Architecture Design

FLIGHT FAIR PREDICTION

Written by	Biswajit Jena
!Version	1.0
Date	25-07-2024

Document Control

Change Record:

Version	Date	Author	Comments
1.0	25-07-2024	Biswajit Jena	

Approval Status:

Version	Review date	Reviewed By	Approved By	Comments

Index

Abstract

The goal of this project is to predict flight prices using machine learning. We will process a dataset containing flight details, perform exploratory data analysis (EDA), preprocess the data, build predictive models, and deploy a web application for real-time flight price predictions.

1. Introduction

1.1What is Architecture Design?

The goal of Architecture Design (AD) is to provide the internal design of the actual program code for the `Flight Price Prediction System`. It describes the class diagrams with methods, relationships between classes, and program specifications. AD details the modules so the programmer can directly code from the document.

1.2 Scope

AD follows a step-by-step refinement process for designing data structures, software architecture, source code, and performance algorithms. The data organization is defined during requirement analysis and refined during data design work.

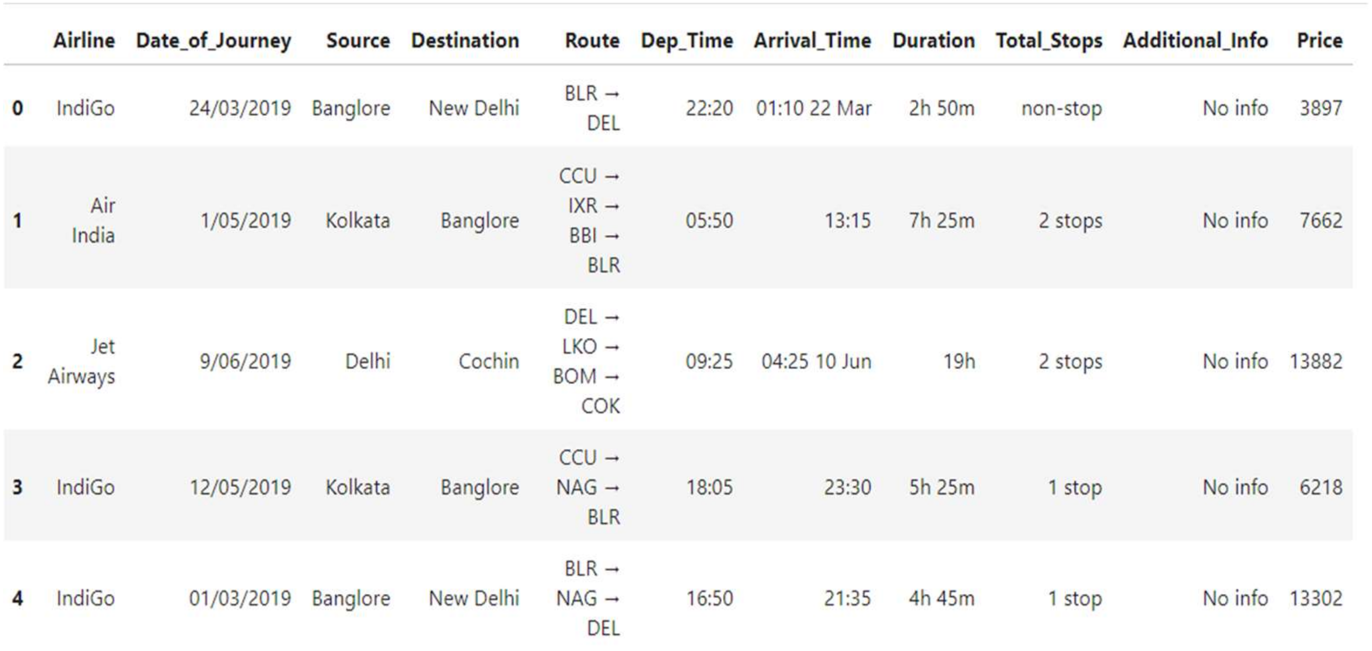
1.3 Constraints

The system predicts flight prices based on the given features such as airline, source, destination, total stops, date and time of journey, and additional info.

2. Technical Specification

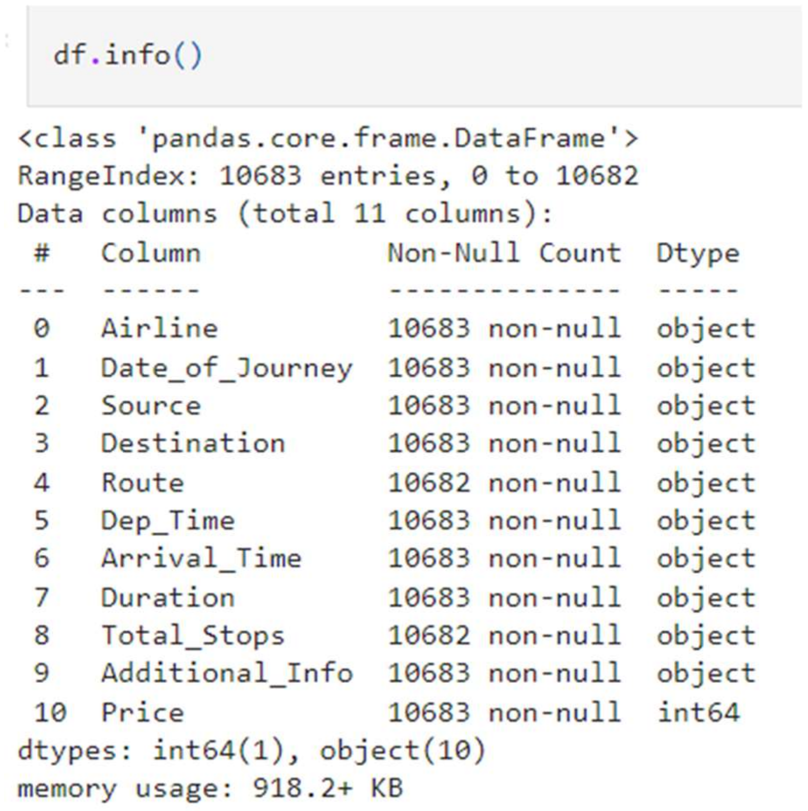
2.1 Dataset

The dataset for flight price prediction includes flight details such as airline, source, destination, total stops, date and time of journey, and flight price. The dataset is preprocessed to handle null values, encode categorical variables, and generate new features such as journey day, month, and duration in minutes.



	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302

The data set consists of various data types from integer , floating and object as shown in Fig.



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Airline          10683 non-null  object
 1   Date_of_Journey  10683 non-null  object
 2   Source           10683 non-null  object
 3   Destination      10683 non-null  object
 4   Route            10682 non-null  object
 5   Dep_Time         10683 non-null  object
 6   Arrival_Time     10683 non-null  object
 7   Duration         10683 non-null  object
 8   Total_Stops      10682 non-null  object
 9   Additional_Info  10683 non-null  object
 10  Price            10683 non-null  int64 
dtypes: int64(1), object(10)
memory usage: 918.2+ KB
```

depth knowledge about the subject of interest and provides insights into the problem. But caution should be observed
with respect to data as it may contain null values, or redundant values, or various types of ambiguity, which also dema nds pre-processing of data. The dataset should therefore be explored as much as possible.
Various factors important by statistical means like mea n, standard deviation, median, count of values and maximum value, etc. are shown below for numerical attributes.

Preprocessing of this dataset includes doing analysis on the independent variables like checking for null values in each column and then replacing or filling them with supported appropriate data types so that analysis and model fitting is not hindered from their way to accuracy. Shown above are some of the representations obtained by using Pandas tools which tell about variable count for numerical columns and model values for categorical columns . Maximum and minimum values in numerical columns, along with their percentile values for median, play an important factor in deciding which value to be chosen at priority for further exploration tasks and analysis. Data types of different columns are used further in label processing and a one-hot encoding scheme during the model building.

2.2 Logging

The system logs every user activity and system flow for debugging and monitoring purposes. Developers can choose logging methods, including database logging, without affecting system performance.

•The system identifies at which step logging require.
•The system should be able to log each and every system flow.
•Developers can choose logging methods. Also can choose database logging.
•The system should be not be hung even after using so much logging. Logging just because we can easily debug issuing so logging is mandatory to do.

2.3 Data Preprocessing

Preprocessing includes handling missing values, encoding categorical variables, and feature engineering. This ensures that the dataset is ready for model building.

2.4 Deployment

For the hosting of the project, we will use GCP.

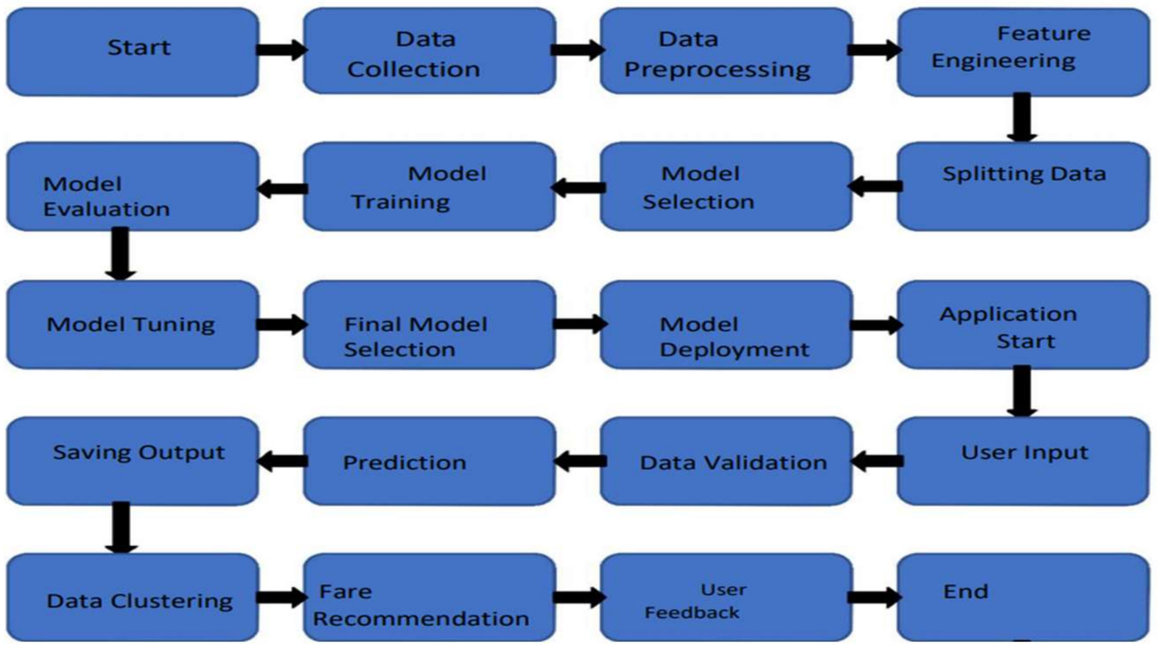
3. Technology Stack

- **Front End**: HTML/JavaScript
- **Backend**: Python
- **Deployment**: GCP

4. Proposed Solution

Perform EDA to find important relationships between features and use machine learning algorithms to predict flight prices. The user inputs flight details through a web application, and the system processes these inputs to provide a price prediction.

5 Architecture detail



5.1 Data Gathering

Data source: https://www.kaggle.com/datasets/nikhilmittal/flight-fare-prediction-mh Train

and Test data are stored in .csv format.

5.2 Raw Data Validation

Validate the data by checking for missing values and zero standard deviations. Remove or fill in these values to ensure they do not affect the prediction model.

5.3 Data Transformation

Before sending the data into the database, data transformation is required so that data are Transform the data into a suitable format for model building. This includes filling missing values with appropriate data to ensure completeness. Categorical features are encoded into numerical values to be compatible with machine learning algorithms. Additionally, features such as the journey day and month are extracted from the journey date, and flight duration is converted into minutes for consistency. The transformation process ensures that the data is structured and clean, making it ready for the next steps in the pipeline.

5.4 New Feature Generation

Generate new features from existing ones to improve model performance. For instance, extracting the day and month from the journey date helps capture seasonal trends in flight prices. Converting flight duration into minutes standardizes the format for easier analysis. Creating binary or ordinal features from categorical data can also reveal hidden patterns. These new features provide additional context and granularity, enabling the model to make more accurate predictions.

5.5 Data Preprocessing

Preprocess the data to handle inconsistencies and prepare it for model training. This involves dealing with missing values by either imputing them or removing affected rows. Categorical variables are encoded using techniques like one-hot encoding to convert them into numerical format. Numerical features are scaled to ensure uniformity across different magnitudes. This step is crucial as it ensures that the data fed into the model is clean, well-structured, and free from biases that could affect the model's performance.

5.6 Feature Engineering

Perform feature engineering to enhance the dataset's predictive power. This includes creating new derived features that can provide additional insights into the relationships between variables. One-hot encoding is applied to categorical variables to avoid introducing ordinal relationships where none exist. Interaction terms between features are created to capture complex dependencies. Feature selection techniques are used to identify and retain the most relevant features, which helps in reducing overfitting and improving the model's generalization capabilities.

5.7 Parameter Tuning

Tune model parameters using GridSearchCV or RandomizedSearchCV to optimize the model's performance. This process involves defining a range of possible values for each hyperparameter and systematically testing each combination to find the best one. Cross-validation is used to evaluate the performance of each parameter set, ensuring that the selected parameters generalize well to unseen data. Proper parameter tuning can significantly enhance model accuracy and robustness.

5.8 Model Building

Build and evaluate multiple models to identify the best-performing algorithm. Models such as Linear Regression, Random Forest, and Gradient Boosting are trained and tested. Each model's performance is evaluated using metrics like RMSE and R^2 score. The best-performing model is selected based on its predictive accuracy and ability to generalize to new data. This step ensures that the most suitable model is chosen for deployment.

5.9 Model Saving

Save the best model using the joblib library in `.joblib` format for efficient serialization. This allows the model to be easily loaded and used in the production environment. Saving the model ensures that the training process does not need to be repeated, saving time and computational resources. It also enables version control, allowing developers to track and manage different iterations of the model..

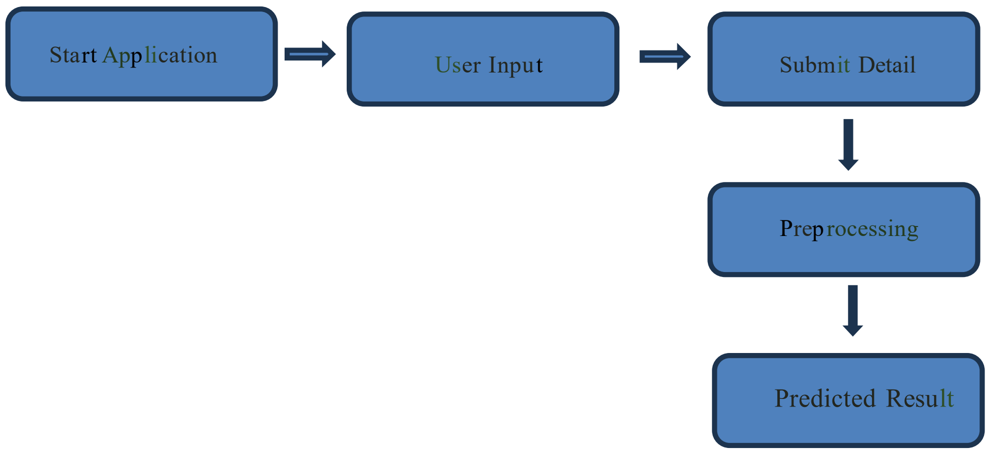
5.10 GitHub

The whole project directory will be pushed into the GitHub repository.

5.11 Deployment

Deploy the project on GCP, using the GitHub repository as the source. This involves setting up a cloud environment and configuring the web server to host the application. The deployment process ensures that the flight price prediction model is accessible to users via a web interface. Continuous deployment practices can be implemented to automatically update the application whenever changes are pushed to the GitHub repository, ensuring that the latest version is always live.

6. User Input / Output Workflow.


Start Application
User Input
Submit Detail
Preprocessing
Predicted Result