**Machine Learning - I**

**Term Project**

**Report**

**Name: Wajeeha Parker**

**ERP: 27505**

**Introduction**

The goal of this project is to predict the price of flights based on various features such as airline, departure and arrival cities, number of stops, departure and arrival times, duration, days left until departure, flight code, and flight number.

Several algorithms, Gradient Boosting, XGBoost, and LightGBM, were utilised to complete this challenge. The model also utilizes an ensemble approach by combining three different regression algorithms: Gradient Boosting, XGBoost, and LightGBM. These algorithms are trained on a training dataset consisting of flight data, and their performance is evaluated using metrics such as R2 score and mean squared error (MSE) on a separate test dataset.

The Voting Regressor is then constructed by combining the individual models. This ensemble model aims to leverage the strengths of each base model to improve the overall prediction accuracy.

The web application, built with Streamlit, allows users to input their flight details and obtain a predicted price based on the trained model. The user interface provides options to select the airline, source and destination cities, number of stops, departure and arrival dates and times, flight code, and flight number. Upon clicking the "Predict" button, the model generates the predicted flight price.

Additionally, the code includes an explanation component using LIME (Local Interpretable Model-Agnostic Explanations) and SHAP (SHapley Additive exPlanations). These techniques help provide interpretability and insights into the model's predictions. LIME generates explanations for individual predictions, while SHAP provides a summary plot of feature importance.

This flight ticket price prediction model and its accompanying web application can be useful for customers, airlines, travel agencies, and other stakeholders in the aviation industry to estimate and understand flight prices based on various factors.

**Dataset:**

The dataset selected for this project contain information about flight details and associated attributes. Here is the breakdown of the columns in the dataset:

1. flightId: A unique identifier for each flight.
2. airline: The name of the airline operating the flight.
3. flight: The flight code, which typically consists of the airline code followed by a dash and a flight number.
4. source\_city: The city from which the flight departs.
5. destination\_city: The city where the flight arrives.
6. departure\_time: The time of departure categorized into different time periods (e.g., Morning, Early Morning, Evening, Night, Afternoon, Late Night).
7. arrival\_time: The time of arrival at the destination categorized into different time periods.
8. stops: The number of stops during the flight categorized into different categories (e.g., zero, one, two or more).
9. duration: The duration of the flight in hours.
10. days\_left: The number of days remaining until the flight departure.
11. price: The price of the flight ticket.

**Feature Selection**

The feature selection method prepares the data by converting categorical variables into numerical representations and dropping irrelevant columns, making it suitable for training a regression model to predict flight ticket prices.

1. Extracting Flight Code and Flight Number: “Flight” column is split into two separate columns “flight\_code” and “flight\_no”.
2. Mapping Categorical Variables:
   * The code performs categorical encoding for the training and test datasets by mapping the categorical variables to numerical values.
   * "airline" is mapped by using a dictionary mapping each airline name to a numerical value.

(Vistara=0, Air\_India=1, GO\_FIRST=2, Indigo=3, AirAsia=4, SpiceJet=5)

* + "source\_city" and "destination\_city" are mapped similarly.

(Mumbai=0, Delhi=1, Bangalore=2, Kolkata=3, Hyderabad=4, Chennai=5)

* + “departure\_time" and "arrival\_time" are mapped using dictionaries that assign numerical values to different time categories

(Morning=0,Early\_Morning=1,Evening=2,Night=3,Afternoon=4,Late\_Night=5)

* + Stops are mapped to their corresponding numbers.

(zero=0, one=1, two\_or\_more=2)

* + Each flight code is assigned a numerical value.

1. Dropping Columns:
   * Unique column “flightId” is dropped.
   * Column ‘Flight’ is split into “flight\_code” and “flight\_no”, thus it is also dropped.

**Machine Learning Algorithms:**

1. **Gradient Boosting:**

*R2=0.8194 and MSE=31,978,343.37*

The Gradient Boosting model exhibits a strong performance with an R2 score of 0.8194, indicating that it captures a significant portion of the target variable's variance. It utilizes an ensemble of weak learners, sequentially building upon each other's shortcomings, to create a powerful predictor. With a maximum depth of 5, learning rate of 0.056 and 2300 estimators, it achieves a balanced trade-off between model complexity and generalization ability. The model's mean squared error (MSE) of 31,978,343.37 indicates a relatively low prediction error, making it a promising choice for applications where accurate price estimation is crucial.

1. **XGBoost:**

*R2=0.814 and MSE=32,941,617.74*

The XGBoost model, leveraging extreme gradient boosting techniques, demonstrates strong predictive capabilities with an R2 score of 0.8140. With 1300 estimators and a learning rate of 0.04, it constructs an ensemble of decision trees that are optimized to minimize the objective function. Its ability to effectively capture non-linear relationships and interactions in the data contributes to its high performance. Despite a slightly higher MSE of 32,941,617.74, the XGBoost model remains a competitive option, especially considering its parallel processing capabilities.

1. **LightGBM:**

*R2=0.7928 and MSE=36,696,315.08*

The LightGBM model, a gradient boosting framework developed by Microsoft, achieves a commendable R2 score of 0.7928. With a learning rate of 0.09, 5800 estimators, and a maximum depth of 4, LightGBM strikes a balance between model complexity and training efficiency. Its ability to handle large datasets and its efficient implementation make it a popular choice for various applications. Although it exhibits a higher MSE of 36,696,315.08 compared to the other models, its computational efficiency and interpretability make it a valuable contender in scenarios where speed and model transparency are prioritized.

1. **Ensemble Model - Voting Regressor:**

*R2=0.8193 and MSE=31,995,161.06*

The Voting Regressor combines the predictions of Gradient Boosting, XGBoost, and LightGBM models to form a robust ensemble. It achieves an R2 score of 0.8193, closely mirroring the performance of the Gradient Boosting model. By leveraging the collective knowledge of the constituent models, the Voting Regressor harnesses the strengths of each model and mitigates individual weaknesses. Although its MSE of 31,995,161.06 is slightly higher than that of the Gradient Boosting model, the Voting Regressor offers a reliable and balanced prediction, making it a compelling choice for applications that demand robustness and model diversity.

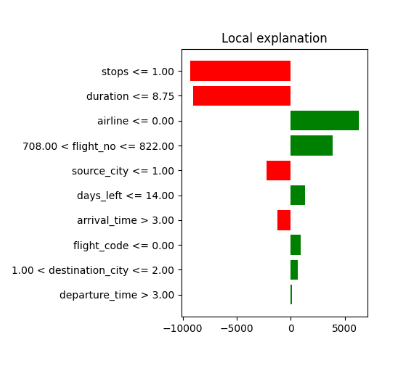
Overall, all four models demonstrate competitive performance in predicting flight prices. The Gradient Boosting model and the Voting Regressor stand out with their strong R2 scores and relatively low MSE values. XGBoost and LightGBM, while exhibiting slightly lower R2 scores and higher MSE values, still offer valuable alternatives with their unique strengths, such as parallel processing and computational efficiency (XGBoost) and interpretability (LightGBM). The choice of the most suitable model depends on the specific requirements and priorities of the application at hand.

**Scenarios**

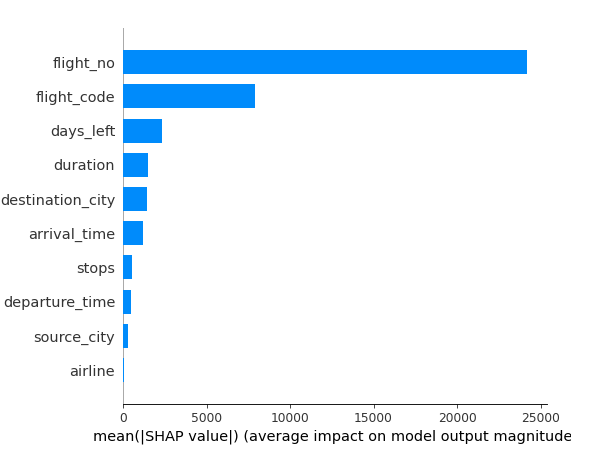
1. Airline=Vistara, Flight=UK-808, City From=Mumbai, City To=Bangalore, Stops=0, Departure Date=2023/05/28, Arrival Date=2023/05/28, Departure Time=12:00, Arrival Time=15:00

*Predicted Price = 36851*

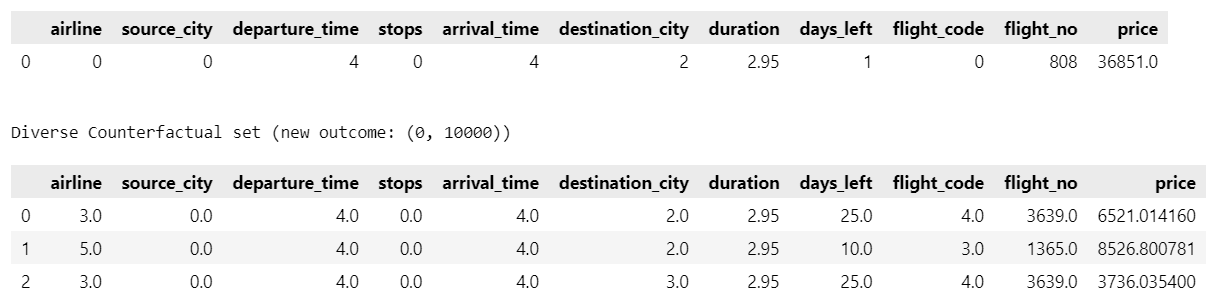
**LIME:**



**SHAP:**



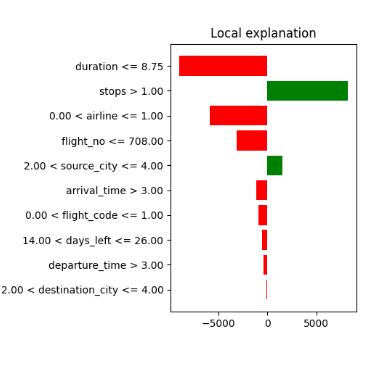
**COUNTERFACTUAL:**



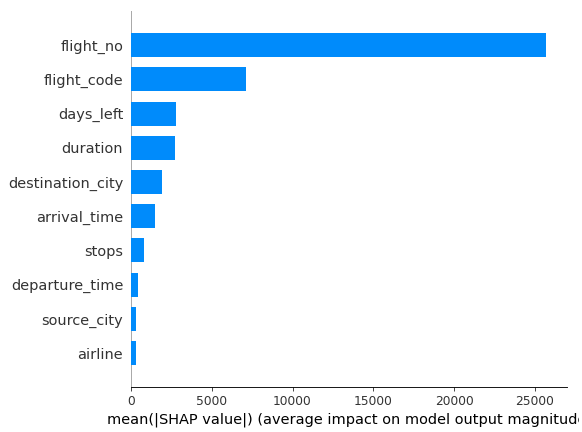
1. Airline=Air\_India, Flight=AI-669, City From=Kolkata, City To=Hyderabad, Stops=2, Departure Date=2023/06/16, Arrival Date=2023/06/17, Departure Time=23:00, Arrival Time=01:00

*Predicted Price = 28766*

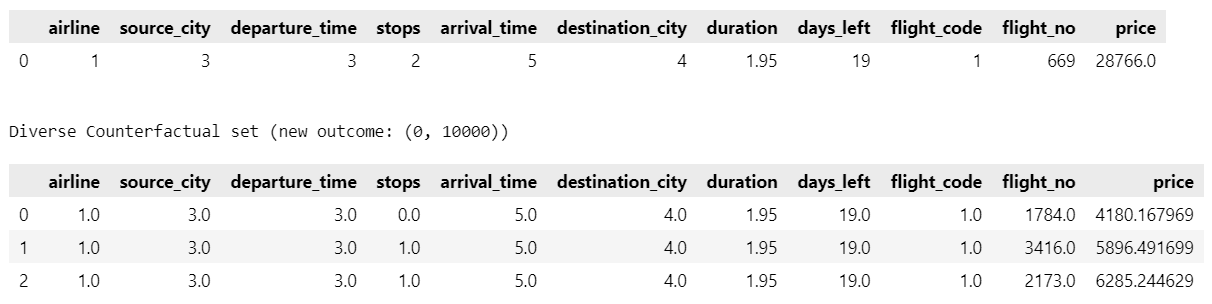
**LIME:**



**SHAP:**



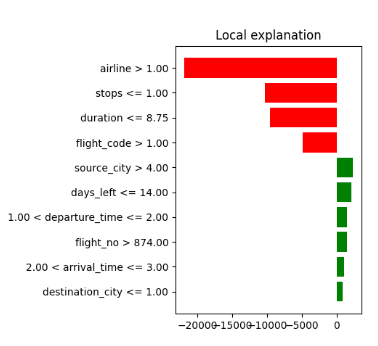
**COUNTERFACTUAL:**



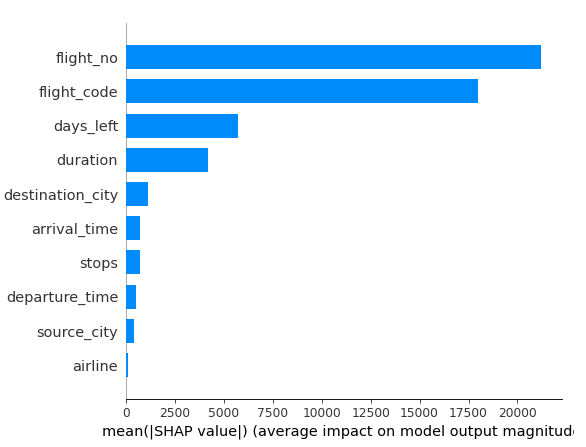
1. Airline=Indigo, Flight=G8-2508, City From=Chennai, City To=Delhi, Stops=1, Departure Date=2023/06/01, Arrival Date=2023/06/01, Departure Time=19:45, Arrival Time=22:45

*Predicted Price = 4618*

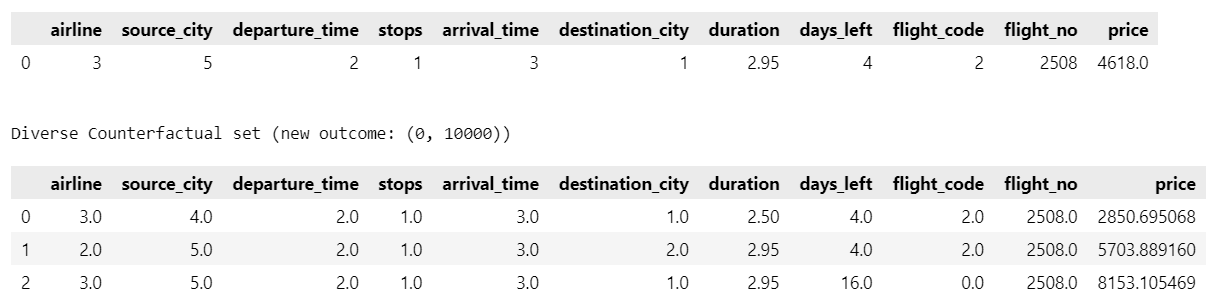
**LIME:**



**SHAP:**



**COUNTERFACTUAL:**



**Programming Interface**

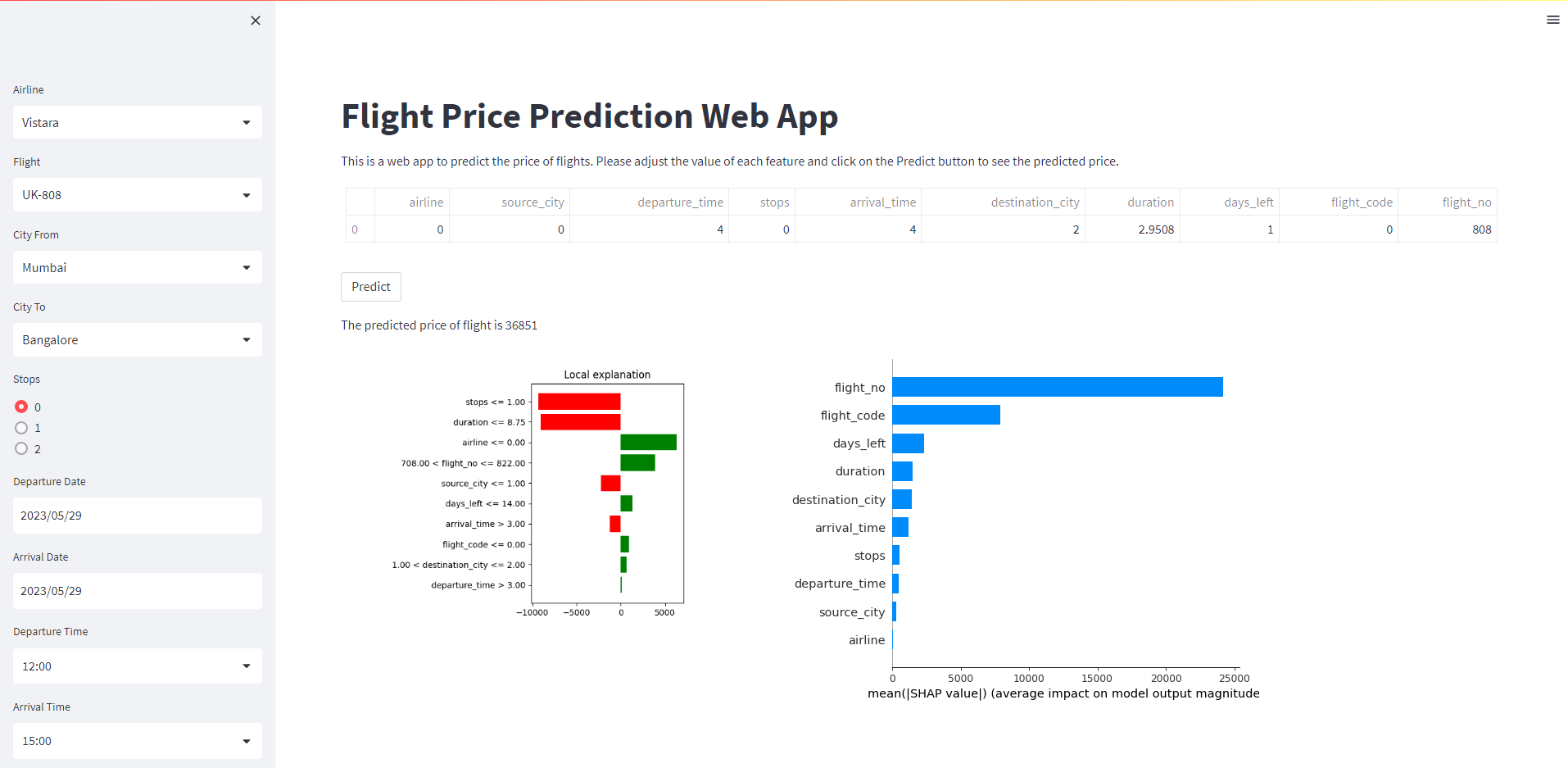
This flight price prediction web application uses Streamlit for programming interface. It provides a user interface where users can adjust various features related to the flight, such as airline, source city, destination city, stops, departure time, arrival time, days left, duration, flight code, and flight number. After adjusting the feature values, users can click on the "Predict" button to see the predicted price of the flight.

The app uses a pre-trained machine learning model, loaded from the "flights\_model.sav" file, to make the price predictions. The model is trained on a dataset represented by the "X\_train.csv" file. The app also displays the selected feature values in a table for easy reference.

In addition to the price prediction, the app provides an explanation of the prediction using the LIME (Local Interpretable Model-Agnostic Explanations) and SHAP (SHapley Additive exPlanations) model.

The Lime function implements the LIME (Local Interpretable Model-agnostic Explanations) algorithm. It takes the input data, a specific instance, and the trained model as inputs. Using the LimeTabularExplainer from the lime\_tabular module, this function generates an explanation for the specific instance by approximating the model locally. The result is an explanation object that contains feature importance values and other relevant information. The LIME explanation is plotted as a graph using Matplotlib and displayed in the streamlit app

The SHAP function utilizes the SHAP (SHapley Additive exPlanations) algorithm to explain model predictions. Similar to the Lime function, it takes the input data, a specific instance, and the trained model as inputs. Using the KernelExplainer from the shap module, this function calculates the SHAP values for the instance, representing the contribution of each feature to the predicted price. Additionally, the expected value of the model is computed to provide further context to the SHAP values. The LIME explanation is visualized as a pyplot figure, showing the feature importance values for the selected flight instance.



Overall, this Streamlit app allows users to interactively explore and predict flight prices while providing interpretability through Lime and Shap explanations.