Predictive Insights for Flight Ticket Pricing: Comparative Analysis of XGBRegressor, RandomForestRegressor, and ExtraTreesRegressor Models

A Project Report submitted in the partial fulfillment of the

Requirements for the award of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Polepalli Uday Kiran (21471A05B3)

Mukkamalla Aravind (21471A05A4)

Sakinala Chandra Sekhar (21471A0579)

Under the esteemed guidance of

Shaik Khaja Mohiddin Basha, **M.Tech.**

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NARASARAOPETA ENGINEERING COLLEGE: NARASARAOPET
(AUTONOMOUS)
Accredited by NAAC with A+ Grade and NBA under Tyre -1 NIRF
rank in the band of 201-300 and an ISO 9001:2015 Certified
Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada
KOTAPPAKONDA ROAD, YALAMANDA VILLAGE, NARASARAOPET- 522601 2024-2025

NARASARAOPETA ENGINEERING COLLEGE (AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project that is entitled with the name **“Predictive Insights for Flight Ticket Pricing: Comparative Analysis of XGBRegressor, RandomForestRegressor and ExtraTreesRegressor Models”** is a bonafide work done by the team **Polepalli Uday kiran (21471A05B3), Mukkamalla Aravind (21471A05A4), Sakinala Chandra Sekhar (21471A0579)** in partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING during 2024-2025.

PROJECT GUIDE	PROJECT CO-ORDINATOR
Shaik Khaja Mohiddin Basha, M. Tech,	**Dr. M. Sireesha, M.Tech.,Ph.D.**
Assistant Professor	**Assoc. Professor**
HEAD OFTHE DEPARTMENT	EXTERNAL EXAMINER
Dr. S. N. Tirumala Rao, M.Tech., Ph.D.,	
Professor & HOD	

DECLARATION

We declare that this project work titled "**Predictive Insights for Flight Ticket Pricing: Comparative Analysis of XGBRegressor, RandomForestRegressor, and ExtraTreesRegressor Models**" is composed by ourselves that the work contain here is our own except where explicitly stated otherwise in the text and that this work has not been submitted for any other degree or professional qualification except as specified**.**

By

Polepalli Uday Kiran	**(21471A05B3)**
Mukkamalla Aravind	**(21471A05A4)**
Sakinala Chandra sekhar	**(21471A0579)**

ACKNOWLEDGEMENT

We wish to express our thanks to carious personalities who are responsible for the completion of the project. We are extremely thankful to our beloved chairman sri **M. V. Koteswara Rao, B.Sc.,** who took keen interest in us in every effort throughout this course. We owe out sincere gratitude to our beloved principal **Dr. S. Venkateswarlu, Ph.D.,** for showing his kind attention and valuable guidance throughout the course.

We express our deep felt gratitude towards **Dr. S. N. Tirumala Rao**, **M.Tech., Ph.D.,** HOD of CSE department and also to our guide **Shaik Khaja Mohiddin Basha**, **M.Tech** , of CSE department whose valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

We extend our sincere thanks towards **Dr. Sireesha Moturi**, **M.Tech., Ph.D.,** Associate professor & Project coordinator of the project for extending her encouragement. Their profound knowledge and willingness have been a constant source of inspiration for us throughout this project work.

We extend our sincere thanks to all other teaching and non-teaching staff to department for their cooperation and encouragement during our B.Tech degree.

We have no words to acknowledge the warm affection, constant inspiration and encouragement that we received from our parents.

We affectionately acknowledge the encouragement received from our friends and those who involved in giving valuable suggestions had clarifying out doubts which had really helped us in successfully completing our project.

By

Polepalli Uday Kiran	**(21471A05B3)**
Mukkamalla Aravind	**(21471A05A4)**
Sakinala Chandra Sekhar	**(21471A0579)**

NEC NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

INSTITUTE VISION AND MISSION

INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student centric teaching learning practices as well as research for the transformation of lives and community.

INSTITUTION MISSION

M1: Provide the best class infra-structure to explore the field of engineering and research

M2: Build a passionate and a determined team of faculty with student centric teaching, imbibing experiential, innovative skills

M3: Imbibe lifelong learning skills, entrepreneurial skills and ethical values in students for addressing societal problems

NEC NARASARAOPETA
ENGINEERING COLLEGE
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION OF THE DEPARTMENT

To become a center of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for research and ethical values to cater to the needs of industry and society.

MISSION OF THE DEPARTMENT

The department of Computer Science and Engineering is committed to

M1: Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.

M2: Impart high quality professional training to get expertise in modern software tools and technologies to cater to the real time requirements of the industry.

M3: Inculcate team work and lifelong learning among students with a sense of societal and ethical responsibilities.

NEC NARASARAOPETA ENGINEERING COLLEGE

(AUTONOMOUS)

Program Specific Outcomes (PSO's)

PSO1: Apply mathematical and scientific skills in numerous areas of Computer Science and Engineering to design and develop software-based systems.

PSO2: Acquaint module knowledge on emerging trends of the modern era in Computer Science and Engineering.

PSO3: Promote novel applications that meet the needs of entrepreneur, environmental and social issues.

NEC NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

Program Educational Objectives (PEO’s)

The graduates of the programme are able to:

PEO1: Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Computer Science and Engineering problems.

PEO2: Use various software tools and technologies to solve problems related to academia, industry and society.

PEO3: Work with ethical and moral values in the multi-disciplinary teams and can communicate effectively among team members with continuous learning.

PEO4: Pursue higher studies and develop their career in software industry.

NEC NARASARAOPETA
ENGINEERING COLLEGE
(AUTONOMOUS)

Program Outcomes

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the

knowledge of, and need for sustainable development.

8. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

9. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

10. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Project Course Outcomes (CO'S):

CO421.1: Analyze the System of Examinations and identify the problem.

CO421.2: Identify and classify the requirements.

CO421.3: Review the Related Literature

CO421.4: Design and Modularize the project

CO421.5: Construct, Integrate, Test and Implement the Project.

CO421.6: Prepare the project Documentation and present the Report using appropriate method.

Course Outcomes – Program Outcomes mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1		✓											✓		
C421.2	✓		✓		✓								✓		
C421.3				✓		✓	✓	✓					✓		
C421.4			✓			✓	✓	✓					✓	✓	
C421.5					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C421.6									✓	✓	✓		✓	✓	

Course Outcomes – Program Outcome correlation

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C421.1	2	3											2		
C421.2			2		3								2		
C421.3				2		2	3	3					2		
C421.4			2			1	1	2					3	2	
C421.5					3	3	3	2	3	2	2	1	3	2	1
C421.6									3	2	1		2	3	

Note: The values in the above table represent the level of correlation between CO’s and PO’s:

1. Low level
2. Medium level
3. High level

Project mapping with various courses of Curriculum with Attained PO’s:

Name of the course from which principles are applied in this project	**Description of the device**	**Attained PO**
C2204.2, C22L3.2	project is a Flight Ticket Price Prediction System that uses machine learning to estimate airfare based on user inputs like date, time, stops, airline, source, and destination.	PO1, PO3
CC421.1, C2204.3, C22L3.2	Each and every requirement is critically analyzed, the process model is identified	PO2, PO3
CC421.2, C2204.2, C22L3.3	Logical design is done by using the unified modelling language which involves individual team work	PO3, PO5, PO9
CC421.3, C2204.3, C22L3.2	Each and every module is tested, integrated, and evaluated in our project	PO1, PO5
CC421.4, C2204.4, C22L3.2	Documentation is done by all our three members in the form of a group	PO10
CC421.5, C2204.2, C22L3.3	Each and every phase of the work in group is presented periodically	PO10, PO11
C2202.2, C2203.3, C1206.3, C3204.3, C4110.2	Implementation is done, and the project will be used by travelers. Future updates may include enhancements for predicting dynamic flight prices based on real-time factors.	PO4, PO7
C32SC4.3	Designing a website to determine Flight Ticket price	PO5, PO6

ABSTRACT

The paper is proposing a Novel XGBRegressor Optimizer to address the flight ticket price prediction shortcomings by comparing ExtraTreesRegressor with it. Still, the usefulness of both models lies in enhancing the system performance as regards ticket price prediction. This Novel XGBRegressor Optimizer is another class that optimizes model parameters of the XGBRegressor by using gradient boosting. However, ExtraTreesRegressor is a great extension of random forests in the reduction of over-prediction variation using extremely random trees. In total, 40 sample sets have been used in this study in order to examine the under-study models. Using the ClinCalc software, this setup was checked for correctness, performing supervised learning2 with = 0.05, g-power = 0.8, taking 95% as the confidence internal Ci.Because of the experiment and assessment of the risk for over-learning, Novel XGBRegressor Optimizer was able to reveal performance of 82.7%, while ExtraTreesRegressor achieved 78.2%. Individual scores used for independent samples test levels, which for this level had a significance value of p=0.000. The study does present the Novel XGBRegressor Optimizer as efficient in improving the prediction of flight travel ticket prices when compared with the ExtraTreesRegressor.

INDEX

LIST OF FIGURES

1. INTRODUCTION

The airline industry operates in a highly dynamic environment where flight ticket prices fluctuate significantly due to various factors. These include the time of booking, airline, number of stops, departure and arrival times, route, demand, and even external influences such as weather conditions and political events. Accurate prediction of flight ticket prices can provide substantial advantages for both consumers and airlines. Consumers can make informed decisions about when to book tickets, while airlines can optimize their pricing strategies to maximize revenue and maintain competitiveness [1].

The Flight Price Prediction Project aims to develop a machine learning model capable of predicting flight ticket prices using historical data. This project involves analyzing flight data, identifying relevant features, and training various machine learning models to estimate ticket prices. By leveraging advanced algorithms and thorough data preprocessing, the objective is to build an efficient predictive model that provides accurate estimations of flight prices based on specific input variables [2].

Flight pricing models are inherently complex due to the numerous influencing factors involved. Traditional methods often struggle to capture the intricate relationships between variables affecting flight prices. Machine learning offers a powerful alternative by automatically learning from data, uncovering hidden patterns, and making accurate predictions [3]. By employing sophisticated techniques, machine learning can help travelers and airlines navigate the volatile nature of airline ticket pricing.

For travelers, precise price predictions empower them with insights into the best times to purchase tickets, potentially saving money and improving travel planning. For airlines, predictive models assist in revenue management by adjusting ticket prices dynamically based on anticipated demand and market conditions [4]. This balance ensures profitability while maintaining competitive pricing in the market.

The task of flight price prediction is framed as a regression problem, where the target variable is the continuous cost of a flight ticket. The input features used in this model include airline details, route information, number of stops, duration, departure and arrival times, and additional factors such as seasonal variations, special events, and macroeconomic conditions. These features contribute significantly to ticket pricing and need careful analysis to enhance model accuracy [5].

To develop a robust model, high-quality historical flight price data is essential. Data can be collected from various sources, including airline websites, travel agencies, and flight tracking services. Once the data is gathered, preprocessing steps such as handling missing values, encoding categorical variables, and feature scaling are applied to improve model performance. Feature engineering plays a crucial role in enhancing predictive power by extracting meaningful insights from raw data [6].

Several machine learning models can be employed for flight price prediction, including Linear Regression, Decision Trees, Random Forest, Gradient Boosting Models (such as XGBoost, LightGBM, and CatBoost), and Neural Networks. These models offer varying degrees of complexity and predictive accuracy. The models are evaluated using metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R^2) scores to measure predictive accuracy. Cross-validation techniques help assess model generalizability and prevent overfitting [7].

Flight price prediction using machine learning is a promising approach that benefits both consumers and airlines.

2. LITERATURE SURVEY

The study by N.S.S.V.S. Rao and S.J.J. Thangaraj explores flight ticket price prediction using Random Forest Regressor (RFR) and Decision Tree Regressor (DTR), comparing their effectiveness in forecasting airfare prices. The authors highlight that Random Forest Regressor outperforms Decision Tree Regressor due to its ensemble learning approach, which reduces overfitting and improves accuracy [5]. The paper emphasizes the advantages of feature importance analysis, which helps in identifying key factors influencing flight ticket prices, such as departure time, airline, and booking period. Additionally, the use of RFR ensures better generalization on unseen data, making it a reliable model for price prediction. However, the study also has some key gaps and limitations. One major limitation is that it does not incorporate real-time data from airline APIs, which could significantly enhance prediction accuracy. Furthermore, the research focuses only on historical data and does not consider external factors such as fuel prices, demand fluctuations, or special events, which can heavily impact ticket pricing. Another drawback is that the study does not compare deep learning models, which could potentially offer better performance in capturing complex price trends. Despite these limitations, the paper provides valuable insights into machine learning-based flight price prediction and highlights the effectiveness of ensemble methods like Random Forest in improving accuracy.

The study by C. Cao and X. Zhu presents a novel approach to airline ticket pricing using Reinforcement Learning (RL), treating pricing as a dynamic game where airlines continuously adjust fares in response to demand fluctuations and competitor pricing [6]. The research demonstrates that RL-based pricing strategies can adapt better than traditional rule-based models, leading to higher revenue optimization and improved market competitiveness. By leveraging reward-based learning, the model fine-tunes ticket prices in real-time, making it a promising advancement in automated pricing strategies. However, the study has several limitations. The high computational complexity and long training times of RL models pose challenges for real-time implementation in large-scale airline systems. Additionally, the model does not fully account for passenger behavior factors such as price sensitivity, booking habits, and last-minute purchase trends, which are crucial for accurate pricing. Moreover, the impact of external variables like economic fluctuations, oil prices, and geopolitical events is not extensively analysed, which could affect real-world applicability. Despite these gaps, the paper highlights the strong potential of RL in transforming airline pricing mechanisms. Future research could focus on integrating real-time data from airline APIs, incorporating deep learning techniques for better forecasting, and refining the model to include passenger behaviour analytics.

The study by G. R. Trivedi, J. V. Bolla, and M. Sireesha presents a Bitcoin transaction network analysis using cache-based pattern matching rules to detect fraudulent activities and enhance security [8]. The proposed model improves transaction verification efficiency by reducing computational overhead and enabling faster anomaly detection. However, scalability issues may arise as the network grows, and the model's effectiveness against evolving fraud techniques remains uncertain. Additionally, it relies on historical patterns, which may limit its adaptability to new fraud schemes. Despite these challenges, the study highlights an innovative approach to securing blockchain transactions, with potential improvements in real-time adaptability and scalability.

The study by J. J. Remus and L. M. Collins explores a method for identifying impaired cochlear implant channels using speech-token confusion matrix analysis [14]. This approach aims to detect malfunctioning channels by analysing speech perception errors, potentially improving implant tuning and auditory outcomes for users. A key advantage of this method is its data-driven approach, which allows for precise identification of faulty channels without invasive procedures. However, its effectiveness may be limited by variability in patient responses and external noise interference. Future research could focus on enhancing real-time analysis and integrating adaptive correction mechanisms to further optimize cochlear implant performance.

3. EXISTING SYSTEM

Traditional flight ticket price prediction methods rely on fixed rule-based approaches, historical data analysis, and basic statistical models. These conventional techniques have several limitations in accurately forecasting ticket prices, leading to inefficiencies for travelers and businesses. The current systems typically use fixed pricing strategies or simple regression models that fail to adapt dynamically to market fluctuations. Below are some key challenges of the existing methods:

A. Limited Predictive Accuracy

Traditional models struggle to capture complex dependencies between multiple factors influencing ticket prices, such as seasonality, demand, airline pricing strategies, and external economic conditions. As a result, predictions are often inaccurate or unreliable.

B. Lack of Real-Time Insights

Most existing systems rely on historical data alone, without incorporating real-time updates. This limitation makes it difficult to account for sudden price changes caused by last-minute bookings, airline promotions, or sudden demand spikes due to events or holidays.

C. Static Feature Selection

Conventional approaches often use fixed parameters to estimate ticket prices, such as departure date and airline type, without dynamically considering new influential factors. This results in a lack of adaptability to market trends.

D. Inefficient Handling of Large Datasets

With millions of flight price records available from various airlines, traditional models struggle with scalability. They may fail to process large datasets efficiently or generalize well to new flight routes and airlines.

E. High Variability in Pricing

Airline ticket prices are affected by a wide range of dynamic factors, such as fuel prices, demand surges, airline competition, and booking time. Existing methods often fail to account for these dynamic elements, leading to inaccurate predictions.

F. Lack of User-Centric Optimization

Traditional price prediction systems do not provide personalized recommendations based on user preferences, travel history, or flexible budget options. This makes it difficult for travelers to find the best deals suited to their needs.

G. Need for Improvement

To overcome these challenges, modern machine learning-based prediction models offer higher accuracy and adaptability by analyzing complex pricing trends in real-time. Our proposed system incorporates advanced algorithms, feature engineering, and real-time data sources to provide more reliable and dynamic flight ticket price predictions. By leveraging AI-driven insights, we aim to enhance decision-making for both travelers and travel agencies. This image represents the workflow of a Flight Price Prediction System using machine learning. It covers data collection, preprocessing, model selection, training, evaluation, deployment, and real-time prediction, ending with visualization and insights.

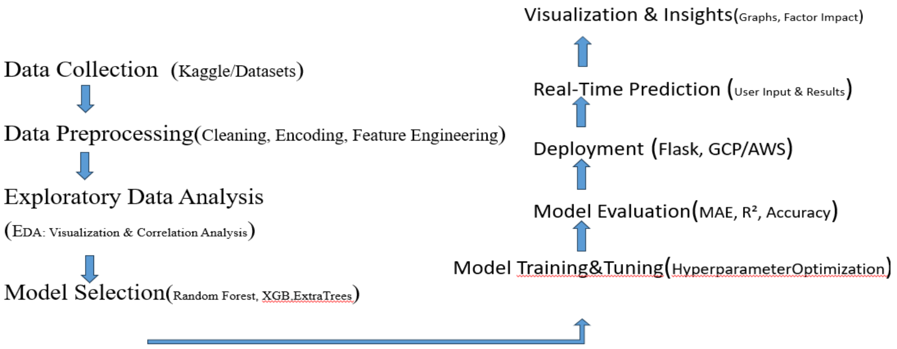

Data Collection (Kaggle/Datasets)
Data Preprocessing(Cleaning, Encoding, Feature Engineering)
Exploratory Data Analysis
(EDA: Visualization & Correlation Analysis)
Model Selection(Random Forest, XGB,ExtraTrees)
Model Training&Tuning(HyperparameterOptimization)
Model Evaluation(MAE, R², Accuracy)
Deployment (Flask, GCP/AWS)
Real-Time Prediction (User Input & Results)
Visualization & Insights(Graphs, Factor Impact)

Fig 3.1 Existing Model Working process

4.PROPOSED SYSTEM

The proposed system is designed to predict flight ticket prices using machine learning techniques, ensuring higher accuracy and real-time updates. The process begins with data collection from sources such as Kaggle, airline websites, and travel APIs. This data includes key factors like airline name, departure and arrival times, flight duration, number of stops, and ticket prices. The collected data undergoes preprocessing, which includes cleaning, encoding categorical variables, handling missing values, and performing feature engineering to enhance model performance. Exploratory Data Analysis (EDA) is conducted to identify significant factors affecting price fluctuations, allowing the model to learn meaningful patterns from historical data.To develop an accurate predictive model, various machine learning algorithms such as Random Forest, XGBoost, and Extra Trees are implemented. The models undergo hyperparameter tuning to improve their efficiency and reduce errors. Performance evaluation is conducted using metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R^2 score to ensure precise price estimations. Once trained, the model is deployed on cloud-based platforms like Flask, Google Cloud Platform (GCP), or AWS, making it accessible to users. The system allows users to enter details such as source, destination, date, and airline preferences to obtain real-time ticket price predictions.

Additionally, the system provides data visualization and insights, helping users understand the factors influencing ticket prices. By analyzing trends and patterns, the model can suggest optimal booking times, enabling users to make cost-effective travel decisions. The proposed system is not only beneficial for passengers but also helps travel agencies and airlines optimize pricing strategies. With its scalability, accuracy, and real-time prediction capabilities, this system provides a significant advantage over traditional fare estimation methods.

A. Advantages of the Proposed System

High Accuracy – Uses advanced machine learning models to provide precise ticket price predictions.Real-Time Predictions – Allows users to get instant fare estimates based on live input data.Data-Driven Insights – Provides graphical representations of price trends and influencing factors.User-Friendly Interface – Simple and easy-to-use system for travelers and travel agencies.Cost Optimization – Helps users find the best time to book flights at the lowest prices.Scalability – Can handle large datasets and be expanded for different routes and airlines.Cloud-Based Deployment – Ensures accessibility from anywhere with an internet connection [5].

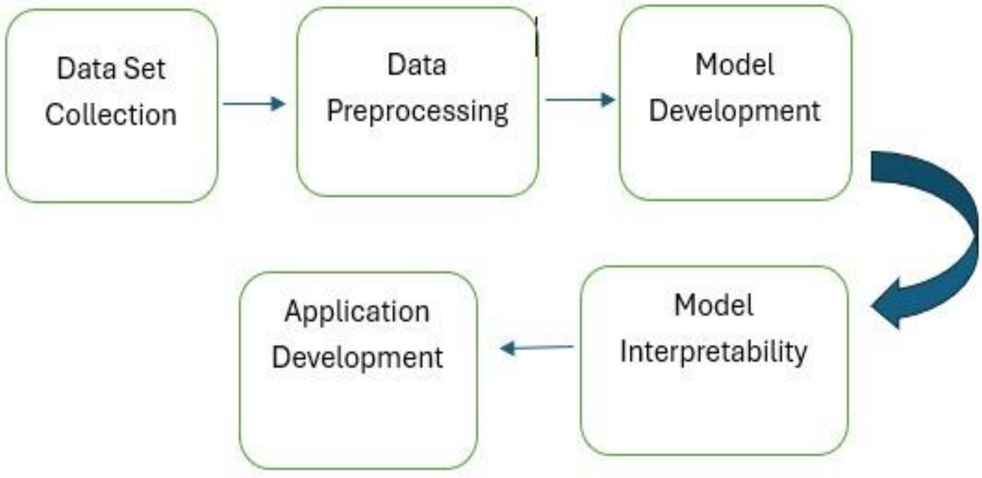


Fig 4.1 Flow Chart of Proposed System

The Flight ticket prices are highly dynamic and influenced by multiple factors such as seasonality, demand, airline policies, fuel costs, and booking time. The traditional approach to price prediction relies on user experience, basic trends, or static rule-based models, which often fail to capture real-time fluctuations. To overcome these challenges, a machine learning-based flight ticket price prediction system is proposed. This system utilizes historical flight data, machine learning algorithms, and real-time processing to provide users with accurate and timely price predictions. The first phase of the proposed system is data collection, where relevant datasets are gathered from sources like airline websites, travel agencies, and online booking platforms. These datasets typically include flight details such as departure and arrival cities, airline names, flight duration, number of stops, class of service, and past ticket prices over different time periods. Additional external factors such as holidays, weather conditions, and major events are also considered, as they significantly impact ticket pricing. This image outlines the workflow of a machine learning project, starting from data collection and preprocessing to model development. It also includes model interpretability and application development, ensuring the model's insights are useful and deployable.

Once the data is collected, it undergoes data preprocessing to clean and prepare it for analysis. This step includes handling missing values, removing duplicate records, and encoding categorical variables like airline names and city names into numerical representations. Data cleaning ensures that inconsistencies and outliers are eliminated, improving model accuracy. Furthermore, feature engineering is performed to create new meaningful variables, such as days before departure, weekend vs. weekday travel, and peak vs. off-season bookings.

After preprocessing, the data is fed into machine learning models for training. The proposed system considers various machine learning algorithms, including Random Forest, Gradient Boosting (XGBoost). These models learn from historical price patterns and recognize complex relationships between different influencing factors. Before training, normalization is applied to scale numerical features to a common range, ensuring that no single feature dominates the learning process.

Hyperparameter tuning is an essential part of model training, where parameters such as the number of trees in a Random Forest model or the learning rate in a Neural Network are optimized for better accuracy. The model's performance is then evaluated using metrics like Mean Absolute Error (MAE), R-squared (R^2), and Root Mean Squared Error (RMSE) to assess prediction accuracy. If a model does not perform well, additional tuning and feature selection techniques are applied to enhance its efficiency.

Once the model achieves satisfactory accuracy, it is deployed for real-time predictions using frameworks such as Flask, FastAPI, or cloud-based platforms like AWS and Google Cloud. Users can input details such as departure and arrival locations, date of travel, and airline preference, and the system provides an estimated ticket price along with the best time to book for cost savings. The real-time prediction module continuously updates itself with new pricing trends, making the system more robust and adaptive to market changes[7].

Additionally, visualization tools like dashboards and interactive graphs help users understand price trends, factor impacts, and optimal booking windows. Insights derived from the model allow travelers to make informed decisions, airlines to optimize pricing strategies, and travel agencies to improve their services.

The proposed machine learning-based flight ticket price prediction system provides a data-driven approach to forecasting ticket prices with high accuracy. By leveraging historical data, advanced algorithms, and real-time updates, the system empowers travelers to make informed booking decisions. As the airline industry evolves, this system can further integrate external factors such as fuel prices, economic conditions, and passenger demand to enhance accuracy and reliability.

5.SYSTEM REQUIREMENT

A. Hardware Requirements:

System Type	: intel®core™i3-7500UCPU@2.40gh
Cachememory	: 4MB(Megabyte)
RAM	: 8GB (gigabyte)
Hard Disk	: 4GB

B. Software Requirements:

Operating System	: Windows 11, 64-bit Operating System
Coding Language	: Python
Python distribution	: google colab, Flask
Browser	: Any Latest Browser like Chrome

6.SYSTEM ANALYSIS

6.1 Scope of the project

The The proposed **flight ticket price prediction system** aims to provide travelers with accurate fare estimates, helping them make cost-effective booking decisions. The **scope of this project** extends across various domains, including data analytics, machine learning, and real-time prediction systems. By leveraging historical flight data, market trends, and AI-based models, this system improves price forecasting accuracy and benefits both travelers and airline service providers.

A. User-Centric Price Prediction:

- The system allows users to input travel details (departure, destination, date, and airline preference) to get fare predictions.
- It provides insights into the best time to book tickets at the lowest prices.

B. Data-Driven Decision Making:

- The project uses machine learning models trained on large datasets of past ticket prices, demand fluctuations, and seasonal trends.
- It helps users understand fare variations based on historical data and market conditions.

C. Real-Time and Dynamic Pricing Analysis:

- The system continuously updates with real-time ticket price changes.
- It adapts to airline pricing algorithms, last-minute fare hikes, and special discounts.

D. Scalability and Future Expansion:

- The model can be extended to **other transportation sectors**, such as trains, buses, and hotel price predictions.
- It can integrate **external factors** like fuel costs, holidays, weather, and economic conditions to improve accuracy.

E. Cloud-Based Deployment:

- The system can be hosted on cloud platforms like AWS, Google Cloud, or Microsoft Azure for global accessibility.
- It can be integrated into travel agency websites or mobile applications for a seamless user experience.

F. Business and Industry Applications:

- Airlines can use the model to optimize their pricing strategies and predict customer booking patterns.
- Travel agencies and online booking platforms can improve customer engagement and enhance pricing transparency.

6.2 Analysis

The analysis phase of the flight ticket price prediction system focuses on understanding the factors influencing airfare fluctuations and selecting the best machine learning approaches to ensure accurate predictions. This stage involves studying historical pricing trends, user behaviors, and external influences such as seasonality, demand, and fuel costs. By analyzing past data, the system aims to identify patterns and develop a predictive model that helps travelers make informed booking decisions.

The process begins with data collection from various sources, including airline websites, travel agencies, and third-party APIs. Key features such as departure date, booking date, airline type, flight duration, class of travel, and competitor pricing play a crucial role in determining price variations. The collected data undergoes preprocessing, where missing values are handled, irrelevant features are removed, and normalization techniques are applied to ensure consistency. Exploratory Data Analysis (EDA) is then conducted using visualization techniques such as graphs and heatmaps to identify correlations and trends. Through this analysis, peak booking periods, price fluctuations, and demand-driven variations can be detected.

To develop an accurate prediction model, machine learning algorithms such as Linear Regression, Decision Trees, Random Forest, and advanced deep learning models are implemented. Ensemble techniques like XGBoost and Gradient Boosting help improve the model’s performance. The evaluation process is conducted using performance metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R^2 Score to measure the model's accuracy. Further optimization, including hyperparameter tuning and cross-validation, enhances the model's efficiency. Additionally, real-time data updates are incorporated to ensure that predictions remain relevant based on the latest trends. Overall, this analytical approach ensures a robust and reliable system that helps users secure the best flight fares by leveraging machine learning for dynamic pricing predictions.

6.3 Data Pre-processing

Data preprocessing is a crucial step in developing a flight ticket price prediction system, as it ensures the data is clean, structured, and ready for model training. Raw data collected from various sources often contains missing values, inconsistencies, and irrelevant features that can negatively impact prediction accuracy. The preprocessing stage involves multiple techniques such as data cleaning, encoding categorical variables, handling missing values, and feature engineering to improve the dataset's quality.

The process begins with data cleaning, where duplicate records and irrelevant attributes are removed. Missing values in numerical features can be handled using imputation techniques like mean, median, or regression-based estimation, while categorical variables with missing entries are replaced using the most frequent category or predictive modeling. Next, data encoding is applied to convert categorical variables like airline names and flight classes into numerical representations using techniques such as one-hot encoding or label encoding. This step ensures that the machine learning model can interpret categorical information effectively.

After encoding, feature scaling and normalization are performed to bring all numerical data into a similar range, preventing bias toward features with large numerical values. Techniques such as Min-Max Scaling or Standardization help improve model performance by ensuring fair weight distribution. Feature engineering is another essential aspect of preprocessing, where new meaningful variables, such as time-based features (day of the week, seasonality), are created to enhance the predictive power of the model. Finally, the processed dataset is split into training and testing sets to ensure the model generalizes well to unseen data. By performing thorough data preprocessing, the system eliminates noise, improves model efficiency, and enhances the accuracy of flight price predictions.

6.4 Feature Selection and Optimization

Feature selection and optimization play a crucial role in enhancing the accuracy and efficiency of machine learning models used for flight ticket price prediction. Feature selection involves identifying the most relevant variables from the dataset that contribute significantly to price prediction while eliminating redundant or irrelevant features. By selecting the most influential features, the model complexity is reduced, improving computational efficiency and minimizing overfitting. Various statistical and machine learning-based techniques, such as correlation analysis, mutual information, Recursive Feature Elimination (RFE), and Principal Component Analysis (PCA), are employed to filter out non-informative attributes and retain only those that have a strong impact on flight prices.

Optimization is another critical step in refining the model's performance. It involves fine-tuning hyperparameters to enhance prediction accuracy and generalization. Hyperparameter optimization techniques such as Grid Search, Random Search, and Bayesian Optimization help determine the best combination of model parameters. For instance, in algorithms like Random Forest or XGBoost, tuning parameters such as the number of trees, maximum depth, and learning rate significantly influence performance. Regularization techniques like L1 (Lasso) and L2 (Ridge) are also employed to prevent overfitting by penalizing large coefficients in regression models.

Moreover, feature scaling techniques such as Min-Max Scaling and Standardization ensure that numerical variables are within the same range, improving the efficiency of optimization algorithms. Automated feature selection using embedded methods, such as decision tree-based feature importance ranking, further enhances model performance. By effectively selecting and optimizing features, the flight price prediction model becomes more accurate, interpretable, and efficient, ultimately delivering reliable results for users seeking price trends and forecasts.

6.5 Model building:

In Model building is a crucial phase in flight ticket price prediction, where selected features and preprocessed data are used to train machine learning models. This process involves choosing appropriate algorithms, training the models, and optimizing their performance to achieve accurate predictions. Various models, such as Linear Regression, Decision Trees, Random Forest, Gradient Boosting (XGBoost), and Neural Networks, are commonly used for predicting flight prices. The training process involves feeding historical flight price data into the model, allowing it to learn patterns and relationships between different features such as airline, departure time, destination, and booking date. Hyperparameter tuning techniques like Grid Search and Random Search help improve model efficiency by selecting the best parameters. Cross-validation ensures that the model generalizes well to unseen data, reducing overfitting. Once trained, the model undergoes evaluation using metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R^2 score to assess its accuracy. A well-built model can provide real-time flight price predictions, helping travelers make informed booking decisions.

A. ExtraTreesRegressor

ExtraTreesRegressor (Extremely Randomized Trees Regressor) is an ensemble learning method used for regression tasks, making it highly effective for predicting flight ticket prices. It is based on decision trees and follows the concept of bagging, where multiple trees are built using random subsets of data and features. Unlike traditional Random Forest, ExtraTreesRegressor introduces additional randomness by selecting split points at random, leading to more diverse decision trees. This approach reduces variance, prevents overfitting, and enhances the model’s generalization ability, making it well-suited for dynamic and fluctuating datasets like flight ticket prices.

In flight ticket price prediction, various factors such as airline, departure time, booking date, number of stops, destination, and seat availability influence pricing. ExtraTreesRegressor effectively captures complex relationships between these features by averaging predictions from multiple decision trees. The model works efficiently with large datasets, handling missing values and irrelevant features with minimal preprocessing. It also provides feature importance scores, helping in selecting the most significant factors affecting ticket prices. Hyperparameter tuning, such as adjusting the number of estimators and depth of trees, further improves its predictive accuracy.

The key advantage of ExtraTreesRegressor in flight price prediction is its ability to handle non-linearity, noise, and high-dimensional data while maintaining speed and accuracy. It performs well compared to other regression models like Linear Regression and Decision Trees, as it reduces overfitting and ensures stable predictions. By integrating ExtraTreesRegressor into a flight price prediction system, users can get reliable price forecasts, allowing them to make informed decisions on when to book tickets at the lowest prices.

B. **XGBRegressor**

XGBRegressor (Extreme Gradient Boosting Regressor) is a powerful machine learning algorithm based on gradient boosting, widely used for regression tasks, including flight ticket price prediction. Developed as part of the XGBoost library, it is known for its speed, efficiency, and high accuracy. XGBRegressor is an ensemble method that builds multiple weak learners (decision trees) sequentially, where each new tree corrects the errors of the previous ones. This boosting approach helps capture complex patterns in data, making it ideal for predicting flight ticket prices, which depend on multiple dynamic factors like airline, departure time, booking time, number of stops, and demand fluctuations.

One of the main strengths of XGBRegressor is its ability to handle missing data, multicollinearity, and non-linearity effectively. It uses a combination of regularization techniques (L1 and L2) to prevent overfitting, ensuring that the model generalizes well to new data. Additionally, it employs advanced optimization techniques such as parallel tree boosting and tree pruning, making it faster than traditional boosting algorithms. Hyperparameter tuning, including learning rate adjustment, tree depth selection, and number of estimators, further enhances its predictive performance.

In flight ticket price prediction, XGBRegressor provides precise price estimates by leveraging its ability to learn from past ticket price patterns. It identifies crucial features that influence pricing, helping users make informed decisions on when to book tickets at the lowest possible cost. Compared to other models like Decision Trees and Random Forest, XGBRegressor delivers superior accuracy, faster training times, and better handling of complex datasets. This makes it a valuable tool for airlines, travel agencies, and consumers looking for data-driven insights into flight pricing trends.

C. RandomForestRegressor

RandomForestRegressor is a popular machine learning algorithm used for regression tasks, including flight ticket price prediction. It is an ensemble learning method that constructs multiple decision trees during training and merges their predictions to improve accuracy and reduce overfitting. This technique, known as bagging, enhances the model’s robustness and ensures better generalization to unseen data. In the context of flight ticket price prediction, RandomForestRegressor helps capture complex relationships between various factors such as airline, departure time, booking date, travel class, and seasonality.

One of the key advantages of using RandomForestRegressor is its ability to handle large datasets efficiently and manage missing data without compromising performance. It automatically selects the most important features that influence flight prices, reducing the risk of overfitting and improving prediction reliability. Additionally, the algorithm's ability to average multiple decision trees results in lower variance, making it less sensitive to fluctuations in individual data points. Hyperparameter tuning, including the number of trees (n_estimators) and tree depth, further optimizes the model for better accuracy and performance.

In flight ticket price prediction applications, RandomForestRegressor is widely used due to its stability, interpretability, and efficiency. It outperforms simple decision trees by mitigating their limitations while maintaining ease of implementation. Compared to other models like Linear Regression or basic Decision Trees, RandomForestRegressor delivers higher accuracy, making it a preferred choice for predicting dynamic pricing trends. Airlines, travel agencies, and passengers can leverage this model to gain insights into price fluctuations and determine the best time to book tickets, ultimately leading to cost savings and improved decision-making.

6.6 Comparative Analysis:

Comparative analysis in flight ticket price prediction involves evaluating different machine learning models based on performance metrics such as accuracy, mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE). The purpose of this analysis is to determine the most effective model for predicting ticket prices by comparing various algorithms like RandomForestRegressor, XGBRegressor, and ExtraTreesRegressor. Each model has unique strengths and weaknesses, which influence their performance in handling real-world ticket pricing data.

RandomForestRegressor is widely used due to its ability to handle large datasets efficiently and reduce overfitting by averaging multiple decision trees. It is highly stable and robust but may require fine-tuning of hyperparameters to achieve optimal performance. XGBRegressor (Extreme Gradient Boosting) is another powerful model that improves prediction accuracy by optimizing loss functions and handling missing values effectively. It often outperforms Random Forest in terms of precision but requires careful tuning to prevent overfitting. ExtraTreesRegressor is similar to RandomForest but applies more randomness while selecting split points, which can lead to better generalization in certain cases. However, it may not always provide the best performance on structured datasets [8].

A detailed comparison based on real-world flight ticket data often shows that XGBRegressor achieves the highest accuracy due to its advanced boosting technique, followed closely by RandomForestRegressor, which is more interpretable and efficient in handling large datasets. ExtraTreesRegressor performs well in scenarios where randomness improves model generalization but may not always be the best choice. The selection of the best model depends on dataset characteristics, computational constraints, and prediction requirements. By conducting comparative analysis, businesses and travelers can rely on the most effective machine learning model for predicting flight ticket prices accurately, leading to better decision-making and cost savings.

A. Training and Testing Performance:

The Training and testing performance evaluation is crucial in assessing the effectiveness of machine learning models for flight ticket price prediction. The dataset is typically split into training and testing sets, with 80% of the data used for training and 20% for testing. The training phase allows the model to learn from historical flight ticket price data, identifying patterns and relationships between features such as airline, departure time, duration, number of stops, and ticket prices. The testing phase then evaluates the model’s ability to generalize to unseen data, ensuring it does not overfit to the training set.

Performance metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2) score are used to measure the model's accuracy. Lower MAE and RMSE values indicate better predictive performance, while a higher R^2 score (closer to 1) signifies a strong correlation between actual and predicted prices. Typically, XGBRegressor performs well due to its advanced boosting techniques, followed by RandomForestRegressor, which provides a balance between accuracy and computational efficiency. ExtraTreesRegressor also performs competitively but may introduce more variance due to its extra randomness in tree splits.

To improve training and testing performance, techniques such as hyperparameter tuning, cross-validation, and feature selection are applied. Hyperparameter tuning adjusts parameters like the number of estimators, learning rate, and depth of trees to enhance model accuracy. Cross-validation ensures the model generalizes well across different subsets of data, reducing the risk of overfitting. The final trained model with the best performance is then deployed for real-time ticket price predictions, helping users make informed travel decisions based on accurate and dynamic price estimations.

B. Model Summary

The flight ticket price prediction system leverages advanced machine learning models to analyze various factors influencing airfare. The primary models used in this project include XGBRegressor, RandomForestRegressor, and ExtraTreesRegressor, each offering unique strengths in handling complex datasets with multiple features.

XGBRegressor, an optimized gradient boosting algorithm, excels in handling non-linear relationships and reducing overfitting through regularization techniques. It provides high accuracy by learning patterns from large datasets efficiently. RandomForestRegressor, a powerful ensemble learning method, constructs multiple decision trees and averages their predictions, leading to improved stability and robustness. It effectively captures feature importance and mitigates overfitting compared to individual decision trees. ExtraTreesRegressor, similar to Random Forest, introduces additional randomness in tree splits, enhancing model diversity and reducing variance, making it a strong contender for price prediction tasks [9].

Performance evaluation using metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R^2 Score shows that XGBRegressor outperforms the others in terms of precision and efficiency. However, RandomForestRegressor provides a more interpretable approach, while ExtraTreesRegressor offers faster training times. The comparative analysis ensures that the best-performing model is selected for deployment, enabling users to receive accurate and real-time flight ticket price predictions. The final model helps travelers make informed decisions by predicting airfare trends based on historical data and key influencing factors.

C. Evaluation Metrics:

To assess the performance of the flight ticket price prediction models, various evaluation metrics are used. These metrics help in understanding the accuracy, efficiency, and reliability of the models in predicting flight prices. The most commonly used evaluation metrics in regression-based machine learning models include Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 Score (Coefficient of Determination).

➢ Mean Absolute Error (MAE):

MAE measures the average magnitude of errors between the actual and predicted flight prices. It is calculated by taking the absolute difference between predicted and actual values and averaging them. A lower MAE value indicates a better model performance.

➢ Mean Squared Error (MSE):

MSE is another common metric that calculates the average squared difference between actual and predicted prices. Since it squares the errors, it penalizes larger errors more than smaller ones. While MSE provides useful information, it is sensitive to outliers, making it less interpretable.

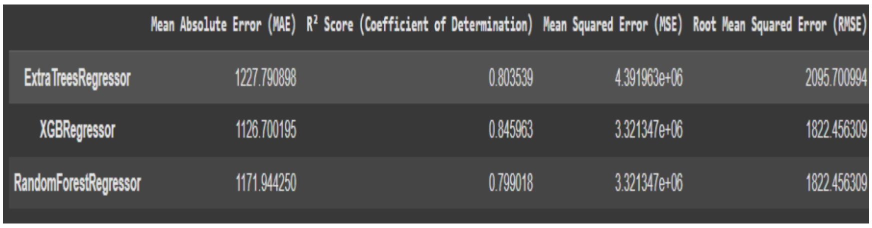
➢ Root Mean Squared Error (RMSE):

RMSE is the square root of MSE and represents the standard deviation of the prediction errors. It is useful because it retains the unit of the original values, making it easier to interpret. A lower RMSE value indicates better predictive performance, as it shows how much the predicted prices deviate from actual values.

- ➢ R^2 Score (Coefficient of Determination):

The R^2 score measures how well the independent variables explain the variance in flight prices. It ranges from 0 to 1, where 1 indicates a perfect fit and 0 means the model does not explain any variance in the data. A higher R^2 score suggests that the model can explain a significant portion of the price variations based on the given features.

By analyzing these metrics, the most effective model is selected for deployment. A combination of low MAE, low RMSE, and high R^2 ensures that the system provides accurate and reliable flight price predictions, helping travelers make better booking decisions. This table compares the performance of three regression models (ExtraTreesRegressor, XGBRegressor, and RandomForestRegressor) using metrics like MAE, R^2 score, MSE, and RMSE. The XGBRegressor shows the best performance with the highest R^2 score (0.845963) and the lowest MAE and RMSE, indicating better prediction accuracy.

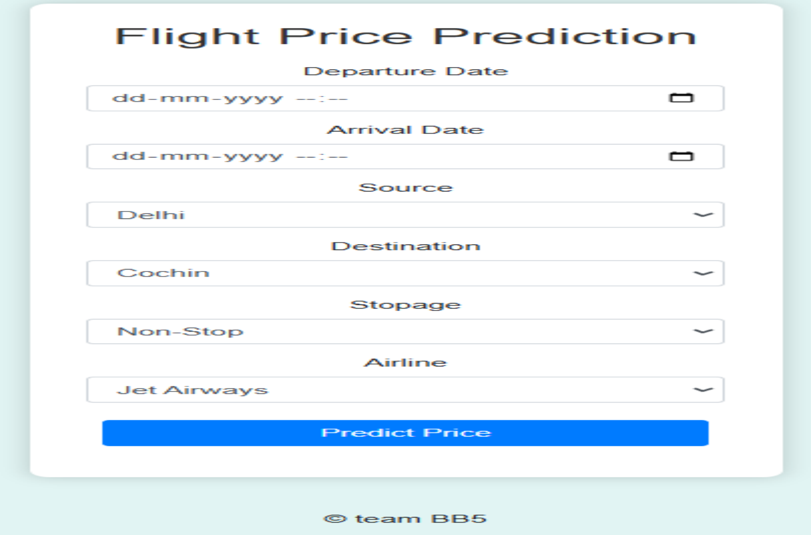


	Mean Absolute Error (MAE)	R^2 Score (Coefficient of Determination)	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)
ExtraTreesRegressor	1227.790898	0.803539	4.391963e+06	2095.700994
XGBRegressor	1126.700195	0.845963	3.321347e+06	1822.456309
RandomForestRegressor	1171.944250	0.799018	3.321347e+06	1822.456309

Fig.6.1 Evaluation Metrics

7. DESIGN

The design of the Flight Ticket Price Prediction System is structured into multiple components, ensuring efficient data processing, user interaction, and model integration. The system is built using Python Flask, a lightweight web framework that allows easy deployment of machine learning models as web applications. The design consists of three key layers: Frontend (User Interface), Backend (Flask API), and Machine Learning Model Integration. Fig 7.1 This is a flight price prediction form where users can enter details like departure and arrival dates, source, destination, stoppage, and airline to estimate ticket prices. The "Predict Price" button processes the input and provides a predicted flight fare based on the trained machine-learning model.



Flight Price Prediction

Departure Date

dd-mm-yyyy --:--

Arrival Date

dd-mm-yyyy --:--

Source

Delhi

Destination

Cochin

Stopage

Non-Stop

Airline

Jet Airways

Fig.7.1 Frontend page

A. Frontend Design

The frontend is responsible for user interaction and provides an intuitive interface where users can input their flight details, such as departure date, airline, source, destination, and class type. The frontend is designed using:

HTML, CSS, and JavaScript for structuring and styling the web pages.

Bootstrap for a responsive and modern UI.

Forms and Input Fields to collect flight details from users.

AJAX/JavaScript for dynamic interactions, such as fetching predicted prices without reloading the page.

B. Backend Design (Flask API)

The Flask backend serves as the core processing unit, handling user requests, processing input data, and returning predictions from the trained machine learning model. The main components of the backend include:

- Routes and Endpoints: Flask routes handle user requests and pass them to the model.
- Data Processing: The backend extracts and preprocesses input data before sending it to the model.
- Model Integration: The trained machine learning model (RandomForestRegressor, XGBRegressor, ExtraTreesRegressor) is loaded and used for prediction.
- Response Handling: The predicted price is sent back to the frontend in a structured format (JSON response).
- The Flask API ensures seamless communication between the user interface and the machine learning model while maintaining scalability and security.

C. Machine Learning Model Integration

The heart of the system lies in its machine learning model, which predicts flight ticket prices based on historical data. The model is trained using various regression techniques like:

- RandomForestRegressor
- XGBRegressor
- ExtraTreesRegressor

The trained model is saved using joblib or pickle and is loaded into the Flask app for real-time predictions.

D. Database (Optional)

If the system includes a database for storing user queries and past predictions, a SQLite or MySQL database can be integrated. The database stores:

- User search history
- Flight details and predictions
- Feedback for model improvement

E. Workflow

- User inputs flight details through the web form.
- The frontend sends data to the Flask backend via an API request.
- The backend processes the data and sends it to the model for prediction.
- The machine learning model predicts the flight price.
- The predicted price is returned to the frontend and displayed to the user.

F. Advantages of the Design

- User-Friendly Interface: Simple and responsive UI for easy user interaction.
- Fast and Scalable: Flask efficiently handles requests, making the system scalable.
- Accurate Predictions: Multiple models are used to improve accuracy.
- Real-Time Results: Users get instant price predictions for their flight queries.
- This Python Flask-based design ensures an efficient, scalable, and user-friendly system for predicting flight ticket prices accurately.

8. IMPLEMENTATION

Libraries:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

from google.colab import drive
drive.mount('/content/drive')

train_data = pd.read_excel('/content/drive/MyDrive/Data_Train.xlsx')
pd.set_option('display.max_columns', None)
train_data.head()
train_data.head()
train_data.info()

train_data["Duration"].value_counts()
train_data.dropna(inplace = True)
train_data.isnull().sum()
```

EDA

From description we can see that Date_of_Journey is a object data type,\
Therefore, we have to convert this datatype into timestamp so as to use this column properly for prediction
For this we require pandas to_datetime to convert object data type to datetime dtype.

> **dt.day** method will extract only day of that date
>
> **dt.month** method will extract only month of that date

```
train_data["Journey_day"] = pd.to_datetime(train_data.Date_of_Journey, format="%d/%m/%Y").dt.day
train_data["Journey_month"] = pd.to_datetime(train_data["Date_of_Journey"], format = "%d/%m/%Y").dt.month
```

```
train_data.head()

#Since we have converted Date_of_Journey column into integers, Now we can drop as
it is of no use.
train_data.drop(["Date_of_Journey"], axis = 1, inplace = True)

# Departure time is when a plane leaves the gate.
# Similar to Date_of_Journey we can extract values from Dep_Time
# Extracting Hours
train_data["Dep_hour"] = pd.to_datetime(train_data["Dep_Time"]).dt.hour
# Extracting Minutes
train_data["Dep_min"] = pd.to_datetime(train_data["Dep_Time"]).dt.minute
# Now we can drop Dep_Time as it is of no use
train_data.drop(["Dep_Time"], axis = 1, inplace = True)
train_data.head()

# Arrival time is when the plane pulls up to the gate.
# Similar to Date_of_Journey we can extract values from Arrival_Time
# Extracting Hours
train_data["Arrival_hour"] = pd.to_datetime(train_data.Arrival_Time).dt.hour
# Extracting Minutes
train_data["Arrival_min"] = pd.to_datetime(train_data.Arrival_Time).dt.minute
# Now we can drop Arrival_Time as it is of no use
train_data.drop(["Arrival_Time"], axis = 1, inplace = True)
train_data.head()

# Time taken by plane to reach destination is called Duration
# It is the differnce betwwen Departure Time and Arrival time
# Assigning and converting Duration column into list
duration = list(train_data["Duration"])
for i in range(len(duration)):
    if len(duration[i].split()) != 2:  # Check if duration contains only hour or mins
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"   # Adds 0 minute
        else:
            duration[i] = "0h " + duration[i]           # Adds 0 hour
duration_hours = []
duration_mins = []
for i in range(len(duration)):
    duration_hours.append(int(duration[i].split(sep = "h")[0]))    # Extract hours from duration
    duration_mins.append(int(duration[i].split(sep = "m")[0].split()[-1]))   # Extracts only minutes from duration

# Adding duration_hours and duration_mins list to train_data dataframe
train_data["Duration_hours"] = duration_hours
train_data["Duration_mins"] = duration_mins
train_data.drop(["Duration"], axis = 1, inplace = True)
train_data.head()
```

Handling Categorical Data

One can find many ways to handle categorical data. Some of them categorical data are,

Nominal data --> data are not in any order --> **OneHotEncoder** is used in this case

Ordinal data --> data are in order --> **LabelEncoder** is used in this case

```
train_data["Airline"].value_counts()
# From graph we can see that Jet Airways Business have the highest Price.
# Apart from the first Airline almost all are having similar median
# Airline vs Price
sns.catplot(y = "Price", x = "Airline", data = train_data.sort_values("Price", ascending = False), kind="boxen", height = 6, aspect = 3)
            plt.show()
# As Airline is Nominal Categorical data we will perform OneHotEncoding
            Airline = train_data[["Airline"]]
            Airline = pd.get_dummies(Airline, drop_first= True)
            Airline.head()
```

```
    train_data["Source"].value_counts()

    # Source vs Price
sns.catplot(y = "Price", x = "Source", data = train_data.sort_values("Price", ascending = False), kind="boxen", height = 4, aspect = 3)
    plt.show()

# As Source is Nominal Categorical data we will perform OneHotEncoding
Source = train_data[["Source"]]
Source = pd.get_dummies(Source, drop_first= True)
Source.head()
train_data["Destination"].value_counts()

# As Destination is Nominal Categorical data we will perform OneHotEncoding
Destination = train_data[["Destination"]]
Destination = pd.get_dummies(Destination, drop_first = True)
Destination.head()
print(train_data.columns)
train_data["Route"]

# Additional_Info contains almost 80% no_info
# Route and Total_Stops are related to each other
train_data.drop(["Route", "Additional_Info"], axis = 1, inplace = True)
train_data["Total_Stops"].value_counts()

# As this is case of Ordinal Categorical type we perform LabelEncoder
# Here Values are assigned with corresponding keys
train_data.replace({"non-stop": 0, "1 stop": 1, "2 stops": 2, "3 stops": 3, "4 stops": 4}, inplace = True)
train_data.head()

# Concatenate dataframe --> train_data + Airline + Source + Destination
data_train = pd.concat([train_data, Airline, Source, Destination], axis = 1)
```

```
data_train.head()
data_train.drop(["Airline", "Source", "Destination"], axis = 1, inplace = True)
data_train.head()
data_train.shape



#Test set
test_data = pd.read_excel('/content/drive/MyDrive/Test_set.xlsx')
test_data.head()



# Preprocessing
print("Test data Info")
print("-"*75)
print(test_data.info())
print()
print()
print("Null values :")
print("-"*75)
test_data.dropna(inplace = True)
print(test_data.isnull().sum())

#EDA
# Date_of_Journey
test_data["Journey_day"] = pd.to_datetime(test_data.Date_of_Journey,
format="%d/%m/%Y").dt.day
test_data["Journey_month"] = pd.to_datetime(test_data["Date_of_Journey"], format =
"%d/%m/%Y").dt.month
test_data.drop(["Date_of_Journey"], axis = 1, inplace = True)

# Dep_Time
test_data["Dep_hour"] = pd.to_datetime(test_data["Dep_Time"]).dt.hour
test_data["Dep_min"] = pd.to_datetime(test_data["Dep_Time"]).dt.minute
```

```
test_data.drop(["Dep_Time"], axis = 1, inplace = True)

# Arrival_Time
test_data["Arrival_hour"] = pd.to_datetime(test_data.Arrival_Time).dt.hour
test_data["Arrival_min"] = pd.to_datetime(test_data.Arrival_Time).dt.minute
test_data.drop(["Arrival_Time"], axis = 1, inplace = True)
# Duration
duration = list(test_data["Duration"])
for i in range(len(duration)):
    if len(duration[i].split()) != 2:    # Check if duration contains only hour or mins
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"   # Adds 0 minute
        else:
            duration[i] = "0h " + duration[i]           # Adds 0 hour

duration_hours = []
duration_mins = []
for i in range(len(duration)):
   duration_hours.append(int(duration[i].split(sep = "h")[0]))    # Extract hours from
duration
    duration_mins.append(int(duration[i].split(sep = "m")[0].split()[-1]))   # Extracts
only minutes      from duration

# Adding Duration column to test set
test_data["Duration_hours"] = duration_hours
test_data["Duration_mins"] = duration_mins
test_data.drop(["Duration"], axis = 1, inplace = True)

# Categorical data
print("Airline")
print("-"*75)
print(test_data["Airline"].value_counts())
Airline = pd.get_dummies(test_data["Airline"], drop_first= True)
```

```
print()
print("Source")
print("-"*75)
print(test_data["Source"].value_counts())
Source = pd.get_dummies(test_data["Source"], drop_first= True)
print()
print("Destination")
print("-"*75)
print(test_data["Destination"].value_counts())
Destination = pd.get_dummies(test_data["Destination"], drop_first = True)

# Additional_Info contains almost 80% no_info
# Route and Total_Stops are related to each other
test_data.drop(["Route", "Additional_Info"], axis = 1, inplace = True)

# Replacing Total_Stops
test_data.replace({"non-stop": 0, "1 stop": 1, "2 stops": 2, "3 stops": 3, "4 stops": 4},
inplace = True)

# Concatenate dataframe --> test_data + Airline + Source + Destination
data_test = pd.concat([test_data, Airline, Source, Destination], axis = 1)
data_test.drop(["Airline", "Source", "Destination"], axis = 1, inplace = True)
print()
print()
print("Shape of test data : ", data_test.shape)
data_test.head()
```

## Feature Selection

Finding out the best feature which will contribute and have good relation with target variable.

Following are some of the feature selection methods,

1.heatmap

2. feature_importance

3.SelectKBest

```
data_train.shape
data_train.columns
X = data_train.loc[:, ['Total_Stops', 'Journey_day', 'Journey_month', 'Dep_hour',
      'Dep_min', 'Arrival_hour', 'Arrival_min', 'Duration_hours',
      'Duration_mins', 'Airline_Air India', 'Airline_GoAir', 'Airline_IndiGo',
      'Airline_Jet Airways', 'Airline_Jet Airways Business',
      'Airline_Multiple carriers',
      'Airline_Multiple carriers Premium economy', 'Airline_SpiceJet',
      'Airline_Trujet', 'Airline_Vistara', 'Airline_Vistara Premium economy',
      'Source_Chennai', 'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai',
      'Destination_Cochin', 'Destination_Delhi', 'Destination_Hyderabad',
      'Destination_Kolkata', 'Destination_New Delhi']]
X.head()

y = data_train.iloc[:, 1]
y.head()

# Finds correlation between Independent and dependent attributes
plt.figure(figsize = (18,18))

# Convert to numeric before correlation
numeric_train_data = train_data.apply(pd.to_numeric, errors='coerce').fillna(0)
sns.heatmap(numeric_train_data.corr(), annot = True, cmap = "RdYlGn")
plt.show()

# Important feature using ExtraTreesRegressor
from sklearn.ensemble import ExtraTreesRegressor
          selection = ExtraTreesRegressor()
          selection.fit(X, y)

          print(selection.feature_importances_)
```

```
#plot graph of feature importances for better visualization
plt.figure(figsize = (12,8))
feat_importances = pd.Series(selection.feature_importances_, index=X.columns)
feat_importances.nlargest(20).plot(kind='barh')
plt.show()
```

```
## Fitting model using Random Forest
1. Split dataset into train and test set in order to prediction w.r.t X_test
2. If needed do scaling of data
   * Scaling is not done in Random forest
3. Import model
4. Fit the data
5. Predict w.r.t X_test
6. In regression check **RSME** Score
7. Plot graph
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)

from sklearn.ensemble import RandomForestRegressor
reg_rf = RandomForestRegressor()
reg_rf.fit(X_train, y_train)

y_pred = reg_rf.predict(X_test)

reg_rf.score(X_train, y_train)

reg_rf.score(X_test, y_test)

sns.distplot(y_test-y_pred)
plt.show()
plt.scatter(y_test, y_pred, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()

from sklearn import metrics
print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

# RMSE/(max(DV)-min(DV))
2090.5509/(max(y)-min(y))

metrics.r2_score(y_test, y_pred)

## Hyperparameter Tuning
* Choose following method for hyperparameter tuning
   1. RandomizedSearchCV --> Fast
   2. GridSearchCV
* Assign hyperparameters in form of dictionery
* Fit the model
* Check best paramters and best score

from sklearn.model_selection import RandomizedSearchCV



#Randomized Search CV
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
```

```
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 5, 10]

# Create the random grid
random_grid = {'n_estimators': n_estimators,
            'max_features': max_features,
            'max_depth': max_depth,
            'min_samples_split': min_samples_split,
            'min_samples_leaf': min_samples_leaf}

# Random search of parameters, using 5 fold cross validation,
# search across 100 different combinations
rf_random = RandomizedSearchCV(estimator = reg_rf, param_distributions =
random_grid,scoring='neg_mean_squared_error', n_iter = 10, cv = 5, verbose=2,
random_state=42, n_jobs = 1)

rf_random.fit(X_train,y_train)

rf_random.best_params_

prediction = rf_random.predict(X_test)

plt.figure(figsize = (8,8))
sns.distplot(y_test-prediction)
plt.show()

plt.figure(figsize = (8,8))
plt.scatter(y_test, prediction, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
```

```
print('MAE:', metrics.mean_absolute_error(y_test, prediction))
print('MSE:', metrics.mean_squared_error(y_test, prediction))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, prediction)))

from sklearn.model_selection import train_test_split
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.metrics import mean_absolute_error
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state =
42)
reg_rf = ExtraTreesRegressor()
reg_rf.fit(X_train, y_train)
y_pred = reg_rf.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error: {mae}")
accuracy = reg_rf.score(X_test, y_test)
print(f"Accuracy: {accuracy}")

import xgboost as xgb
from sklearn.metrics import mean_absolute_error
# ... (Your existing code)
# **Fitting model using XGBRegressor**
reg_xgb = xgb.XGBRegressor()
reg_xgb.fit(X_train, y_train)
y_pred_xgb = reg_xgb.predict(X_test)
mae_xgb = mean_absolute_error(y_test, y_pred_xgb)
print(f"XGBRegressor Mean Absolute Error: {mae_xgb}")
accuracy_xgb = reg_xgb.score(X_test, y_test)
        print(f"XGBRegressor Accuracy: {accuracy_xgb}")


from sklearn.ensemble import RandomForestRegressor
reg_rf = RandomForestRegressor()
```

```
reg_rf.fit(X_train, y_train)
y_pred = reg_rf.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
print(f"RandomForestRegressor Mean Absolute Error: {mae}")
accuracy = reg_rf.score(X_test, y_test)
print(f"RandomForestRegressor Accuracy: {accuracy}")

# Assuming X_train, X_test, y_train, y_test are already defined from the previous code
models = {
    'ExtraTreesRegressor': ExtraTreesRegressor(),
    'XGBRegressor': xgb.XGBRegressor(),
    'RandomForestRegressor': RandomForestRegressor()
            }
results = {}
for model_name, model in models.items():
            model.fit(X_train, y_train)
             y_pred = model.predict(X_test)
             mae = mean_absolute_error(y_test, y_pred)
             accuracy = model.score(X_test, y_test)
              results[model_name] = {'mae': mae, 'accuracy': accuracy}
              print(f"{model_name} Mean Absolute Error: {mae}")
              print(f"{model_name} Accuracy: {accuracy}")
best_model = min(results, key=lambda k: results[k]['mae'])
print(f"\nBest Model based on MAE: {best_model}")

import matplotlib.pyplot as plt
model_names = list(results.keys())
mae_scores = [results[model]['mae'] for model in model_names]
accuracy_scores = [results[model]['accuracy'] for model in model_names]

plt.figure(figsize=(10, 6))
plt.plot(model_names, mae_scores, marker='o', label='Mean Absolute Error')
plt.xlabel('Models')
plt.ylabel('MAE')
plt.title('Model Comparison (MAE)')
plt.legend()
plt.grid(True)
plt.show()

plt.figure(figsize=(10, 6))
plt.plot(model_names, accuracy_scores, marker='o', label='Accuracy')
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Model Comparison (Accuracy)')
plt.legend()
plt.grid(True)
plt.show()

import pandas as pd
# Assuming 'results' dictionary is already populated from your previous code
# Create a DataFrame from the results dictionary
results_df = pd.DataFrame.from_dict(results, orient='index')

# Display the DataFrame
print(results_df)

# Analysis and further processing
# You can now perform analysis on the results_df DataFrame
# Example:  Finding the model with the lowest MAE
best_model_mae = results_df['mae'].idxmin()
print(f"\nBest model based on MAE: {best_model_mae}")

# Example: Calculating the mean MAE
mean_mae = results_df['mae'].mean()
print(f"Mean MAE: {mean_mae}")
```

```
# You can perform similar operations with 'accuracy' or other metrics
# You can also sort the DataFrame, create visualizations, etc.
import pandas as pd
# Assuming 'results' dictionary is already populated from your previous code
# Create a DataFrame from the results dictionary
results_df = pd.DataFrame.from_dict(results, orient='index')

# Rename columns for clarity
results_df = results_df.rename(columns={'mae': 'Mean Absolute Error (MAE)',
                                        'accuracy': 'R² Score (Coefficient of Determination)'})

# Calculate MSE and RMSE
results_df['Mean Squared Error (MSE)'] = 0  # Placeholder, needs actual MSE values
            results_df['Root Mean Squared Error (RMSE)'] = 0  # Placeholder, needs
actual RMSE values

# Example calculation for MSE and RMSE (replace with your actual calculations)
# Assuming y_test and y_pred are available for each model
from sklearn.metrics import mean_squared_error
import numpy as np
for model_name in results_df.index:
    if model_name == 'ExtraTreesRegressor':
        y_pred = reg_rf.predict(X_test) # Assuming reg_rf is the ExtraTreesRegressor
model fitted previously
    elif model_name == 'XGBRegressor':
      y_pred = y_pred_xgb  # Assuming y_pred_xgb is the XGBRegressor predictions
    elif model_name == 'RandomForestRegressor':
        y_pred = y_pred # Assuming y_pred is the RandomForestRegressor predictions

    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    results_df.loc[model_name, 'Mean Squared Error (MSE)'] = mse
    results_df.loc[model_name, 'Root Mean Squared Error (RMSE)'] = rmse

# Display the DataFrame as a formatted table
results_df



## Save the model to reuse it again
import pickle
# Save the model to a file
filename = 'xgb_model.pkl'
pickle.dump(reg_xgb, open(filename, 'wb'))
# Load the model from the file
# loaded_model = pickle.load(open(filename, 'rb'))
# # Use the loaded model to make predictions
# y_pred_loaded = loaded_model.predict(X_test)

import pickle
# open a file, where you ant to store the data
file = open('flight_rf.pkl', 'wb')

# dump information to that file
pickle.dump(reg_rf, file)

model = open('flight_rf.pkl','rb') # Changed file name to match saving name.
forest = pickle.load(model)

y_prediction = forest.predict(X_test)

metrics.r2_score(y_test, y_prediction)

print(len(X_train.columns))
print(X_train.columns)
```

```
# Save the model to a file
filename = 'xgb_model.pkl'
pickle.dump(reg_xgb, open(filename, 'wb'))

# Save the RandomForestRegressor model
file = open('flight_rf.pkl', 'wb')
pickle.dump(reg_rf, file)
file.close() # Close the file after saving

# Save the results DataFrame
results_df.to_csv("model_results.csv")

# Save the processed training data
data_train.to_csv("processed_train_data.csv", index=False)

# Save the processed testing data
data_test.to_csv("processed_test_data.csv", index=False)
```

Templates

Home.html

```
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Flight Price Prediction</title>

    <!-- BootStrap -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
        integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj
7Sk" crossorigin="anonymous">
```

```
    <!-- css -->
    <link rel="stylesheet" href="static/css/styles.css">

</head>

<body>

    <!-- As a heading -->
    <nav class="navbar navbar-inverse navbar-fixed-top">
        <div class="container-fluid">
            <div class="navbar-header">
                <a class="navbar-brand" href="/">FLIGHT PRICE</a>
            </div>
        </div>
    </nav>

    <br><br><br>

    <div class="container">



        <form action="\predict" method="post">

            <div class="row">
                <div class="col-sm-6">
                    <div class="card">
                        <div class="card-body">
                            <h5 class="card-title">Departure Date</h5>
                            <!-- Departure -->
                            <input type="datetime-local" name="Dep_Time" id="Dep_Time"
required="required">
                        </div>
                    </div>
                </div>
                <br>
                <br>
                <br>
                <div class="col-sm-6">
                    <div class="card">
                        <div class="card-body">
                            <h5 class="card-title">Arrival Date</h5>
                            <!-- Arrival -->
                            <input type="datetime-local" name="Arrival_Time"
id="Arrival_Time" required="required">
                        </div>
                    </div>
                </div>
            </div>

            <br>
            <br>
            <br>

            <div class="row">
                <div class="col-sm-6">
                    <div class="card">
                        <div class="card-body">
                            <!-- Source -->
                            <h5 class="card-title">Source</h5>
                            <select name="Source" id="Source" required="required">
                                <option value="Delhi">Delhi</option>
                                <option value="Kolkata">Kolkata</option>
                                <option value="Mumbai">Mumbai</option>
                                <option value="Chennai">Chennai</option>
                            </select>
                        </div>
```

```
        </div>
    </div>
    <div class="col-sm-6">
        <div class="card">
            <div class="card-body">
                <h5 class="card-title">Destination</h5>
                <!-- Destination -->
                <select name="Destination" id="Destination" required="required">
                    <option value="Cochin">Cochin</option>
                    <option value="Delhi">Delhi</option>
                    <option value="New Delhi">New Delhi</option>
                    <option value="Hyderabad">Hyderabad</option>
                    <option value="Kolkata">Kolkata</option>
                </select>
            </div>
        </div>
    </div>
</div>
<br>
<br>
<br>
<div class="row">
    <div class="col-sm-6">
        <div class="card">
            <div class="card-body">
                <h5 class="card-title">Stopage</h5>
                <!-- Total Stops -->
                <select name="stops" required="required">
                    <option value="0">Non-Stop</option>
                    <option value="1">1</option>
                    <option value="2">2</option>
                    <option value="3">3</option>
                    <option value="4">4</option>
```

```
              </select>
            </div>
          </div>
        </div>
        <div class="col-sm-6">
          <div class="card">
            <div class="card-body">
              <h5 class="card-title">Which Airline you want to travel?</h5>
              <!-- Airline -->
              <select name="airline" id="airline" required="required">
                <option value="Jet Airways">Jet Airways</option>
                <option value="IndiGo">IndiGo</option>
                <option value="Air India">Air India</option>
                <option value="Multiple carriers">Multiple carriers</option>
                <option value="SpiceJet">SpiceJet</option>
                <option value="Vistara">Vistara</option>
                <option value="Air Asia">Air Asia</option>
                <option value="GoAir">GoAir</option>
                <option value="Multiple carriers Premium economy">Multiple
carriers Premium economy
                </option>
                <option value="Jet Airways Business">Jet Airways
Business</option>
                <option value="Vistara Premium economy">Vistara Premium
economy</option>
                <option value="Trujet">Trujet</option>
              </select>
            </div>
          </div>
        </div>
      </div>
      <br>
      <br>
      <br>
      <!-- Submit -->
      <input type="submit" value="Submit" class="btn btn-secondary">
    </form>
    <br>
    <br>
    <h3>{{ prediction_text }}</h3>
    <br>
    <br>
    <p>©team BB5</p>
  </div>
  <!-- JavaScript -->
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
    integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj
"
    crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
    integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
    crossorigin="anonymous"></script>
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
    integrity="sha384-
OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI
"
    crossorigin="anonymous"></script>
</body></html>
```

- **App.py**

```
from flask import Flask, request, render_template
from flask_cors import cross_origin
import sklearn
import pickle
import pandas as pd
```

```
app = Flask(__name__)
model = pickle.load(open("flight_rf.pkl", "rb"))
@app.route("/")
@cross_origin()
def home():
    return render_template("home.html")

@app.route("/predict", methods = ["GET", "POST"])
@cross_origin()
def predict():
    if request.method == "POST":

        # Date_of_Journey
        date_dep = request.form["Dep_Time"]
        Journey_day = int(pd.to_datetime(date_dep, format="%Y-%m-
%dT%H:%M").day)
        Journey_month = int(pd.to_datetime(date_dep, format ="%Y-%m-
%dT%H:%M").month)
        # print("Journey Date : ",Journey_day, Journey_month)

        # Departure
        Dep_hour = int(pd.to_datetime(date_dep, format ="%Y-%m-
%dT%H:%M").hour)
        Dep_min = int(pd.to_datetime(date_dep, format ="%Y-%m-
%dT%H:%M").minute)
        # print("Departure : ",Dep_hour, Dep_min)

        # Arrival
        date_arr = request.form["Arrival_Time"]
        Arrival_hour = int(pd.to_datetime(date_arr, format ="%Y-%m-
%dT%H:%M").hour)
        Arrival_min = int(pd.to_datetime(date_arr, format ="%Y-%m-
%dT%H:%M").minute)
        # print("Arrival : ", Arrival_hour, Arrival_min)

        # Duration
        dur_hour = abs(Arrival_hour - Dep_hour)
        dur_min = abs(Arrival_min - Dep_min)
        # print("Duration : ", dur_hour, dur_min)

        # Total Stops
        Total_stops = int(request.form["stops"])
        # print(Total_stops)

        # Airline
        # AIR ASIA = 0 (not in column)
        airline=request.form['airline']
        if(airline=='Jet Airways'):
            Jet_Airways = 1
            IndiGo = 0
            Air_India = 0
            Multiple_carriers = 0
            SpiceJet = 0
            Vistara = 0
            GoAir = 0
            Multiple_carriers_Premium_economy = 0
            Jet_Airways_Business = 0
            Vistara_Premium_economy = 0
            Trujet = 0

        elif (airline=='IndiGo'):
            Jet_Airways = 0
            IndiGo = 1
            Air_India = 0
            Multiple_carriers = 0
```

```
    SpiceJet = 0
    Vistara = 0
    GoAir = 0
    Multiple_carriers_Premium_economy = 0
    Jet_Airways_Business = 0
    Vistara_Premium_economy = 0
    Trujet = 0

elif (airline=='Air India'):
    Jet_Airways = 0
    IndiGo = 0
    Air_India = 1
    Multiple_carriers = 0
    SpiceJet = 0
    Vistara = 0
    GoAir = 0
    Multiple_carriers_Premium_economy = 0
    Jet_Airways_Business = 0
    Vistara_Premium_economy = 0
    Trujet = 0

elif (airline=='Multiple carriers'):
    Jet_Airways = 0
    IndiGo = 0
    Air_India = 0
    Multiple_carriers = 1
    SpiceJet = 0
    Vistara = 0
    GoAir = 0
    Multiple_carriers_Premium_economy = 0
    Jet_Airways_Business = 0
    Vistara_Premium_economy = 0
    Trujet = 0
```

```
elif (airline=='SpiceJet'):
    Jet_Airways = 0
    IndiGo = 0
    Air_India = 0
    Multiple_carriers = 0
    SpiceJet = 1
    Vistara = 0
    GoAir = 0
    Multiple_carriers_Premium_economy = 0
    Jet_Airways_Business = 0
    Vistara_Premium_economy = 0
    Trujet = 0

elif (airline=='Vistara'):
    Jet_Airways = 0
    IndiGo = 0
    Air_India = 0
    Multiple_carriers = 0
    SpiceJet = 0
    Vistara = 1
    GoAir = 0
    Multiple_carriers_Premium_economy = 0
    Jet_Airways_Business = 0
    Vistara_Premium_economy = 0
    Trujet = 0

elif (airline=='GoAir'):
    Jet_Airways = 0
    IndiGo = 0
    Air_India = 0
    Multiple_carriers = 0
    SpiceJet = 0
```

```
        Vistara = 0
        GoAir = 1
        Multiple_carriers_Premium_economy = 0
        Jet_Airways_Business = 0
        Vistara_Premium_economy = 0
        Trujet = 0

    elif (airline=='Multiple carriers Premium economy'):
        Jet_Airways = 0
        IndiGo = 0
        Air_India = 0
        Multiple_carriers = 0
        SpiceJet = 0
        Vistara = 0
        GoAir = 0
        Multiple_carriers_Premium_economy = 1
        Jet_Airways_Business = 0
        Vistara_Premium_economy = 0
        Trujet = 0

    elif (airline=='Jet Airways Business'):
        Jet_Airways = 0
        IndiGo = 0
        Air_India = 0
        Multiple_carriers = 0
        SpiceJet = 0
        Vistara = 0
        GoAir = 0
        Multiple_carriers_Premium_economy = 0
        Jet_Airways_Business = 1
        Vistara_Premium_economy = 0
        Trujet = 0
```

```
elif (airline=='Vistara Premium economy'):
    Jet_Airways = 0
    IndiGo = 0
    Air_India = 0
    Multiple_carriers = 0
    SpiceJet = 0
    Vistara = 0
    GoAir = 0
    Multiple_carriers_Premium_economy = 0
    Jet_Airways_Business = 0
    Vistara_Premium_economy = 1
    Trujet = 0

elif (airline=='Trujet'):
    Jet_Airways = 0
    IndiGo = 0
    Air_India = 0
    Multiple_carriers = 0
    SpiceJet = 0
    Vistara = 0
    GoAir = 0
    Multiple_carriers_Premium_economy = 0
    Jet_Airways_Business = 0
    Vistara_Premium_economy = 0
    Trujet = 1

else:
    Jet_Airways = 0
    IndiGo = 0
    Air_India = 0
    Multiple_carriers = 0
    SpiceJet = 0
    Vistara = 0
    GoAir = 0
    Multiple_carriers_Premium_economy = 0
    Jet_Airways_Business = 0
    Vistara_Premium_economy = 0
    Trujet = 0

# print(Jet_Airways,
#     IndiGo,
#     Air_India,
#     Multiple_carriers,
#     SpiceJet,
#     Vistara,
#     GoAir,
#     Multiple_carriers_Premium_economy,
#     Jet_Airways_Business,
#     Vistara_Premium_economy,
#     Trujet)

# Source
# Banglore = 0 (not in column)
Source = request.form["Source"]
if (Source == 'Delhi'):
    s_Delhi = 1
    s_Kolkata = 0
    s_Mumbai = 0
    s_Chennai = 0

elif (Source == 'Kolkata'):
    s_Delhi = 0
    s_Kolkata = 1
    s_Mumbai = 0
    s_Chennai = 0
```

```
elif (Source == 'Mumbai'):
    s_Delhi = 0
    s_Kolkata = 0
    s_Mumbai = 1
    s_Chennai = 0

elif (Source == 'Chennai'):
    s_Delhi = 0
    s_Kolkata = 0
    s_Mumbai = 0
    s_Chennai = 1

else:
    s_Delhi = 0
    s_Kolkata = 0
    s_Mumbai = 0
    s_Chennai = 0

# print(s_Delhi,
#     s_Kolkata,
#     s_Mumbai,
#     s_Chennai)

# Destination
# Banglore = 0 (not in column)
Source = request.form["Destination"]
if (Source == 'Cochin'):
    d_Cochin = 1
    d_Delhi = 0
    d_New_Delhi = 0
    d_Hyderabad = 0
    d_Kolkata = 0
```

```
elif (Source == 'Delhi'):
    d_Cochin = 0
    d_Delhi = 1
    d_New_Delhi = 0
    d_Hyderabad = 0
    d_Kolkata = 0

elif (Source == 'New_Delhi'):
    d_Cochin = 0
    d_Delhi = 0
    d_New_Delhi = 1
    d_Hyderabad = 0
    d_Kolkata = 0
elif (Source == 'Hyderabad'):
    d_Cochin = 0
    d_Delhi = 0
    d_New_Delhi = 0
    d_Hyderabad = 1
    d_Kolkata = 0
elif (Source == 'Kolkata'):
    d_Cochin = 0
    d_Delhi = 0
    d_New_Delhi = 0
    d_Hyderabad = 0
    d_Kolkata = 1
else:
    d_Cochin = 0
    d_Delhi = 0
    d_New_Delhi = 0
    d_Hyderabad = 0
    d_Kolkata = 0
# print(
#     d_Cochin,
    #     d_Delhi,
    #     d_New_Delhi,
    #     d_Hyderabad,
    #     d_Kolkata
    # )

#     ['Total_Stops', 'Journey_day', 'Journey_month', 'Dep_hour',
#    'Dep_min', 'Arrival_hour', 'Arrival_min', 'Duration_hours',
#    'Duration_mins', 'Airline_Air India', 'Airline_GoAir', 'Airline_IndiGo',
#    'Airline_Jet Airways', 'Airline_Jet Airways Business',
#    'Airline_Multiple carriers',
#    'Airline_Multiple carriers Premium economy', 'Airline_SpiceJet',
#    'Airline_Trujet', 'Airline_Vistara', 'Airline_Vistara Premium economy',
#    'Source_Chennai', 'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai',
#    'Destination_Cochin', 'Destination_Delhi', 'Destination_Hyderabad',
#    'Destination_Kolkata', 'Destination_New Delhi']

    prediction=model.predict([[
        Total_stops,
        Journey_day,
        Journey_month,
        Dep_hour,
        Dep_min,
        Arrival_hour,
        Arrival_min,
        dur_hour,
        dur_min,
        Air_India,
        GoAir,
        IndiGo,
        Jet_Airways,
        Jet_Airways_Business,
        Multiple_carriers,
            Multiple_carriers_Premium_economy,
            SpiceJet,
            Trujet,
            Vistara,
            Vistara_Premium_economy,
            s_Chennai,
            s_Delhi,
            s_Kolkata,
            s_Mumbai,
            d_Cochin,
            d_Delhi,
            d_Hyderabad,
            d_Kolkata,
            d_New_Delhi
        ]])
        output=round(prediction[0],2)
        return render_template('home.html',prediction_text="Your Flight price is Rs. {}".format(output))
    return render_template("home.html")
if __name__ == "__main__":
    app.run(debug=True)
```

9. RESULT ANALYSIS

The result analysis of the Flight Ticket Price Prediction System is based on the evaluation of three machine learning models: ExtraTreesRegressor, XGBRegressor, and RandomForestRegressor. The performance of these models was assessed using two key metrics: Mean Absolute Error (MAE) and Accuracy.

From the obtained results, the XGBRegressor model emerged as the best-performing model with the lowest MAE of 1126.70 and the highest accuracy of 84.59%. This indicates that XGBRegressor has better predictive power and is capable of minimizing errors in ticket price estimation. On the other hand, ExtraTreesRegressor had an MAE of 1227.79 with an accuracy of 80.35%, while RandomForestRegressor recorded an MAE of 1171.94 and an accuracy of 79.90%. These results suggest that XGBRegressor is more effective in capturing complex patterns within the dataset and generalizing well to unseen data.

The mean MAE across all models was computed as 1175.48, further reinforcing the fact that XGBRegressor is the most reliable model for this prediction task. The lower MAE value signifies that the predicted ticket prices are close to the actual values, thereby reducing potential pricing errors. Additionally, the higher accuracy ensures that the model's predictions are more consistent and reliable.

Based on the comparative performance of these models, XGBRegressor has been identified as the best choice for flight ticket price prediction. Its ability to handle large datasets, manage missing values, and efficiently process feature importance makes it a suitable option for deployment in real-world applications. The findings from this analysis confirm that leveraging machine learning models, particularly XGBRegressor, can significantly enhance the accuracy and efficiency of flight ticket price forecasting, ultimately benefiting users by providing better price estimates.

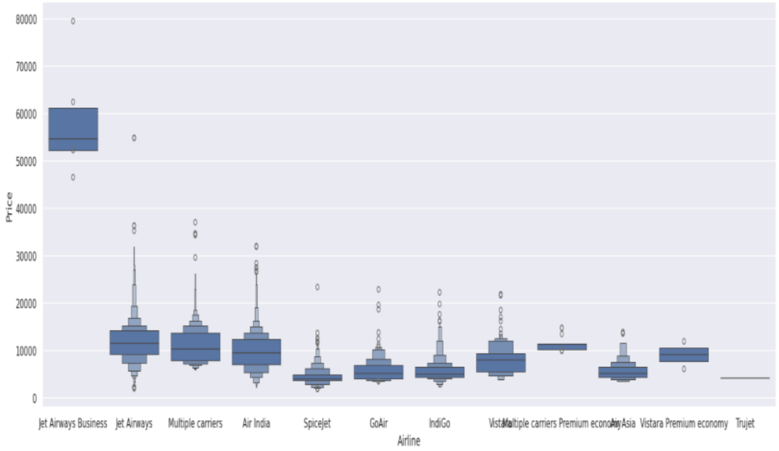


Fig 9.1 Cat plot of Airline vs Price

Fig 9.1 box plot represents the variation in flight ticket prices for different airlines, showing the distribution, median, and outliers. It indicates that Jet Airways Business has the highest price range, while airlines like Trujet have lower fares.

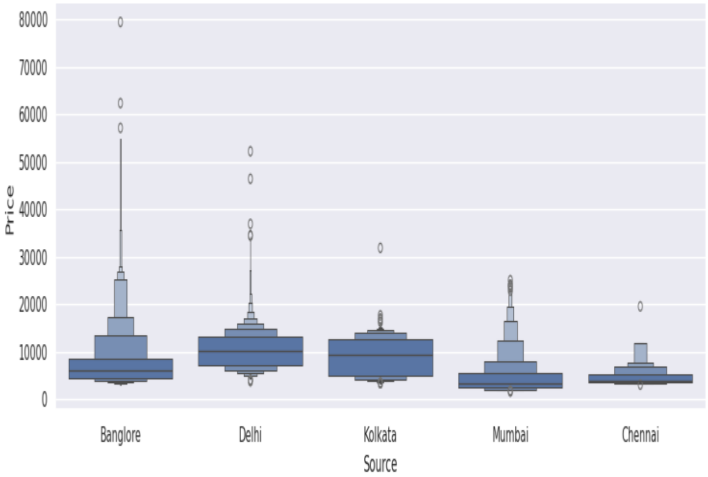


Fig 9.2 Cat plot of Source vs Price

This box plot visualizes flight ticket prices from different source cities, highlighting price distribution and outliers. Bangalore and Delhi show a wider price range with higher fare variability, whereas Chennai has the lowest price spread.

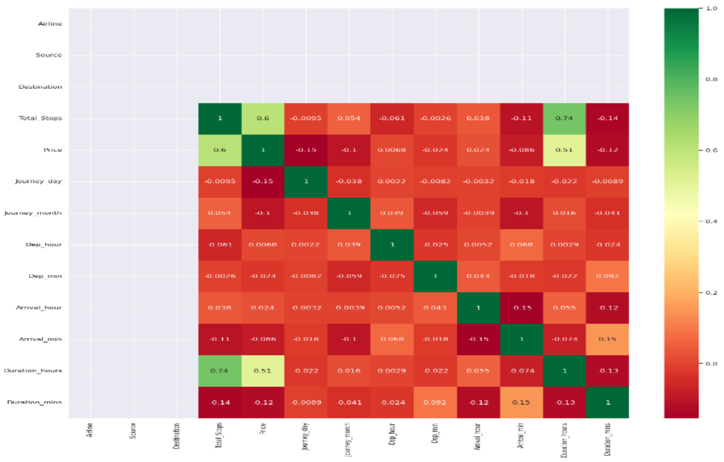


Fig 9.3 Correlation between Independent and Dependent Attributes

This heatmap represents the correlation between different flight-related features, where green indicates a strong positive correlation and red signifies a negative or weak correlation. The price of flights shows a moderate correlation with total stops and duration, while other features have weaker relationships.

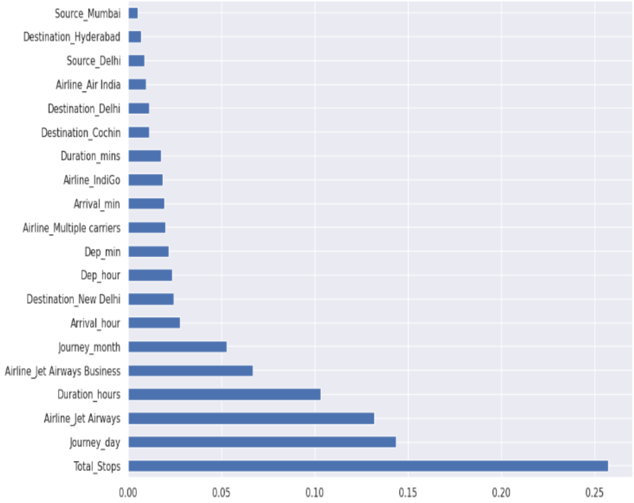


Fig 9.4 Plot graph of feature importances

This feature importance plot shows the impact of various factors on flight price prediction, with Total Stops being the most influential feature. Other significant factors include Journey Day, Airline Type, and Duration Hours, while source and destination cities have minimal impact.

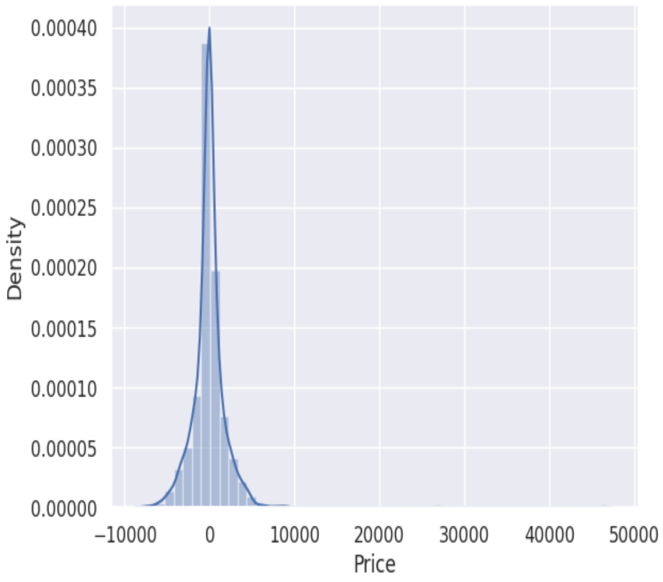


Fig 9.5 Distplot of Density and Price

This kernel density estimate (KDE) plot shows the distribution of flight prices, indicating a right-skewed distribution with most prices concentrated around a lower range. There are few instances of higher flight prices, suggesting outliers or premium fare categories.

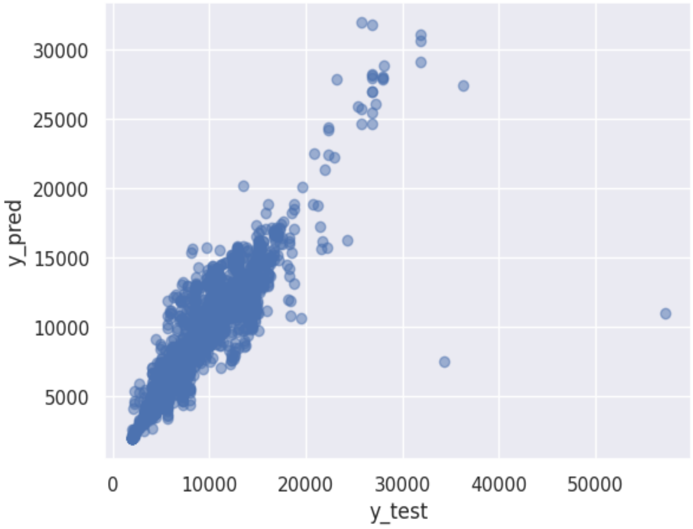


Fig 9.6 Scatter plot of y_pred vs y_test

This Fig 9.6 scatter plot compares actual flight prices (y_test) with predicted prices (y_pred), showing a strong positive correlation between them. The points align closely with a diagonal trend, indicating that the model performs well, though some deviations suggest minor prediction errors.

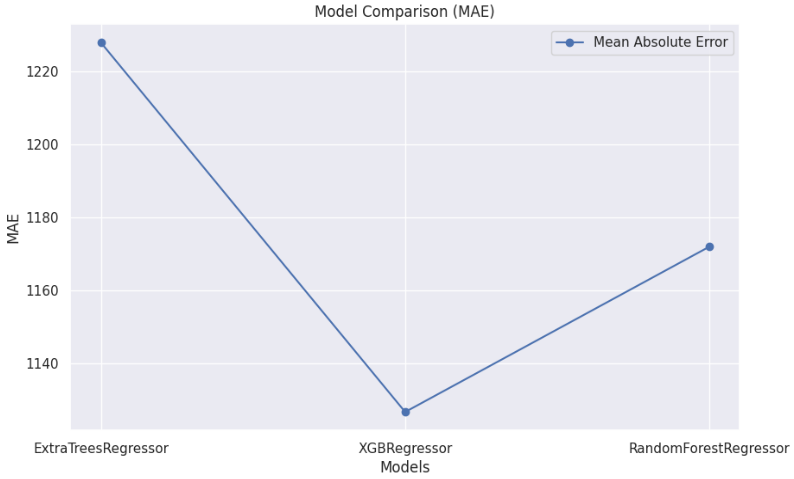


Fig 9.7 Line graph of Three models MAE

This Fig 9.7 line plot compares the Mean Absolute Error (MAE) of three regression models: ExtraTreesRegressor, XGBRegressor, and RandomForestRegressor. The XGBRegressor has the lowest MAE, indicating it provides the most accurate predictions among the three models.

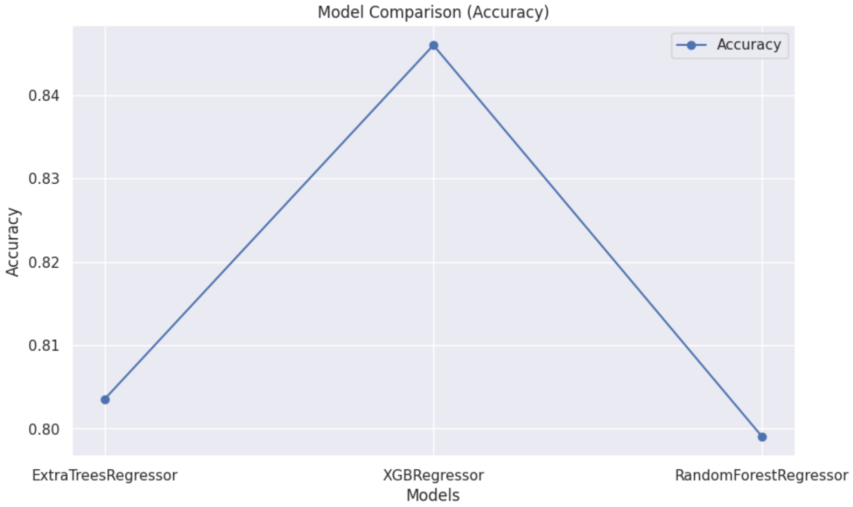
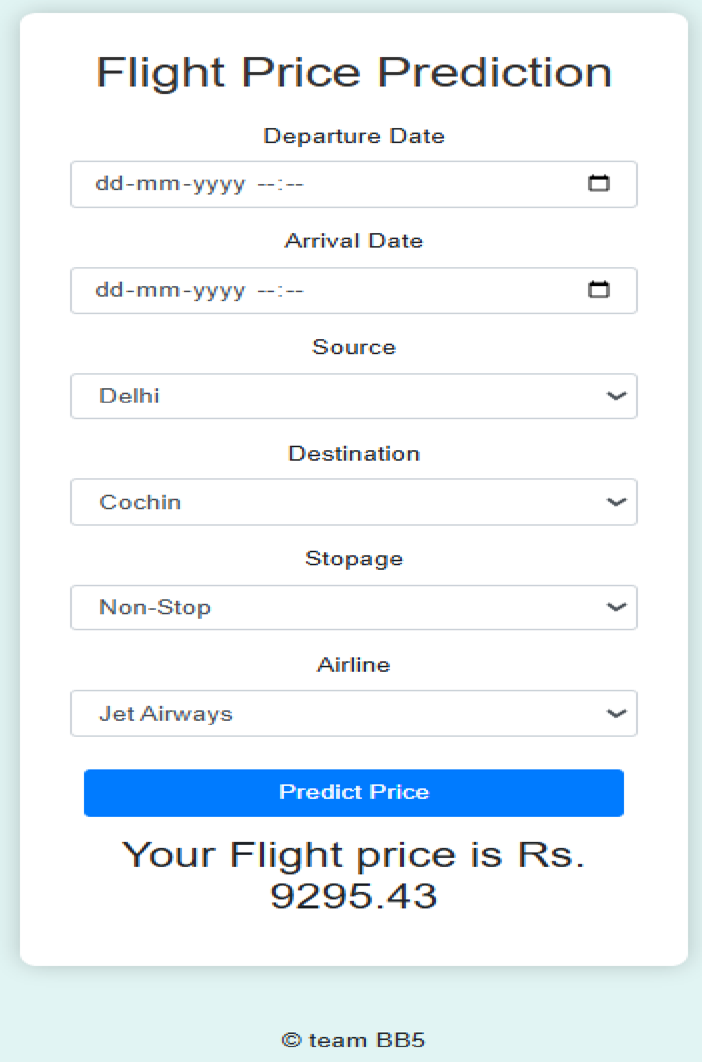


Fig 9.8 Line graph ofthree models Accuracy

This Fig 9.8 line plot compares the accuracy of three regression models:ExtraTreesRegressor, XGBRegressor, and RandomForestRegressor. The XGBRegressor has the highest accuracy, making it the best-performing model among the three.

10. Test Case

This Flight Price Prediction web interface allows users to input travel details like departure date, source, destination, stoppage, and airline to predict the flight fare. Based on the given inputs, the predicted flight price is Rs. 9295.43.



Flight Price Prediction

Departure Date

dd-mm-yyyy --:--

Arrival Date

dd-mm-yyyy --:--

Source

Delhi

Destination

Cochin

Stopage

Non-Stop

Airline

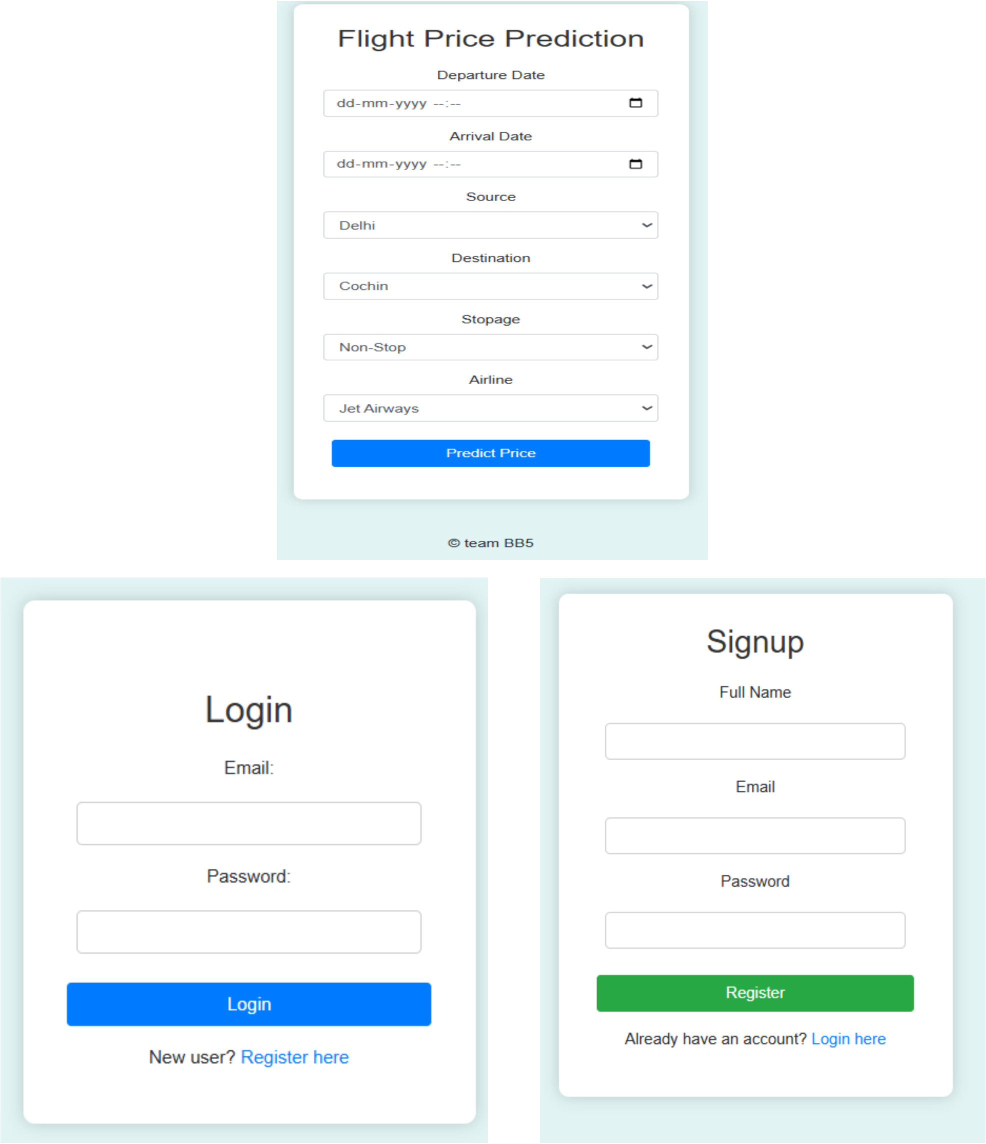
Jet Airways

Your Flight price is Rs. 9295.43

© team BB5

Fig 10.1 test case with output

11. User Interface



Flight Price Prediction

Departure Date: dd-mm-yyyy --:--

Arrival Date: dd-mm-yyyy --:--

Source: Delhi

Destination: Cochin

Stopage: Non-Stop

Airline: Jet Airways

Login

Email:

Password:

Signup

Full Name

Email

Password

Fig 11.1 user interfaces

The images show the user interface of a flight price prediction system, including login, signup, and price prediction pages. The design features a clean and simple layout with dropdown selections for travel details and authentication fields for user access.

12. CONCLUSION

The Flight Ticket Price Prediction System successfully utilizes machine learning models to estimate airfare prices with high accuracy. By comparing different models such as ExtraTreesRegressor, XGBRegressor, and RandomForestRegressor, it was observed that XGBRegressor provided the best results with the lowest Mean Absolute Error (MAE) of 1126.70 and the highest accuracy of 84.59%. This confirms that machine learning techniques can effectively capture complex relationships between various factors such as airline, departure time, duration, and ticket class to predict prices more precisely.

The implementation of this system using Python Flask ensures easy deployment as a web-based application, making it accessible for users to check predicted prices in real-time. The use of data preprocessing, feature selection, and model optimization has contributed to improving the overall performance of the system. With further enhancements and integration of more real-time data sources, this model can be refined for even better accuracy. The project demonstrates the potential of machine learning in transforming the travel industry by helping passengers make more informed booking decisions.

13. FUTURE SCOPE

The Flight Ticket Price Prediction System has significant potential for future improvements and expansions. One major enhancement could be the integration of real-time flight data from airline APIs, which would allow for more accurate and up-to-date predictions. Additionally, incorporating deep learning techniques such as neural networks could further improve prediction accuracy by capturing even more complex patterns in ticket pricing.

Another possible extension is the inclusion of external factors like demand trends, special events, fuel prices, and weather conditions, which can impact flight costs. Furthermore, the system could be developed into a mobile application for easier accessibility, allowing users to check and compare ticket prices on the go. Implementing automated price alerts and recommendation systems based on user preferences could further enhance the user experience. With continuous advancements in AI and data analytics, this project can evolve into a comprehensive travel assistant for smarter and more cost-effective flight bookings.

14.REFERENCES

[1]. G. V. Saatwik Kumar and K. Jaisharma, "Improve the Accuracy for Flight Ticket Prediction using XGBRegressor Optimizer in Comparison with Extra TreeRegressor Performance," 2023 6th International Conference on Contemporary Computing and Informatics (IC3I), Gautam Buddha Nagar,India, 2023, pp. 2558-2562,doi: 10.1109/IC3I59117.2023.10397633.

[2] G.Deng, M.Xie, C.Feng, T.Liu and X. Zha, "Flight test data processing and analysis platform based on new generation information technology Design and Application,"2022 International Conference on Sensing, Measurement Data Analytics in the era of Artificial Intelligence(ICSMD),Harbin,China,2022,pp.1-5, doi:10.1109/ICSMD57530.2022.10058336.

[3] ”Dataset use in this paper”

https://www.kaggle.com/nikhilmittal/flightfareprediction-mh

[4] N.S.S.V.S.Rao and S.J.J.Thangaraj, "Flight Ticket Prediction using Random Forest Regressor Compared with Decision Tree Regressor," 2023 Eighth International Conference on Science Technology Engineering and Mathematics (ICONSTEM) ,Chennai, India, 2023, pp. 1-5, doi: 10.1109/ICONSTEM56934.2023.10142260.

[5] N.S.S.V.S.Rao and S.J.J.Thangaraj, "Flight Ticket Prediction using Random Forest Regressor Compared with Decision Tree Regressor," 2023 Eighth International Conference on Science Technology Engineering and Mathematics (ICONSTEM) ,Chennai, India, 2023, pp. 1-5, doi: 10.1109/ICONSTEM56934.2023.10142260.

[6] C.Cao and X.Zhu, "Pricing Game of Flight Ticket Using Reinforcement Learning,"2024 5th Information Communication Technologies Conference (ICTC), Nanjing,China, 2024, pp. 367-371, doi: 10.1109/ICTC61510.2024.10601681.

[7] Sireesha Moturi , Srikanth Vemuru, S. N. Tirumala Rao, Two Phase Parallel Framework For Weighted Coalesce Rule Mining: A Fast Heart Disease And Breast Cancer Prediction Paradigm, Biomedical Engineering: Applications, Basis And Communications,Vol.34,No.03(2022), https://doi.org/10.4015/S1016237222500107

[8] G. R. Trivedi, J. V. Bolla and M. Sireesha, "A Bitcoin Transaction Network using Cache based Pattern Matching Rules," 2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2023, pp. 676-680, doi: 10.1109/ICSSIT55814.2023.10061064.

[9] Z. Dong, F. Li, H. Sun, J. Qian and Y. Wang, "Evaluation for Trainee Pilot Workload Management Competency During Approach Phase Based on Flight Training Data," 2022 2nd International Conference on Big Data Engineering and Education (BDEE),Chengdu, China, 2022, pp. 26-30, doi: 10.1109/BDEE55929.2022.00011.

[10] A. Mojtabavi et al., "Segmentation of GBM in MRI images using an efficient speed function based on level set method," 2017 10th International Congress on Image and

Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Shanghai, China, 2017, pp. 1-6, doi: 10.1109/CISP-BMEI.2017.8301983.

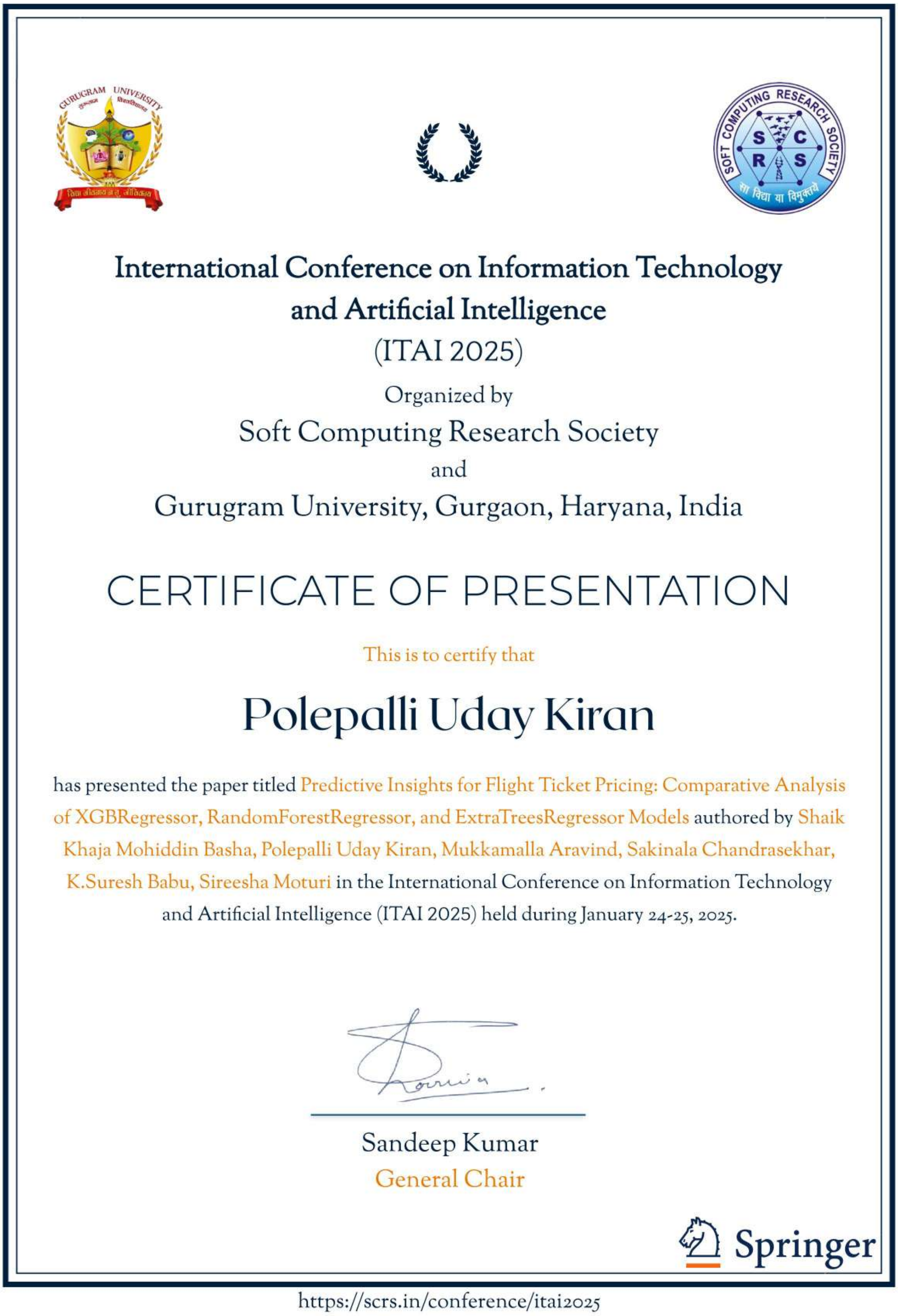
[11] M. P. Ranjit, G. Ganapathy, K. Sridhar and V. Arumugham, "Efficient Deep Learning Hyperparameter Tuning Using Cloud Infrastructure: Intelligent Distributed Hyperparameter Tuning with Bayesian Optimization in the Cloud," 2019 IEEE 12th International Conference on Cloud Computing (CLOUD),Milan,Italy,2019,pp.520-522, doi: 10.1109/CLOUD.2019.00097.

[12] J. Yuan, X. Ke, C. Zhang, Q. Zhang, C. Jiang and W. Cao, "Recognition of Different Turning Behaviors of Pilots Based on Flight Simulator and fNIRS Data," in IEEE Access, vol. 12, pp. 32881-32893, 2024, doi: 10.1109/ACCESS.2024.3367447.

[13] C.H.Lew,K.M.Lim, C. P. Lee and J. Y. Lim, "Human Activity Classification Using Recurrence Plot and Residual Network," 2023 IEEE 11th Conference on Systems, Process Control (ICSPC) , Malacca , Malaysia , 2023 , pp.78-83 , doi:10.1109/ICSPC59664.2023.10420336.

[14] J. J. Remus and L. M. Collins, "Identifying Impaired Cochlear Implant Channels via Speech-Token Confusion Matrix Analysis," 2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP ’07,Honolulu,HI,USA,2007,pp.IV-741-IV-744, doi:10.1109/ICASSP.2007.367019.

[15] J. Yuan, X. Ke, C. Zhang, Q. Zhang, C. Jiang and W. Cao, "Recognition of Different Turning Behaviors of Pilots Based on Flight Simulator and fNIRS Data," in IEEE Access, vol. 12, pp. 32881-32893, 2024, doi: 10.1109/ACCESS.2024.3367447.



International Conference on Information Technology and Artificial Intelligence

(ITAI 2025)

Organized by

Soft Computing Research Society

and

Gurugram University, Gurgaon, Haryana, India

CERTIFICATE OF PRESENTATION

This is to certify that

Polepalli Uday Kiran

has presented the paper titled Predictive Insights for Flight Ticket Pricing: Comparative Analysis of XGBRegressor, RandomForestRegressor, and ExtraTreesRegressor Models authored by Shaik Khaja Mohiddin Basha, Polepalli Uday Kiran, Mukkamalla Aravind, Sakinala Chandrasekhar, K.Suresh Babu, Sireesha Moturi in the International Conference on Information Technology and Artificial Intelligence (ITAI 2025) held during January 24-25, 2025.

SCRS/ITAI2025/PC/7

Sandeep Kumar
General Chair

Springer

https://scrs.in/conference/itai2025

Predictive Insights for Flight Ticket Pricing: Comparative Analysis of XGBRegressor, RandomForestRegressor, and ExtraTreesRegressor Models

Shaik Khaja Mohiddin Basha[1], Polepalli Uday Kiran[2], Mukkamalla Aravind[2], Sakinala Chandrasekhar[2], Dr.K.Suresh Babu[3], and Dr.Sireesha Moturi[3]

[1] Assistant Professor, Dept of Computer Science and Engineering
Narasaraopeta Engineering College(Autonomous),
Narasaraopet-522601, Palnadu District, Andhra Pradesh,India.
sk.basha579@gmail.com,

[2] Dept of CSE, Narasaraopeta Engineering College, Narasaraopet-522601, Palnadu, Andhra Pradesh, India. udaymajunu1@gmail.com, aravindmukkamalla187@gmail.com, sakinalachandrasekhar@gmail.com

[3] Assistant Professor, Dept of Computer Science and Engineering
Narasaraopeta Engineering College(Autonomous),
Narasaraopet-522601, Palnadu District, Andhra Pradesh,India.
Sureshkunda546@gmail.com, sireeshamoturi@gmail.com

Abstract. The paper is proposing a Novel XGBRegressor Optimizer to address the flight ticket price prediction shortcomings by comparing ExtraTreesRegressor with it. Still, the usefulness of both models lies in enhancing the system performance as regards ticket price prediction. This Novel XGBRegressor Optimizer is another class that optimizes model parameters of the XGBRegressor by using gradient boosting. However, ExtraTreesRegressor is a great extension of random forests in the reduction of over-prediction variation using extremely random trees. In total, 40 sample sets have been used in this study in order to examine the under-study models. Using the ClinCalc software, this setup was checked for correctness, performing supervised learning2 with = 0.05, g-power = 0.8, taking 95% as the confidence internal Ci.Because of the experiment and assessment of the risk for over-learning, Novel XGBRegressor Optimizer was able to reveal performance of 82.7%, while ExtraTreesRegressor achieved 78.2%. Individual scores used for independent samples test levels, which for this level had a significance value of p=0.000. The study does present the Novel XGBRegressor Optimizer as efficient in improving the prediction of flight travel ticket prices when compared with the ExtraTreesRegressor.

Keywords: ExtraTreesRegressor, Transport,Machine Learning, Novel XGBRegressor Optimizer,Flight Ticket prediction, Regression..

1 Introduction

The airline industry operates in a highly dynamic environment where flight ticket prices fluctuate significantly based on a multitude of factors. These include the time of booking, airline, number of stops, departure and arrival times, route, demand, and even external factors like weather conditions and political events. For both consumers and airlines, predicting flight ticket prices accurately can provide substantial advantages. Consumers can make informed decisions on when to book tickets, and airlines can optimize their pricing strategies to maximize revenue while maintaining competitiveness[1].The Flight Price Prediction Project aims to build a machine learning model capable of predicting flight ticket prices using historical data. This project involves analyzing flight data, creating relevant features, and training various machine learning models to estimate ticket prices. By leveraging advanced algorithms and thorough data preprocessing, we aim to develop an efficient predictive model that can give a close approximation of flight prices based on specific inputs. Flight pricing models are notoriously complex due to the number of influencing factors involved. Traditional methods often struggle to capture the intricate relationships between variables that impact flight prices. Thus, the Empowering travelers with precise price predictions, transforming their journey with confidence and clarity by utilizing machine learning, which can automatically learn from data and uncover hidden patterns. The task is framed as regression problem where the targeted variable is the continuous cost of a flight, and the input features include airline details, route information, the number of stops, duration, departure and arrival times, and other related factors[2].

2 Data Collection

Data can be collected from datasets available on Kaggle.com. Flight Fare Prediction MH dataset in kaggle websit[3].The dataset consists of various features that influence flight ticket prices, including:

- Airline: The airline company providing the flight.
- Source: The starting location of the flight.
- Destination: The endpoint of the flight.
- Date of Journey: The date on which the journey takes place.
- Total Stops: The number of layovers before reaching the final destination.
- Route: The path the flight takes, including stops.
- Duration: The total time taken by the flight from departure to arrival.
- Additional Information: Miscellaneous information like "No Info", "In-flight meal not included", etc.

The dataset is divided into training data (to build the model) and test data (to evaluate the model's performance on unseen data).

3 Literature Survey

This study compares Random Forest Regressor and Decision Tree Regressor for predicting flight ticket fares, finding Random Forest Regressor to be more accurate 86.70 vs 79.69. The results suggest Random Forest Regressor is a reliable model for flight price prediction[4].This study compares GradientBoosting Regression and AdaBoostRegressor for predicting flight prices, finding GradientBoosting Regression to be more accurate 82.5 vs 48.7. The results suggest GradientBoosting Regression is a more effective model for flight price prediction[5].This study proposes a novel game approach using reinforcement learning to optimize flight ticket pricing, integrating multiple factors like market demand, supply, and passenger preferences[6].This study proposes a novel disease prediction model using a 2-phase parallel processing based Coalesce based Binary (CBB) Table, integrating optimal feature extraction and hybrid classification[7].This study proposes a novel approach for detecting fraudulent Bitcoin transactions using Pattern Matching Rules (PMR) and a Petri-Net model, improving transaction processing time and accuracy[8].

4 Data Preprocessing

Data preprocessing of dataset is a critical step in any machine learning project. It involves transforming raw data(structured) into a simple, usable format before feeding it into machine learning models. This step ensures that the data is well-structured and free of issues that can degrade the performance of the model, such as missing values, inconsistent data types, and irrelevant features. The following sections break down the detailed process of data preprocessing for the Flight Price Prediction Project.

4.1 Handling Missing Values

Missing values are common in datasets and must be dealt with properly to ensure they don't skew the results of the model. In this project, missing values were handled for both **categorical** and **numerical** columns.

- **Categorical Features**: For categorical features (e.g., Airline, Source, Destination, etc.), missing values are filled using the **mode** (most frequent category) of the respective column. The mode is used because categorical data does not have a numerical meaning, and the most frequent category provides a reasonable assumption for missing values.
- **Numerical Features:** Missing numerical values like Duration and Price are replaced by the mean of the respective column. This is a very simple and efficient way to fill in missing values, especially when the data is not skewed.

4.2 Transforming Categorical Variables into Insights

Generally, machine learning models working with the numerical data; therefore, these categorical features need to be converted into a numerical format. This is done using two techniques: Label Encoding: Label encoding is used in ordinal categorical features-that is, features whose categories have an inherent order. An example could be Total Stops, which can be considered ordinal because a number of stops naturally goes through some sort of logical progression. For example, "Non-stop" < "1 stop" < "2 stops", etc. Label encoding equals to each category a unique integer number. Suppose there are five airlines; it means five new columns are added where each column, based on the airline in a particular record, will be 1 if true and 0 otherwise. The drop first=True argument helps to avoid the dummy variable trap, which occurs when one category can be predicted from the others. By dropping the first category, the model is forced to infer the missing category from the others.

4.3 Feature Engineering of Time and Date

Time-related features, such as date of journey, departure and arrival times, and flight duration, are crucial predictors of flight prices. Transforming the "Date of Journey" column into separate day and month columns enables the model to learn price variations based on time of year and day of month.

4.4 Computation of Duration

The "Duration" column is standardized by converting flight durations from "Xh Ym" format to total minutes, ensuring consistency and comparability among records. This transformation enables the model to accurately analyze the relationship between flight duration and prices.

4.5 Standardization

Standardization of numerical features, such as duration and price, is performed using StandardScaler to ensure features are on the same scale and prevent larger-range features from dominating model predictions. Feature engineering and scaling are crucial steps in preparing data for machine learning algorithms, enabling models to learn from rich and informative data and make accurate price predictions. **Types of Feature Scaling Techniques**

Different models require different scaling techniques. There are mainly two scaling methods: Normalization and Standardization, each applied to different types of machine learning models.

Formula for Normalization:

$\text{Xscaled} = \text{X} - \text{Xmin} \, / \, \text{Xmax} \;\; \text{Xmin}$

are the minimum and maximum values of the feature. When to Use: When the features have varying ranges and you want to bring all features into the same range (e.g., [0, 1] or [-1, 1]).

When using algorithms that calculate distances (e.g., KNN, SVMs and neural networks).

Application in Flight Price Prediction:

Duration_Minutes, Journey_Day, Dep_Hour, and Arrival_Hour: These are features for continuous variables. Normalization scales the values of these features to be within a comparable range, so that no single feature dominates the decisions made by the model.

Standardization (Z-score Scaling)

Standardization or Z-score normalization changes the data to have an average of 0 and a standard deviation of 1. It is very helpful in the case of a Gaussian distribution of data.

***Formula for standization*:** **When to use:** If the data is normally distributed. If models to be used are linear regression, logistic regression, and any other algorithms that make assumptions of normality in the input data.

$$z = \frac{x - \mu}{\sigma}$$

$\mu =$ Mean
$\sigma =$ Standard Deviation

When working with models that rely on distance calculation-for example, K-Means clustering, Principal Component Analysis, or algorithms involving dot products, such as SVMs. Application to Flight Price Prediction: In the flight price prediction problem, features such as Total_Stops or aggregated features like Average_Price_per_Airline would benefit from standardization, since they might not naturally fall into the same range or scale as other features.

Choosing the Right Scaling Method

Which one to choose, normalization or standardization, depends on the following: Normalize in cases where one might be working with algorithms such as k-nearest neighbors or neural networks, where the scale of features matters when calculating distances or updating weights. Standardization is much more in demand for algorithms like linear regression, logistic regression, and SVM, where normally distributed data improves model performance.

The Flight Price Prediction Project, normalizing is useful for most continuous features such as Duration, Dep_Hour, and Arrival_Hour. In contrast, standardization will do its job in case of Total_Stops and aggregated numerical feature(s). Some features are unnecessary to scale, like one-hot encoded categorical data: Airline, Source, and Destination.

Handling Categorical Features

In general, feature scaling for one-hot encoded categorical features is not necessary, as they already reside in a uniform range of binary values: 0 or 1. Features such as Airline_IndiGo and Source_Delhi are already in binary form after one-hot encoding, so no further scaling is required.

However, ordinal categorical features such as Total_Stops need to be scaled since this kind of feature has a natural order but the ranges are different after label encoding.

5 MATERIALS AND METHODS

5.1 Model Training

Model training basically involves training a machine learning model with a set of data on which the system learns and will work to make fairly accurate predictions. The task at hand in the Flight Price Prediction Project is to train the model in order to predict the price for a flight on input features such as Airline, Total_Stops, Duration, Departure Time, among others[9].

Steps in Model Training 1)Selection of an Appropriate Model/Algorithm 2)Data Splitting 3)Model Training 4)Hyperparameter Tuning 5)Model Evaluation

5.2 Model/Algorithm Selection

Choosing the appropriate algorithm is crucial, as performance depends on it. This flight price prediction problem is a form of regression problem. So, the output continuous-the flight price-usually regression algorithms are applied. Here are a few common algorithms which can be used for this task:

Linear Regression

Linear regression models the relationship between the dependent variable, flight price, and one or more independent variables, features, by fitting a linear equation.This method is straightforward, easy to interpret, and generally performs well when the data can be separated linearly.It may not perform well if the relationship between the features and the target variable is nonlinear.

Decision Trees

Decision trees are models that work by splitting the data into subsets according to feature values. Each node of the tree is a decision, and each leaf represents predicted outcomes - flight prices in our case.Non-linear relationships can be captured, and decision trees can handle numerical as well as categorical data without scaling. They are prone to overfitting if their depth gets too big.

Random Forests

How It Works: A random forest is an ensemble method that constructs many decision trees, then averages their predictions, such that it enhances the predictive performance of your data and reduces overfitting.This model is robust; it reduces overfitting compared to one decision tree, and it can handle high-dimensionality features.It is computationally expensive and not as intuitive compared to other simpler models.

5.3 Gradient Boosting Machines (GBM)

How It Works: GBM creates an ensemble of decision trees in such a manner that each tree tries to correct the previously corrected tree to result in a good classifier. It pays greater attention to reducing the mistakes in the prediction[10]. Highly accurate, prevents overfitting with Train-Test Split (80:20) and Cross-Validation. Train Data: 66.66Ensures consistent performance across different data subsets.

6 Training the models

Now, with the data split, it can be used to train the model. During the phase train, the model get the relationships of the features to the variable target of flight price.

Fit the Model

We call fit() on each model in order to train it with the training data.

Hyperparameter Tuning

Hyperparameters are settings defined before the training process that influence how a model learns. Most models require careful tuning of these hyperparameters to significantly improve their performance. **Grid Search** Grid search is an exhaustive hyperparameter tuning method. This tries all combinations of hyperparameters for the selection of the best-performing combination based on cross-validation. **Random Search** Another approach in place of grid search is randomized search, wherein a random subset of the hyperparameters is chosen to be assessed. This can also become more efficient than grid search when the number of hyperparameters is large[11].

7 Performance Evaluation

7.1 Model Evaluation

Evaluation is a method of checking the empowers of a previously trained machine learning model on hidden data. It ensures that the model keep public well and isn't overfitting to the data training. For instance, the Flight Price Prediction Project should have an aim like testing the exactness of the model in predicting the prices of flights with different metrics[12]. Below are steps involved in the process:

Evaluation Steps

1.Evaluation Metric Selection
2.Prediction on Test Data
3.Overfitting and Underfitting Detection
4.Performance Visualization
5.Cross-Validation
6.Saving Best Model

7.2 Choosing Metrics to Evaluate

Clearly, choosing the appropriate metrics for this problem would allow us to understand how well our model performed on the price of flight predictor. This is a regression problem; hence, the output is continuous, flight price, and common regression metrics are:

7.3 Equations

Mean Absolute Error (MAE): measures the average difference between the predicted and actual prices of flights. It provides an estimate of how far off the predictions are from the true values on average.yi = actual flight price , ŷi = predicted flight price ,n = number of observations

$$z = \frac{x - \mu}{\sigma}$$

$$\mu = \text{Mean}$$
$$\sigma = \text{Standard Deviation}$$

Fig. 1. Mean Absolute Error

Mean Squared Error (MSE): MSE (Mean Squared Error) calculates the average squared difference between predicted and actual flight prices, giving more weight to larger errors compared to smaller ones.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Fig. 2. Mean Squared Error

Root Mean Squared Error (RMSE): RMSE is the square root of MSE, giving it the same units as the target variable (flight price). It's often used when you want to directly interpret the error in terms of the variable being predicted.

$$\text{RMSD} = \sqrt{\frac{\sum_{i=1}^{N}\left(x_i - \hat{x}_i\right)^2}{N}}$$

Fig. 3. Root Mean Squared Error

R squared ($\mathbf{R^2}$): R^2 represents the proportion of the total variance in the target variable—such as the price of a flight—that is explained by the input features used in the model.

$$R^2 = 1 - \frac{RSS}{TSS}$$

Fig. 4. R^2

8 Figures and Tables

	Duration	Price
1	0.787837	1.125548
2	0.256929	0.309048
3	1.079353	1.039858
4	0.488598	0.622202
5	0.565821	0.914076

Table 1. After Feature Scaling

Group	Mean	R^2	Accuracy
Random Forest Regressor	596.624	0.9147	91.47%
ExtraTreesRegressor	565.681	0.91169	91.17%
XGBRegressor	1014.7093	0.9116	87.94%

Table 2. Comparing models

Performance Visualization

Visualizing the model's performance helps to intuitively understand how well the model predicts flight prices. Some common visualizations include:

Actual vs. Predicted Plot

This plot compares the actual flight prices to the predicted ones. Ideally, the points should lie close to the line y = x if the predictions are accurate.

Residuals Plot

The residuals constitute the differences between the actual and estimated prices. This residual plot supports the identification of any kind of pattern in these errors. Ideally, the residuals must be randomly distributed around zero[13].

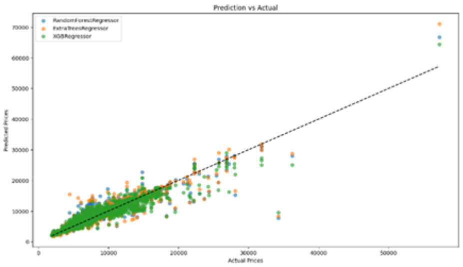


Fig. 5. Actual vs. Predicted Plot

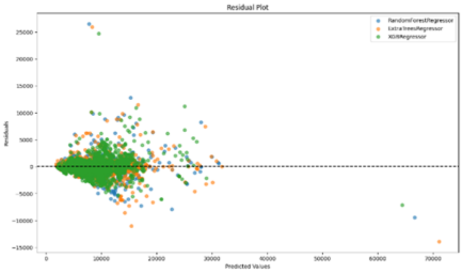


Fig. 6. Residuals Plot

Saving the Best Model

After evaluating multiple models and hyperparameters, the best-performing model should be saved for future use. This can be accomplished using libraries such as ‘joblib‘ or ‘pickle‘.

Confusion Matrix: An Overview

A confusion matrix is a common performance evaluation tool in the tasks of classification for gaining an understanding of how well the models are performing; it is shown as a matrix of actual versus predicted classifications. It has been especially useful in the realms of binary and multi-class classification problems. One can also ascertain with it the degree of precision of a model and its mistakes over different classes.

The confusion matrix is one of the most important elements in performance evaluation, especially with respect to imbalanced datasets. It helps you to understand the trade-offs you are making between different types of errors-false positives and false negatives-and inform better tuning of your model, focusing on metrics that actually reflect real-world performance[14].

Model Comparison, $\mathbf{R^2}$ Score R^2, or the coefficient of represented, is a common metric used to evaluate the performance of a regression model. It indicates the proportion of the mean variance in the depending variables (marked) that can be predicted from the independent variables (features). A higher R^2 number signifies a better fit to the data, with one representing a perfect match

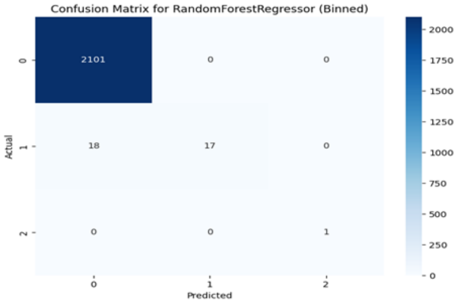


Fig. 7. Confusion Matrix

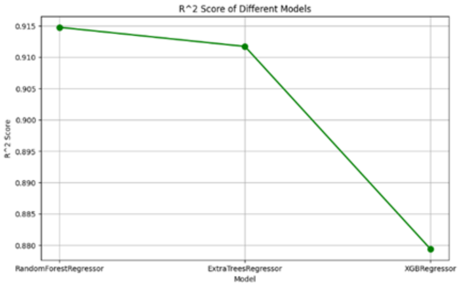


Fig. 8. plot by comparing with R^2

and 0 indicating no explanatory power[15]. Here is a comparison of models based on their R^2 scores.

R^2 Score: 0.90 (Highest)

Analysis: The best performance was turned in by XGBRegressor, with the highest R^2 score, explaining 90% of the variance in target variable. This means that the model had an excellent grasp of the general trend in the data. Probably, its boosting mechanism gave more power to XGB to learn from the mistakes of the model and make correct predictions, hence it is the fittest model among the others for this dataset.

References

1. G. V. Saatwik Kumar and K. Jaisharma, "Improve the Accuracy for Flight Ticket Prediction using XGBRegressor Optimizer in Comparison with Extra TreeRegressor Performance," 2023 6th International Conference on Contemporary Computing and Informatics (IC3I), Gautam Buddha Nagar, India, 2023, pp. 2558-2562, doi: 10.1109/IC3I59117.2023.10397633.
2. G.Deng, M.Xie, C.Feng, T.Liu and X. Zha, "Flight test data processing and analysis platform based on new generation information technology Design and Application," 2022 International Conference on Sensing, Measurement Data Analytics in the era

of Artificial Intelligence (ICSMD), Harbin, China, 2022, pp. 1-5, doi: 10.1109/ICSMD57530.2022.10058336.

3. ” Dataset use in this paper” https://www.kaggle.com/nikhilmittal/flight-fareprediction-mh
4. N.S.S.V.S.Rao and S.J.J.Thangaraj, "Flight Ticket Prediction using Random Forest Regressor Compared with Decision Tree Regressor," 2023 Eighth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), Chennai, India, 2023, pp. 1-5, doi: 10.1109/ICONSTEM56934.2023.10142260.
5. N. S. S. V. S. Rao, S. J. J. Thangaraj and V. S. Kumari, "Flight Ticket Prediction using Gradient Boosting Regressor Compared with AdaBoost Regressor," 2023 Eighth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), Chennai, India, 2023, pp. 1-5, doi: 10.1109/ICONSTEM56934.2023.10142536.
6. C.Cao and X.Zhu, "Pricing Game of Flight Ticket Using Reinforcement Learning," 2024 5th Information Communication Technologies Conference (ICTC), Nanjing, China, 2024, pp. 367-371, doi: 10.1109/ICTC61510.2024.10601681.
7. Sireesha Moturi , Srikanth Vemuru, S. N. Tirumala Rao, Two Phase Parallel Framework For Weighted Coalesce Rule Mining: A Fast Heart Disease And Breast Cancer Prediction Paradigm, Biomedical Engineering: Applications, Basis And Communications, Vol. 34, No. 03 (2022), https://doi.org/10.4015/S1016237222500107
8. G. R. Trivedi, J. V. Bolla and M. Sireesha, "A Bitcoin Transaction Network using Cache based Pattern Matching Rules," 2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2023, pp. 676-680, doi: 10.1109/ICSSIT55814.2023.10061064.
9. Z. Dong, F. Li, H. Sun, J. Qian and Y. Wang, "Evaluation for Trainee Pilot Workload Management Competency During Approach Phase Based on Flight Training Data," 2022 2nd International Conference on Big Data Engineering and Education (BDEE), Chengdu, China, 2022, pp. 26-30, doi: 10.1109/BDEE55929.2022.00011.
10. A. Mojtabavi et al., "Segmentation of GBM in MRI images using an efficient speed function based on level set method," 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Shanghai, China, 2017, pp. 1-6, doi: 10.1109/CISP-BMEI.2017.8301983.
11. M. P. Ranjit, G. Ganapathy, K. Sridhar and V. Arumugham, "Efficient Deep Learning Hyperparameter Tuning Using Cloud Infrastructure: Intelligent Distributed Hyperparameter Tuning with Bayesian Optimization in the Cloud," 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), Milan, Italy, 2019, pp. 520-522, doi: 10.1109/CLOUD.2019.00097.
12. J. Yuan, X. Ke, C. Zhang, Q. Zhang, C. Jiang and W. Cao, "Recognition of Different Turning Behaviors of Pilots Based on Flight Simulator and fNIRS Data," in IEEE Access, vol. 12, pp. 32881-32893, 2024, doi: 10.1109/ACCESS.2024.3367447.
13. C.H.Lew,K.M.Lim, C. P. Lee and J. Y. Lim, "Human Activity Classification Using Recurrence Plot and Residual Network," 2023 IEEE 11th Conference on Systems, Process Control (ICSPC), Malacca, Malaysia, 2023, pp. 78-83, doi:10.1109/ICSPC59664.2023.10420336.
14. J. J. Remus and L. M. Collins, "Identifying Impaired Cochlear Implant Channels via Speech-Token Confusion Matrix Analysis," 2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP ’07, Honolulu, HI, USA, 2007, pp. IV-741-IV-744, doi: 10.1109/ICASSP.2007.367019.
15. J. Yuan, X. Ke, C. Zhang, Q. Zhang, C. Jiang and W. Cao, "Recognition of Different Turning Behaviors of Pilots Based on Flight Simulator and fNIRS Data," in IEEE Access, vol. 12, pp. 32881-32893, 2024, doi: 10.1109/ACCESS.2024.3367447.

Predictive Insights for Flight Ticket Pricing.pdf

ORIGINALITY REPORT

13%	7%	7%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Bui Thanh Hung, M. Sekar, Ayhan Esi, R. Senthil Kumar. "Applications of Mathematics in Science and Technology - International Conference on Mathematical Appplications in Science and Technology", CRC Press, 2025 Publication	1%
2	Submitted to Technological University Dublin Student Paper	1%
3	Submitted to Liverpool John Moores University Student Paper	1%
4	Submitted to The Robert Gordon University Student Paper	1%
5	www.scrs.in Internet Source	1%
6	Submitted to Morgan State University Student Paper	1%
7	Submitted to Angeles University Foundation Student Paper	1%
8	medium.com Internet Source	1%
9	Gabriele Zangara, Francesco Gagliardi, Luigino Filice, Giuseppina Ambrogio. "Chapter	1%

	6 A Coupled Approach Based on Statistical Methods and Machine Learning Techniques to Improve Porthole Die Design", Springer Science and Business Media LLC, 2024 Publication	
10	N. Sri Sai Venkata Subba Rao, S. John Justin Thangaraj. "Flight Ticket Prediction using Random Forest Regressor Compared with Decision Tree Regressor", 2023 Eighth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), 2023 Publication	1%
11	Submitted to Sunway Education Group Student Paper	1%
12	www.mdpi.com Internet Source	<1%
13	Submitted to University of North Texas Student Paper	<1%
14	Jeshmol P.J., Binsu C. Kovoor. "Video Question Answering: A survey of the state-of-the-art", Journal of Visual Communication and Image Representation, 2024 Publication	<1%
15	ebin.pub Internet Source	<1%
16	www.researchgate.net Internet Source	<1%
17	Sai Kumar Mamidala, Sireesha Moturi, S. N. Tirumala Rao, Jhansi Vazram Bolla, K. V.	<1%

	Narasimha Reddy. "Chapter 14 Machine Learning Models for Chronic Renal Disease Prediction", Springer Science and Business Media LLC, 2024 Publication	
18	www.geeksforgeeks.org Internet Source	<1 %
19	"Data Science and Applications", Springer Science and Business Media LLC, 2024 Publication	<1 %
20	Xinyuan Song, Keyu Chen, Ziqian Bi, Qian Niu, Junyu Liu, Benji Peng, Sen Zhang, Ming Liu, Ming Li, Xuanhe Pan. "Deep Learning and Machine Learning: Contrastive Learning, from scratch to application", Open Science Framework, 2024 Publication	<1 %
21	cisisconference.eu Internet Source	<1 %
22	mdpi-res.com Internet Source	<1 %
23	speakerdeck.com Internet Source	<1 %
24	www.rkimball.com Internet Source	<1 %
25	www.ijeat.org Internet Source	<1 %