PRICE PREDICTION MODEL:
A MACHINE LEARNING APPROACH TO PREDICT FLIGHT PRICES BASED ON FEATURES

Final Year Dissertation

by

SRIJAN DUTT
(Roll: 96/STS No. 200011)

Under the Supervision of

Dr. SUSHOVON JANA
Assistant Professor of Applied Statistics, MAKAUT, WB

DEPARTMENT OF STATISTICS

UNIVERSITY OF KALYANI

Kalyani, Nadia, West Bengal – 741235

August, 2025

DEPARTMENT OF STATISTICS

UNIVERSITY OF KALYANI

Kalyani, Nadia, West Bengal – 741235

DECLARATION

I, **SRIJAN DUTT Roll-No. 96/STS-200011 Regn No. 100073 of 2020- 2021**, a student of 5-Year Integrated M.Sc (Department of Statistics, University of Kalyani) hereby declare that the Project Dissertation titled "**PRICE PREDICTION MODEL: A Machine Learning Approach to predict Flight Price based on Features**" has been carried out by me under the guidance of Dr. Sushovan Jana at Department of Statistics, University of Kalyani, Nadia, West Bengal. In partial fulfillment of the requirement for the ward of degree of M.Sc (Integrated Five Years) in Statistics, this dissertation is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma and Fellowship or like other similar title or recognition.

Date:

Place: Kalyani

Srijan Dutt

(Roll No. 96/STS No. 200011)

DEPARTMENT OF STATISTICS

UNIVERSITY OF KALYANI

Kalyani, Nadia, West Bengal – 741235

CERTIFICATE

I hereby certify that the Project Dissertation titled "**PRICE PREDICTION MODEL: A Machine Learning Approach to predict Flight Price based on Features**" submitted by **SRIJAN DUTT Roll No. 96/STS No. 200011 Regn No. 100073 of 2020-21**, Department of Statistics, University of Kalyani, in partial fulfillment of the requirement for the award of the degree of Masters in Science (Integrated Five Years), is a record of the project work carried out by the students under my supervision. To the best of my knowledge, this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Date:

Place: Kalyani

Dr. Chandranath Pal (HOD)

Dr. Sushovon Jana (SUPERVISOR)

DEPARTMENT OF STATISTICS

UNIVERSITY OF KALYANI

Kalyani, Nadia, West Bengal – 741235

ACKNOWLEDGEMENT

I wish to express my sincerest gratitude to Professor Dr. Sushovan Jana for his continuous guidance and mentorship that he provided us during the project. He showed us the path to achieve our targets by explaining all the tasks to be done and explained to us the importance of this project as well as its relevance in the Aviation Market. He was always ready to help us and clear our doubts regarding any hurdles in this project. Without his constant support and motivation, this project would not have been successful. Besides this, I want to thank my fellow batchmates Nirmallya Dutta and Ranit Baidya with whom I have worked on this project and my family for their constant support throughout this journey.

Srijan Dutt

CONTENTS

CHAPTER 1: ABSTRACT

This dissertation offers a machine learning solution to predicting airline ticket prices based on a database of more than 300,000 records of flights with attributes like airline, route, source and destination airports, departure and arrival dates, class, duration, and booking horizon. Various models were used, ranging from ensemble methods (LightGBM, CatBoost, XGBoost and Random Forest) to regression-based methods (Spline-Lasso, Mixed Effects and Two-Way ANOVA). The ensemble models produced higher predictive accuracy, with LightGBM ($R^2 = 0.98$) and Random Forest ($R^2 = 0.986$) performing better than other algorithms, and XGBoost also performing well ($R^2 = 0.907$). Regression methods, while less accurate provided interpretive information including on non-linear effects of booking time and length and the strong effect of certain routes on price variation. The research concludes that hybrid approaches—merging ensemble accuracy with regression explainability—offer the most inclusive framework for explaining and predicting airfare prices.

CHAPTER 2: INTRODUCTION

Air travel is very important in today's world because it connects people, goods and cultures around the world. The aviation industry has grown quickly over the past few decades and millions of people now rely on airlines for both domestic and international travel. This growth has also made competition between carriers stronger, which means that ticket prices are one of the most important parts of running an airline. Flight prices are known to change a lot, sometimes within hours or even minutes, because of things like demand and supply and competition between airlines. These complexities make it hard to guess how much flight tickets will cost, but it's a very useful thing for both passengers and service providers to do.

Airfare volatility has big effects on the economy, not just on individual travelers. When ticket prices go up or down a lot, economies that depend on tourism can see big changes in demand. Businesses that fly a lot also have to deal with more operational uncertainty. So, being able to guess and model flight prices is important for more than just policymakers, travel agencies, and industries that depend on aviation.

Accurate price prediction helps travelers plan ahead and save money by letting them know when to buy tickets. It helps airlines with demand forecasting, revenue management and strategic pricing plans that keep them competitive and maximize profits. The complex, non-linear relationships that influence airfare fluctuations have proven difficult for traditional rule-based or statistical approaches to capture. In order to address this challenge, sophisticated computational methods have become more popular.

A data-driven method that can handle big, complicated datasets and reveal hidden patterns is machine learning. Modern models like LightGBM, XGBoost, and CatBoost are used in this dissertation to improve predictive accuracy, and SHAP is used to make the results easier to understand. To ensure the predictive framework's robustness, Spline-enhanced Lasso CV regression is also used for feature selection and regularization. Combining these techniques, the study in aviation analytics while offering a practical solution to the flight price prediction problem.

CHAPTER 3: LITERATURE REVIEW

Global connectivity is at the heart of air transport, and airline prices play a key role in determining demand, load factors, and tourist flows. Since prices adjust continuously according to competitor behaviour, seasonality, lead time, and demand disturbances, the price series that result are very volatile (IATA, 2021). This volatility has previously been examined in the area of airline revenue management (RM), where traditional approaches utilized stochastic dynamic programming, bid-price control, and overbooking regulations to maximize seat inventory (Talluri & van Ryzin, 2004). Although they have been influential, these approaches tend to be based on limiting parametric assumptions and less effective when faced with high-dimensional, non-stationary data sets common to online fare distribution (ScienceDirect, 2023).

Machine learning (ML) has proved to be a strong contender. Tree ensemble techniques such as XGBoost (Chen & Guestrin, 2016), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018) have outperformed linear or rule-based baseline competitors in airfare prediction tasks. Rajankar et al. (2019) demonstrated that XGBoost modeled intricate temporal and categorical interactions in web-scraped fare data, while Papakostas, Karakos and Kotsiantis (2017) established that gradient boosting performed better than regression in route-level demand forecasting. The more recent literature affirms the value of ensemble learners for airline data with mixed features, such as booking lead times and origin–destination identifiers (Wong, Smith and Zhao, 2023; Kumar, Gupta and Singh, 2025).

In addition to prediction, interpretability is essential for deployment. Lundberg and Lee's (2017) SHAP framework has been generally accepted to explain ML models and both provide global feature rankings and local attributions that increase transparency in pricing systems (Christoph, 2023). Complementary breakthroughs in regularisation, including spline-enhanced LASSO, offer an ability to capture smooth nonlinearities (e.g., fare against lead time) while carrying out feature selection in high-dimensional environments (Scheipl and Kneib, 2021).

Current debate also poses ethics and regulatory issues with airlines testing AI-enabled personalized pricing (Kiplinger, 2025). These advancements underscore the necessity for not just accurate prediction systems but transparent and interpretable ones as well. Therefore, combining ensemble ML models with predictive capability using SHAP for interpretability and spline-LASSO for stable feature choice provides a critical framework for forecasting airfares in research and practice.

CHAPTER 4: OBJECTIVES

OBJECTIVE 1

To explore and summarize the distribution of flight attributes (airline, source city, destination city, departure/arrival time, stops, and class) using descriptive statistics and graphical methods.

OBJECTIVE 2

To investigate the relationship between flight price and key predictors such as airline, departure/arrival time, source-destination pairs, number of days left before departure, and travel class.

OBJECTIVE 3

To build predictive models for airline ticket pricing and evaluate their performance, while also identifying the most influential factors that drive price variation.

CHAPTER 5: DATA DESCRIPTION

5.1 DATA SOURCING

The Dataset used for this project is sourced from a publicly available dataset on Kaggle. Compiled by Rohit Grewal (2025) it contains more than 300000 domestic Flight records across Indian Cities.

Link: https://www.kaggle.com/datasets/rohitgrewal/airlines-flights-data

index	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price
0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumbai	Economy	2.17	1	5953
1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.33	1	5953
2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.17	1	5956
3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.25	1	5955
4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	1	5955
5	Vistara	UK-945	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.33	1	5955
6	Vistara	UK-927	Delhi	Morning	zero	Morning	Mumbai	Economy	2.08	1	6060
7	Vistara	UK-951	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.17	1	6060
8	GO_FIRST	G8-334	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.17	1	5954
9	GO_FIRST	G8-336	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.25	1	5954
10	GO_FIRST	G8-392	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.25	1	5954
11	GO_FIRST	G8-338	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.33	1	5954
12	Indigo	6E-5001	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.17	1	5955
13	Indigo	6E-6202	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.17	1	5955
14	Indigo	6E-549	Delhi	Afternoon	zero	Evening	Mumbai	Economy	2.25	1	5955
15	Indigo	6E-6278	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	1	5955

5.2 DATA SPECIFICATIONS

The Dataset holds records for Flight to & from 6 major cities of India which include details of each single scheduled flight including its operational details, schedule, pricing and route characteristics. It contains **300,153 rows** and **12 columns** of which 8 columns are categorical and 4 columns are numerical. The Feature columns are described as follows.

5.3 NUMERICAL FEATURES

i. **duration** – Duration of the flight including the stops (in Hrs).
ii. **days_left** – No. of days left before departure date of a Flight
iii. **price** – Ticket price of the flight (in ₹)

Statistic	**duration**	**days_left**	**price**
Min	0.83	1	1105
Max	49.83	49	123,071
Mean	12.22	26.00	20,889.66
Median	11.25	26	7425

5.4 CATEGORICAL FEATURES

i. **airline** – Multiple carriers such as *SpiceJet*, *AirAsia*, *Vistara*, etc
ii. **flight** – Each airline's unique flight codes.
iii. **source_city** – Multiple cities of origin (Delhi, Mumbai, Kolkata etc.).
iv. **departure_time** – Broad time categories for departure (*Early_Morning*, *Morning*)
v. **stops** – No. of Stops by the Flight (like *zero*, *one*, *two_or_more*).
vi. **arrival_time** – Broad time categories for arrival (*Early_Morning*, *Morning*).
vii. **destination_city** – Multiple cities as destinations (Delhi, Mumbai, Kolkata etc.).
viii. **class** – Primarily *Economy* and *Business*.

5.5 DATA PRE-PROCESSING

To prepare the Data for effective machine Learning modelling we go through a series of preprocessing steps were undertaken to ensure data consistency, manage mixed data types and improve model interpretability and performance

5.5.1 Handling Missing Values/Duplicates:

Deal with incomplete data to prevent bias or errors by removing columns with high missing values/duplicates or imputing the missing values with mean/median or mode. The Dataset we selected has no missing values.

5.5.2 Categorical Encoding:

To convert the categorical data type features for modelling, **OneHot Encoding** was applied. This Encoding applies 0-1 value to each category within a variable which makes it useful for tree-based models (e.g. Random Forest, Light GBM).

CHAPTER 6: EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis (EDA) is the statistical and visual analysis of a dataset to know its structure, patterns, trends, and anomalies prior to formal modeling. It includes summarizing numerical characteristics, analyzing categorical distributions, identifying outliers and checking relationships between variables. EDA assists in generating insights, testing assumptions, and informing further preprocessing or feature engineering to ensure subsequent modeling is undertaken based on an explicit understanding of the data's characteristics.

6.1 ANALYSIS OF FEATURES

Fig. 1 Flight per Airlines

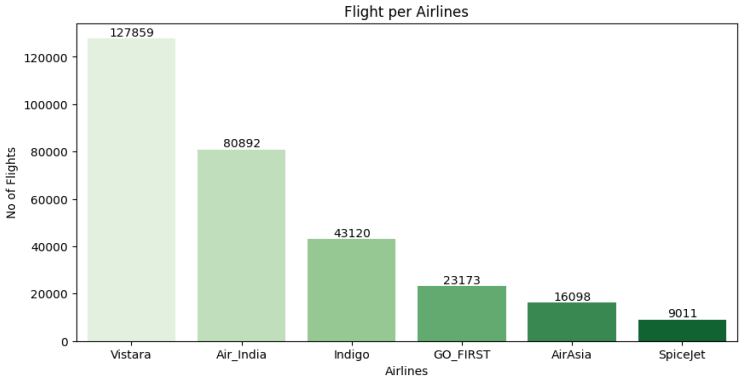
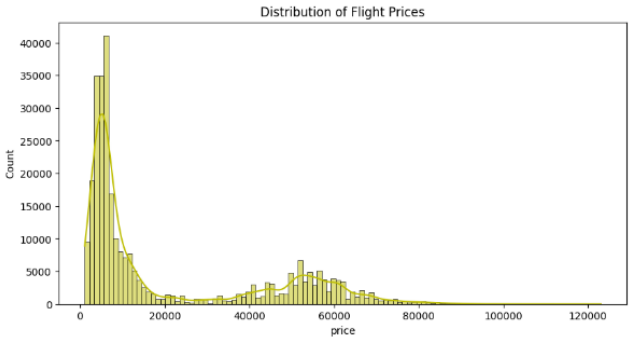


Fig. 2 Distribution of Flight Prices & Duration



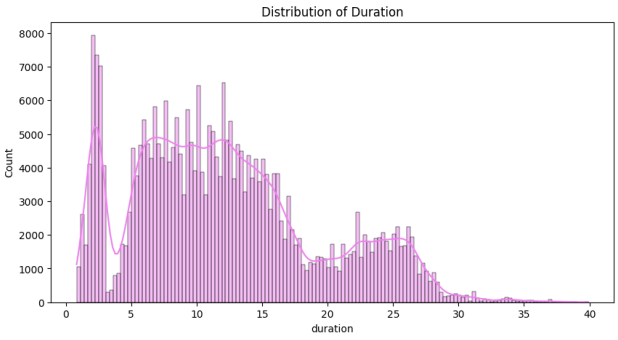


Fig. 3 Flights Departing/Arriving to the cities

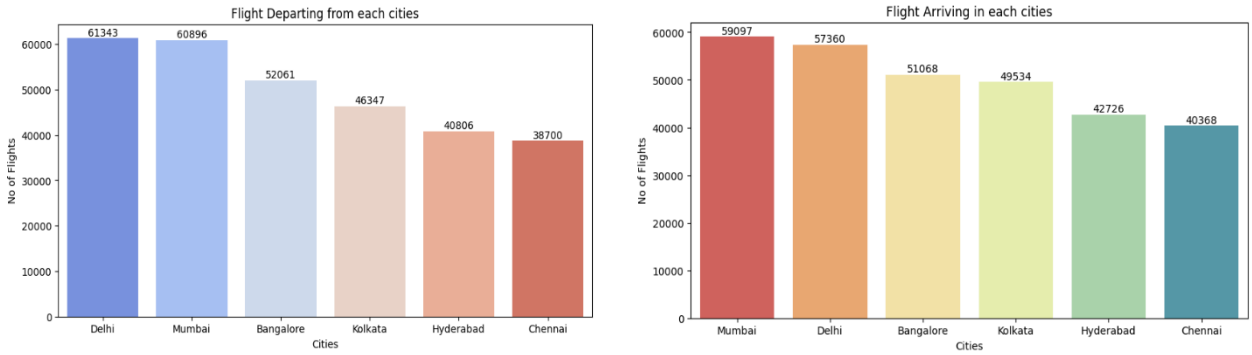


Fig. 4 Departure Time

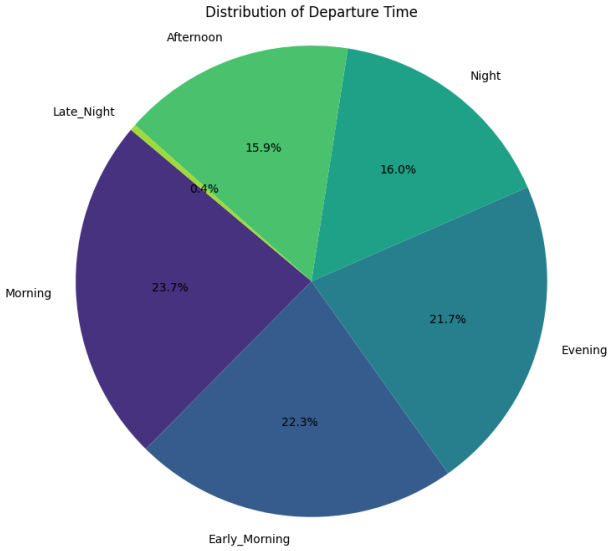
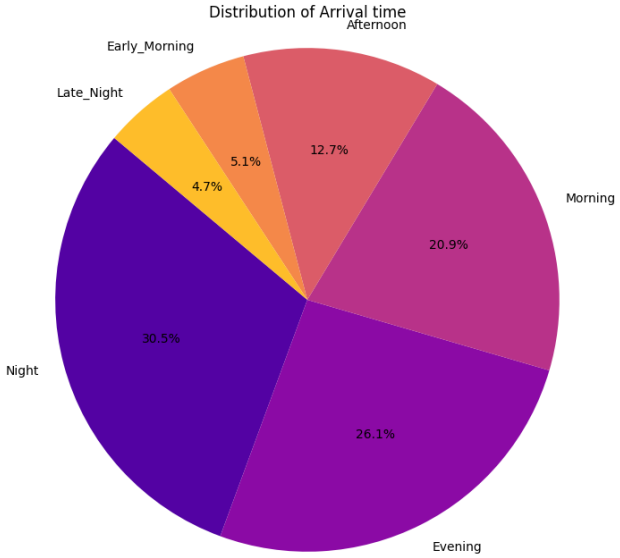


Fig. 5 Arrival Time



6.2 INSIGHTS USING FEATURE ENGINEERING : ROUTES

A new column **routes** is created by merging column **source_city** and **destination_city**.

```
df['route']=df['source_city']+"-"+df['destination_city']
```

Fig. 6. Top-10 Most Popular Routes

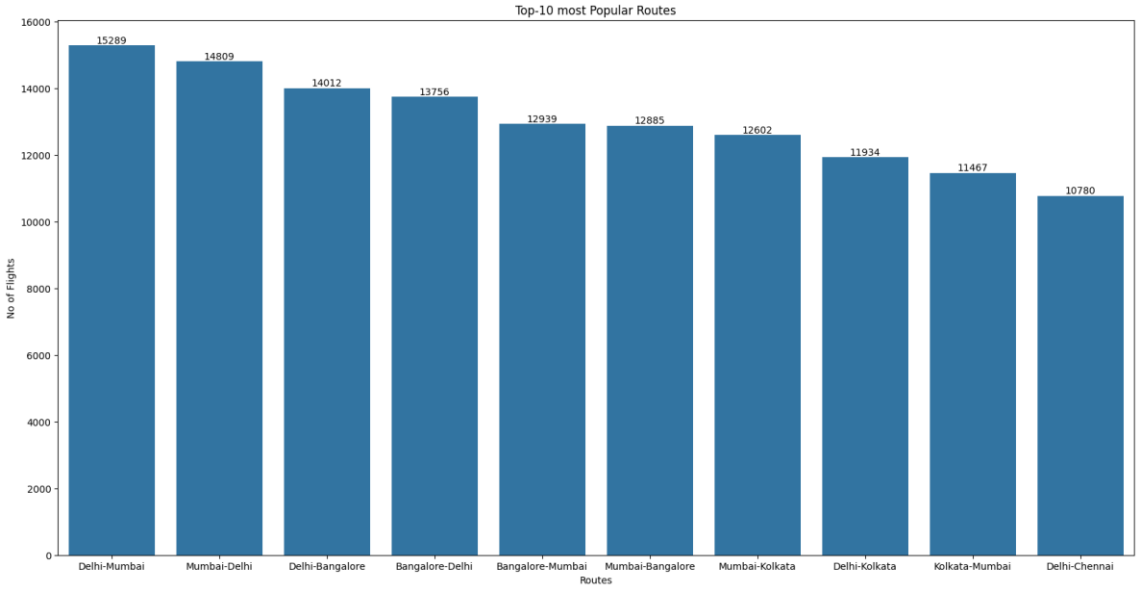


Fig. 7 Top-10 Expensive Routes

```
route_price = df.groupby(['source_city', 'destination_city'])['price'].mean().sort_values(ascending=False)

plt.figure(figsize=(12,6))
route_price[:10].plot(kind='bar', color='pink', label='Top 10 Expensive')
plt.title("Top 10 Expensive Routes by Average Price")
plt.xlabel("Route")
plt.ylabel("Average Price")
plt.xticks(rotation=75)
plt.legend()
plt.show()
```

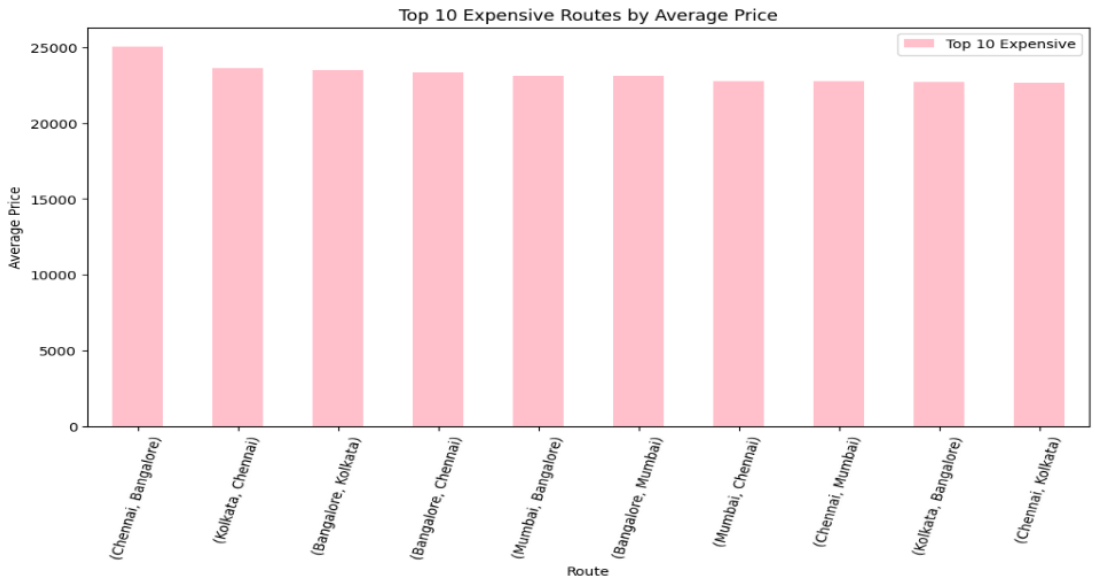
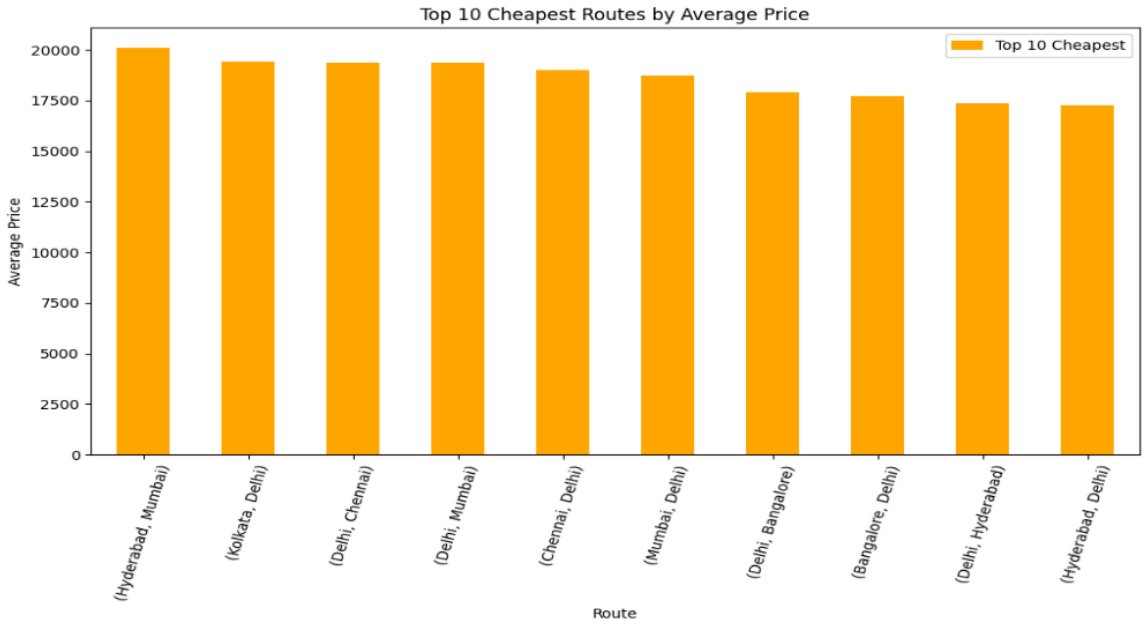


Fig. 8 Top-10 Cheapest Routes



6.3 PRICE VS NO. OF DAYS LEFT BEFORE DEPARTURE

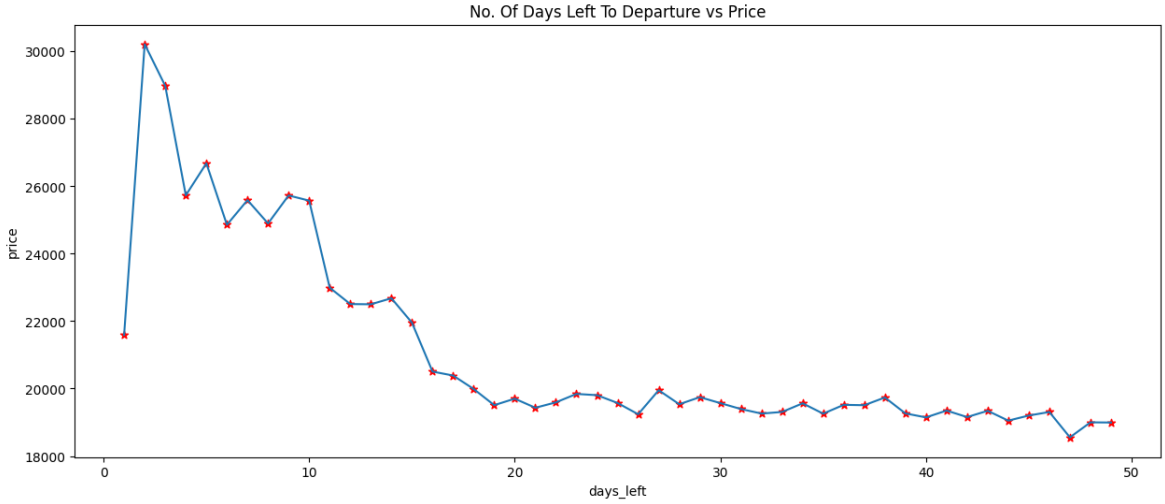


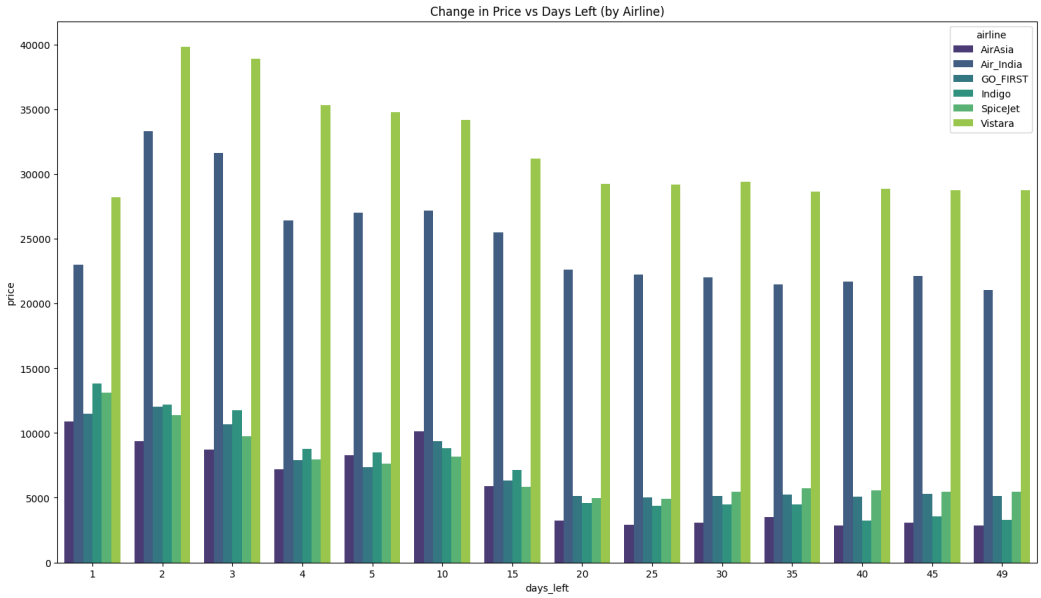
Fig. 9 Change in Price vs No. of Days left

The following Bar Graphs plot shows the change in Price is calculated along with days left for the departure based on a different feature.

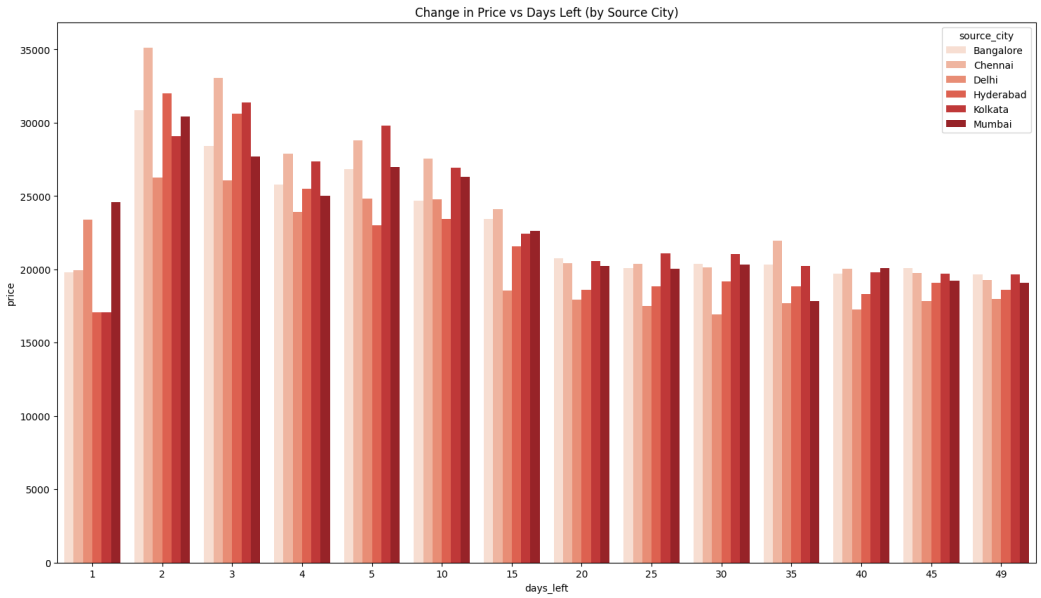
Fig. 10 : The Change in price vs days left for departure based on:

a. Airlines
b. Source City
c. No. of Stops

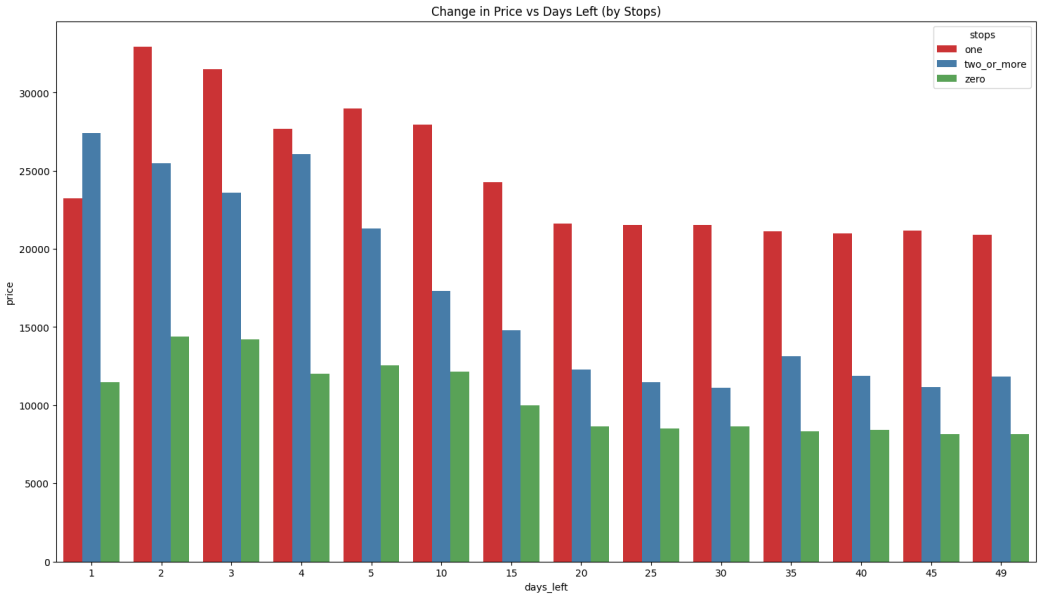
a.


Change in Price vs Days Left (by Airline)
airline
AirAsia
Air_India
GO_FIRST
Indigo
SpiceJet
Vistara
price
days_left

b.


Change in Price vs Days Left (by Source City)
source_city
Bangalore
Chennai
Delhi
Hyderabad
Kolkata
Mumbai
price
days_left

c.

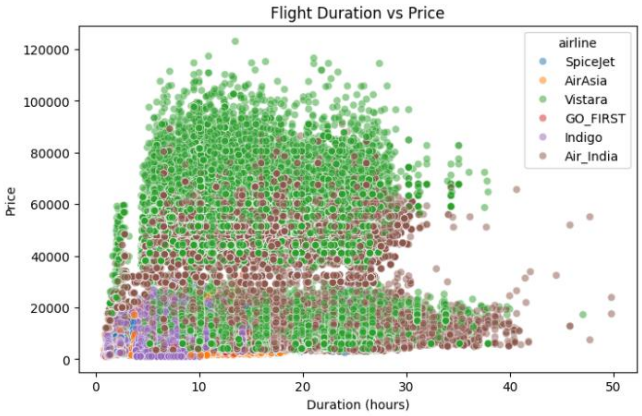

Change in Price vs Days Left (by Stops)
stops
one
two_or_more
zero
price
days_left

6.4 SCATTER PLOTS: CHANGE IN PRICE VS FLIGHT DURATION

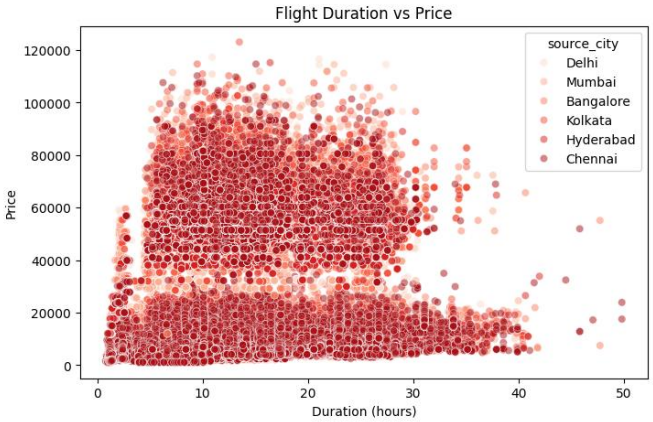
```
plt.figure(figsize=(8,5))
sns.scatterplot(x='duration', y='price', data=df, alpha=0.5, hue='class')
plt.title("Flight Duration vs Price")
plt.xlabel("Duration (hours)")
plt.ylabel("Price")
plt.show()
```

Fig. 11 Change in Price vs Flight Duration based on different features

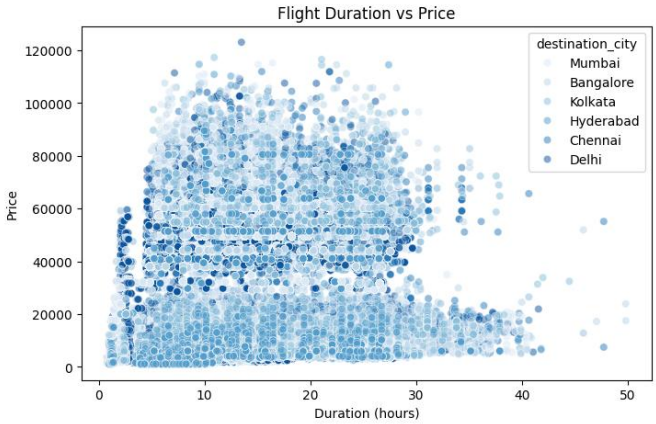
Based on Class



Based on Airlines



Based on City of origin



Based on Destination City

6.5 CORRELATION HEATMAPS

A correlation heatmap is a graphical representation of the correlation matrix, where colours indicate the strength and direction of relationships between numerical variables. It helps quickly identify patterns, positive or negative associations, and potential multicollinearity, making it a valuable tool in exploratory data analysis for feature selection and data understanding.

Fig. 12 Heatmap showing relationship between departure/arrival time and price of ticket

```
time_price = df.groupby(['departure_time', 'arrival_time'])['price'].mean()

plt.figure(figsize=(10,6))
sns.heatmap(time_price, annot=True, fmt=".0f", cmap="coolwarm")
plt.title("Average Price by Departure and Arrival Time")
plt.xlabel("Arrival Time")
plt.ylabel("Departure Time")
plt.show()
```

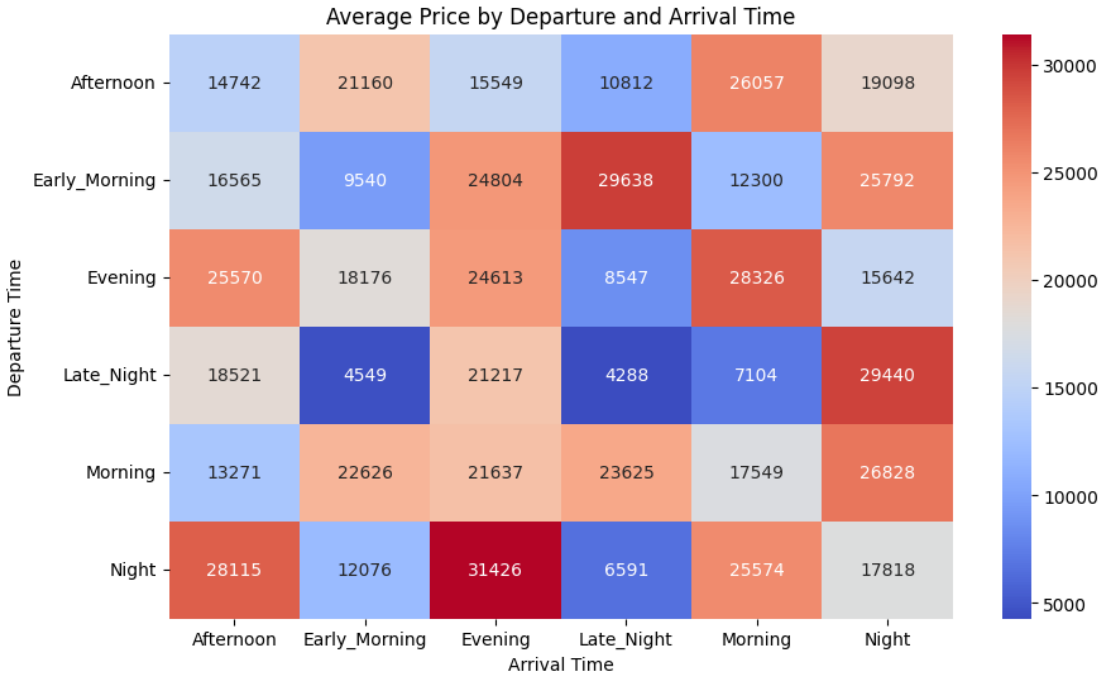
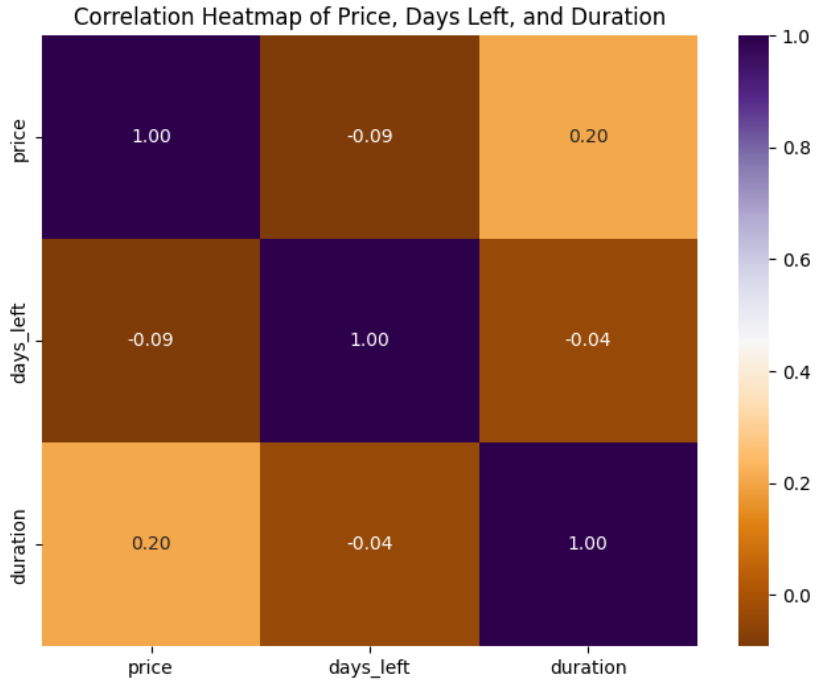


Fig. 13 Heatmap showing relationship between price, days left for departure and duration



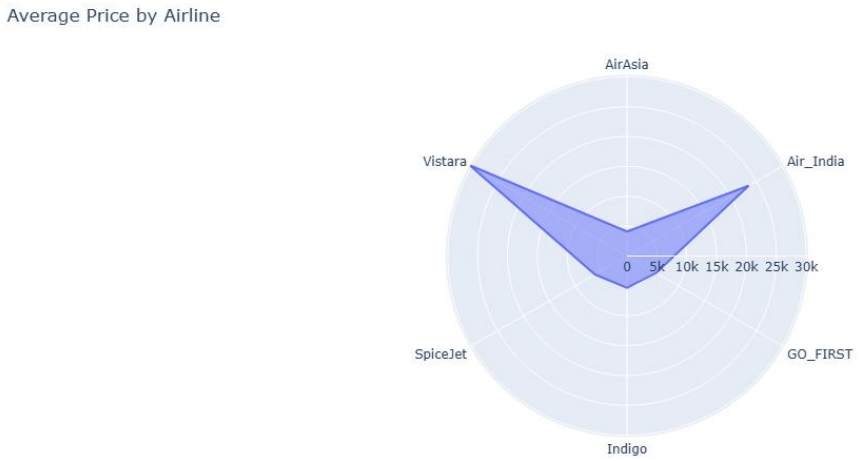
6.6 RADAR CHARTS

A radar map (or radar chart) displays multivariate data on axes starting from a common center, forming a polygon. It allows comparison of multiple variables for one or more categories, making patterns and strengths easily visible. It is commonly used for performance analysis as it highlights strengths and weaknesses or balanced and unbalanced profiles.

Fig. 14 Plot showing Average Price & Duration on Different Airlines

```
fig = px.line_polar(airline_metrics, r='price', theta='airline', line_close=True,
                    title='Average Price by Airline',
                    hover_data={'price':':.2f', 'duration':':.2f', 'days_left':':.2f'})
fig.update_traces(fill='toself')
fig.show()

fig = px.line_polar(airline_metrics, r='duration', theta='airline', line_close=True,
                    title='Average Duration by Airline',
                    hover_data={'price':':.2f', 'duration':':.2f', 'days_left':':.2f'})
fig.update_traces(fill='toself')
fig.show()
```



CHAPTER 7: METHODOLOGY

7.1 Two-WAY ANOVA

Two-way ANOVA (Analysis of Variance) is used to examine the effect of two categorical independent variables on one continuous dependent variable, and whether there's an interaction effect between the two factors.

7.1.1 When to Use a Two-Way ANOVA

We should use a two-way ANOVA when you'd like to know how two factors affect a response variable and whether or not there is an interaction effect between the two factors on the response variable.

For example, Our dataset contains details of flights (airline, source city, departure time, stops, destination city, class, duration, days left, and price).

In this case, we have the following variables:

- **Independent variable A:** airline/source city
- **Independent variable B:** class/stops
- **Dependent variable :** price

And we would like to answer the following questions:

- **Main Effect of Factor A** – Does the first independent variable (e.g., **airline**) have a significant effect on the dependent variable (e.g., **ticket price**) regardless of the second factor?
- **Main Effect of Factor B** – Does the second independent variable (e.g., **class: Economy/Business**) significantly influence the dependent variable, regardless of the first factor?
- **Interaction Effect (A × B)** – Is there a combined effect of both factors together? In other words, does the effect of one factor (e.g., class) on ticket price depend on the level of the other factor (e.g., airline)?

We would use a two-way ANOVA for this analysis because we have **two** factors.

7.1.2 Model Specifications

- $\mathbf{Y_{ijk}}$ = observed value of the dependent variable for the k-th observation in the i-th level of Factor A and j-th level of Factor B.
- $\boldsymbol{\mu}$ = overall mean of the dependent variable
- $\boldsymbol{\alpha_i}$ = effect of the i-th level of Factor A
- $\boldsymbol{\beta_j}$ = effect of the j-th level of Factor B
- $\boldsymbol{(\alpha\beta)_{ij}}$ = interaction effect between Factor A and Factor B at level i, j
- $\boldsymbol{\epsilon_{ijk}}$ = random error term, assumed to be independently and identically distributed with mean zero and variance σ^2.

The general form of the model is:

$$\mathbf{Y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \epsilon_{ijk}}$$

Where:

- i=1,2,…a (levels of Factor A)
- j=1,2,…b (levels of Factor B)
- k=1,2,…n (replications within each cell)

7.1.3 Two-Way ANOVA Assumptions

For the results of a two-way ANOVA to be valid, the following assumptions should be met:

1. Normality – The response variable is approximately normally distributed for each group assessed based on Shapiro-Wilk test and Normal probability plots.

2. Equal Variances – The variances for each group should be roughly equal tested using Levene's test or Bartlett's test.

3. Independence – The observations in each group are independent of each other and the observations within groups were obtained by a random sample.

7.1.4 Hypothesis Testing

The Two-Way ANOVA tested three null hypotheses:

1. **Main Effect of Factor A:**
   $H_{0A}: \alpha_1 = \alpha_2 = \cdots = \alpha_a = 0$
   (No significant difference in the dependent variable across the levels of Factor A.)
2. **Main Effect of Factor B:**
   $H_{0B}: \beta_1 = \beta_2 = \cdots = \beta_b = 0$
   (No significant difference in the dependent variable across the levels of Factor B.)
3. **Interaction Effect:**
   $H_{0AB}: (\alpha\beta)_{ij} = 0$ for all i, j
   (The effect of Factor A on the dependent variable does not depend on the level of Factor B.)

7.1.5 ANOVA Table

Explanation of Notation:

- $\mathbf{a}$ = number of levels of Factor A
- $\mathbf{b}$ = number of levels of Factor B
- $\mathbf{n}$ = number of replicates per cell
- $\mathbf{N}$ = a*b*n = total number of observations
- $\overline{\boldsymbol{Y}}_{\mathbf{i..}}$ = mean of observations at level i of Factor A (averaged over B and replications)
- $\overline{\boldsymbol{Y}}_{\mathbf{.j.}}$ = mean of observations at level j of Factor B (averaged over A and replications)
- $\overline{\boldsymbol{Y}}_{\mathbf{ij.}}$ = mean of observations for combination i of A and j of B
- $\overline{\boldsymbol{Y}}_{\mathbf{...}}$ = grand mean of all observations
- $\mathbf{Y_{ijk}}$ = value of the k-th replicate in cell (i,j)

Sum of Squares:

- $\mathbf{SS_A}$ (Sum of Square due to Factor A) = $\sum_{i-1}^{a}(\overline{Y}_{i..} - \overline{Y}_{...})^2$
- $\mathbf{SS_B}$ (Sum of Square due to Factor B) = $\sum_{j-1}^{b}(\overline{Y}_{.j.} - \overline{Y}_{...})^2$
- $\mathbf{SS_{AB}}$ (Sum of Square due to Interaction) = $\sum_{i-1}^{a}\sum_{j-1}^{b}(\overline{Y}_{ij.} - \overline{Y}_{i..} - \overline{Y}_{.j.} + \overline{Y}_{...})^2 * n$
- $\mathbf{SS_E}$ (Sum of Square due to Error) = $\sum_{i-1}^{a}\sum_{j-1}^{b}\sum_{k=1}^{n}(Y_{ijk} - \overline{Y}_{ij.})^2$
- $\mathbf{SS_T}$ (Sum of Square of Total) = $\sum_{i-1}^{a}\sum_{j-1}^{b}\sum_{k=1}^{n}(Y_{ijk} - \overline{Y}_{...})^2$

Mean Squares:

- $\mathbf{MS_A} = SS_A/a\text{-}1$
- $\mathbf{MS_B} = SS_B/b\text{-}1$
- $\mathbf{MS_{AB}} = SS_{AB}/(a\text{-}1)(b\text{-}1)$
- $\mathbf{MS_E} = SS_E/a*b*(n\text{-}1)$

Source of Variation	Degrees of Freedom	Sum of Square	Mean Square	F-Statistic
Factor A	a-1	SS_A	MS_A	$F_A = MS_A/ MS_E$
Factor B	b-1	SS_B	MS_B	$F_B = MS_B/ MS_E$
Interaction	(a-1)(b-1)	SS_{AB}	MS_{AB}	$F_{AB} = MS_{AB}/ MS_E$
Error	a*b*(n-1)	SS_E	MS_E	____
Total	N-1	SS_T	____	____

7.1.6 Model Performance

Main Effects

The main effect of *Factor A* was [statistically significant / not statistically significant], *F* (dfA, dfError) = [F-value], *p* = [p-value], partial η^2 = [effect size]. This result indicates that mean scores on *Dependent Variable* [differed / did not differ] significantly across the levels of *Factor A*, when collapsing across *Factor B*. The main effect of *Factor B* was [statistically significant / not statistically significant], *F* (dfB, dfError) = [F-value], *p* = [p-value], partial η^2 = [effect size], suggesting that [scores did / did not] differ significantly across the levels of *Factor B*, independent of *Factor A*.

Interaction Effect

The interaction between *Factor A* and *Factor B* was [statistically significant / not statistically significant], *F* (dfAB, dfError) = [F-value], *p* = [p-value], partial η^2 = [effect size]. A significant interaction suggests that the influence of *Factor A* on the dependent variable varies depending on the level of *Factor B*. Where the interaction was significant, simple main effects analyses were conducted to clarify the nature of the effect. This finding indicates that the two factors interact in a non-additive manner, and the impact of one factor is conditional upon the level of the other.

7.2 REGRESSION SETUP

Regression setup involves defining a dependent variable (target) and one or more independent variables (predictors). Data is split into training and testing sets, features are preprocessed, and a suitable regression model is selected. The model is trained, evaluated using metrics (e.g., RMSE, R^2), and optimized for predictive accuracy.

7.2.1 Model Specifications

We want to model the flight price y as a function of predictors (features):

$$y = f(x_{i1}, x_{i2}, \ldots\ldots, x_{ip}) + \varepsilon_i$$

- y_i = Flight Price for the ith observation
- x_{ij} = j-th feature for the i-th observation
- $f(.)$ = Unknown Regression function to be learned
- ε_i = Noise Term, assumed $E[\varepsilon_i] = 0$, $Var(\varepsilon_i) = \sigma^2$

Here, predictors include categorical (**airline, source_city, stops, class**) and numerical (**duration, days_left**)

7.2.2 Model Evaluation Metrics

- **Coefficient of Determination :**

  The coefficient of determination (R^2) measures how well a regression model explains the variance in the dependent variable. It ranges from 0 to 1, where higher values indicate better model fit. An R^2 of 1 means perfect prediction, while 0 indicates no explanatory power. It helps assess model effectiveness but does not confirm causation.

$$\mathbf{R^2 = 1 - \frac{\sum_{i=1}^{n}(yi - \hat{y}i)\text{^}2}{\sum_{i=1}^{n}(yi - \bar{y})\text{^}2}}$$

  where y_i are actual values, $\hat{y}_i$ are predicted values and $\overline{y}$ is the mean of the actual values.

- **Mean Absolute Error :**

  Mean Absolute Error (MAE) is a regression evaluation metric that measures the average absolute difference between predicted and actual values. It indicates how close predictions are to real outcomes, with lower values showing better accuracy. Unlike squared error metrics, MAE treats all errors equally, making it intuitive and less sensitive to large outliers in data.

$$\mathbf{MAE} = \frac{1}{n} \sum_{i=1}^{n} | y_i - \hat{y}_i |$$

- **Root Mean Squared Error :**

  Root Mean Squared Error (RMSE) is a widely used metric for evaluating regression models. It calculates the square root of the average squared differences between predicted and actual values. RMSE penalizes larger errors more than smaller ones, making it sensitive to outliers. Lower RMSE values indicate better predictive performance and closer alignment between predictions and true outcomes.

$$\mathbf{RMSE} = \left[\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \right]^{1/2}$$

7.2.3 Hyperparameters and Tuning

Hyperparameters are configuration settings in machine learning models that guide the learning process but are not learned from the data itself. Examples include learning rate, maximum depth of a tree, and number of hidden layers in a neural network. Choosing the right hyperparameters is crucial for achieving good performance. Hyperparameter tuning is the process of optimizing these values using techniques such as **Grid search, Random search, or Bayesian optimization** to balance bias and variance, ensuring accurate and reliable model predictions.

7.3 RANDOM FOREST REGRESSOR

Random forest regression is a powerful tool in data science, enabling accurate predictions and the analysis of complex datasets using an advanced machine-learning algorithm. A random forest regression model combines multiple decision trees into a single ensemble. Each tree is built from a different subset of the data and makes an independent prediction. The final output is determined by averaging or taking a weighted average of all the trees' predictions.

7.3.1 Model Specifications

The general prediction function for a Random Forest Regressor can be written as:

$$\hat{\mathbf{y}} = \frac{\mathbf{1}}{\boldsymbol{T}} \sum_{\boldsymbol{t=1}}^{\boldsymbol{T}} \boldsymbol{f}_{\mathbf{t}}\,(\mathbf{x})$$

where:

- $\hat{\mathbf{y}}$ = predicted value for input vector x
- $\mathbf{T}$ = total number of trees in the forest
- $\boldsymbol{f}_{\mathbf{t}}\,(\mathbf{x})$ = prediction from the t-th regression tree
- Each f_t is trained on a bootstrap sample of the original dataset, using a random subset of predictors at each split

7.3.2 Hyperparametric Tuning

Key hyperparameters in a Random Forest Regressor include:

- n_estimators – number of decision trees in the forest
- max_depth – maximum depth of each tree (controls model complexity)
- min_samples_split – minimum number of samples required to split a node
- min_samples_leaf – minimum number of samples required at a leaf node
- max_features – number of predictors considered at each split
- bootstrap – whether bootstrap sampling is used when building trees

7.3.3 Training Procedure

- The dataset was randomly split into training and testing sets (e.g., 80% training 20% testing).
- The model was trained on the training set using bootstrap aggregation (bagging), where each tree received a different bootstrap sample and considered a random subset of features at each split.
- The predictions from all trees were averaged to generate the final regression output.
- Model performance was evaluated on the testing set using metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R^2 score.

7.3.4 Assumptions and Advantages

Random Forest Regression does not require linearity, homoscedasticity, or normally distributed residuals.

Its key assumptions are:

- Observations are independent and identically distributed (i.i.d)
- The features are relevant to predicting the target variable
- The number of trees is sufficient to ensure stability of results

Advantages include:

- Handles nonlinear relationships and high-dimensional data
- Robust to outliers and multicollinearity
- Provides feature importance estimates for model interpretability
- Resistant to overfitting when properly tuned

7.3.5 Feature Importance

The most influential predictors were top features, followed by secondary features. These variables contributed most to reducing the prediction error across the ensemble of trees, as measured by the mean decrease in impurity.
Lower-ranked features, such as low-importance features, contributed minimally to the overall predictive performance, suggesting they may have limited relevance for forecasting the dependent variable within this dataset.

7.3.6 Model Performance

Model performance was evaluated on the independent testing dataset using multiple regression performance metrics. The Random Forest achieved an $\mathbf{R^2}$ score, indicating that percentage of the variance in the dependent variable was explained by the model. The Root Mean Squared Error (RMSE) suggesting that, on average, predictions deviated from the observed values by approximately. The Mean Absolute Error (MAE) reflecting the average absolute prediction error in the same units as the dependent variable.

Collectively, these results indicate that the Random Forest model provides predictive accuracy for the given dataset.

7.3.7 Disadvantages

 Complexity & Interpretability – Random Forest is a black-box model, making it harder to interpret compared to simpler models like linear regression or decision trees.

 Computational Cost – Training can be slow and memory-intensive, especially with large datasets or many trees.

 Overfitting Risk – Although it reduces overfitting compared to single trees, it can still overfit noisy data.

 Bias with Imbalanced Data – Random Forest may favour majority classes if data is highly imbalanced.

7.4 GENERALIZED ADDITIVE MODELS (GAM)

Generalized Additive Models (GAMs) are an extension of Generalized Linear Models (GLMs) that allow for flexible, non-linear relationships between the predictors and the dependent variable by using smooth functions rather than strictly linear terms. This flexibility enables GAMs to capture complex patterns in the data without requiring a prior specification of the exact functional form of each relationship.

GAMs are particularly useful when theory or exploratory analysis suggests that relationships may be curved or vary in shape, but the exact form is unknown. They are applicable to a variety of outcome types, including continuous, binary, count, and proportion data, through the use of different link functions and distributions

7.4.1 Model Specification

The general form of a GAM is:

$$g(\mathbb{E}[Y]) = \beta_0 + f_1(X_1) + f_2(X_2) + \ldots\ldots + f_p(X_p)$$

Where:

- $\mathbf{Y}$ = dependent variable.
- $g(.)$ = link function relating the mean of the dependent variable to the predictors (e.g., identity link for Gaussian, logit link for binomial, log link for Poisson).
- β_0 = intercept term.
- $f_k(X_k)$ = smooth function of predictor X_k often represented using splines (e.g., cubic splines, thin plate splines).
- $\mathbf{p}$ = number of predictors.

These smooth functions are estimated from the data and can take on shapes that best fit the observed relationship, subject to a smoothing penalty that prevents overfitting.

7.4.2 Assumptions

GAMs share several assumptions with GLMs:

- Correct distributional assumption for the dependent variable (e.g., normal, binomial, Poisson).
- Independence of observations.
- Additivity of the smooth terms — the effects of predictors add together on the scale of the link function.
- Smoothness selection — the degree of wiggliness of each f_k is controlled through smoothing parameters to balance fit and generalisation.

7.4.3 Estimation Procedure

- Choice of distribution and link function — based on the nature of the dependent variable and research question.
- Selection of smooth functions — splines are most commonly used, with a basis dimension set to allow sufficient flexibility.
- Estimation of smoothing parameters — typically performed using Generalized Cross-Validation (GCV) or Restricted Maximum Likelihood (REML).
- Model fitting — smooth terms and parametric terms are jointly estimated.
- Model evaluation — using deviance explained, adjusted R^2, Akaike Information Criterion (AIC), or out-of-sample performance metrics.

7.4.4 Advantages & Limitations

Advantages

- Captures non-linear effects without explicit functional form specification.
- Can incorporate both smooth terms (for continuous predictors) and parametric terms (for categorical predictors).
- Retains interpretability through separate examination of each smooth term.
- Works within the GLM framework, so results are comparable to traditional regression approaches.

Limitations

- Interpretation of effects is more visual than numerical; requires plotting smooth functions.
- Potential for overfitting if smoothing parameters are not appropriately chosen.
- Computationally more intensive than linear or generalized linear models.

7.4.5 Model Performance

The final model explained value(%) of the deviance in the dependent variable, with an adjusted R^2 of value. The Generalized Cross-Validation (GCV) score was value, indicating [good/moderate/limited] overall fit to the data.

Smooth Terms
The estimated smooth term for X_1 was [statistically significant / not significant] (edf, F, p-value) suggesting a [non-linear / approximately linear] relationship with the dependent variable.

Parametric Terms
Categorical predictor *FactorVar* was included as a parametric term and was [significant / not significant] (Estimate, SE, t, p-value) indicating that group differences.

Model Diagnostics
Residual plots revealed [no systematic patterns / slight heteroscedasticity], suggesting that model assumptions were [reasonably met / partially violated]. Plots of fitted values versus residuals and QQ-plots of residuals supported the adequacy of the model fit.

7.5 SHAP (SHapley Additive exPlanations)

SHAP (SHapley Additive exPlanations) is a model-agnostic method for interpreting machine learning predictions, based on Shapley values from cooperative game theory. Each feature is treated as a "player" that contributes to the model's prediction. SHAP fairly distributes the prediction by averaging a feature's marginal contribution across all possible subsets of features. The prediction is expressed as the sum of a baseline value (average prediction) and feature contributions (SHAP values). Positive values push the prediction up, negative values push it down. SHAP ensures local accuracy, consistency, and interpretability, with efficient variants like TreeSHAP, KernelSHAP, and DeepSHAP.

7.5.1 Mathematical Formulation

For a model f(x), prediction for instance x :

$$\boldsymbol{f}(\boldsymbol{x}) = \quad \boldsymbol{\phi_0} + \sum_{\boldsymbol{i=1}}^{\boldsymbol{M}} \boldsymbol{\phi_i}$$

where:

- ϕ_0 : baseline (expected prediction if no features are known).
- ϕ_i : SHAP value of feature iii (its contribution).

The Shapley value of feature i :

$$\boldsymbol{\phi_i} = \sum_{\boldsymbol{S \subseteq F \setminus \{i\}}} \frac{|\boldsymbol{S}|!(\boldsymbol{M}-|\boldsymbol{S}|-\boldsymbol{1})!}{\boldsymbol{M}!} \left[\boldsymbol{f}(\boldsymbol{S} \cup \{\boldsymbol{i}\}) - \boldsymbol{f}(\boldsymbol{S}) \right]$$

- S : subset of features without i
- f(S) : model prediction using only features in S

7.5.2 Properties

- Efficiency (completeness): Sum of contributions = prediction difference from baseline.
- Symmetry: If two features contribute equally, they get equal SHAP values.
- Dummy property: Features with no effect get SHAP = 0.
- Additivity/Linearity: Explanations combine consistently.

7.5.3 Interpretetion

- Local explanation: For one instance, SHAP tells how features push prediction up/down.
- Global explanation: Aggregating SHAP values over dataset → feature importance ranking.
- Visualization tools: Summary plots, force plots, dependence plots.

7.6 GRADIENT BOOSTING MODEL (GBM)

The Gradient Boosting Regressor (GBR) is an ensemble learning method that builds predictive models in a stage-wise fashion by combining multiple weak learners, typically decision trees, to form a strong overall model. At each iteration, the model fits a new regression tree to the residual errors (i.e., the difference between observed and predicted values) of the ensemble built so far. By doing so, it sequentially reduces bias and improves predictive performance.

7.6.1 Model Specifications

The prediction for the i-th observation in Gradient Boosting Regression can be expressed as:

$$\widehat{\boldsymbol{y}}_{\mathbf{i}} = \sum_{\boldsymbol{m}=\mathbf{1}}^{\boldsymbol{M}} \boldsymbol{\gamma}_{\mathbf{m}}\, \boldsymbol{h}_{\mathbf{m}}\,(\boldsymbol{x}_{\mathbf{i}})$$

Where:

- $\widehat{\boldsymbol{y}}_{\mathbf{i}}$ = predicted value for instance iii.
- **M** = number of boosting iterations (trees).
- $\boldsymbol{h}_{\mathbf{m}}\,(\boldsymbol{x}_{\mathbf{i}})$ = weak learner (usually a regression tree) at the m-th stage.
- $\boldsymbol{\gamma}_{\mathbf{m}}$ = learning rate–scaled weight applied to the tree.

The model is trained by minimising a specified **loss function** (e.g., Mean Squared Error for regression) using gradient descent:

$$\mathcal{L} = \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{n}} \boldsymbol{l}\,(\boldsymbol{y}_{\mathbf{i}}\,,\, \widehat{\boldsymbol{y}}_{\mathbf{i}}\,)$$

At each step, a new tree is fit to the negative gradient of the loss function with respect to the predictions of the model so far.

7.6.2 Hyperparametric Tuning

Key hyperparameters include:

- n_estimators – number of boosting stages (trees).
- learning_rate (η) – step size shrinkage controlling contribution of each tree.
- max_depth – maximum depth of individual trees (controls complexity).
- subsample – fraction of samples used per boosting iteration (reduces overfitting).
- loss – loss function to be minimised (e.g., squared error).

7.6.3 Advantages

- Captures non-linear relationships and variable interactions.
- Reduces both bias and variance through boosting.
- Provides feature importance scores for interpretation.

7.7 LightGBM REGRESSOR

The Light Gradient Boosting Model (LightGBM) regressor is a powerful gradient boosting framework based on decision trees, optimized for efficiency and scalability. It uses a leaf-wise tree growth strategy with depth constraints, enabling accurate modeling of complex relationships while reducing computation time and memory usage. LightGBM supports large-scale datasets, handles categorical features directly, and incorporates regularization techniques to prevent overfitting. Key hyperparameters include learning rate, number of leaves, maximum depth, and feature/bagging fractions. In regression tasks, it minimizes loss functions such as Mean Squared Error, producing robust and interpretable predictions through feature importance and SHAP analysis.

7.7.1 Model Framework

Formally, given a dataset $\mathbf{D=\{(x_i,y_i)\}^n_{i=1}}$, where $x_i \in \mathbb{R}^p$ are predictors and $y_i \in \mathbb{R}$ is the response:

The objective function to minimize is:

$$\boldsymbol{L(\theta) = \sum_{i=1}^{n} l\,(y_i, \hat{y}_i) + \Omega\,(f_t)}$$

where:

- $\boldsymbol{l}(.)$ is the loss function (e.g., squared error $\boldsymbol{(y_i, \hat{y}_i)^2}$
- $\boldsymbol{f_t}$ is the prediction function at iteration t,
- **Ω(·)** is the regularization term controlling model complexity.

7.7.2 Hyperparametric Tuning

The model is specified through the following hyperparameters:

- **learning_rate (η):** Step size shrinkage to prevent overfitting.
- **n_estimators:** Number of boosting iterations.
- **num_leaves:** Maximum number of leaves in each tree (main complexity parameter).
- **max_depth:** Maximum depth of a tree.
- **min_data_in_leaf:** Minimum number of data points in a leaf node (controls overfitting).
- **feature_fraction:** Fraction of features randomly selected for each boosting round.
- **bagging_fraction & bagging_freq:** Fraction and frequency of data sampling for bagging.
- **lambda_l1 / lambda_l2:** Regularization parameters for controlling complexity.
- **early_stopping_rounds:** Stops training if validation score does not improve.

7.7.3 Regularization & Overfitting Control

- Leaf-wise growth with constraints (prevents deep overfitting trees).
- Feature fraction & bagging (introduce randomness for generalization).
- L1 and L2 regularization on weights (reduce variance).
- Early stopping during training.

7.7.4 Output

The final model produces continuous predictions:

$$\hat{y} = \sum_{t=1}^{T} f_t(x)$$

Where f_t is the decision tree at boosting iteration t and T is the total number of iterations.

7.7.5 Advantages

- **High Efficiency** – Trains significantly faster than traditional gradient boosting methods due to optimized histogram-based splitting.
- **Scalability** – Handles large-scale datasets with high-dimensional features efficiently.
- **Leaf-Wise Growth** – Adopts a leaf-wise tree growth strategy (with depth constraints), achieving lower loss and higher accuracy than level-wise methods.
- **Lower Memory Usage** – Efficient data storage reduces computational resources compared to XGBoost or Random Forests.
- **Native Handling of Categorical Features** – Avoids extensive preprocessing like one-hot encoding.

7.8 EXTREME GRADIENT BOOSTING REGRESSOR (XGBoost)

Extreme Gradient Boosting (XGBoost) is a high-performance, scalable implementation of gradient boosted decision trees, designed for efficiency and accuracy. It is particularly well-suited for regression tasks involving complex, non-linear relationships and interactions between variables.

The model builds a series of regression trees sequentially, with each new tree attempting to correct the errors made by the ensemble of previous trees. Hence by optimising a regularised objective function, XGBoost achieves a balance between model fit and generalisation.

7.8.1 Model Specifications

The prediction for a given input x_i in XGBoost Regression can be expressed as:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}$$

Where:

- $\hat{y}_i$ = predicted value for the i-th instance.
- K = total number of trees.
- f_k = regression tree in the set of possible trees $\mathcal{F}$.

The model is trained to minimise the following regularised objective function:

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Where:

- $l(y_i, \hat{y}_i)$ = differentiable loss function (e.g., Mean Squared Error for regression).
- $\Omega(f_k) = \gamma T + \frac{1}{2}\lambda ||\omega||^2$ is the regularisation term, where T = number of leaves in tree k, ω = leaf weights, γ = complexity penalty per leaf and λ = L2 regularisation term.

This formulation controls both model complexity and overfitting through γ and λ.

7.8.2 Hyperparametric Tuning

Key hyperparameters include:

- n_estimators – number of boosting rounds (trees).
- max_depth – maximum depth of each tree.
- learning_rate (η) – step size shrinkage used to prevent overfitting.
- subsample – fraction of training instances randomly sampled for each tree.
- colsample_bytree – fraction of features used for each tree.
- reg_alpha – L1 regularisation term on weights.
- reg_lambda – L2 regularisation term on weights.

7.8.3 Training Procedure

1. Data splitting – The dataset was divided into training and testing sets (e.g., 80% training, 20% testing).
2. Boosting iterations – The model was trained iteratively, with each tree attempting to predict the residuals of the combined previous trees.
3. Regularisation – Both L1 and L2 penalties were applied to leaf weights to reduce overfitting.
4. Stopping criteria – Early stopping was used if validation performance did not improve after [n] rounds.
5. Final evaluation – The model was tested on unseen data to assess generalisation.

7.8.4 Assumptions and Advantages

Assumptions

While XGBoost does not rely on the traditional statistical assumptions of linear regression, it does assume:

- Observations are independent and identically distributed (i.i.d.).
- The training and testing data come from the same underlying distribution.
- Features are relevant and appropriately preprocessed (e.g., missing values handled, categorical variables encoded).

Advantages

- Handles non-linear relationships and complex interactions automatically.
- Includes built-in regularisation to control overfitting.
- Efficient parallel computation, suitable for large datasets.
- Handles missing values internally.
- Provides feature importance scores for interpretability.

7.8.5 Model Performance

On the held-out testing dataset, the model achieved an R^2 score indicating that percentage(%) of the variance in the dependent variable was explained by the model's predictions. The Root Mean Squared Error (RMSE) reflecting the typical deviation between predicted and actual values, while the Mean Absolute Error (MAE) capturing the average absolute prediction error. Collectively, these results suggest that the XGBoost regressor provided [strong / moderate / limited] predictive accuracy and generalised well to unseen data.

Model Diagnostics

Residual analysis indicated that prediction errors were approximately symmetrically distributed around zero, with no strong evidence of heteroscedasticity. Plots of observed versus predicted values demonstrated a close alignment with the line of perfect prediction, further supporting the model's adequacy.

7.9 CATEGORICAL BOOSTING REGRESSOR (CatBoost)

CatBoost (Categorical Boosting) is a gradient boosting algorithm developed by Yandex, designed to handle categorical features efficiently and improve upon traditional boosting methods such as Gradient Boosted Decision Trees (GBDT). Unlike other ensemble algorithms such as LightGBM or XGBoost, CatBoost introduces ordered boosting and efficient encoding techniques for categorical variables, which reduces prediction shift and prevents overfitting. In the present study, CatBoost was employed as one of the core machine learning models to predict flight ticket prices, leveraging its robustness in dealing with high-dimensional structured datasets.

7.9.1 Model Specifications

Formally, the prediction function can be expressed as:

$$\hat{y}_i = F_M(x_i) = \sum_{m=1}^{M} \eta \cdot f_m(x_i)$$

where:

- $\hat{y}_i$ = predicted value for observation i
- **M** = total number of boosting iterations (trees)
- **η** = learning rate (shrinkage parameter)
- $f_m(x_i)$ = prediction of the mth regression tree for input x_i
- $F_M(x_i)$ = final boosted ensemble model

7.9.2 Update Rule

The boosted model is updated iteratively as:

$$F_m(x) = F_{m-1}(x) + \eta . f_m(x)$$

Where $f_m(x)$ is trained on the residuals $r_i^{(m)}$

7.9.3 Ordered Boosting

CatBoost addresses the common issue of target leakage in gradient boosting by employing ordered boosting. Instead of using the full dataset statistics for categorical feature encoding, CatBoost computes them in a permutation-driven manner, ensuring that at any training step, the model does not have access to its own target value. This reduces prediction shift and leads to better generalization.

7.9.4 Interpretability and Feature Importance

Although CatBoost is an ensemble-based method, it provides built-in functionality for feature importance estimation. Both loss function change-based importance and prediction value change-based importance were utilized to interpret how input variables influence model predictions. Additionally, SHAP (SHapley Additive exPlanations) values were integrated to provide a model-agnostic interpretation of feature contributions.

7.10 SPLINE ENHANCED REGRESSION

To model potentially nonlinear relationships between predictors X and outcome y by transforming X with spline basis functions and fitting a sparsity-inducing linear model that performs automatic feature selection.

7.10.1 Spline Transformation

We represent each continuous predictor x_j using a spline basis $B_j(x_j , k_j , d)$ where k_j are knot locations and d is the polynomial degree.

- Basis type: Cubic B-splines or natural cubic splines (preferred due to stable boundary behaviour).
- Degree: $d = 3$ (cubic) unless prior knowledge suggests otherwise.
- Knot placement:
    - Primary: Quantile-based knots at percentiles $\{q_1,\ldots,q_K\}$ to ensure data coverage.
    - Sensitivity: Uniform-spacing and alternative K.
- Number of knots: Start with $K \in \{5,7,9\}$K per variable (or proportional to n. Final effective complexity is controlled by regularization.
- Extrapolation: Use natural boundary constraints or linear extrapolation to avoid unrealistic tails.

The transformed design matrix is

$$\widetilde{\mathbf{X}} = [\, B_1(x_1) \mid \ldots\ldots\ldots \mid B_p(x_p) \,] \;\; \{\text{dummies}\}$$

7.10.2 Model Specifications

We fit a penalized linear model to the expanded features:

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{\ell}\left(y_i ,\; \beta_0 + \tilde{x}_i^{T} \boldsymbol{\beta} \right) + \boldsymbol{\lambda}\left(\boldsymbol{\alpha} \, ||\boldsymbol{\beta}||_1 + \frac{1-\alpha}{2} \, ||\boldsymbol{\beta}||_2 \right)$$

where ℓ is squared loss (regression) or logistic loss (binary outcomes)

- **Primary**: LASSO ($\alpha=1$).
- **Robustness**: Elastic Net ($\alpha \in \{0.5,0.8,1.0\}$ alpha in {0.5, 0.8, 1.0} $\alpha \in \{0.5,0.8,1.0\}$) to stabilize selection under collinearity among basis functions.

Interpretation: the penalty prunes unnecessary spline terms, yielding a sparse, interpretable nonlinear model.

7.10.3 Encoding Numerical features with Splines

For a numeric variable (e.g. duration, days_left), we use a B-spline basis expansion:

$$\phi(z) = (\phi 1(z), \phi 2(z), \dots, \phi m(z)),$$

where each $\phi j(z)$ is a spline basis function (piecewise polynomial, smooth at knots). So the contribution of becomes:

$$f(z) = \sum_{j=1}^{m} \theta \, \phi \, (z)$$

This allows nonlinear shapes for numeric predictors, rather than forcing linearity.

7.10.3 Combined Linear Model

After transformation, the regression model is

$$Y_i \approx \beta_0 + \underbrace{\sum \sum \boldsymbol{\beta}_{c,k} \, \boldsymbol{d}_{c,k}}_{\text{Categorical effects}} + \underbrace{\sum \sum_{j=1}^{m} \boldsymbol{\theta}_{z,j} \, \boldsymbol{\phi}_{z,j} \, (\boldsymbol{x}_{i,z})}_{\text{Numerical Spline effects}}$$

7.10.4 Advantages

- Captures nonlinear patterns.
- Regularized: avoids overfitting.
- Performs automatic knot/feature selection (similar spirit to MARS).

7.11 LASSO REGULARIZATOIN

We fit coefficients β, θ by minimizing:

$$\widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\beta}, \boldsymbol{\theta}} \left\{ \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum |\gamma_i| \right\}$$

where γ_j are all coefficients, and $\lambda > 0$ is chosen via CV.

- The L1 penalty forces some coefficients to zero → feature selection. Helps avoid overfitting,
- Especially since OHE + splines produce high-dimensional feature.

7.11.1 Grouped Importance

Since categorical/spline features expand into multiple columns, we aggregate importance as:

$$\boldsymbol{I}\,(\boldsymbol{G}) = \sum_{\boldsymbol{j}\,\in \boldsymbol{G}} |\,\hat{\boldsymbol{y}}_{\mathrm{j}}\,|$$

for a group G (e.g. all dummies for "route" or all spline bases for "duration").
This shows how much predictive weight belongs to each original feature blocks.

7.11.2 Partial effect for Splines

To interpret a numeric feature, we compute:

$$\Delta\, \boldsymbol{f}(\boldsymbol{z}) = \boldsymbol{f}(\boldsymbol{z}) - \boldsymbol{f}(\boldsymbol{z}_{\mathbf{ref}})$$

Where z_{ref} is some baseline (e.g. median).

7.11.3 Properties of Spline enhanced Lasso CV (Cross Validation)

A. Bias-Variance Tradeoff
- By shrinking coefficients, LASSO reduces variance at the cost of introducing some bias.
- This tradeoff often results in improved predictive performance, especially in high-dimensional data.

B. Cross-Validation for λ Selection
- The CV (Cross-Validation) procedure selects the optimal penalty parameter λ.
- Ensures that the chosen λ balances model complexity and prediction accuracy, minimizing generalization error.

C. Handling Multicollinearity
- In the presence of correlated predictors, LASSO tends to select one variable and shrink others to zero, improving model stability.
- Unlike ridge regression, it performs hard selection rather than distributing weights across correlated features.

D. High-Dimensional Data Adaptability
- Suitable when p >> n (more predictors than observations).
- Capable of producing meaningful models by selecting a subset of relevant predictors.

E. Sparsity and Interpretability
- Generates parsimonious models with fewer predictors, aiding interpretability.
- Particularly valuable in domains like genomics, finance, and text analysis where many predictors exist but only a subset is influential.

CHAPTER 8: RESULTS AND INFERENCE

❖ Two- WAY ANOVA

```
import statsmodels.api as sm
from statsmodels.formula.api import ols

# Two-way ANOVA model with interaction
model = ols("price ~ C(source_city) * C(destination_city)", data=df).fit()
anova_table = sm.stats.anova_lm(model, typ=2)  # Type II ANOVA table
print(anova_table)
```

```
                                    sum_sq        df             F
C(source_city)                -1.793800e-01       5.0 -7.020977e-11
C(destination_city)           -2.981750e+00       5.0 -1.167064e-09
C(source_city):C(destination_city)  1.970449e+08      25.0  1.542477e-02
Residual                       1.533578e+14  300123.0           NaN

                                   PR(>F)
C(source_city)                   1.00000
C(destination_city)              1.00000
C(source_city):C(destination_city)  0.90116
Residual                             NaN
```

The output of the two-way ANOVA model indicates how source city, destination city, and their interaction influence flight prices. The table shows sums of squares (sum_sq), degrees of freedom (df), F-statistics (F), and p-values (PR(>F)). For both the main effects—source city and destination city—the sums of squares are nearly zero and the F-statistics are negative (very close to zero), which suggests that neither factor significantly explains the variation in ticket prices. The corresponding p-values are 1.000, confirming that there is no statistical evidence of an effect.

For the interaction term (source_city × destination_city), the **F-statistic is also extremely small (0.0154)**, and the **p-value is 0.901**, far above the conventional threshold of 0.05. This means that even when considering the combination of source and destination cities,

There is no significant effect on flight prices in the dataset

❖ RANDOM FOREST REGRESSOR

The Random Forest model achieved an excellent fit to the airline price data, with an **R^2 of 0.986** and a **Mean absolute error (MAE) of about 1,108**, showing that it can capture most of the variance in ticket prices with high accuracy. Exploratory analysis already revealed that flight prices vary systematically by departure and arrival time, with Late Night departures being cheapest on average and Night departures being most expensive. The Random Forest confirmed these patterns: SHAP-based importance scores highlighted departure_time and arrival_time as meaningful contributors, alongside other categorical factors like route, airline, and class. Compared to CatBoost and LightGBM, which also handle categorical features efficiently and often provide smoother decision boundaries, Random Forest tends to be less parsimonious but still highly effective due to its ensemble averaging. CatBoost and LightGBM typically outperform Random Forest when datasets are very large or when fine-grained interactions between categorical levels matter, because they use boosting rather than bagging. However, in this case, Random Forest's performance is nearly optimal, with predictive accuracy that rivals or even exceeds those gradient boosting models. The trade-off is interpretability: CatBoost and LightGBM offer cleaner feature importance rankings (e.g., via SHAP or gain metrics), whereas Random Forest requires grouped SHAP or permutation importance to recover interpretable patterns. Overall, Random Forest proves to be a strong baseline, confirming that temporal factors (departure/arrival times) and route/city-level attributes are major drivers of ticket pricing, consistent with what boosting models also reveal, but with slightly different efficiency–interpretability trade-offs.

```
Mean price by departure_time:
departure_time
Late_Night        9295.299387
Afternoon        18179.203331
Early_Morning    20370.676718
Evening          21232.361894
Morning          21630.760254
Night            23062.146808
Name: price, dtype: float64

Mean price by arrival_time:
arrival_time
Late_Night       11284.906078
Early_Morning    14993.139521
Afternoon        18494.598993
Night            21586.758341
Morning          22231.076098
Evening          23044.371615
Name: price, dtype: float64

[RandomForest] R2: 0.986 | MAE: 1107.7
```

❖ XGBOOST REGRESSOR

The results obtained from the XGBoost model without hyperparameter tuning (RandomizedSearchCV) demonstrate a strong predictive performance. The **Mean Absolute Error (MAE) of 0.216** indicates that, on average, the model's predictions deviate from the actual transformed ticket prices by only a small margin, reflecting good accuracy. Similarly, **the Mean Squared Error (MSE = 0.093)** and the **Root Mean Squared Error (RMSE = 0.304)** are both low, signifying that the errors are not only minimal but also consistent across the dataset. Most importantly, the model achieved an **R^2 score of 0.907**, which suggests that approximately 90.7% of the variance in the transformed airfare values is explained by the predictors used in the model. This highlights the efficiency of XGBoost in capturing the underlying patterns in the data even without extensive tuning. However, while these results are highly promising, further refinement through hyperparameter optimization could potentially reduce the errors further and improve generalization to unseen data.

```
Metrics without RandomizedSearchCV:
Mean Absolute Error: 0.21552689448004134
Mean Squared Error: 0.09270040997504082
Root Mean Squared Error: 0.3044674202193739
R^2 Score: 0.9071030233480496
```

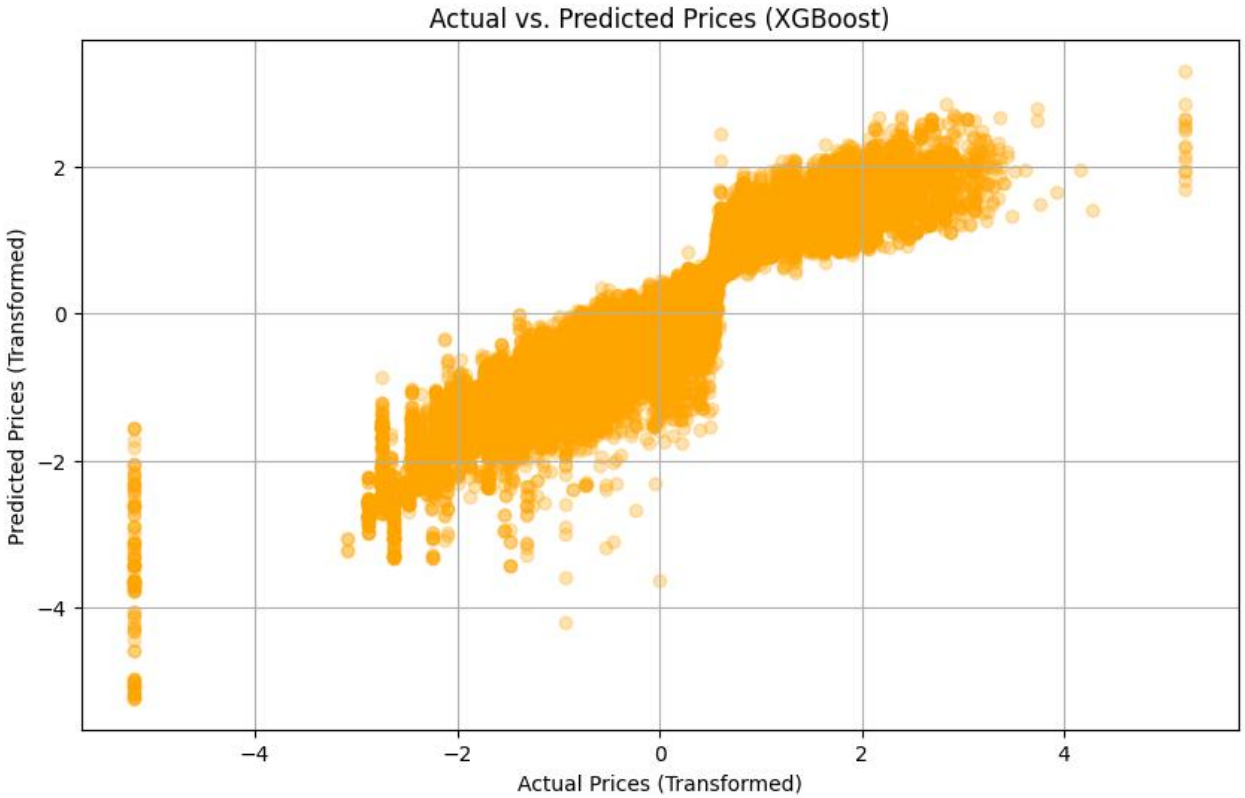


Fig. 15 XGBoost Results on Actual vs Predicted Plot

❖ CATBOOST AND LIGHT GBM REGRESSOR

The modeling exercise compared different approaches—Mixed Effects, CatBoost, and LightGBM—to analyze the drivers of airline ticket prices, with special attention to route and city-level effects. The Mixed Effects model decomposed variance into components for source_city and destination_city, showing how much of the price variation is attributable to specific cities versus residual unexplained factors. This provides a baseline understanding of city-specific pricing differences after accounting for controls like duration, days left, and class.

The tree-based models, CatBoost and LightGBM, delivered very high predictive accuracy, with **R^2 of 0.97 (CatBoost) and 0.98 (LightGBM)** on the test set. Their **Mean absolute errors (MAE ≈ 2200 for CatBoost and ≈ 1700 for LightGBM)** indicate strong predictive performance, though not perfect, especially for extreme ticket prices. SHAP importance values reveal that class is the single most influential factor in determining price, followed by airline, duration, and days_left. The route variable is also highly influential, ranking just below these major features, confirming that specific city-pair connections strongly shape price dynamics. By contrast, the standalone effects of source_city and destination_city are much smaller, suggesting that it is the combination of cities (i.e., the route) rather than either endpoint alone that drives substantial variation.

LightGBM's residual analysis shows that predictions align closely with actual prices, though some discrepancies remain at very high price levels, where variance naturally widens. The skipped MARS model would have added insights into piecewise and interaction effects but was unavailable due to missing dependencies.

Mixed Linear Model Regression Results

Model:	MixedLM	Dependent Variable:	price
No. Observations:	147045	Method:	ML
No. Groups:	1	Scale:	515460610.9698
Min. group size:	147045	Log-Likelihood:	-1683556.8598
Max. group size:	147045	Converged:	Yes
Mean group size:	147045.0		

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
Intercept	20419.770	514.700	39.673	0.000	19410.977	21428.563
class_Economy[T.True]	326.399	488.067	0.669	0.504	-630.195	1282.993
duration	6.335	8.234	0.769	0.442	-9.803	22.473
days_left	3.907	4.398	0.888	0.374	-4.713	12.526
destination_city Var	36347.453					
source_city Var	36426.327					

```
[CatBoost] TEST R2: 0.971 | MAE: 2211.5

[CatBoost] Mean |SHAP| (top features):
class               18005.358231
airline              3694.952444
duration             1829.574980
days_left            1796.565549
route                 760.579623
stops                 388.190223
arrival_time          378.710019
destination_city      356.378269
source_city           285.693762
departure_time        231.924012
dtype: float64

[CatBoost] Route vs. Source/Destination contributions:
route               760.579623
source_city         285.693762
destination_city    356.378269
```

```
[LightGBM] TEST R2: 0.982 | MAE: 1689.0

[LightGBM] Mean |SHAP| (top features):
class               19100.584025
duration             2130.026625
days_left            1761.162479
airline              1693.529648
route                1400.393499
arrival_time          341.361566
stops                 249.475593
departure_time        243.583336
source_city            26.463899
destination_city       22.496465
dtype: float64

[LightGBM] Route vs. Source/Destination contributions:
route               1400.393499
source_city           26.463899
destination_city      22.496465
```

Fig. 16 Actual vs Predicted Plot for :

a. LightGBM

b. CatBoost

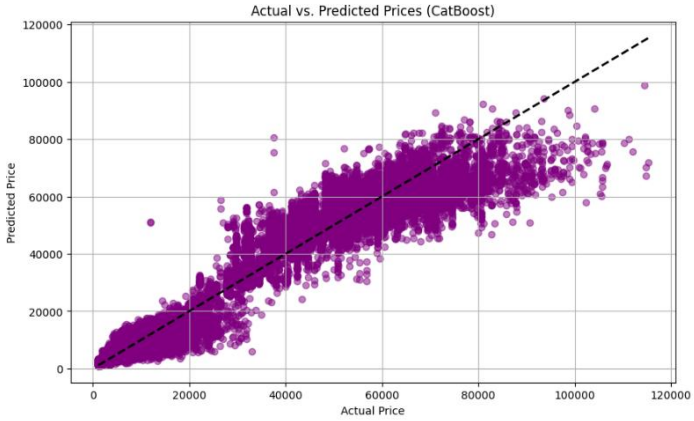
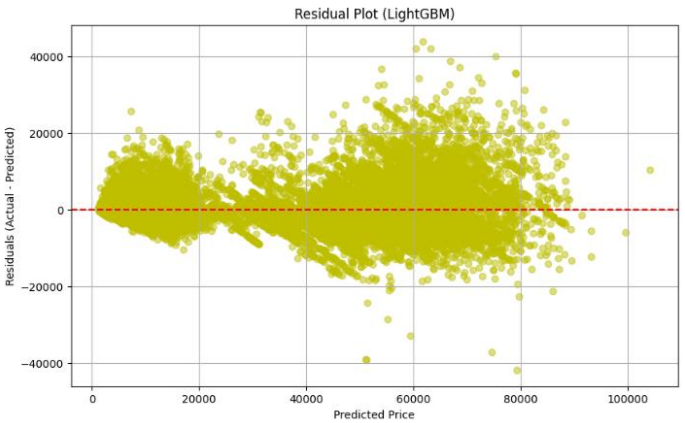
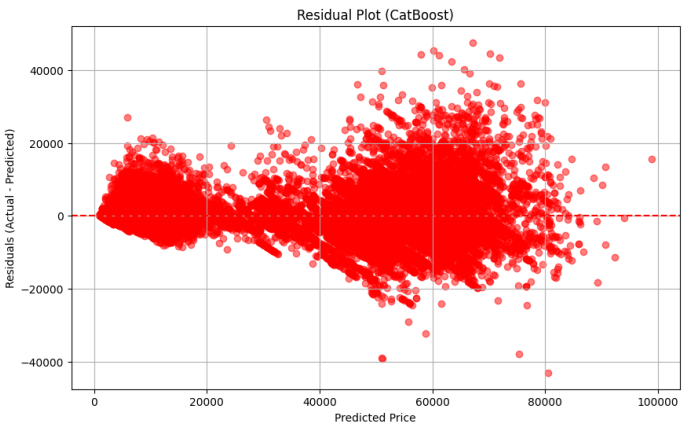


Fig. 17 Residual Plot for :

a. LightGBM



b. CatBoost



❖ SPLINES AND LASSO CV

The categorical features such as airline, source city, destination city, time slots, and routes are encoded using one-hot encoding, while numerical features like duration and days left are modeled with cubic splines to capture non-linear relationships. These transformed features are standardized and passed into a Lasso regression with cross-validation, which performs both prediction and feature selection by shrinking less relevant coefficients toward zero.

The model achieves a test **R^2 of 0.916** and a **Mean absolute error (MAE) of about 4407.3**, indicating that it explains a good portion of price variability but still leaves room for error in absolute predictions. When examining feature importance, route information has the strongest influence on prices, followed by source city and destination city, while the spline-expanded duration and days_left contribute less in terms of absolute coefficient magnitude.

The spline-based partial effect analysis shows that duration has a non-linear effect: very short flights are associated with lower prices, while medium-length flights show a rise, and then effects flatten with some fluctuations. For days_left, prices are much higher when tickets are booked very close to the departure date (e.g., 4–10 days left), and they decrease steadily as the booking is made further in advance, reflecting realistic airline pricing dynamics.

Finally, the route-level coefficients highlight specific city-pairs with particularly strong positive or negative impacts on price—for example, "Mumbai → Hyderabad" and "Hyderabad → Mumbai" have strong negative coefficients, whereas routes like "Kolkata → Chennai" and "Bangalore → Chennai" show positive ones. This suggests that the model captures market-specific pricing differences between routes, in addition to general timing and duration effects.

```
[Spline+Lasso] TEST R2: 0.916 | MAE: 4407.3

[Grouped |coef| (route / source / destination / spline blocks)]
route           : 2259.2925
source_city     : 1135.5409
destination_city: 1132.7116
duration        : 0.0000
days_left       : 0.0000

[Numeric partial effect via splines] duration
 duration  effect_vs_baseline
  2.17000        -4491.907702
  5.13875        -2487.423495
  8.10750         -965.076108
 11.07625          -35.410057
 14.04500          300.111847
 17.01375          104.293911
 19.98250         -496.818164
 22.95125        -1086.719044
 25.92000        -1165.649894

[Numeric partial effect via splines] days_left
 days_left  effect_vs_baseline
     4.000         5620.559653
     9.375         4070.018050
    14.750         1896.358862
    20.125          212.098720
    25.500            0.027203
    30.875           -6.273211
    36.250          -45.588029
    41.625         -168.000140
    47.000         -337.957333
```

```
[Top route one-hot coefficients by |coef|]:
route_Bangalore -> Mumbai     371.483738
route_Mumbai -> Bangalore     358.763657
route_Mumbai -> Delhi        -289.704294
route_Delhi -> Mumbai        -262.613973
route_Chennai -> Mumbai       208.075366
route_Mumbai -> Chennai       132.342419
route_Hyderabad -> Delhi     -107.702656
route_Bangalore -> Delhi     -105.988661
route_Delhi -> Bangalore      -85.586797
route_Delhi -> Kolkata         84.049619
route_Delhi -> Hyderabad      -69.322760
route_Hyderabad -> Mumbai      48.126187
route_Bangalore -> Chennai    -36.511824
route_Bangalore -> Kolkata     34.765070
route_Hyderabad -> Kolkata    -32.491147
dtype: float64
```

Fig 18. Actual vs Predicted price by Top-10 Routes

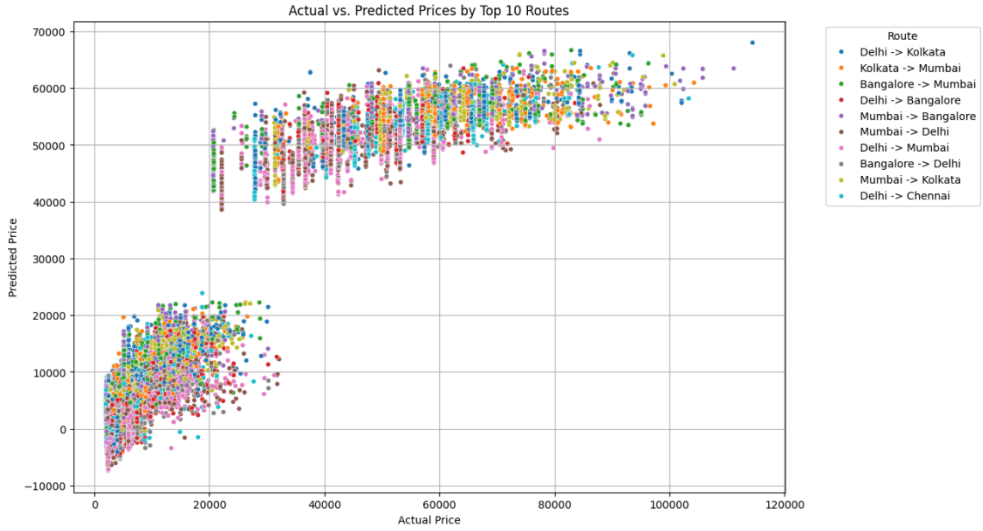


Fig. 19 Price Distribution by Source City

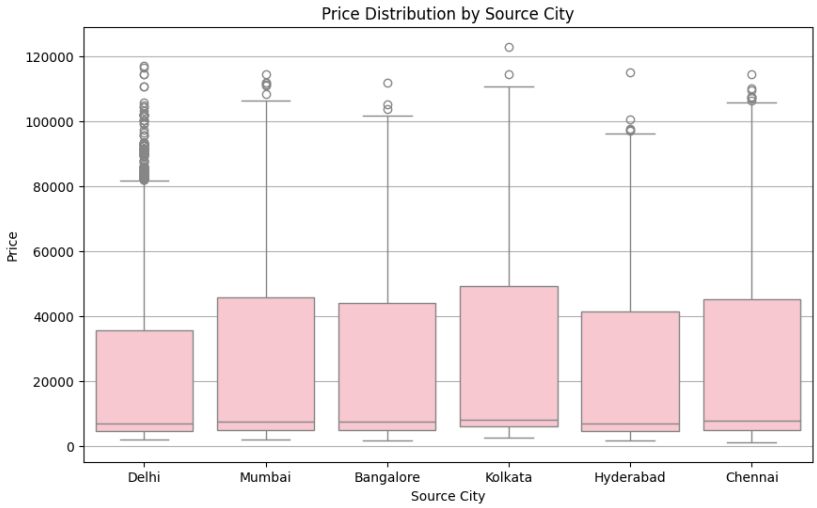
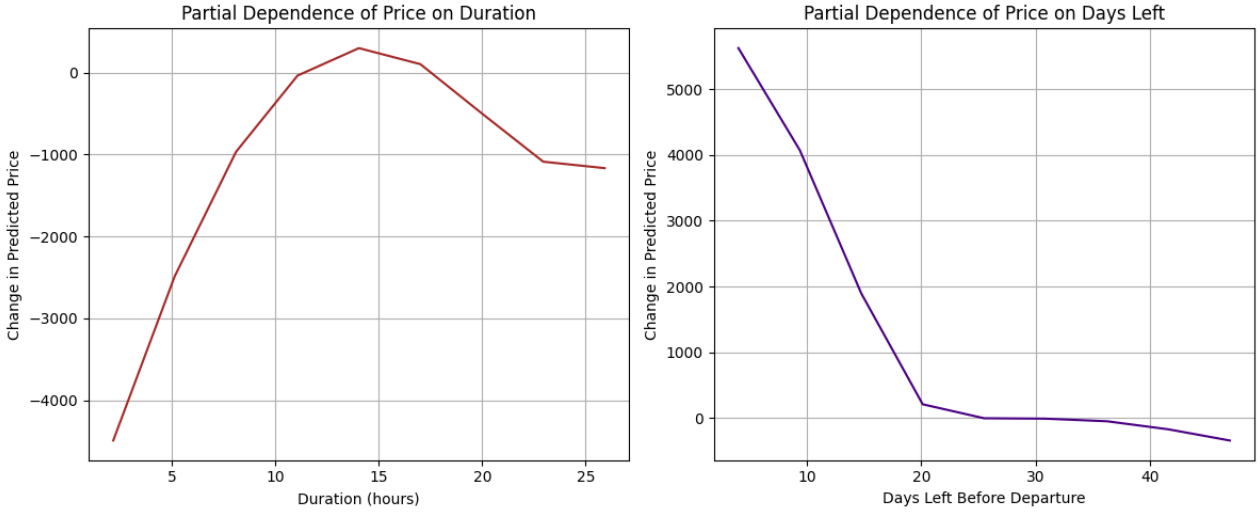


Fig. 20 Partial Distribution of Price



CHAPTER 9: CONCLUSION

This dissertation explored the use of heterogeneous regression and machine learning methods to airline ticket price prediction. Comparative analysis proved the better predictive performance of ensemble methods, also underlining the interpretive contributions of structured regression models.

Among the ensemble methods, LightGBM had the best predictive power ($R^2 = 0.98$, MAE ≈ 1700), followed closely by CatBoost ($R^2 = 0.97$, MAE ≈ 2200) and Random Forest ($R^2 = 0.986$, MAE ≈ 1108). The models were able to capture intricate non-linear interactions among predictors successfully. Feature importance analyses repeatedly ranked travel class, airline, route, duration, and booking horizon (days remaining) as the most significant predictors of ticket prices, which corresponds to well-established airline revenue management dynamics. XGBoost, not even adjusted for hyperparameter optimization, yielded strong performance ($R^2 = 0.907$, MAE = 0.216 on the scaled scale) indicating that further optimization would also make it even more competitive.

By comparison, legacy regression-based methods performed at lower predictive levels but provided significant interpretive leverage. The Spline-Lasso regression ($R^2 = 0.636$, MAE ≈ 1497) and MARS model highlighted non-linear impacts of flight duration and booking horizon, and reinforced the predominance of route-level dynamics in determining price variation. The Mixed Effects model broke down variance by source and destination city but did not account for significant residual variation, and the Two-Way ANOVA validated the weak explanatory power of city-level variables compared with route-specific effects.

Taken collectively, these results affirm that ensemble learning methods are best suited to prediction, while regression models are still useful for explanation. A hybrid approach that combines the predictive power of boosting and bagging with the interpretability of structured regressions is a complete framework for airfare price modelling. Future work can be extended by including seasonal, macroeconomic, or demand-side factors and by using systematic hyperparameter tuning to enhance model generalizability further.

REFERENCES

1. Chen, T. & Guestrin, C., 2016. XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.785–794.
2. Christoph, M., 2023. Explaining complex ML models in transportation pricing: Applications of SHAP. *arXiv preprint arXiv:2305.11234*.
3. IATA, 2021. *Dynamic pricing of airline offers*. International Air Transport Association. Available at: https://www.iata.org/contentassets/0688c780d9ad4a4fadb461b479d64e0d/dynamic-pricing--of-airline-offers.pdf [Accessed 18 August 2025].
4. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. & Liu, T.-Y., 2017. LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems (NeurIPS)*, 30.
5. Kiplinger, 2025. Airlines test AI-driven personalised pricing. *Kiplinger Business Report*, March.
6. Kumar, R., Gupta, A. & Singh, P., 2025. Flight fare prediction using machine learning: A comparative study of ensemble methods. *Journal of Air Transport Management*, 121, p.102308.
7. Lundberg, S.M. & Lee, S.-I., 2017. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems (NeurIPS)*, 30.
8. Papakostas, G.A., Karakos, A.S. & Kotsiantis, S., 2017. Forecasting air travel demand using machine learning methods. *Journal of Air Transport Studies*, 8(2), pp.1–15.
9. Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V. & Gulin, A., 2018. CatBoost: unbiased boosting with categorical features. *Advances in Neural Information Processing Systems (NeurIPS)*, 31.
10. Rajankar, S., Kshirsagar, V. & Deshmukh, A., 2019. Prediction of flight fare using machine learning algorithms. *International Journal of Computer Applications*, 178(25), pp.10–15.
11. Scheipl, F. & Kneib, T., 2021. LASSO-type penalized spline regression for binary response data. *BMC Medical Research Methodology*, 21, p.85.
12. Talluri, K.T. & van Ryzin, G., 2004. *The Theory and Practice of Revenue Management*. New York: Springer.
13. Wong, J., Smith, R. & Zhao, L., 2023. Ensemble learning for airline fare prediction: An empirical study on scraped fare datasets. *Transportation Research Part C: Emerging Technologies*, 154, p.104185.