Flight Fare Price Prediction: Architecture

Team members:

- Yash Keshari
- Md Ehsanul Haque Kanan

Date: 25/10/2023

Document Version Control

Date Issued	Version	Description	Author
19/10/2023	1	Initializing - V 0.1	Yash Keshari
23/10/2023	2	Major changes – V 0.2	Yash Keshari

Contents

1. Introduction:

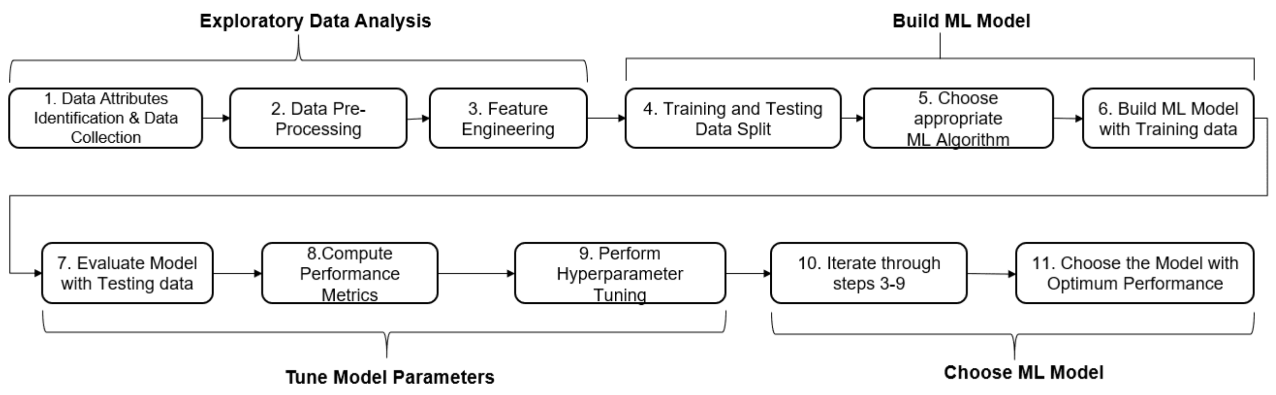
The Flight Fare Price Prediction System represents a comprehensive solution designed to simplify and enhance the process of planning air travel. This Architecture Document outlines the structural and design aspects of the system, providing valuable insights into how the various components work together to deliver accurate fare predictions.

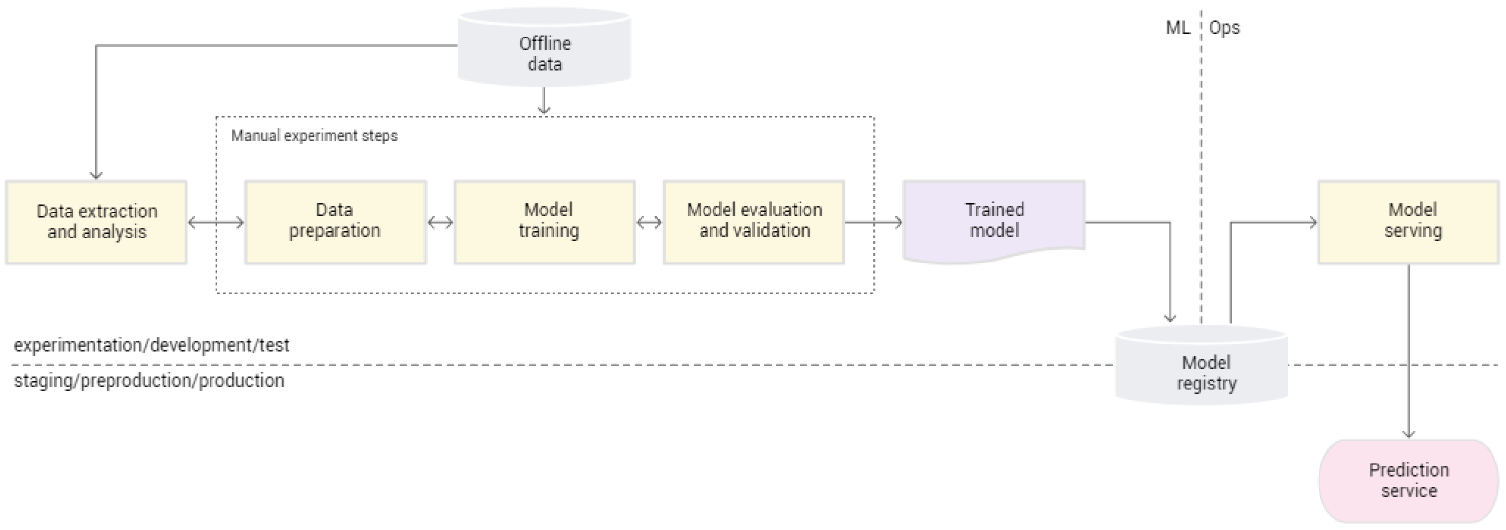
2. Scope:

The scope of this document extends to the architecture and design of the Flight Fare Price Prediction System. It outlines the high-level components, data flow, interfaces, technology stack, quality attributes, and deployment strategies. While this document provides a comprehensive view, it is not intended to replace detailed technical documentation, which may be produced separately.

3. Architecture Diagram

The architecture of the Flight Fare Price Prediction project is designed to deliver accurate and real-time fare estimates to travelers. The system collects data from various sources, processes it for modeling, and employs a Random Forest Regressor to predict fare prices. Users interact with the system through a user-friendly front-end, and the entire solution is deployed on a scalable cloud platform which is AWS for accessibility and responsiveness. This architecture ensures travelers receive up-to-date and precise fare predictions, enhancing their booking experience.


Exploratory Data Analysis
Build ML Model
1. Data Attributes Identification & Data Collection
2. Data Pre-Processing
3. Feature Engineering
4. Training and Testing Data Split
5. Choose appropriate ML Algorithm
6. Build ML Model with Training data
7. Evaluate Model with Testing data
8.Compute Performance Metrics
9. Perform Hyperparameter Tuning
10. Iterate through steps 3-9
11. Choose the Model with Optimum Performance
Tune Model Parameters
Choose ML Model



4. **Architecture Description:**

- ### Data Description:

The project leverages a comprehensive dataset that encompasses various flight-related attributes, including airline details, routes, historical pricing data, total stops, duration and additional factors influencing fare prices.

- ### Data Ingestion into the Artifact:

Processed data is ingested into the project's artifact repository for efficient storage and retrieval. This repository is a crucial resource for model training and user interactions.

- ### Export Data from Artifact:

Historical data can be exported from the artifact repository for model training and validation, ensuring the model is updated with the latest information.

- ### Data Pre-processing:

Data pre-processing encompasses tasks like scaling features, handling outliers, and encoding categorical variables to improve the quality of data used for model training.

- ### Data Transformation:

The collected data undergoes transformation to prepare it for modeling. This stage includes tasks like converting data types, and creating engineered features to enhance predictive accuracy.
Feature engineering is performed to create and select relevant features that improve the model's ability to predict fare prices accurately.

- ## Model Building:

The central component of the architecture is the machine learning model development. A machine learning algorithm, such as the Random Forest Regressor, is employed to construct a predictive model that establishes relationships between flight features and fare prices.

- ## Model Evaluation:

The trained model is evaluated for its performance using various metrics, ensuring it meets the required accuracy standards. R2-score is employed in our case.

- ## Model Export:

Once validated, the model is exported for deployment, making it accessible to users for real-time fare predictions. We have used Joblib for this task.

- ## Front-end Development:

A user-friendly front-end application, such as a web or mobile app, is developed using "Streamlit" to smoothen user interactions. This interface enables users to input their flight details and receive fare predictions.

- ## Deployment (on AWS):

The entire system, including the predictive model and user interface, is deployed on a cloud platform AWS EC2 for scalability and accessibility. This deployment allows users to access the service from various devices and locations.

5. Technology Stack:

- **Language**: Python
- **Preprocessing**: Numpy, Pandas
- **Machine Learning:** Scikit-learn
- **Data Processing:** NumPy, Pandas
- **Visualization:** Matplotlib, Seaborn
- **Cloud:** AWS EC2
- **Frontend:** Streamlit library
- **IDE**: VS-Code

6. Project Structure:

```
|- artifacts/
| (Include trained models or other important artifacts)
|
|- notebooks/
|  |- notebook.ipynb
|   (Your Jupyter Notebook for the project)
|  |- data/
|    (Data files or datasets)
|
|- logs/
  (Log files or logs directory)
```

```
|- src/
|  |- logger.py
|    (Logging utilities)
| |- exception.py
|    (Custom exception handling)
|  |- utils.py
|    (Utility functions)
|  |- __init__.py
|
|  |- components/
|    - __init__.py
|    - data_ingestion.py
|      (Data loading functions)
|    - data_transformation.py
|      (Data preprocessing and feature engineering)
|    - model_training.py
|     (Machine learning model training)
|
|  |- pipelines/
|    - __init__.py
|    - prediction_pipeline.py
|      (Fare prediction pipeline)
|    - training_pipeline.py
|      (Model training pipeline)
|
|- visuals/
|  |- _image.jpg
|    (Images, plots, or visualization files)
|
|- requirements.txt
|  (List of project dependencies)
|
|- setup.py
|  (Setup script for the project)
|
|
|- app.py
|  (Main application script)
|
|- .gitignore
     (Specify files or directories to ignore in version control)
```

7. References:

- Google Images
- Google
- https://www.datarevenue.com/en-blog/machine-learning-project-architecture
- Pw documentation.