FLIGHT FARE PREDICTION

Objective

The primary objective of the flight fare prediction project is to develop an accurate and user-friendly model that forecasts airline ticket prices using historical data and machine learning. This tool aims to help users save money by identifying the best times to purchase tickets, analyze factors influencing fare changes, and continuously improve prediction accuracy. Additionally, it seeks to provide travel businesses with insights to optimize pricing strategies, ultimately enhancing customer satisfaction and offering a competitive edge in the travel industry.

Benefits:

1) Cost Savings
2) Informed Decisions
3) Convenience
4) Trend Awareness
5) Competitive Advantage
6) Customer Satisfaction
7) Resource Optimization

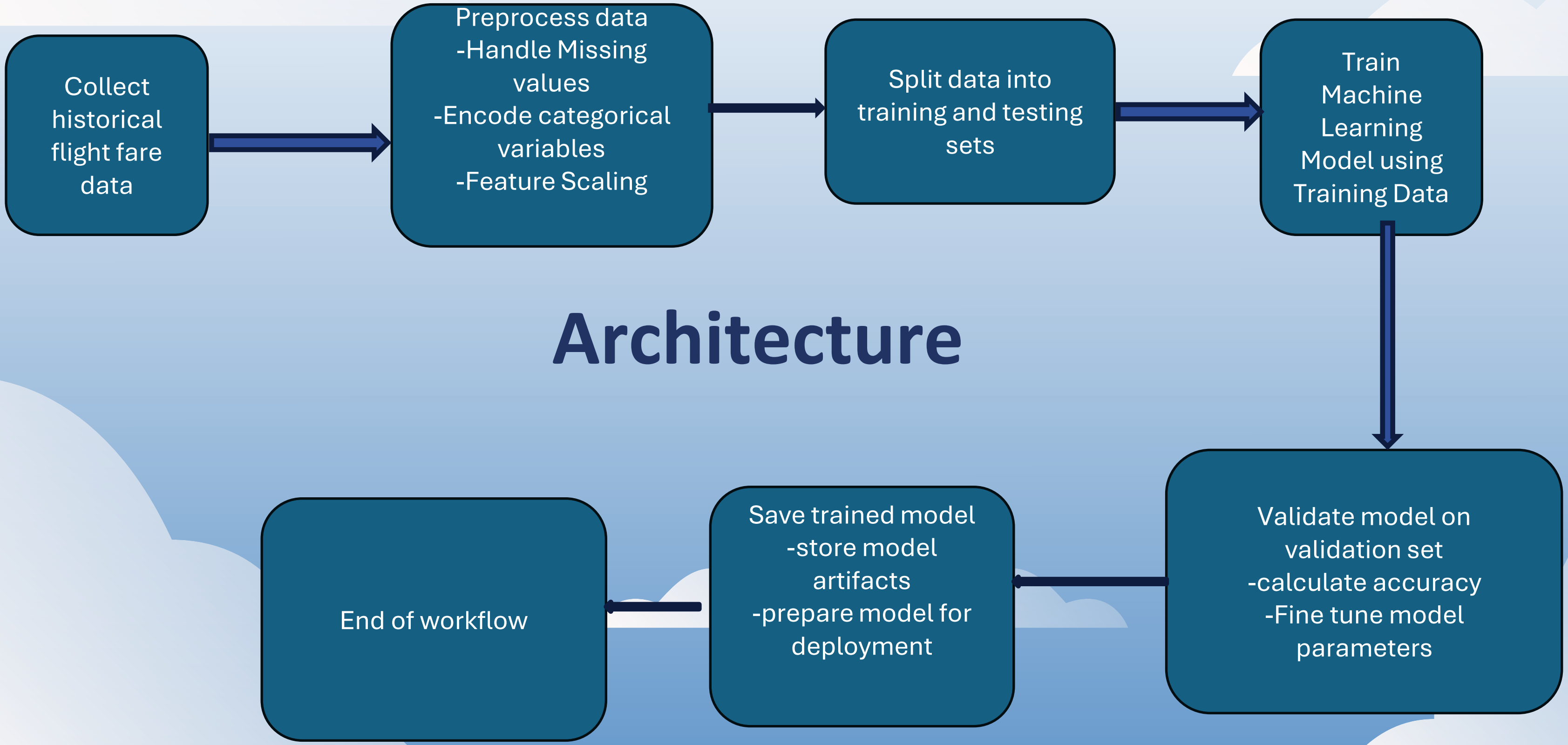
Data sharing agreement

Sample file name	:airline_data.xlsx
Length of date stamp	:8 digits(yyyymmdd)
Length of time stamp	:6 digits(hhmmss)
Number of Columns	: 11

Column names and data type:

- Airline (Object)
- Date_of_Journey (Date)
- Source (Object)
- Destination (Object)
- Route (Object)
- Arrival_Time (Time)
- Dep_Time (Time)
- Duration (Object)
- Total_Stops (Object)
- Additional Info (Object)
- Price (Integer)

Architecture


Collect historical flight fare data
Preprocess data
-Handle Missing values
-Encode categorical variables
-Feature Scaling
Split data into training and testing sets
Train Machine Learning Model using Training Data
Validate model on validation set
-calculate accuracy
-Fine tune model parameters
Save trained model
-store model artifacts
-prepare model for deployment
End of workflow

Data Validation& Data Transformation

Name Validation: Check file names against DSA regex. Move to "Good_Data_Folder" if valid, else to "Bad_Data_Folder".

Number of Columns: Validate against schema. Move to "Bad_Data_Folder" if mismatched.

Name of Columns: Match against schema. Move to "Bad_Data_Folder" if not identical.

Data Type of Columns: Validate data types against schema. Move to "Bad_Data_Folder" if incorrect.

Null Values: Discard files with all NULL values. Move to "Bad_Data_Folder".

Data Transformation: Convert data types and preprocess. Move to "Good_Data_Folder" for further processing.

Data insertion in database

Table Creation:

The script sets up the Cassandra connection using environment variables for authentication. It then executes a CQL query to create a table named "trainingdata" if it doesn't already exist in the keyspace "flightprice". The table has several columns such as id, date_of_journey, airline, destination, duration, price, etc., with their respective data types.

Insertion of Data:

After creating the table, the script executes another CQL query to insert sample data into the "trainingdata" table for testing purposes. The inserted data includes various attributes such as the date of journey, airline, destination, departure and arrival times, price, route, etc. This sample data insertion demonstrates how actual flight data could be populated into the database table.

Model Training

The model training process involves preparing the dataset, selecting appropriate features, and training a machine learning model to predict flight fares. The dataset is preprocessed to handle missing values, convert categorical data into numerical form, and normalize or scale numerical features. Once the data is ready, it is split into training and testing sets to evaluate the model's performance. Various algorithms, such as linear regression, decision trees, or gradient boosting, can be used to train the model. The trained model is then evaluated using metrics like mean absolute error, mean squared error, or R-squared to ensure it accurately predicts flight fares. This trained model can be fine-tuned and optimized to improve its predictive accuracy before deployment.

Model Selection

Two regression models, Random Forest and XGBoost, were trained and compared for flight fare prediction. The data was first split into training and testing sets, and outliers in the training data were identified and removed using IsolationForest. Both models were then trained on the cleaned data and evaluated based on their Mean Absolute Error (MAE) on the test set. XGBoost provided better accuracy, making it the preferred model for this task.

Prediction

The testing files are shared in batches, and the same validation operations, data transformation, and data insertion procedures are applied to them. The accumulated data from the database is then exported in CSV format for prediction.

We perform data preprocessing techniques on the exported data to ensure it is clean and ready for prediction. The XGBoost model, which was determined to be the most accurate during the training phase, is loaded for making predictions.

Q & A

Q1) What’s the source of data?

The data for training is provided by the client in multiple batches, and each batch contains multiple files.

Q2) What was the type of data?

The data was a combination of numerical and categorical values.

Q3) What’s the complete flow you followed in this project?

Refer to slide 5 for a detailed understanding of the complete project flow.

Q4) After file validation, what do you do with incompatible files or files that didn’t pass the validation?

Files that do not pass validation are moved to the Archive Folder. A list of these files is shared with the client, and the "Bad_Data_Folder" is removed to ensure only valid data is processed.

Q & A

Q5) How do you handle missing or null values in the data?

Missing or null values are handled during the data preprocessing stage. Depending on the extent and nature of the missing values, we either impute them with appropriate values or remove the affected rows or columns.

Q6) Which models did you use for prediction and why?

We used the Random Forest and XGBoost models for prediction. After comparing their performance, we found that XGBoost provided better accuracy and was thus chosen as the final model for predicting flight fares.

Q7) How do you ensure the accuracy of your predictions?

We ensure accuracy by splitting the data into training and testing sets, training the model on the training set, and evaluating its performance on the testing set using metrics like Mean Absolute Error (MAE). We also perform cross-validation to further validate the model's performance.

Q & A

Q8) What do you do with the prediction results?

The prediction results are saved in a CSV format and shared with the relevant stakeholders. This allows for easy analysis and integration into their systems for decision-making purposes.

Q9) How do you handle outliers in the data?

Outliers are detected using the Isolation Forest algorithm. Once detected, they are removed from the training set to improve the model's performance.

Q10) How do you process and use the data for prediction?

The accumulated data from the database is exported in CSV format, preprocessed, and then used for prediction. The trained XGBoost model is loaded, and predictions are made on the preprocessed data. The results are then saved and shared as needed.

THANK YOU