram representing the main classes and their relationships:

3. Detailed Class Specifications

3.1 DataIngestion

fetch\_data()

Description: Retrieves flight fare data from various sources (APIs, web scraping).

Parameters: None

Returns: Raw data in a suitable format (e.g., DataFrame).

load\_data()

Description: Loads data from local or remote storage.

Parameters: file\_path (str)

Returns: DataFrame

3.2 DataPreprocessing

clean\_data()

Description: Cleans the raw data by removing duplicates and correcting inconsistencies.

Parameters: raw\_data (DataFrame)

Returns: Cleaned DataFrame

handle\_missing()

Description: Handles missing values through imputation or removal.

Parameters: data (DataFrame)

Returns: DataFrame with missing values handled

transform\_data()

Description: Transforms data types and formats as required.

Parameters: data (DataFrame)

Returns: Transformed DataFrame

3.3 FeatureEngineering

extract\_features()

Description: Extracts relevant features from the data (e.g., day of the week, flight duration).

Parameters: data (DataFrame)

Returns: DataFrame with new features

encode\_features()

Description: Encodes categorical variables into numerical formats.

Parameters: data (DataFrame)

Returns: DataFrame with encoded features

3.4 ModelTraining

train\_model()

Description: Trains the Random Forest Regression model.

Parameters: training\_data (DataFrame), target (Series)

Returns: Trained model

tune\_hyperparameters()

Description: Tunes model hyperparameters for optimal performance.

Parameters: training\_data (DataFrame), target (Series)

Returns: Best hyperparameters

3.5 ModelEvaluation

evaluate\_model()

Description: Evaluates the model using metrics like MAE, MSE, and R-squared.

Parameters: model, test\_data (DataFrame), test\_target (Series)

Returns: Evaluation metrics (dict)

validate\_model()

Description: Validates model performance using cross-validation.

Parameters: model, data (DataFrame), target (Series)

Returns: Cross-validation scores (dict)

3.6 APIService

predict\_fare()

Description: Predicts flight fare using the trained model.

Parameters: input\_features (dict)

Returns: Predicted fare (float)

create\_endpoints()

Description: Creates API endpoints for model prediction.

Parameters: None

Returns: None

3.7 WebInterface

render\_page()

Description: Renders the web page for user interaction.

Parameters: template\_name (str), context (dict)

Returns: Rendered HTML page

get\_user\_input()

Description: Collects user input from the web interface.

Parameters: request (Request object)

Returns: User input (dict)

Program Specifications

Data Flow

Data Ingestion: Fetch and load data into the system.

Data Preprocessing: Clean, handle missing values, and transform data.

Feature Engineering: Extract and encode features from the preprocessed data.

Model Training: Train and tune the Random Forest model.

Model Evaluation: Evaluate and validate the trained model.

API Development: Develop API for real-time predictions.

Web Interface: Create a web interface for user interaction and fare prediction.

API Endpoints

/predict

Method: POST

Description: Takes input features and returns the predicted fare.

Parameters: JSON object with input features.

Returns: JSON object with the predicted fare.

Web Interface

Home Page: Displays input form for users to enter flight details.

Result Page: Displays the predicted fare based on user input.

Deployment Specifications

Environment: Use Docker for containerization.

Cloud Service: Deploy on AWS, GCP, or Azure.

CI/CD Pipeline: Implement CI/CD using GitHub Actions or similar tools.

Conclusion

This Architecture Design document provides a detailed blueprint for the Flight Fare Prediction system, outlining the internal structure, class diagrams, and method specifications. This document serves as a guide for developers to implement the system effectively and ensure all components work seamlessly together.