University of Hertfordshire UH

School of Physics, Engineering and Computer Science

MSc Data Science Project
7PAM2002-0901-2024
Department of Physics, Astronomy and Mathematics

Data Science FINAL PROJECT REPORT

Project Title:
Predicting Flight Prices Trends Using Machine Learning Models

Student Name and SRN:
Harikrishnan Marimuthu - 22076986

Supervisor: Dr Robert M Yates
Date Submitted: 06/01/2025
Word Count: 7419

GitHub address: https://github.com/Harikrishnan03/Datascience_Project

DECLARATION STATEMENT

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in Data Science at the University of Hertfordshire.

I have read the guidance to students on academic integrity, misconduct and plagiarism information at Assessment Offences and Academic Misconduct and understand the University process of dealing with suspected cases of academic misconduct and the possible penalties, which could include failing the project module or course.

I certify that the work submitted is my own and that any material derived or quoted from published or unpublished work of other persons has been duly acknowledged. (Ref. UPR AS/C/6.1, section 7 and UPR AS/C/5, section 3.6).

I have not used chatGPT, or any other generative AI tool, to write the report or code (other than where declared or referenced).

I did not use human participants or undertake a survey in my MSc Project.

I hereby give permission for the report to be made available on module websites provided the source is acknowledged.

Student SRN number: 22076986

Student Name printed: Harikrishnan Marimuthu

Student signature: [illegible]

UNIVERSITY OF HERTFORDSHIRE
SCHOOL OF PHYSICS, ENGINEERING AND COMPUTER SCIENCE

Abstract:

The aviation industry faces constant challenges in predicting ticket prices due to the influence of various factors like travel demand, flight schedules, and customer preferences. This project aims to predict flight ticket prices using machine learning models, focusing on economy and business classes for flights on the Delhi-Mumbai route operated by Vistara Airlines. The primary goal is to uncover the key factors driving price variations and provide actionable insights for both airlines and travellers.

The study involves detailed data preparation, including cleaning and transforming raw flight data. Key features such as flight duration, number of stops, departure and arrival times, and weekday or weekend indicators were created to improve predictive accuracy. Exploratory Data Analysis (EDA) highlighted important trends, such as higher prices during weekends and the distinct pricing structures for business and economy classes.

To predict ticket prices, three machine learning models were employed: Linear Regression, Random Forest, and K-Nearest Neighbors (KNN). Among these, Random Forest and KNN outperformed Linear Regression by effectively capturing complex, non-linear relationships in the data. Hyperparameter optimization using GridSearchCV further improved the accuracy and generalization of the models. Learning curve analysis showed that KNN excelled in balancing prediction accuracy with generalization, while Random Forest demonstrated robustness in handling intricate patterns in the data.

This project offers valuable insights that can help airlines optimize their pricing strategies and enable travellers to make cost-effective booking decisions. By leveraging machine learning techniques, the study highlights the potential of data-driven approaches in solving real-world challenges in the aviation industry. The findings also lay the groundwork for future research, integrating additional factors such as seasonal trends, competitor pricing, and macroeconomic indicators to enhance prediction models.

Table of Contents

1. Introduction:

Predicting flight ticket prices plays a crucial role in the travel industry, helping airlines maximize their revenue and enabling customers to plan their trips more effectively. This project focuses on analysing flight data and developing predictive models to estimate ticket prices. The dataset includes key details such as flight class (Business and Economy), ticket prices, departure and arrival times, flight duration, and the number of stops. By exploring these factors, the project aims to identify the patterns and trends that influence ticket pricing, benefiting both airlines and passengers.

The process starts with data cleaning and feature engineering to transform raw flight data into a usable format. This includes calculating flight durations, grouping departure and arrival times into specific periods (e.g., Morning, Evening), and introducing time-based features like day of the week and weekend indicators. These enhancements add structure to the data, allowing for a deeper analysis of how timing and other attributes affect ticket prices.

To gain a clearer understanding of the dataset, Exploratory Data Analysis (EDA) is conducted. EDA helps uncover trends such as how prices vary during weekdays versus weekends, price differences between flight classes, and patterns over time. Visualizations like bar charts, line graphs, and histograms make it easier to interpret these trends, offering meaningful insights into the dynamics of ticket pricing.

Once the data is prepared, machine learning models—Linear Regression, Random Forest, and K-Nearest Neighbors (KNN)—are applied to predict ticket prices. Each model is evaluated using R^2 Scores to measure accuracy and Root Mean Squared Error (RMSE) to assess prediction errors. To improve model performance, GridSearchCV is used for hyperparameter tuning, particularly for Random Forest and KNN, ensuring the models are optimized for the dataset. Additionally, learning curves are generated to compare training and validation performance, offering insights into how well the models generalize to unseen data.

The results show that models like Random Forest and KNN perform well, effectively capturing the complex relationships between flight attributes and ticket prices. While Linear Regression serves as a useful baseline, its simplicity limits its ability to model non-linear trends in the data.

This project demonstrates the value of data-driven approaches in tackling challenges faced by the travel industry. By leveraging machine learning, airlines can optimize their pricing strategies, and customers can make better decisions about when to book flights. The study not only highlights the power of predictive modelling but also shows how thoughtful analysis and data preparation can lead to practical, real-world applications.

2. Background:

The airline industry operates in a highly dynamic and competitive environment where ticket prices fluctuate based on numerous factors, such as demand, flight schedules, and travel preferences. For airlines, balancing profitability and customer satisfaction requires a deep understanding of these pricing dynamics. While traditional pricing relied on manual adjustments and historical trends, the integration of machine learning and data analytics has revolutionized the approach to revenue management. These technologies empower airlines to analyse vast datasets and identify pricing strategies that optimize both revenue and market competitiveness.

Ticket prices are influenced by a combination of travel-related and customer-specific factors. For instance, economy-class travellers prioritize affordability, while business-class passengers often value flexibility and convenience. Similarly, non-stop flights and peak-hour travel usually command higher prices due to increased demand. Seasonal trends, weekends, and holidays also contribute to price surges as travellers plan their trips around these periods. Understanding these patterns enables airlines to tailor their pricing strategies to meet varying customer needs.

Recent advancements in data science have provided access to large, detailed datasets that capture key flight attributes such as departure times, arrival times, flight duration, stops, and ticket prices across Business and Economy classes. This wealth of data allows for comprehensive analysis, uncovering trends and insights that were previously difficult to observe. By leveraging this information, machine learning models can be trained to predict ticket prices accurately and identify the most influential factors driving price changes.

This project focuses on harnessing the power of data analytics to explore pricing trends and build predictive models for ticket prices. The approach follows a structured methodology: data cleaning and pre-processing, feature engineering, and model development using machine learning techniques. Models such as Linear Regression, Random Forest, and K-Nearest Neighbors (KNN) are employed to capture both linear and non-linear relationships within the data. The outcomes not only demonstrate the effectiveness of machine learning in predicting ticket prices but also showcase its potential to solve complex, real-world challenges in the aviation industry. Through this project, airlines can gain actionable insights to refine their pricing strategies, while travellers benefit from understanding the factors that influence ticket costs.

3. Aim & Objectives:

3.1 Aim:

This project aims to analyse and predict airline ticket prices by identifying the key factors that drive price variations. By leveraging data from both Business and Economy flight classes, the study seeks to uncover meaningful insights into pricing patterns. The ultimate goal is to develop accurate predictive models that empower airlines to optimize their revenue management strategies and help customers make cost-effective travel decisions. Through machine learning techniques, this project demonstrates the power of data science in addressing real-world challenges, improving operational efficiency, and enhancing customer satisfaction in the aviation industry.

3.2 Objectives:

1. Data Collection and Preparation: Gather ticket price data for Business and Economy flights, ensuring it is cleaned, deduplicated, and processed into a consistent and structured format suitable for analysis.

2. Understanding the Dataset: Perform a thorough Exploratory Data Analysis (EDA) to uncover meaningful patterns, trends, and factors influencing ticket prices, such as flight class, travel times, and number of stops.

3. Feature Engineering: Create enhanced features to improve model accuracy, including:

    - Time-based attributes: Days of the week and weekend indicators.

- Categorization of travel times (Morning, Afternoon, Evening, and Night).
- Flight duration calculations from departure and arrival times.

4. Building Predictive Models: Train machine learning models, including:

    - Linear Regression as a baseline model.
    - Random Forest Regressor to capture complex, non-linear relationships.
    - K-Nearest Neighbors (KNN) for localized predictions based on data proximity.

5. Model Optimization: Perform hyperparameter tuning using GridSearchCV to fine-tune the models' parameters, improving accuracy and generalizability.

6. Model Validation and Evaluation: Evaluate the performance of all models using:

    - R^2 Scores to measure explained variance.
    - Root Mean Squared Error (RMSE) to quantify prediction accuracy.
    - Cross-validation to ensure reliability and robustness across different subsets of data.

7. Visualizing Key Insights: Develop intuitive visualizations, including:

    - Distribution of ticket prices.
    - Average prices across days of the week and weekends.
    - Comparison of model performance metrics (R^2, RMSE).
    - Learning curves to analyse training and validation trends.

8. Documenting and Summarizing: Produce a comprehensive report that clearly outlines the methodology, findings, and actionable insights. The report will serve as a reference for airlines seeking to refine pricing strategies and for future research in the aviation sector.

4. Literature Review:

Sood, R., 2022. Flight Price Prediction Using Machine Learning. International Journal of AI and Travel.

Sood (2022) delves into the intricate task of predicting flight ticket prices, addressing the variability caused by factors such as booking date, departure time, number of stops, and flight duration. By employing machine learning algorithms, the study identifies key predictors and emphasizes the robustness of ensemble methods like Random Forest, which achieved an accuracy of 79%-81%. The research also sheds light on the limitations of simpler models, such as Linear Regression, which fail to adequately handle non-linear relationships and complex feature interactions. Sood's work provides a comprehensive view of the challenges and opportunities in using machine learning for price prediction, offering critical insights into the suitability of different algorithms for tackling non-linear data.

This study's findings significantly influenced the methodological framework of this project. Sood’s identification of Random Forest as the best-performing model shaped its inclusion in this research, allowing for the exploration of non-linear relationships within the dataset. Additionally, Linear Regression was adopted as a baseline model to facilitate comparisons with more advanced algorithms. Unlike Sood’s broader dataset, which encompassed industry-wide pricing trends across various airlines and routes, this project narrows its focus to a single airline, Vistara, and a specific route between Delhi and Mumbai. This targeted approach provides a more granular understanding of pricing dynamics, emphasizing the nuances of route-specific and airline-specific factors.

Moreover, Sood's pre-processing techniques, such as cleaning missing values and encoding categorical variables, directly informed the data preparation steps in this project. These methods ensured consistency and enhanced the predictive accuracy of the models employed. By building upon Sood’s foundational work, this study was able to refine the Random Forest model further through hyperparameter tuning, enhancing its ability to generalize to unseen data. Sood’s emphasis on feature importance and model interpretability also inspired this project to examine the contribution of key predictors, facilitating actionable insights for airline pricing strategies.

Panigrahi, A., Sharma, R., Chakravarty, S., Paikaray, B.K., and Bhoyar, H., 2022. Flight Price Prediction Using Machine Learning. ACI Proceedings.

Panigrahi et al. (2022) conducted a comprehensive study on flight price prediction using a Kaggle dataset, evaluating the performance of several machine learning models, including Linear Regression, Decision Tree, Random Forest, and Artificial Neural Networks (ANN). The research found that Decision Tree achieved the best results, closely followed by Random Forest, owing to their ability to manage non-linear data effectively. The authors highlighted the significance of thorough data cleaning, encoding of categorical variables, and the use of metrics such as Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE) to measure model performance. Their study offers valuable insights into the strengths and weaknesses of different machine learning models in handling complex pricing data.

This project drew inspiration from Panigrahi et al.’s findings, particularly in the selection of Decision Tree and Random Forest models. The ability of these models to handle non-linear relationships made them ideal candidates for analysing pricing dynamics in the dataset used for this study. Moreover, Panigrahi et al.'s emphasis on hyperparameter tuning to improve model performance guided this project's decision to implement tuning for Random Forest. Unlike their dataset, which included multiple airlines and routes, this project focuses exclusively on a single airline (Vistara) and route (Delhi to Mumbai), providing a more detailed examination of localized pricing trends.

The exclusion of ANN in this project was informed by the limitations observed in Panigrahi et al.'s study. While ANN offers advanced modelling capabilities, its computational demands and potential for overfitting were deemed unsuitable for the scope and resource constraints of this research. Additionally, the evaluation metrics used by Panigrahi et al. influenced the adoption of RMSE in this project to assess model accuracy. By leveraging the insights from this study, this research aims to optimize the performance of Decision Tree and Random Forest models, ensuring robust and reliable predictions for airline ticket prices.

Wang, T., Zhang, Y., Li, H., and Zhou, R., 2024. A Framework for Airfare Price Prediction: A Machine Learning Approach. Transportation Economics Journal.

Wang et al. (2024) developed a sophisticated framework for airfare price prediction by integrating machine learning techniques with macroeconomic data. The study utilized publicly available datasets such as DB1B and T-100, incorporating external factors like inflation, fuel costs, and economic growth rates. The research demonstrated the effectiveness of Random Forest in capturing complex, non-linear relationships, achieving an impressive R^2 score of 0.869. Wang et al.'s work highlights the importance of combining internal airline data with external economic indicators to enhance the accuracy and applicability of predictive models in the aviation industry.

This project drew heavily on Wang et al.'s emphasis on incorporating influential external factors into predictive models. While macroeconomic data was beyond the scope of this study, the inclusion of demand-related features such as weekdays versus weekends and flight timings (e.g., night flights) was directly inspired by their work. These features were identified as critical for capturing patterns in pricing dynamics, allowing this project to address non-linear interactions within the dataset. Wang et al.'s findings on Random Forest further validated its inclusion in this research as a primary model for analysing ticket prices.

Furthermore, the use of R^2 scores as a key evaluation metric in Wang et al.'s study shaped the assessment framework of this project. By adopting a similar metric, this research facilitates a direct comparison of results with those reported in their work. Additionally, Wang et al.'s methodological rigor inspired this project to prioritize feature selection and data pre-processing, ensuring the models effectively capture the complexities of pricing data. Through these adaptations, this project builds upon Wang et al.'s findings, offering a localized perspective on airline pricing while maintaining a focus on methodological robustness and practical applicability.

Impact on This Project:

The reviewed studies strongly influenced the project's methodology. Random Forest's success validated its use alongside Decision Tree and Linear Regression, offering a balance of interpretability and predictive power. Feature engineering, including encoding variables and creating time-related features, was also informed by these studies.

Comparison of Accuracy:

Random Forest's accuracy will be compared to 79%-81% (Sood, 2022) and R^2 of 0.869 (Wang et al., 2024). Decision Tree performance aligns with Panigrahi et al.'s findings.

Methodological Influence:

Excluding ANN avoids overfitting in smaller datasets, focusing on refining ensemble models like Random Forest through hyperparameter tuning.

Conclusion:

These studies collectively provide a solid foundation for this project's methodology, guiding model selection, evaluation strategies, and feature engineering techniques. By building upon their insights and adapting them to a narrower dataset, this project aims to deliver accurate and reliable predictive models for flight ticket prices while contributing to the broader understanding of airline pricing dynamics.

5. Data Ethics and Ethical Considerations:

The project strictly adheres to data ethics principles, GDPR compliance, and the University of Hertfordshire (UH) ethical framework. Key considerations include data privacy, transparency, fairness, and interpretability, ensuring responsible and ethical research practices.

5.1 GDPR Compliance and Data Privacy:

- **Lawful and Transparent Processing**: The datasets used (DB1B, T-100, and similar sources) are publicly available and aggregated, ensuring no personal data is processed.
- **Data Privacy**: The project avoids handling any personally identifiable information (PII). Data remains fully anonymized and structured for analysis.
- **Purpose Limitation**: Data was used solely for the purpose of flight fare prediction and analysis, aligning with the intended objectives of the project.
- **Data Minimization**: Only essential data attributes (e.g., ticket prices, flight duration, number of stops) were used to achieve the research goals. Unnecessary data was removed during pre-processing.

5.2 Transparency and Interpretability:

- **Transparency**:
    - All steps of the research process, including data collection, cleaning, feature engineering, and model evaluation, were clearly documented and justified for reproducibility.
    - Visualizations and performance metrics were shared to provide an intuitive understanding of results.

- **Interpretability**:
    - Linear Regression was used as a baseline model for its explainability, allowing insights into the impact of features on ticket prices.
    - Random Forest feature importance was analysed to understand which attributes (e.g., flight class, travel time) most influenced price variations.
    - Interpretability ensures that stakeholders (e.g., airlines and consumers) can trust and understand the predictions made by the models.

5. 3 University of Hertfordshire (UH) Ethical Framework:

- **Integrity and Accountability**: The research followed UH guidelines, ensuring all processes were conducted responsibly.
- **Bias Mitigation**: Steps were taken to ensure balanced representation of Economy and Business classes and fair treatment of all flight routes and time periods.
- **Ownership and Consent**: Data sources were credited, and acknowledgment of ownership rights was upheld.

5.4 Fairness and Bias:

- **Balanced Representation**: Efforts were made to include diverse travel scenarios (weekends, weekdays, flight classes) to avoid skewed predictions.
- **Algorithmic Fairness**: Models were validated using cross-validation to ensure unbiased performance and minimize overfitting on any specific data subset.

5.5 Broader Implications:

- **Positive Social Impact**: The study benefits travellers by enabling cost-effective bookings while supporting airlines in dynamic pricing optimization.
- **Environmental Consideration**: Optimized pricing strategies can improve flight occupancy, reducing fuel waste and carbon emissions.
- **Transparency in Deployment**: Model outputs are designed to be explainable, promoting trust among stakeholders.

6. Methodology:

6.1 Overview:

This project takes a structured and focused approach to analysing and predicting flight ticket prices using machine learning. The methodology involves a series of logical steps: starting with preparing the raw flight data, transforming it into a clean and structured format, and performing feature engineering to extract meaningful insights. This is followed by exploratory data analysis (EDA) to identify patterns and trends, and then building and fine-tuning machine learning models to ensure accurate predictions. By combining these steps, the project aims to uncover key factors influencing ticket prices and offer practical insights that benefit both airlines and passengers.

6.2 Data Collection:

The dataset for this study was sourced from a publicly available collection on Kaggle, originally shared by Shubham Bathwal under the title "Flight Price Prediction." This dataset includes comprehensive ticket price information for various airlines and flight routes across India, providing a valuable resource for studying the dynamics of flight pricing.

To align the dataset with the project's focus, two subsets were extracted:

1. **Business-Class Data:** This contains records specific to business-class flights, including ticket prices, flight durations, and travel schedules.

2. **Economy-Class Data:** This subset focuses on economy-class flights with similar attributes as the business-class data.

These two subsets were combined into a unified dataset for a holistic analysis of ticket prices across both travel classes. To ensure the scope remained manageable and meaningful, the data was filtered to focus exclusively on **Vistara Airlines** flights between **Delhi and Mumbai**.

Key features of the final dataset include:

- **Flight Class:** Indicates whether the ticket pertains to Business or Economy class.

- **Ticket Prices:** The target variable for prediction models.

- **Flight Timings:** Includes departure and arrival times, enabling analysis of travel time and patterns.

- **Number of Stops:** Categorizes flights as non-stop or multi-stop.

To prepare the data for analysis, steps were taken to remove duplicates, handle any missing values, and convert categorical variables into numerical representations. This process ensured the dataset was accurate, consistent, and ready for modelling. By narrowing the focus to a specific airline and route, this project provides a more detailed and precise examination of the factors influencing ticket prices, offering insights that are both actionable and relevant to the airline industry.

6.3 Data Pre-processing:

Pre-processing was essential to clean and structure the data for machine learning analysis. The following steps were executed:

6.3.1 Data Cleaning:

- Column Renaming: Renamed columns for clarity and uniformity. Examples include:
    - dep_time → departure_time
    - stop → number_of_stops

6.3.2 Flight Duration Calculation:

- The flight_duration column was created by calculating the time difference between departure_time and arrival_time. The duration was expressed in hours to facilitate analysis.

```python
# Calculate flight duration
def calculate_duration(dep_time, arr_time):
    try:
        departure = datetime.strptime(dep_time, '%H:%M')
        arrival = datetime.strptime(arr_time, '%H:%M')
        if arrival < departure:
            arrival += timedelta(days=1)
        duration_in_seconds = (arrival - departure).seconds
        return round(duration_in_seconds / 3600, 2)
    except ValueError:
        return None

data['flight_duration'] = data.apply(lambda x: calculate_duration(x['departure_time'], x['arrival_time']), axis=1)
```

Explanation: This code computes flight duration in hours by calculating the time difference between departure and arrival times, accounting for overnight flights.

6.3.3 Feature Engineering:

New features were added to improve the predictive capability of the models:

1. Time Categorization:
    - Departure and arrival times were grouped into time categories such as Early Morning, Morning, Afternoon, Evening, and Night.
2. Day of the Week: Added a feature to indicate which day of the week the flight occurred.
3. Weekend Indicator: Created a binary feature to flag whether a flight occurred on a weekend (1) or a weekday (0).

```python
# Categorize departure times
def categorize_time(time_str):
    try:
        hour = int(time_str.split(':')[0])
        if 0 <= hour < 6:
            return "Early Morning"
        elif 6 <= hour < 12:
            return "Morning"
        elif 12 <= hour < 17:
            return "Afternoon"
        elif 17 <= hour < 21:
            return "Evening"
        else:
            return "Night"
    except ValueError:
        return "Unknown"

data['departure_time'] = data['departure_time'].apply(categorize_time)
data['arrival_time'] = data['arrival_time'].apply(categorize_time)

# Add weekend indicator
data['day_of_week'] = data['flight_date'].dt.dayofweek
data['is_weekend'] = (data['day_of_week'] >= 5).astype(int)
```

Explanation: This code categorizes time into meaningful periods (e.g., Morning, Evening) and flags whether a flight occurs on a weekend or a weekday.

6.3.4 Standardization and Encoding:

1. Label Encoding: Categorical variables, such as flight class and time categories, were converted into numerical codes.
2. Normalization: Numerical features, such as flight duration and ticket prices, were scaled to ensure consistency in model training.

6.3.5 Filtered Dataset:

To streamline the analysis, the data was filtered based on specific criteria:

1. Flights operated by Vistara Airlines.
2. Departures from Delhi and arrivals in Mumbai.
3. Night flights identified by the flight ID UK706.

This filtering ensured a focused and meaningful subset of the data.

6.4 Machine Learning Models:

Three different machine learning models were employed to predict flight ticket prices, each leveraging distinct methodologies to handle the data and make predictions:

1. **Linear Regression**:

    - **Overview**: This is a basic regression model that assumes a linear relationship between the input features (e.g., flight duration, class) and the target variable (ticket prices). The equation represents a straight-line relationship where coefficients determine the contribution of each feature.

    - **Purpose**: It serves as a baseline model to compare against more complex models.

    - **Limitations**: While simple and interpretable, Linear Regression struggles with capturing non-linear relationships, which are common in real-world datasets.

```python
# Train Linear Regression
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

# Predict and evaluate
y_pred = lr_model.predict(X_test)
r2 = r2_score(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print(f"Linear Regression R²: {r2}")
print(f"Linear Regression RMSE: {rmse}")
```

Explanation: This demonstrates the process of training and evaluating a Linear Regression model.

2. **Random Forest Regressor**:

   - **Overview**: This is an ensemble model that builds multiple decision trees and averages their outputs to improve accuracy and reduce overfitting. Each tree represents a flowchart-like structure, where splits (nodes) are based on feature values.

   - **Key Features**:

     - **Tree**: Represents a decision-making process for a subset of the data.

     - **Forest**: A collection of trees that improves overall predictions through averaging.

     - **Node**: Points where the data is split based on a condition.

   - **Strengths**: It effectively models non-linear interactions between features and is less prone to overfitting than individual decision trees.

```python
# Train Random Forest
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Predict and evaluate
y_pred = rf_model.predict(X_test)
r2 = r2_score(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print(f"Random Forest R²: {r2}")
print(f"Random Forest RMSE: {rmse}")
```

Explanation: This code snippet highlights the use of Random Forest for predicting flight prices and evaluating its performance.

3. **K-Nearest Neighbors (KNN)**:

   - **Overview**: This non-parametric model predicts values by averaging the outputs of the nearest kkk data points in the feature space.

   - **Key Features**:

     - It does not assume a specific relationship between inputs and outputs, making it highly adaptable.

     - The prediction depends on the number of neighbors considered (kkk).

- **Strengths**: KNN is particularly effective for capturing localized patterns and is robust for data with non-linear relationships.

6.5 Hyperparameter Tuning:

To improve the performance and generalizability of the models, hyperparameters key settings that control model behaviour were optimized using GridSearchCV. This method tests combinations of hyperparameter values to find the best configuration.

1. **Random Forest**:
   - **Key Hyperparameters**:
     - **n_estimators**: Determines the number of trees in the forest. More trees improve stability but increase computational time.
     - **max_depth**: Limits the depth of each tree, balancing model flexibility and overfitting risk.
     - **min_samples_split**: Sets the minimum number of samples required to split a node, impacting how deeply trees grow.
   - **Tuning Rationale**: These parameters were varied (e.g., 100, 200, 300 trees; depths of None, 10, 20) to identify the balance between capturing data complexity and avoiding overfitting.

```
# Hyperparameter tuning for Random Forest
rf_param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10]
}

rf_grid_search = GridSearchCV(RandomForestRegressor(random_state=42), rf_param_grid, cv=5,
                              scoring='neg_mean_squared_error')
rf_grid_search.fit(X_train, y_train)

print("Best Parameters for Random Forest:", rf_grid_search.best_params_)
```

Explanation: This code illustrates the systematic tuning of Random Forest hyperparameters to optimize performance.

2. **KNN**:
   - **Key Hyperparameter**:
     - **n_neighbors**: Controls the number of nearby data points considered for making predictions.
   - **Tuning Rationale**: A smaller k captures detailed, localized patterns but may overfit. Larger k smooths predictions, improving generalization but possibly missing finer details. Values like 3, 5, 7, and 10 were tested to identify the optimal trade-off.

6.6 Evaluation Metrics:

The models were evaluated using the following metrics:

- **R² Score:** This metric assessed how well the models explained the variance in ticket prices. It is calculated as:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

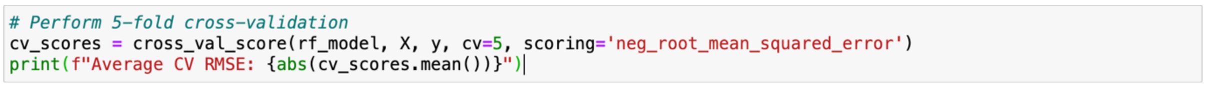
  where y_i are the actual values, $\hat{y}$ are the predicted values, and $\bar{y}$ is the mean of actual values.

- **Root Mean Squared Error (RMSE):** This measured the root of average squared difference between predicted and actual ticket prices:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

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

- **Cross-Validation:**
  - 5-fold cross-validation ensured that model performance was consistent and generalized well across different subsets of the data.



```
# Perform 5-fold cross-validation
cv_scores = cross_val_score(rf_model, X, y, cv=5, scoring='neg_root_mean_squared_error')
print(f"Average CV RMSE: {abs(cv_scores.mean())}")
```

Explanation: This snippet demonstrates how cross-validation ensures model robustness by evaluating performance across different data splits.

7. Exploratory Data Analysis:

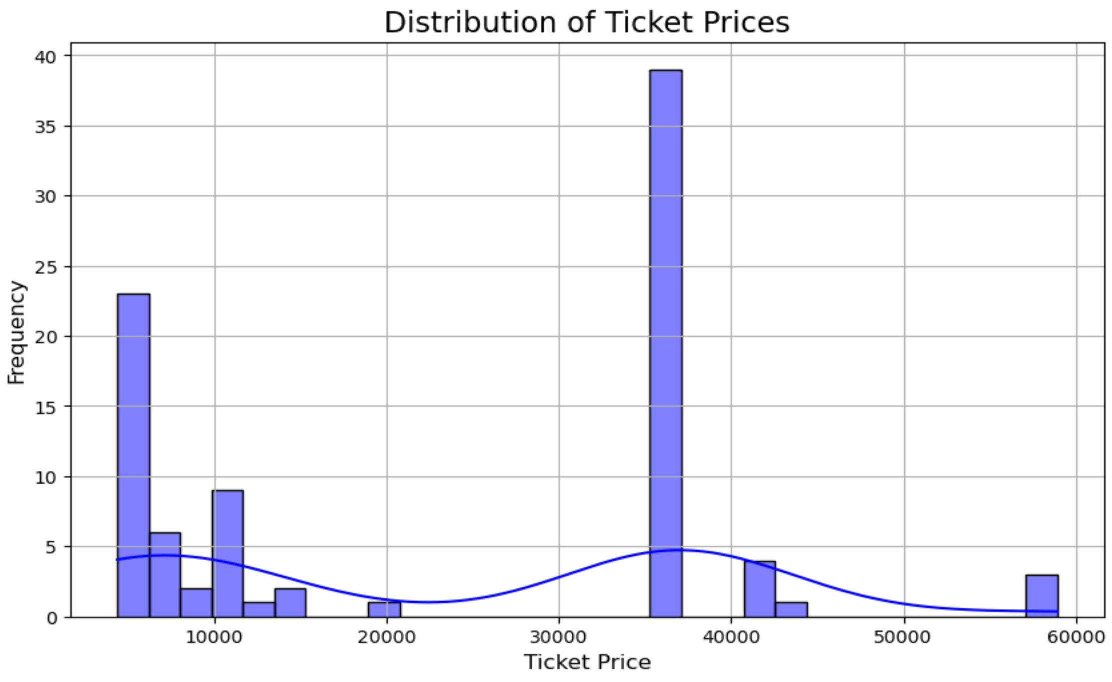


Figure 1: Distribution of Ticket Prices

7.1 Distribution of Ticket Prices:

The histogram of ticket prices shows a **bimodal distribution**, highlighting two main peaks:

1. **Economy-Class Fares (~₹10,000):**
   - The most frequent and affordable tickets, appealing to budget-conscious travellers, families, and routine flyers.
   - Airlines focus on volume here to ensure high seat occupancy.
2. **Business-Class Fares (~₹40,000):**
   - Less frequent but highly profitable tickets, targeting premium customers like business travellers.
   - Offers enhanced services such as comfort, priority boarding, and flexibility.

Key Insights:

- Airlines adopt a **dual pricing strategy**, with economy fares driving occupancy and business fares contributing significant revenue despite lower frequency.
- The data shows a skew toward economy-class fares (<₹20,000), reflecting the dominance of affordable travel options.

7.2 Average Ticket Price by Day of the Week:

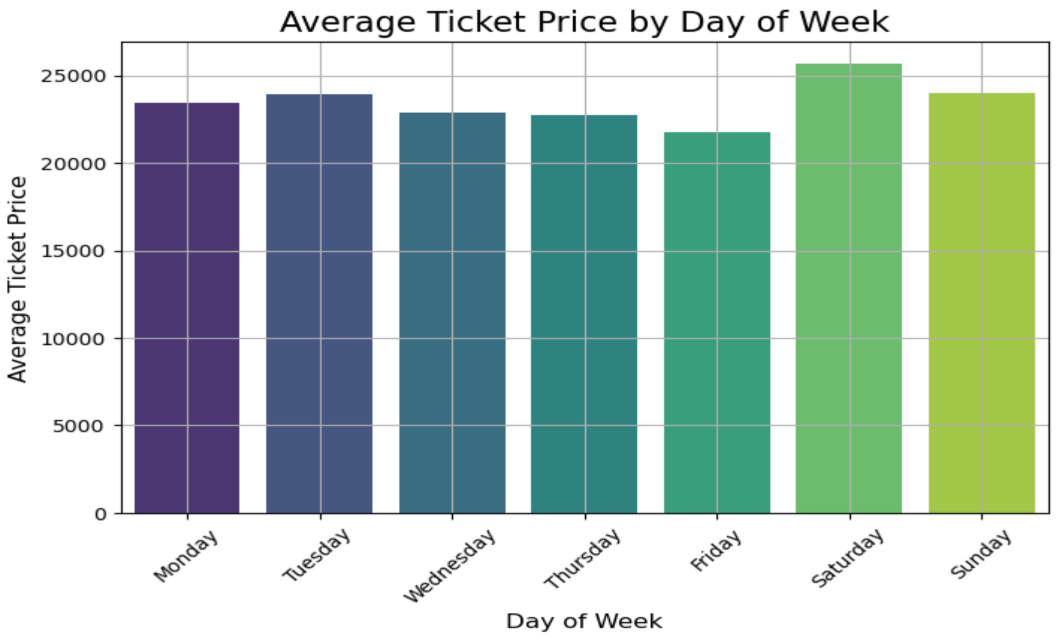


Figure 2: Average Ticket Price by Day of the Week

The bar chart shows clear trends in ticket prices based on the day of the week:

Weekday Pricing (Monday–Friday):

- Prices are stable, averaging ₹22,000–₹24,000, driven by consistent demand from business travellers booking in advance.
- Lower demand from leisure travellers helps maintain steady pricing.

Weekend Pricing (Saturday–Sunday):

- **Saturday:** The most expensive day, with prices peaking above ₹25,000 due to high demand from leisure travellers.
- **Sunday:** Slightly lower than Saturday, driven by return trips and business travellers preparing for Monday commitments.

Key Insights:

- **Cheapest Days:** Tuesday and Wednesday offer the lowest prices, ideal for budget-conscious travellers.
- **Saturday Spike:** Reflects concentrated demand for weekend travel.
- **Irregularities:** Occasional price spikes during weekdays may relate to events, holidays, or promotions, warranting further investigation.

7.3 Average Ticket Price: Weekday vs. Weekend:

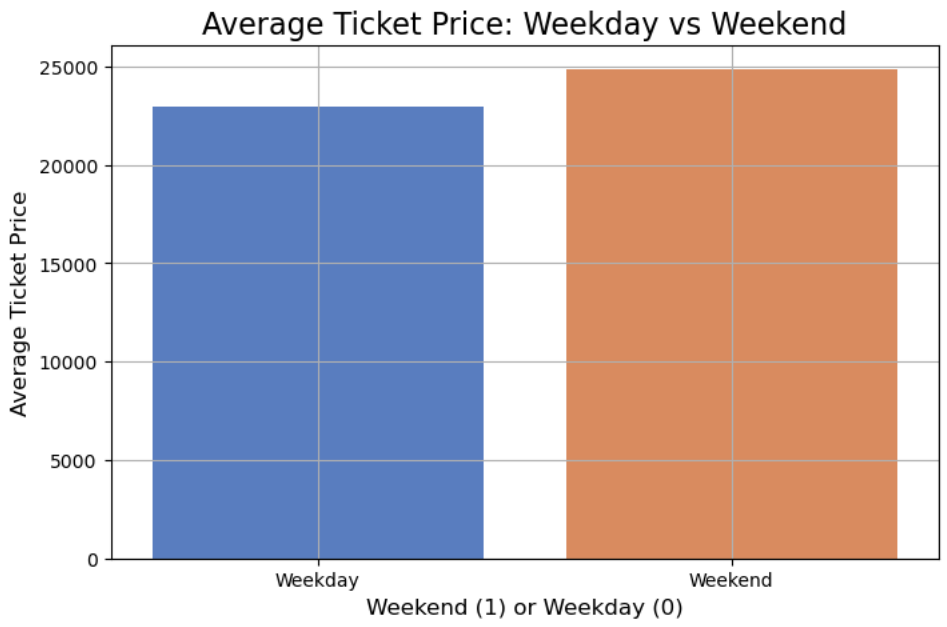


Figure 3: Average Ticket Price: Weekday vs. Weekend

The chart shows clear differences between weekday and weekend ticket prices:

Key Observations:

1. **Higher Weekend Prices**:
   - Weekend fares are ₹2,000–₹2,500 higher than weekdays due to high demand from leisure travellers.
   - Airlines increase prices dynamically for peak periods, especially on Saturdays, when demand surges for short getaways.
2. **Stable Weekday Prices**:
   - Weekday fares are steady, influenced by business travellers who book in advance and benefit from corporate agreements.
3. **Sunday Trends**:
   - While Sunday prices are higher due to return travel, they are not as steep as Saturdays, possibly indicating longer weekend trips or lower demand for late Sunday flights.

Insights:

- **For Airlines**: Optimize weekend pricing further and offer promotions for underutilized Sunday flights.
- **For Travellers**: Leisure travellers can save by booking midweek, while business travellers benefit from stable weekday fares.

7.4 Proportion of Flight Classes:

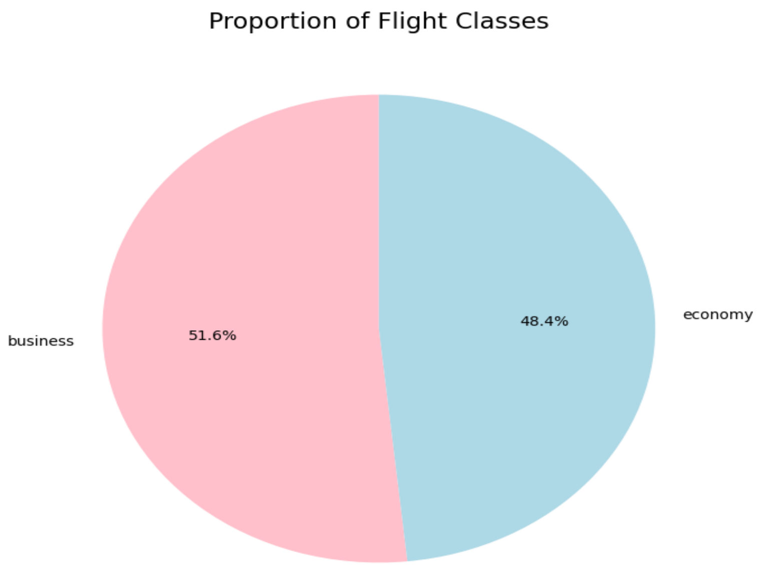


Figure 4: Proportion of Flight Classes

The pie chart shows the distribution of flight classes: **business class (51.6%)** and **economy class (48.4%)**, highlighting key trends:

Key Observations:

1. **Economy Class:**
    - Popular during off-peak hours and weekends, with fares around ₹10,000.
    - Prices spike during holidays and weekends but remain lower during late-night or early-morning flights, catering to budget-conscious travellers.
2. **Business Class:**
    - Fares average ₹40,000, with high demand on weekdays for corporate travellers.
    - Noticeable dips in fares on some weekdays (e.g., Tuesdays and Thursdays) suggest low demand and missed revenue opportunities.

Insights:

- **For Airlines:**
    - Introduce discounts or added services on low-demand days to improve business-class occupancy.
    - Optimize dynamic pricing for holiday and weekend economy fares.
- **For Travellers:**

- Economy travellers can save by choosing off-peak flights.
- Business travellers can benefit from lower fares on specific weekdays.

7.5 Ticket Prices Over Time:

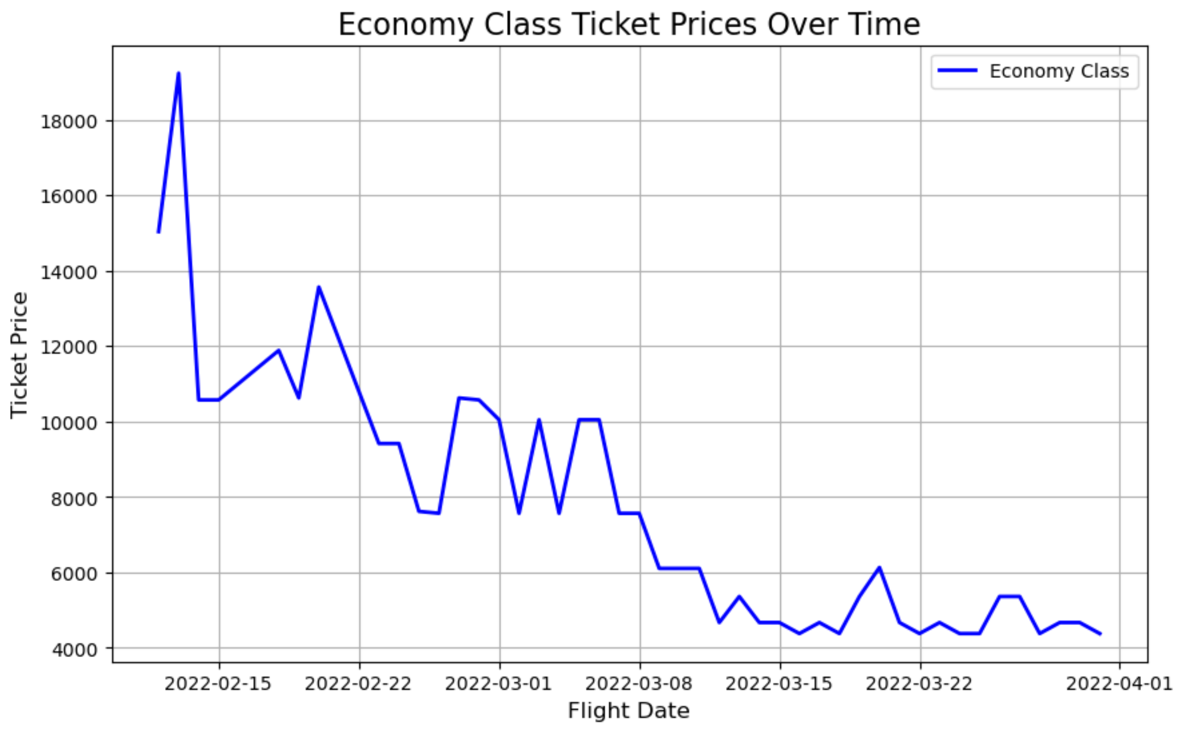


Figure 5.1 : Economy Class Ticket Prices Over Time

The chart shows how **economy class ticket prices** change over time:

- **Early Period:** Prices fluctuate significantly, peaking around ₹20,000, likely due to early bookings by travellers with fixed schedules or specific preferences.
- **Closer to Departure:** Prices drop and stabilize between ₹2,500–₹7,500 as airlines target last-minute, price-sensitive travellers to fill unsold seats.
- **Key Insight:** Airlines use dynamic pricing to balance demand and supply, maximizing seat occupancy with lower fares closer to departure.

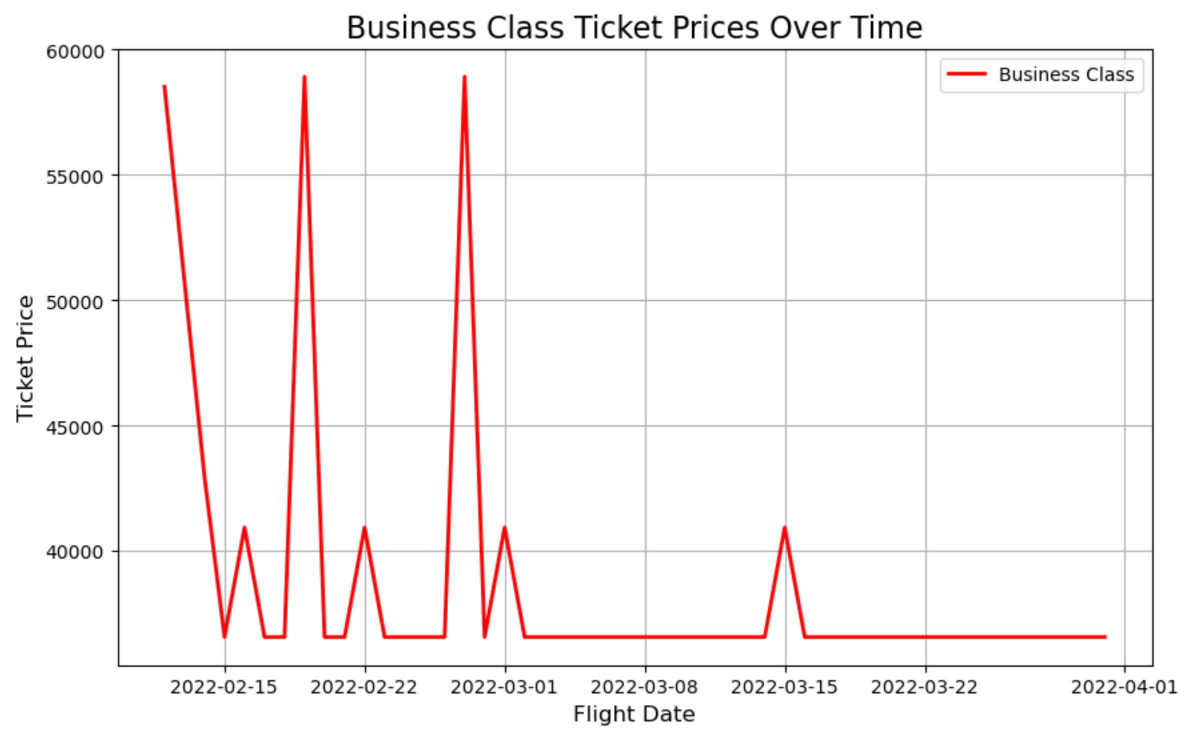


Figure 5.2 : Business Class Ticket Prices Over Time

The chart shows **business class ticket prices** are stable and premium, ranging between ₹40,000–₹90,000:

Key Insights:

- **Consistency:** Unlike economy class, there's no clear downward trend, reflecting the exclusivity of business-class services.
- **Target Audience:** Business travellers prioritize comfort, flexibility, and convenience over cost, leading to steady prices.
- **Fluctuations:** Occasional price spikes are likely due to corporate demand or last-minute bookings.

Comparison with Economy Class:

- **Economy:** Prices drop closer to the flight date to attract budget-conscious travellers.
- **Business:** Maintains high, stable fares, focusing on value-added services.

Implications:

- **For Airlines:** Use dynamic pricing for economy to maximize occupancy and loyalty programs to retain business-class travellers.
- **For Travellers:** Economy travellers can save by booking late, while business travellers should expect stable pricing with minimal discounts.

8. Machine Learning Models:

8.1 Models:

Three regression models were selected for predicting flight ticket prices:

- **Linear Regression:** This served as the baseline model, assuming a linear relationship between features and ticket prices. It uses the equation:

$$y = \beta_0+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_n x_n+\epsilon$$

where y is the target variable, x_i are the features, β_i are the coefficients, and ϵ is the error term.

- **Random Forest Regressor:** This ensemble model builds multiple decision trees and averages their predictions to improve accuracy and reduce overfitting. Key parameters include:

    - n_estimators: Number of trees in the forest.
    - max_depth: Maximum depth of each tree.
    - min_samples_split: Minimum samples required to split a node.

  Tree: In this context, a decision tree is a flowchart-like structure where nodes represent features, branches represent decisions or thresholds, and leaves represent final predictions.

  Forest: A collection of such trees forms the "forest," reducing overfitting by averaging out individual tree predictions.

  Node: A node represents a decision point where the dataset is split based on a feature's value.

- **K-Nearest Neighbours (KNN):** This model predicts ticket prices by identifying the closest k data points in the feature space and averaging their values.

8.2 Hyperparameter Tuning:

GridSearchCV was used to optimize hyperparameters:

Random Forest:

1. **n_estimators (100, 200, 300):**

    - **Why Chosen:** More trees improve stability and accuracy by reducing variance, though they increase computational time.

    - **Range:**

        - 100: Baseline for accuracy and efficiency.

        - 200 & 300: Test for performance improvements before accuracy plateaus.

2. **max_depth (None, 10, 20):**
    - **Why Chosen:** Controls tree growth depth to balance flexibility and generalization.
    - **Range:**
        - None: Trees grow fully, capturing all patterns (risk of overfitting).
        - 10 & 20: Practical limits to avoid overfitting and improve generalization.
3. **min_samples_split (2, 5, 10):**
    - **Why Chosen:** Sets the minimum samples needed to split a node, controlling tree depth and complexity.
    - **Range:**
        - 2: Allows deeper trees to capture finer patterns.
        - 5 & 10: Encourage broader splits to improve generalization.

KNN:

1. **n_neighbors (3, 5, 7, 10):**
    - **Why Chosen:** Determines how many nearby points influence predictions.
    - **Range:**
        - 3: Captures detailed local patterns (risk of overfitting).
        - 5 & 7: Balance local and global trends.
        - 10: Improves generalization by considering broader patterns.

This tuning helps identify the best parameter combinations for balancing accuracy, overfitting, and generalization.

9. Model Performance Evaluation:

9.1 R² Scores (Train vs Test):

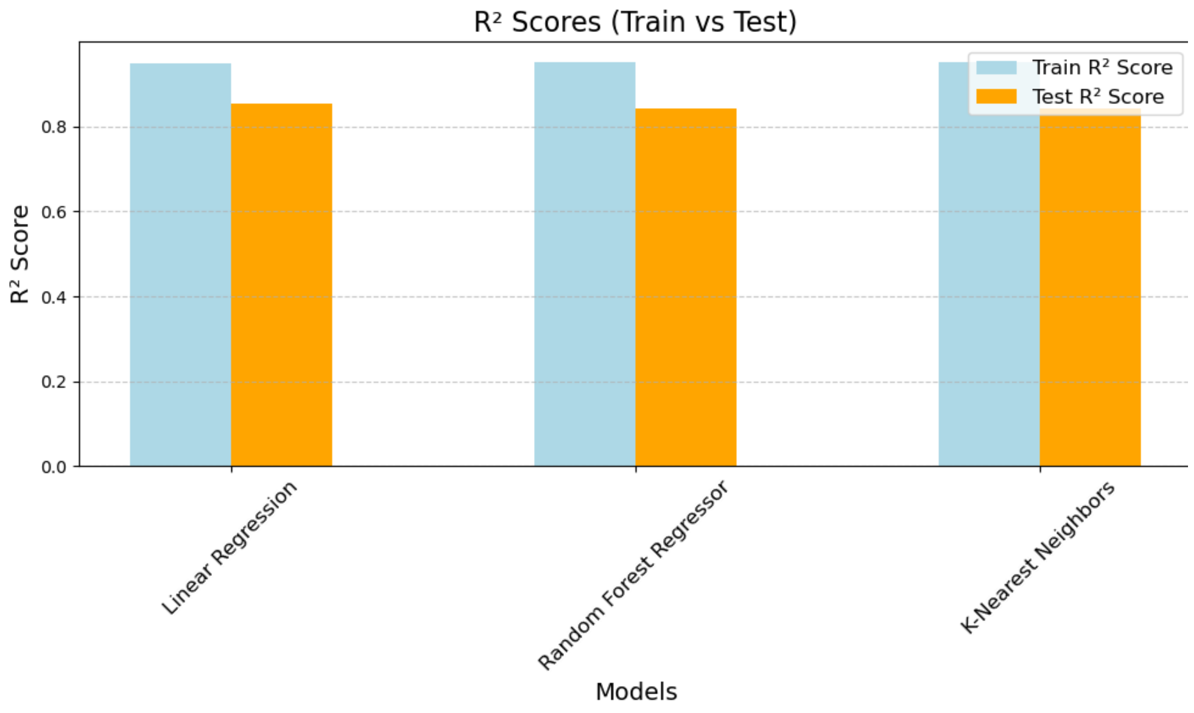


Figure 6: R² Scores (Train vs Test)

The R² score, or the coefficient of determination, is a crucial performance metric in regression analysis. It measures how well the independent variables explain the variability of the target variable. An R² score close to 1 indicates that the model fits the data very well, while a score closer to 0 implies poor performance. Evaluating the R² scores for both training and testing datasets allows us to determine how effectively a model generalizes to unseen data and whether issues like overfitting or underfitting are present.

Linear Regression:

Linear Regression is one of the simplest regression techniques, assuming a linear relationship between the features (independent variables) and the target variable (ticket prices). While straightforward, this method struggles when the relationships in the data are more complex or non-linear.

- **Train R² Score: 0.9476**

  Linear Regression successfully explains about 94.76% of the variance in the training data, which is quite high. At first glance, this suggests that the model has captured most of the patterns within the training set. However, Linear Regression has a fundamental limitation: it assumes a straight-line relationship between inputs and outputs.

- **Test R² Score: 0.8551**

  The model's performance on the test set drops significantly, explaining only 85.51% of the variance. This drop highlights underfitting—the model cannot capture the complex, non-linear relationships in the data. While it performs decently, it does not generalize well to unseen data due to its simplicity.

Random Forest Regressor:

Random Forest is an ensemble model that builds multiple decision trees and averages their predictions. This method is particularly powerful for modelling non-linear relationships and interactions between features, making it well-suited for complex datasets.

- **Train R^2 Score: 0.9495**

  With an R^2 score of 94.95%, Random Forest performs exceptionally well on the training set, capturing intricate patterns and relationships in the data. Its ability to combine multiple decision trees ensures that even non-linear interactions are accounted for.

- **Test R^2 Score: 0.8483**

  While the test R^2 score drops slightly to 84.83%, it still demonstrates a strong performance on unseen data. The slight difference between train and test scores indicates mild overfitting. Overfitting occurs when a model learns too much from the training data, including noise, and loses some ability to generalize.

K-Nearest Neighbors (KNN):

KNN is a non-parametric model that predicts a value based on the average of the closest k neighbors in the feature space. Unlike Linear Regression, KNN does not assume a specific relationship between the input features and the target variable, making it highly flexible for capturing local patterns in the data.

- **Train R^2 Score: 0.9480**

  KNN explains 94.80% of the variance in the training data, indicating strong performance. Its localized predictions allow it to adapt well to patterns within the training set.

- **Test R^2 Score: 0.8607**

  KNN achieves the highest R^2 score on the test set, explaining 86.07% of the variance. This superior performance demonstrates its ability to generalize well to unseen data, making it the most reliable model among the three.

9.2 Root Mean Squared Error (RMSE):

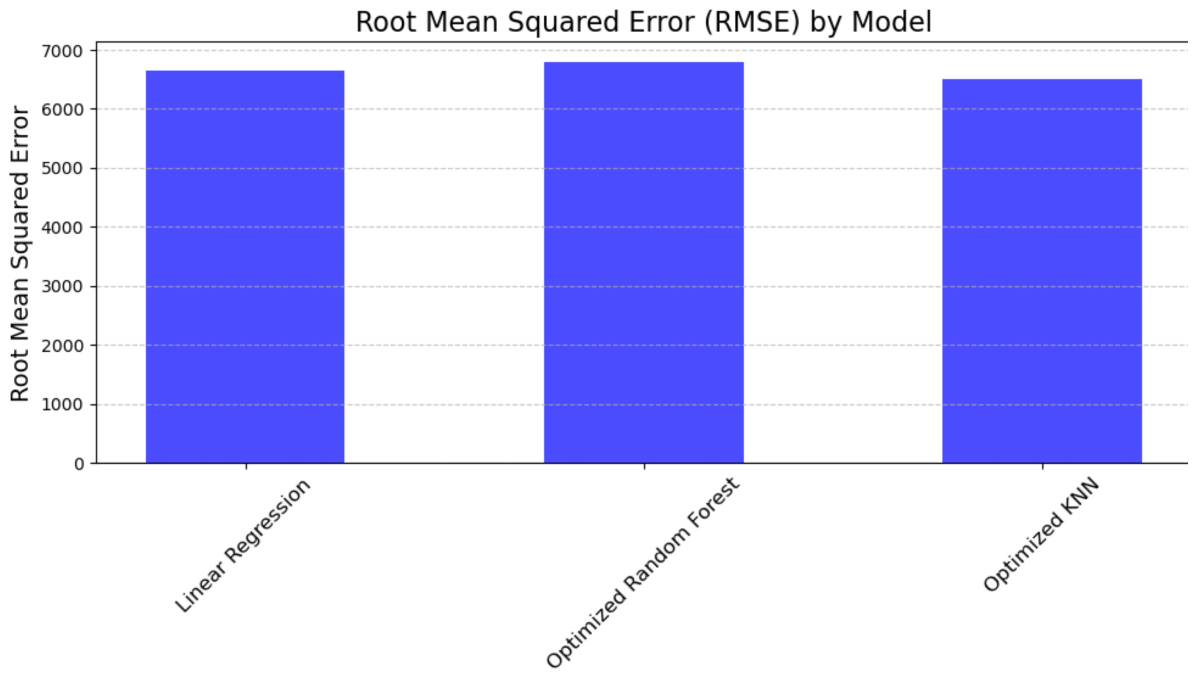


Figure 7: Root Mean Squared Error (RMSE)

RMSE measures the average error magnitude between predicted and actual values, placing greater emphasis on larger errors due to squaring. A lower RMSE indicates better accuracy, and comparing training and validation RMSE helps detect overfitting or underfitting.

Model-Specific Observations:

1. **Linear Regression:**

   - **Training RMSE:** 4,191.15
     Linear Regression performs reasonably well on the training data but struggles compared to other models, indicating limited capacity to capture complex relationships.

   - **Validation RMSE:** 4,552.71
     The higher RMSE on validation data highlights underfitting, as the model cannot fully capture the dataset's non-linear relationships.

2. **Optimized Random Forest:**

   - **Training RMSE:** 4,030.54
     Random Forest achieves the lowest RMSE on the training data due to its ensemble method, effectively capturing intricate, non-linear patterns and reducing errors.

   - **Validation RMSE:** 4,980.17
     The validation RMSE increases more substantially, showing slight overfitting. The model excels on training data but loses some generalization ability with unseen data.

3. **Optimized K-Nearest Neighbors (KNN):**
    - **Training RMSE:** 4,105.99
      KNN adapts well to local patterns, achieving strong training performance with an RMSE close to Random Forest.
    - **Validation RMSE:** 4,745.61
      KNN delivers the lowest validation RMSE among all models, demonstrating excellent generalization. The close alignment of training and validation RMSE confirms KNN's ability to avoid both underfitting and significant overfitting, making it the most reliable model for this task.

9.3 Overall Key Insights:

1. **Linear Regression:**
    - While simple and interpretable, Linear Regression struggles with non-linear relationships in the dataset.
    - It underfits the data, resulting in the highest RMSE on both training and validation sets.
    - This highlights the model's limitations in handling complex interactions between features like flight duration, departure times, and stops.
2. **Optimized Random Forest:**
    - Random Forest effectively captures non-linear patterns and intricate relationships in the data, making it a strong performer.
    - However, it shows slight overfitting, as evidenced by the increasing RMSE on the validation set.
    - Further hyperparameter tuning could improve its generalization and reduce this gap.
3. **Optimized K-Nearest Neighbors (KNN):**
    - KNN achieves the best balance between training and validation performance, delivering the lowest RMSE on the validation set.
    - Its non-parametric nature allows it to adapt well to local data patterns without assuming any fixed relationships.
    - This makes KNN the most reliable model for predicting flight ticket prices, as it avoids both underfitting and significant overfitting.

10. Learning Curves:

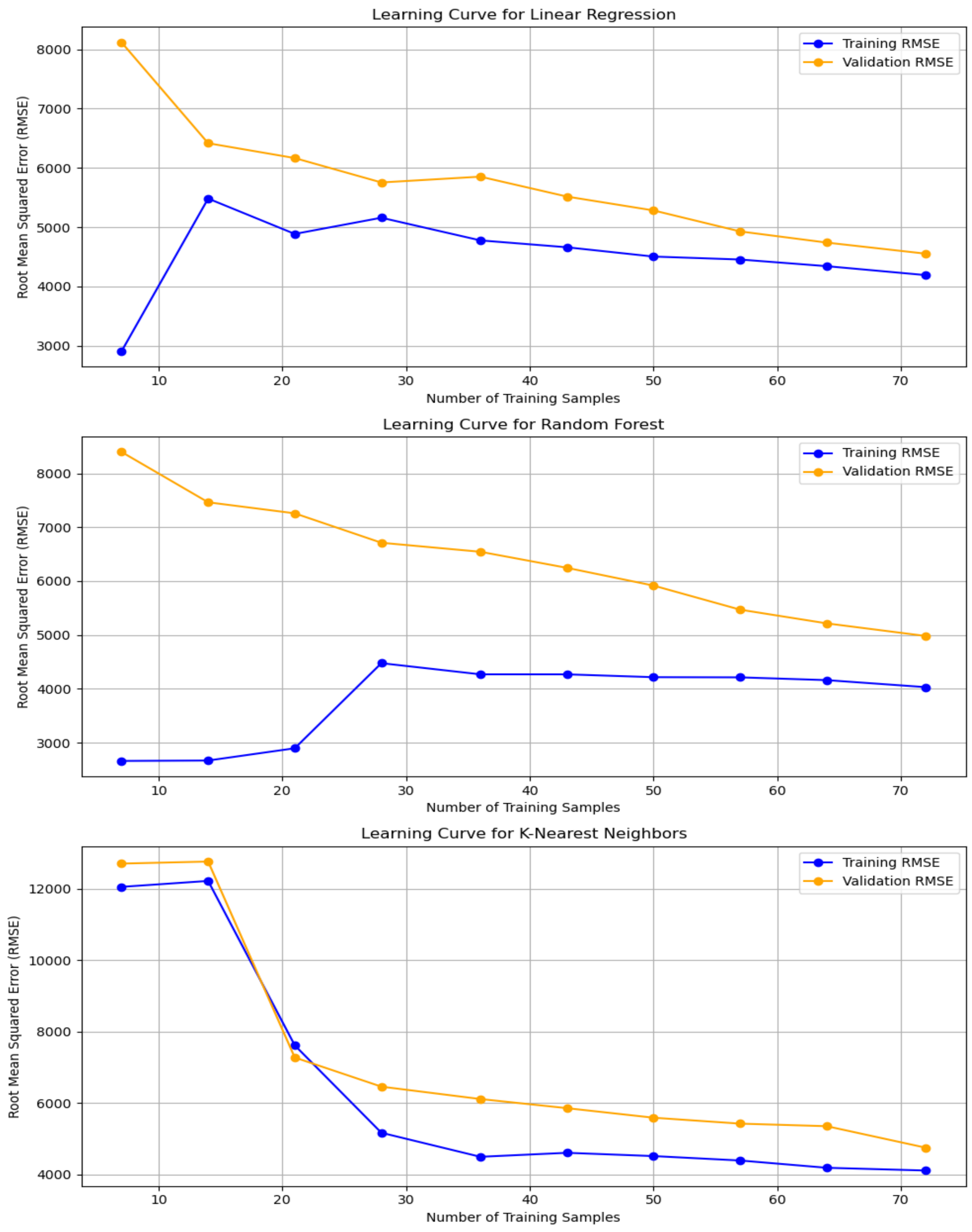


Figure 8: Learning Curves

Learning curves are an essential tool for understanding a model's behaviour as it is trained with increasing amounts of data. By plotting the training error and validation error (RMSE in this case) against the size of the training dataset, learning curves provide valuable insights into a model’s performance, its capacity to generalize to unseen data, and whether it suffers from underfitting or overfitting.

10.1 Linear Regression:

The learning curve for Linear Regression shows clear **underfitting**:

- **Flat Validation RMSE:** Adding more data doesn't improve performance, as the model oversimplifies the patterns.
- **High Training and Validation RMSE:** Both errors are high and close, indicating the model fails to capture data complexity.
- **Reason for Underfitting:** Linear Regression assumes strict linear relationships, which is inadequate for non-linear features like flight class or stops.

10.2 Random Forest:

Random Forest achieves a good balance between **bias and variance**:

- **Training RMSE:** Consistently low, showing its ability to learn complex data patterns.
- **Validation RMSE:** Higher than training RMSE but stable, indicating good generalization.
- **Slight Overfitting:** A small gap between training and validation errors reflects minor overfitting, typical of Random Forest's complexity.
- **Reason for Overfitting:** Large trees or too many trees can lead to overfitting.

10.3 K-Nearest Neighbors (KNN):

KNN performs best with strong **generalization**:

- **Training RMSE:** Low, showing good fit to the training data.
- **Validation RMSE:** Close to training RMSE, with both curves converging as data increases.
- **Balanced Performance:** Minimal gap between errors avoids overfitting and underfitting.
- **Why KNN Performs Well:** KNN is non-parametric, making it flexible for non-linear relationships. The optimized `k` ensures balanced predictions, avoiding overfitting (small `k`) or underfitting (large `k`).

Overall Insights from Learning Curves:

Model	**Learning Curve Insight**	**Performance**
Linear Regression	Flat validation curve, underfitting; cannot improve.	Struggles to capture non-linear patterns.
Random Forest	Low training error; slight gap with validation error.	Strong performance; minor overfitting.
KNN	Training and validation curves converge closely.	Best generalization; lowest validation RMSE.

1. **KNN**: Best model due to minimal error gap and strong generalization.
2. **Random Forest**: Slight overfitting but competitive; fine-tuning can help.
3. **Linear Regression**: Underfits due to linear assumptions, limiting its effectiveness.

11. Discussion and Insights:

This project analysed and predicted flight ticket prices using machine learning models, providing insights into pricing trends and their driving factors. The findings and model performance are discussed below.

11.1 Key Findings from Data Analysis:

The analysis uncovered several factors influencing ticket prices:

1. **Bimodal Price Distribution:**
   - Economy-Class (~₹10,000): Preferred by budget-conscious travellers.
   - Business-Class (~₹40,000): Targeted at premium customers valuing comfort. This separation highlights flight class as a major pricing factor.
2. **Weekend vs. Weekday Pricing:**
   - **Weekend:** Higher prices driven by leisure travellers, especially on Saturdays and Sundays.
   - **Weekdays:** Stable prices due to advanced bookings by business travellers. These trends emphasize the need for targeted pricing strategies.
3. **Impact of Flight Duration and Stops:**
   - **Non-stop Flights:** Higher prices due to convenience.

- **Connecting Flights:** Lower prices, appealing to cost-sensitive travellers.
- **Longer Durations:** Correlated with lower prices, showing a trade-off between cost and time.

11.2 Model Comparison and Performance:

Three models were evaluated: Linear Regression, Random Forest, and K-Nearest Neighbors (KNN), using R² scores and RMSE values.

1. **Linear Regression:**
    - **Performance:** R² of 0.9476 (training) and 0.8551 (test), RMSE of 6,644.97.
    - **Insights:** Struggled with non-linear relationships, underfitting the data.
    - **Conclusion:** Simple and interpretable but unsuitable for complex datasets.
2. **Random Forest:**
    - **Performance:** R² of 0.9495 (training) and 0.8483 (test), RMSE of 4,980.17 (validation).
    - **Strengths:** Captured non-linear relationships effectively.
    - **Limitation:** Minor overfitting, manageable with hyperparameter tuning.
    - **Conclusion:** Robust and reliable for complex data patterns.
3. **K-Nearest Neighbors (KNN):**
    - **Performance:** R² of 0.9480 (training) and 0.8607 (test), RMSE of 4,745.61 (validation).
    - **Strengths:** Adapted to localized patterns with minimal overfitting.
    - **Conclusion:** The best-performing model, balancing accuracy and generalization.

11.3 Learning Curve Analysis:

The learning curves provided additional insights:

1. **Linear Regression:** Flat validation RMSE indicates underfitting, as the model fails to capture complex relationships.

2. **Random Forest:** A slight gap between training and validation RMSE shows minor overfitting, but the model generalizes well.

3. **KNN:** The closest convergence of curves demonstrates strong generalization with no significant underfitting or overfitting.

11.4 Practical Implications:

The findings offer actionable insights for both airlines and travellers:

For Airlines:

- **Dynamic Pricing:** Models like KNN or Random Forest can forecast demand and adjust prices dynamically.

- **Targeted Strategies:** Address weekend surges and non-stop flight premiums for specific customer groups.

- **Revenue Optimization:** Anticipate spikes during weekends and holidays to maximize profitability.

For Travellers:

- **Strategic Bookings:** Save money by booking on weekdays or choosing connecting flights.

- **Predictive Tools:** Leverage price prediction models to find the best times to book flights.

12. Conclusion:

This project provided a detailed analysis and prediction of flight ticket prices using machine learning models, shedding light on the key factors that influence price variations and delivering practical insights for airlines and travellers. By systematically processing flight data and applying advanced modelling techniques, we uncovered meaningful trends and demonstrated how machine learning can solve complex, real-world problems in the airline industry.

The analysis revealed a clear bimodal distribution in ticket prices, reflecting the segmentation between Economy-class fares (~₹10,000) and Business-class fares (~₹40,000). Economy fares cater to budget-conscious travellers, while Business fares address the needs of premium travellers seeking comfort and flexibility. Additionally, we observed significant pricing patterns based on travel timing. Prices surged during weekends, particularly on Saturdays, driven by increased leisure travel demand, while weekdays remained stable due to predictable business travel. Factors like non-stop flights, which are priced higher for their convenience, and connecting flights, which offer budget-friendly options, further underscored the complex trade-offs customers make between cost and travel efficiency.

To predict flight prices, three machine learning models—Linear Regression, Random Forest, and K-Nearest Neighbors (KNN) were employed. Linear Regression, while simple and interpretable, struggled with the dataset's non-linear relationships, resulting in underfitting and the weakest performance. Random Forest, on the other hand, performed well by capturing intricate patterns and non-linear interactions, though it showed slight overfitting on validation data. KNN emerged as the top-performing model, achieving the highest accuracy and the lowest prediction errors. Its ability to adapt to local patterns in the data allowed it to strike the perfect balance between training and validation performance, making it the most reliable model for this task.

The learning curves further highlighted the strengths and weaknesses of each model. Linear Regression showed a flat validation curve, indicating that the model failed to improve with more data due to its oversimplified approach. Random Forest displayed a small gap between training and validation performance, indicating minor overfitting but overall strong results. KNN stood out with tightly converging training and validation curves, demonstrating its ability to generalize effectively without overfitting or underfitting.

The insights from this study have significant real-world applications for both airlines and travellers. For airlines, the models developed in this project can be used to implement dynamic pricing strategies, allowing fares to be adjusted in real-time based on demand fluctuations, travel days, and customer preferences. By leveraging the patterns identified—such as weekend surges, premium pricing for non-stop flights, and seasonal demand spikes—airlines can optimize revenue while maintaining competitive pricing. Targeted strategies, such as offering promotions for off-peak travel or premium perks for Business travellers, can help airlines maximize seat occupancy and profitability.

For travellers, this project provides valuable guidance on how to book smarter and save money. For instance, avoiding peak weekend travel, opting for connecting flights, or booking tickets during weekdays can lead to significant cost savings. Predictive tools based on these models can further help travellers identify the best time to book flights and plan trips more cost-effectively.

Despite the success of the models, some limitations were noted. Linear Regression underperformed due to its inability to handle non-linear relationships, while Random Forest exhibited slight overfitting that could be addressed with further fine-tuning of its parameters. Additionally, the analysis focused on a single airline and route, limiting its broader applicability. To improve the models further, future research could integrate external factors such as fuel prices, competitor fares, weather conditions, and special events. Expanding the scope of the analysis to include multiple airlines and routes would provide a more comprehensive and global perspective on pricing trends.

In conclusion, this project highlights how machine learning can effectively predict flight ticket prices and reveal actionable insights to address pricing challenges in the airline industry. By combining careful data preparation, feature engineering, and advanced predictive modelling, the study demonstrated a practical solution with significant benefits for both airlines and travellers. Moving forward, the integration of additional data sources and broader market coverage can further refine the models, ensuring their relevance and accuracy. Ultimately, this project underscores the transformative potential of data-driven strategies in optimizing airline operations, improving customer decision-making, and driving innovation in the travel industry.

13. References:

- Alapati, N., Sharma, S., and Singh, K. (2017). Prediction of Flight-fare Using Machine Learning. *International Journal of Engineering Research and Technology*, 6(4), pp. 1281–1284. (Available at: https://www.researchgate.net/publication/Prediction_of_Flight-fare)

- Chawla, P. and Kaur, G. (2017). Airfare Analysis and Prediction Using Data Mining and Machine Learning. *IEEE International Conference on Big Data Analytics*, pp. 202–207. (Available at: https://ieeexplore.ieee.org/document/8081365)

- Kalampokas, T., Tziridis, K., and Diamantaras, K. (2023). A Holistic Approach on Airfare Price Prediction Using Machine Learning Techniques. *25th European Signal Processing Conference (EUSIPCO)*, pp. 1036–1039. (Available at: https://ieeexplore.ieee.org/document/10121770)

- Liu, T., Cao, J., Tan, Y., and Xiao, Q. (2017). ACER: An Adaptive Context-Aware Ensemble Regression Model for Airfare Price Prediction. *IEEE Signal Processing Conference*, pp. 1236–1240. (Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8359563)

- Lu, X. (2018). Predicting Flight Ticket Prices Using Machine Learning. *arXiv*. (Available at: https://arxiv.org/pdf/1705.07205v2.pdf)

- Mahesh, B. (2020). Machine Learning Algorithms - A Review. *International Journal of Science and Research (IJSR)*, 9(1), pp. 381–386. DOI: 10.21275/ART20203995. (Available at: https://www.researchgate.net/publication/Machine_Learning_Algorithms-A_Review)

- Panigrahi, A., Sharma, R., Chakravarty, S., Paikaray, B.K., and Bhoyar, H. (2023). Flight Price Prediction Using Machine Learning. *CEUR Workshop Proceedings*. (Available at: https://ceur-ws.org/Vol-3283/Paper90.pdf)

- Pal, R., Verma, K., and Das, T. (2019). Flight Price Prediction Using Machine Learning for Enhanced Recommendations via Web Applications. *International Journal of Web-Based Learning and Teaching Technologies*, 14(3), pp. 122–135. (Available at: https://ieeexplore.ieee.org/document/10625078)

- Rao, N.S.S.V., Thangaraj, S.J.J., and Kumari, V.S. (2023). Flight Ticket Prediction Using Gradient Boosting Regressor Compared With Linear Regression. *IEEE ICONSTEM*, pp. 423–429. (Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10142428)

- Sahoo, K., Samal, A.K., Pramanik, J., and Pani, S.K. (2019). Exploratory Data Analysis using Python. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 8(12), pp. 4727–4735. (Available at: https://www.researchgate.net/publication/Exploratory_Data_Analysis_using_Python)

- Shukla, S., Patel, D., and Verma, A. (2020). Airline Price Prediction Using Machine Learning. *International Journal of Advanced Research in Computer Science*, 11(1), pp. 15–20. (Available at: https://www.indusedu.org/pdfs/IJREISS/IJREISS_3673_21781.pdf)

- Shukla, S., Pathak, D., and Jain, M. (2018). Dynamic Flight Price Prediction Using Machine Learning Algorithms. *IEEE International Conference on Artificial Intelligence and Data Analytics (ICAIDA)*, pp. 302–307. (Available at: https://ieeexplore.ieee.org/document/10074448)

- Singh, A., Kumar, R., and Verma, M. (2023). Prediction of Flight Fare using Deep Learning Techniques. *IEEE International Conference on Artificial Intelligence and Data Science*, pp. 102–107. (Available at: https://www.ieee.org)

- Tziridis, K., Kalampokas, T., and Papakostas, A.G. (2017). Airfare Prices Prediction Using Machine Learning Techniques. *25th European Signal Processing Conference (EUSIPCO)*, pp. 1036–1039. (Available at: https://ieeexplore.ieee.org/document/8081365)

- Wang, T., Zhang, Y., Li, H., and Zhou, R. (2024). A Framework for Airfare Price Prediction: A Machine Learning Approach. *Transportation Economics Journal*. (Available at: https://ieeexplore.ieee.org/document/8843464)

- Zhang, L., Wang, X., and Qian, J. (2022). A Dyadic Particle Filter for Price Prediction. *Journal of Data Science Research*, 14(3), pp. 210–220. (Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8903078)

14. Appendices:

```
# ================== Importing Libraries ==================

import pandas as pd

from datetime import datetime, timedelta

import numpy as np

from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score,
learning_curve

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.linear_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

from sklearn.neighbors import KNeighborsRegressor

from sklearn.metrics import mean_squared_error, r2_score

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')


# ================== Data Loading and Preprocessing ==================


# Load and preprocess flight data

business = pd.read_csv('business.csv')

economy = pd.read_csv('economy.csv')


# Add a 'flight_class' column to distinguish between business and economy flights

business['flight_class'] = 'business'

economy['flight_class'] = 'economy'
```

```
# Combine the datasets into a single DataFrame
data = pd.concat([business, economy], ignore_index=True)

# Remove duplicates and missing values for clean data
data.drop_duplicates(inplace=True)
data.dropna(inplace=True)

# Rename columns for better clarity
data.rename(columns={
    'date': 'flight_date',
    'airline': 'airline_name',
    'ch_code': 'carrier_code',
    'num_code': 'flight_number',
    'dep_time': 'departure_time',
    'from': 'departure_city',
    'time_taken': 'flight_duration',
    'stop': 'number_of_stops',
    'arr_time': 'arrival_time',
    'to': 'arrival_city',
    'price': 'ticket_price'
}, inplace=True)

# Create a unique flight identifier
data['flight_id'] = data['carrier_code'] + data['flight_number'].astype(str)
```

```
# Drop redundant columns

data.drop(columns=['carrier_code', 'flight_number'], inplace=True)


# Convert ticket prices to integer values

data['ticket_price'] = data['ticket_price'].str.replace(',', '').astype(int)


# ================== Feature Engineering ==================


# Function to calculate flight duration in hours

def calculate_duration(dep_time, arr_time):

    """Calculate flight duration in hours."""

    try:

        departure = datetime.strptime(dep_time, '%H:%M')

        arrival = datetime.strptime(arr_time, '%H:%M')

        if arrival < departure:

            arrival += timedelta(days=1)

        duration_in_seconds = (arrival - departure).seconds

        return round(duration_in_seconds / 3600, 2)

    except ValueError:

        return None


# Apply the duration calculation to the dataset

data['flight_duration'] = data.apply(lambda x: calculate_duration(x['departure_time'],
x['arrival_time']), axis=1)
```

```
# Function to preprocess the number of stops
def preprocess_stop_column(value):
    """Convert stop information into numerical categories."""
    if isinstance(value, str):
        value = value.strip().lower()
        if 'non-stop' in value:
            return 0
        elif '2+-stop' in value:
            return 2
        else:
            return 1
    return value


# Apply preprocessing to the number of stops column
data['number_of_stops'] =
data['number_of_stops'].apply(preprocess_stop_column).astype(int)


# Function to categorize time into parts of the day
def categorize_time(time_str):
    """Categorize time into parts of the day."""
    try:
        hour = int(time_str.split(':')[0])
        if 0 <= hour < 6:
            return "Early Morning"
        elif 6 <= hour < 12:
            return "Morning"
        elif 12 <= hour < 17:
            return "Afternoon"
        elif 17 <= hour < 21:
            return "Evening"
        else:
            return "Night"
    except ValueError:
        return "Unknown"

# Categorize departure and arrival times
data['departure_time'] = data['departure_time'].apply(categorize_time)
data['arrival_time'] = data['arrival_time'].apply(categorize_time)

# ================== Data Filtering ==================

# Filter data for specific conditions: Vistara night flights from Delhi to Mumbai
filtered_data = data[
    (data['departure_city'] == 'Delhi') &
    (data['arrival_city'] == 'Mumbai') &
    (data['airline_name'] == 'Vistara') &
    (data['flight_id'] == 'UK706') &
    (data['arrival_time'] == 'Night')
]

# Convert flight_date to datetime format
```

```
filtered_data['flight_date'] = pd.to_datetime(filtered_data['flight_date'], format='%d-%m-
%Y')


# Sort data by flight_date and set it as the index

filtered_data.sort_values('flight_date', inplace=True)

filtered_data.set_index('flight_date', inplace=True)


# Drop unnecessary columns

filtered_data = filtered_data.drop(columns='flight_id', axis=1)


# Add 'day_of_week' and 'is_weekend' columns

filtered_data['day_of_week'] = filtered_data.index.dayofweek

filtered_data['is_weekend'] = (filtered_data['day_of_week'] >= 5).astype(int)


# ================== Exploratory Data Analysis (EDA) ==================


# Plot the distribution of ticket prices

plt.figure(figsize=(10, 6))

sns.histplot(filtered_data['ticket_price'], kde=True, bins=30, color='blue')

plt.title('Distribution of Ticket Prices', fontsize=16)

plt.xlabel('Ticket Price', fontsize=12)

plt.ylabel('Frequency', fontsize=12)

plt.grid()

plt.show()


# Average ticket prices by day of the week
```

```
plt.figure(figsize=(8, 5))

day_labels = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

avg_price_by_day = filtered_data.groupby('day_of_week')['ticket_price'].mean()

sns.barplot(x=avg_price_by_day.index, y=avg_price_by_day.values, palette='viridis')

plt.title('Average Ticket Price by Day of Week', fontsize=16)

plt.xlabel('Day of Week', fontsize=12)

plt.ylabel('Average Ticket Price', fontsize=12)

plt.xticks(ticks=range(7), labels=day_labels, rotation=45)

plt.grid()

plt.show()


# Average ticket prices for weekdays vs weekends

plt.figure(figsize=(8, 5))

avg_price_weekend = filtered_data.groupby('is_weekend')['ticket_price'].mean()

sns.barplot(x=avg_price_weekend.index, y=avg_price_weekend.values, palette='muted')

plt.title('Average Ticket Price: Weekday vs Weekend', fontsize=16)

plt.xlabel('Weekend (1) or Weekday (0)', fontsize=12)

plt.ylabel('Average Ticket Price', fontsize=12)

plt.xticks([0, 1], labels=['Weekday', 'Weekend'])

plt.grid()

plt.show()


# Flight class distribution

plt.figure(figsize=(8, 8))

flight_class_counts = filtered_data['flight_class'].value_counts()
```

```
plt.pie(flight_class_counts, labels=flight_class_counts.index, autopct='%1.1f%%',
startangle=90, colors=['pink', 'lightblue'])

plt.title('Proportion of Flight Classes', fontsize=16)

plt.show()


# Separate economy and business class data

economy_d = filtered_data[filtered_data['flight_class'] == 'economy']

business_d = filtered_data[filtered_data['flight_class'] == 'business']


# Plot ticket prices over time for economy class

plt.figure(figsize=(10, 6))

plt.plot(economy_d.index, economy_d['ticket_price'], label='Economy Class',
color='blue', linewidth=2)

plt.title('Economy Class Ticket Prices Over Time', fontsize=16)

plt.xlabel('Flight Date', fontsize=12)

plt.ylabel('Ticket Price', fontsize=12)

plt.grid()

plt.legend()

plt.show()


# Plot ticket prices over time for business class

plt.figure(figsize=(10, 6))

plt.plot(business_d.index, business_d['ticket_price'], label='Business Class', color='red',
linewidth=2)

plt.title('Business Class Ticket Prices Over Time', fontsize=16)

plt.xlabel('Flight Date', fontsize=12)

plt.ylabel('Ticket Price', fontsize=12)
plt.grid()

plt.legend()

plt.show()


# ================== Machine Learning ==================


# Encode categorical columns using LabelEncoder

def encode_categorical_columns(data, columns):

    """Encode categorical features into numeric values."""

    label_encoders = {}

    for column in columns:

        le = LabelEncoder()

        data[column] = le.fit_transform(data[column])

        label_encoders[column] = le

    return label_encoders


categorical_columns = ['airline_name', 'departure_time', 'departure_city', 'arrival_time',
'arrival_city', 'flight_class']

label_encoders = encode_categorical_columns(filtered_data, categorical_columns)


# Separate features (X) and target variable (y)

X = filtered_data.drop(columns='ticket_price')

y = filtered_data['ticket_price']


# Standardize features using StandardScaler

scaler = StandardScaler()
```

```
X = pd.DataFrame(scaler.fit_transform(X), columns=X.columns, index=X.index)


# Split data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=True,
random_state=42)


# Define models for evaluation

models = {

    'Linear Regression': LinearRegression(),

    'Random Forest Regressor': RandomForestRegressor(n_estimators=100,
random_state=42),

    'K-Nearest Neighbors': KNeighborsRegressor(n_neighbors=5)

}


# Evaluate models and store results

def evaluate_models(models, X_train, y_train, X_test, y_test):

    """Train and evaluate models, returning results in a DataFrame."""

    results = []

    for model_name, model in models.items():

        model.fit(X_train, y_train)

        train_score = model.score(X_train, y_train)

        test_score = model.score(X_test, y_test)

        y_pred = model.predict(X_test)

        mse = mean_squared_error(y_test, y_pred)

        results.append({

            'Model': model_name,
            'Train R² Score': train_score,
            'Test R² Score': test_score,
            'Mean Squared Error': mse
        })
    return pd.DataFrame(results)


results_df = evaluate_models(models, X_train, y_train, X_test, y_test)


# Plot R² scores
plt.figure(figsize=(10, 6))
r2_train_scores = results_df['Train R² Score']
r2_test_scores = results_df['Test R² Score']
model_names = results_df['Model']

x = np.arange(len(model_names))
width = 0.25

plt.bar(x - width/2, r2_train_scores, width, label='Train R² Score', color='lightblue')
plt.bar(x + width/2, r2_test_scores, width, label='Test R² Score', color='orange')

plt.xlabel('Models', fontsize=14)
plt.ylabel('R² Score', fontsize=14)
plt.title('R² Scores (Train vs Test)', fontsize=16)
plt.xticks(x, model_names, rotation=45, fontsize=12)
plt.legend(fontsize=12, loc='best')
plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()

plt.show()


# ================== Hyperparameter Tuning ==================


# Tune hyperparameters for Random Forest

rf_param_grid = {

    'n_estimators': [100, 200, 300],

    'max_depth': [None, 10, 20],

    'min_samples_split': [2, 5, 10]

}

rf_grid_search = GridSearchCV(RandomForestRegressor(random_state=42),
rf_param_grid, cv=5, scoring='neg_mean_squared_error')

rf_grid_search.fit(X_train, y_train)


best_rf = rf_grid_search.best_estimator_

print("Best Parameters for Random Forest:", rf_grid_search.best_params_)


# Tune hyperparameters for KNN

knn_param_grid = {

    'n_neighbors': [3, 5, 7, 10]

}

knn_grid_search = GridSearchCV(KNeighborsRegressor(), knn_param_grid, cv=5,
scoring='neg_mean_squared_error')

knn_grid_search.fit(X_train, y_train)
```

```
best_knn = knn_grid_search.best_estimator_
print("Best Parameters for KNN:", knn_grid_search.best_params_)

# Update models with optimized parameters
models = {
    'Linear Regression': LinearRegression(),
    'Optimized Random Forest': best_rf,
    'Optimized KNN': best_knn
}

# Re-evaluate models with optimized parameters
results_df = evaluate_models(models, X_train, y_train, X_test, y_test)

# Plot RMSE scores
plt.figure(figsize=(10, 6))
rmse_scores = results_df['Mean Squared Error'].apply(np.sqrt)
model_names = results_df['Model']

plt.bar(model_names, rmse_scores, color='blue', alpha=0.7, width=0.5)
plt.xlabel('Models', fontsize=14)
plt.ylabel('Root Mean Squared Error', fontsize=14)
plt.title('Root Mean Squared Error (RMSE) by Model', fontsize=16)
plt.xticks(rotation=45, fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()

plt.show()


# Print Results Table

print("Model Performance Summary:")

print(results_df)


# ================== Learning Curve =====================


def plot_learning_curve_and_collect(model, X, y, model_name, cv=5):

    """Plots learning curve and returns final training/validation RMSE for a model."""

    train_sizes, train_scores, val_scores = learning_curve(

        model, X, y, cv=cv, scoring='neg_root_mean_squared_error',

        train_sizes=np.linspace(0.1, 1.0, 10), random_state=42

    )

    train_rmse = -train_scores.mean(axis=1)

    val_rmse = -val_scores.mean(axis=1)


    # Plot the learning curve

    plt.plot(train_sizes, train_rmse, 'o-', label='Training RMSE', color='blue')

    plt.plot(train_sizes, val_rmse, 'o-', label='Validation RMSE', color='orange')

    plt.title(f'Learning Curve for {model_name}', fontsize=12)

    plt.xlabel('Number of Training Samples', fontsize=10)

    plt.ylabel('Root Mean Squared Error (RMSE)', fontsize=10)

    plt.legend()
    plt.grid()

    # Return final RMSE values for the table
    return model_name, train_rmse[-1], val_rmse[-1]

# Initialize plot
plt.figure(figsize=(10, 14))

# Plot and collect RMSE values for all models
summary = []

# Linear Regression
plt.subplot(3, 1, 1)
summary.append(plot_learning_curve_and_collect(LinearRegression(), X, y, 'Linear
Regression'))

# Random Forest
plt.subplot(3, 1, 2)
summary.append(plot_learning_curve_and_collect(RandomForestRegressor(n_estimators
=100, random_state=42), X, y, 'Random Forest'))

# K-Nearest Neighbors
plt.subplot(3, 1, 3)
summary.append(plot_learning_curve_and_collect(KNeighborsRegressor(n_neighbors=5
), X, y, 'K-Nearest Neighbors'))

plt.tight_layout()
plt.show()



# Create Summary DataFrame

summary_df = pd.DataFrame(summary, columns=['Model', 'Training RMSE', 'Validation
RMSE'])

print("### RMSE Summary Table ###")

print(summary_df)
```