

HARNESSING COGNITIVE ALGORITHMS FOR FLIGHT FARE DETECTION

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

LAKSHMI PRASANNA R (Reg.No - 40111052)
RENUKA DEVI MEDAPATI (Reg.No - 40111055)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

CATEGORY -1 UNIVERSITY BY UGC

Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

APRIL – 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **LAKSHMI PRASANNA R(40111052)** and **RENUKA DEVI MEDAPATI(40111055)** who carried out the Project Phase-2 entitled **"HARNESSING COGNITIVE ALGORITHMS FOR FLIGHT FARE DETECTION"** under my supervision from November 2023 to April 2024.

Internal Guide

Dr. J. REFONAA M.E., Ph.D.

HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Chennai - 600 119.

Dr. L. LAKSHMANAN M.E., Ph.D.

Chennai - 600 119.

Submitted for Viva voce Examination held on 13-04-2024

Internal Examiner

External Examiner

DECLARATION

I, **LAKSHMI PRASANNA R (40111052)**, hereby declare that the Project Report entitled **“HARNESSING COGNITIVE ALGORITHMS FOR FLIGHT FARE DETECTION”** done by me under the guidance of **Dr. J. REFONAA, M.E., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 13-04-2024

PLACE: CHENNAI

R. Lakshmi Prasanna

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of Sathyabama Institute Of Science And Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph. D, Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.J.Refonaa, M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

In terms of convenience in Mode of Transportation, Airplanes the primary methods for long as well as short-distance travel. But flight prices are dynamic and to predict them for normal people is so hectic. As the price keeps on changing on the demand, flight prices are different during morning time and evening time of the day. It also keeps on changing by seeing the holidays as well as the special festive season. The price of flight tickets gets affected by so many factors. Most distributors have data or information on many parameters, but purchasers only have limited knowledge, which is insufficient to anticipate flight prices. we will explore a machine learning model that will take into account features such as departure location, number of stops, departure time, arrival time, the number of days till departure, and time of day to determine the optimal time to purchase tickets. The goal of this research work is to identify the elements that influence changes in travel prices and how these factors are linked to changes in values. Flight ticket prices increase or decrease every now and then depending on various factors like the timing of the flights, destination, and duration of flights. In the proposed system a predictive model will be created by applying machine learning algorithms to the collected historical data of flights. With the explosive growth of the Internet and e-commerce, air passengers nowadays can check the airfare and availability of any airline around the world easily. When satisfied with airfare, these customers can purchase their desired tickets online through official airline or agent websites. To help customers to buy the most inexpensive airfare, there have been a number of prediction models to predict airfare prices. Since the airline industry's deregulation, airfare pricing strategy has evolved into a complex structure of sophisticated rules and mathematical models that drive airfare pricing strategies. Even though these rules are still largely unknown, studies have revealed that they are influenced by a variety of factors. Traditional variables such as distance, while still important, are no longer the sole determinants of pricing strategy. Economic, marketing, and societal trends have all played a growing role in determining airfare prices. Flight Ticket prices for the same flight fluctuate dramatically

TABLE OF CONTENTS

Chapter No	TITLE		Page No.
	ABSTRACT		v
	LIST OF FIGURES		viii
	LIST OF ABBREVIATIONS		ix
1	INTRODUCTION		1
2	LITERATURE SURVEY		2
	2.1	Inferences from Literature Survey	4
	2.2	Open problems in Existing System	4
3	REQUIREMENTS ANALYSIS		6
	3.1	Feasibility Studies/Risk Analysis of the Project	6
	3.2	Software Requirements Specification Document	7
4	DESCRIPTION OF PROPOSED SYSTEM		9
	4.1	Flow Chart	9
	4.2	Selected Methodology or Historical Data	10
	4.3	Architecture / Overall Design of Proposed System	13
	4.4	Description of Software for Implementation and Testing Plan of the Proposed Model/System	14
	4.5	Project Management Plan	22

5	IMPLEMENTATION DETAILS		25
	5.1	Development and Deployment Setup	25
	5.2	Algorithms	27
	5.2.1	Random Forest	27
	5.2.2	Decision Tree	28
	5.2.3	K-Nearest Neighbors.	30
	5.3	Testing Plan	31
6	RESULTS AND DISCUSSION		33
	6.1	Data Analysis And Interpretation	33
	6.2	Future Enhancement	37
	6.3	Performance Analysis	38
7	CONCLUSION		41
	REFERENCES		43
	APPENDIX		45
	A.	SOURCE CODE	45
	B.	SCREENSHOTS	55
	C.	RESEARCH PAPER	70
	D.	CERTIFICATE	80

LIST OF FIGURES

FIGURE NO	FIGURE NAME	Page No.
4.1	Flow chart	8
4.2	overview of the model	11
4.3	System Architecture	12
6.1.1	Train Data set	33
6.1.2	Test Data set	34
6.1.3	Most Preferred Airlines	34
6.1.4	Heat Map	35
6.1.5	Price Differences	35
6.1.6	Airline company vs flight Ticket Price	36
6.1.7	Accuracy Values	36

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
KNN	K-Nearest Neighbors.
MSE	Mean Squared Error
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
UAT	User Acceptance Testing

CHAPTER 1

INTRODUCTION

The primary goal of this project is to create a machine learning model which work will predict the prices of flights based on certain circumstances like departure place, Number of stops, Arrival and departure Time, and many other Features. In this project, we have described the problems to be tackled, and using machine learning we have come near to the solution to these problems. Flight prices for every airline corporation vary dynamically and the prediction of the same by a human is not only difficult but quite in accurate. Hence, inflation in ticket price is quite a regular scenario while traveling via flights. This uncertain or unpredictable nature of ticket pricing tends to be quite a hurdle for regular travelers. The price of flights depends on many factors like Departure place, Number of stops, arrival and Departure Time, and many such Features. The final goal is to secure the optimal price of the flight at a certain time from a certain departure to a certain arrival place. Hence, inflation in the ticket price is quite a regular scenario while traveling via flights. This uncertain or unpredictable nature of ticket pricing tends to be quite a hurdle for regular travelers. The price of flights depends on many factors like Departure place, Number of stops, Arrival and Departure Time, and many such Features. The final goal is to secure the optimal price of the flight at a certain time from a certain departure place to a certain arrival place. The passenger receives a rough estimate of plane prices based on the date during detailed monitoring, allowing them to select Hence, inflation in the ticket price is quite a regular scenario while traveling via flights. This uncertain or unpredictable nature of ticket pricing tends to be quite a hurdle for regular travelers. The price of flights depends on many factors like Departure place, Number of stops, Arrival and Departure Time, and many such Features. The final goal is to secure the optimal price of the flight at a certain time from a certain departure place to a certain arrival place.

CHAPTER 2

LITERATURE SURVEY

This problem statement has been extensively studied over the past 5 years by researchers about Flight price prediction plays a crucial role in the airline industry and travel planning. Accurate predictions can benefit both airlines and consumers. This literature survey reviews research in the domain of dynamic flight price prediction using machine learning techniques

Tziridis.k.et.al [17] proposed that the research work they have used dataset consisting of 1814 data flights of the Aegean Airlines collected and used to train machine learning model They have used various algorithms such as Multilayer Perceptron (MLP), Generalized Regression Neural Network, Extreme Learning Machine (ELM), Random Forest Regression Tree. The best results came from Bagging regression tree.

William Groves.et.al[21]introduced which is able to optimize purchase timing on behalf of customers. Partial least square regression technique is used to build a model. Initially they have used various techniques for feature selection. The improved performance can be attributed to a natural resistance to collinear and irrelevant variables.

Santos Dominguez-Menchero.J.et.al[12] provided customers with the relevant information they need to decide the best time to purchase a ticket, striking a balance between the desire to save money and any time restraints the buyer may have. Their study shows how non-parametric isotonic regression techniques, as opposed to standard parametric techniques, are particularly useful.

Supriya Rajankar.et.al[13]predicted on flight fare prediction using machine learning algorithm uses small data set consisting of flights between Delhi and Bombay. Survey on flight fare prediction using machine learning algorithm uses small data set consisting of algorithms are: Support Vector Machine (SVM), Linear regression, K-Nearest Neighbor, Decision tree, Multilayer Perceptron, Gradient Boosting and Random Forest Algorithm.

Tianyi wang.et.al[18]proposed framework where two databases are combined together with macroeconomic data and machine learning algorithms such as support vector machine, XG Boost are used to model the average ticket price based on source and destination pairs. The framework achieves a high prediction accuracy 0.869 with the adjusted R squared performance metrics. They had the result of lowest error rate of 0.92 with the XG Boost Algorithm.

Janssen.T.et.al[7]predicted that they have used Naive Bayes and Stacked Prediction Model. the research a desired model is implemented using the Linear Quantile Blended Regression methodology for San Francisco–New York course where each day airfares are given by an online website. Two features such as number of days for departure and whether departure is on a weekend or weekday are considered to develop the model.

Wohlfarth.et.al[22] have used the various data mining techniques. It is the goal of this paper to consider the design of decision-making tools in the context of varying travel prices from the customer's perspective. Terms used in the research are machine techniques/ algorithms mentioned as Clustering.

Vinod Kimbhaune.et.al[20]used to determine ideal purchase time for flight ticket. There project aims to develop an application which will predict the flight prices for various flights using machine learning model. The techniques they have used are mentioned as Linear Regression, Decision Tree and random Forest. The performance metrics techniques used are MAE, MSE and RSME. The outcome for their project was not fully accurate but by adding more real time data set will give more accurate results.

Groves.W.et.al[4]Proposed the extended version of the research paper [3] exploited Partial Least Square Regression (PLSR) for building up a model. The information was gathered from major travel adventure booking sites from 22 February 2011 to 23 June 2011. Extra information was additionally gathered and are utilized to check the correlations of the exhibitions of the last model.

2.1 INFERENCES FROM LITREATURE SURVEY

Although Many studies emphasize the importance of historical flight price data analysis as a fundamental component of prediction models. Inferences can be drawn regarding the significance of historical trends in shaping dynamic pricing. Researchers may discuss how economic factors (fuel prices, inflation) and external events (natural disasters, geopolitical events) are integrated into pricing models. You can infer from the literature which data sources are commonly used, such as global distribution systems (GDS), and how data quality and reliability impact prediction accuracy Literature might reveal the development of real-time pricing models, indicating the importance of keeping pace with dynamic pricing changes as they happen.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

Open problems in existing system for dynamic flight price prediction Dynamic flight price prediction systems have come a long way, but there are still several open problems and challenges in the field. Some of these include:

- **Data Quality and Availability:** Ensuring high-quality, up-to-date, and consistent data from various sources can be a challenge. Many factors affect flight prices, and obtaining reliable data for all these variables is not always easy.
- **Feature Engineering:** Identifying and incorporating the right features that affect flight prices is an ongoing challenge. There are many variables to consider, such as weather, airline-specific factors, and global events.
- **Temporal Factors:** Predicting flight prices accurately over time, considering seasonality, holidays, and events, remains a challenge. The dynamic nature of flight prices makes it essential to adapt to changing patterns.

- **Demand-Supply Dynamics:** Predicting how demand and supply for flights change and influence prices can be complex. This can involve understanding airport capacity, route popularity, and airline strategies.
- **Machine Learning Models:** Developing robust machine learning models that can handle the complexity of flight price prediction is an ongoing area of research. Finding models that generalize well across different routes and airlines is a challenge.
- **User-Specific Predictions:** Providing personalized price predictions for users based on their historical preferences and behavior is a goal, but it raises privacy and data availability concerns.
- **Interactions with Third Parties:** Integrating with airline reservation systems and travel agencies while adhering to their policies and systems is challenging. APIs and data access issues may arise.
- **Economic and Geopolitical Events:** Predicting how global economic and geopolitical events impact flight prices requires sophisticated modeling and access to relevant data sources.
- **Explain ability:** Making flight price predictions more transparent and interpretable is important, especially for consumers who want to understand why a particular price is offered.
- **Evaluation Metrics:** Defining appropriate evaluation metrics for dynamic flight price prediction that capture user satisfaction and model performance accurately is an ongoing challenge.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

Performing a risk analysis for a dynamic flight price prediction system is crucial to identify potential challenges and mitigate them effectively. Here are some key risks to consider:

1) Data Quality and Availability:

Risk: Inaccurate or incomplete data can lead to poor predictions.

Mitigation: Implement data validation and cleansing processes and have a backup data sources and redundancy.

2) Model Accuracy:

Risk: Predictive models may not perform as expected due to changing market dynamics.

Mitigation: Regularly retrain models with updated data. Monitor model performance and adjust algorithms as needed.

3) Environmental Factors:

Risk: Weather-related disruptions affecting flight prices.

Mitigation: Integrate real-time weather data into the prediction system and account for weather patterns.

4) Legal Challenges:

Risk: Legal disputes with airlines or other stakeholders.

Mitigation: Establish clear agreements and legal terms with relevant parties and have legal counsel on standby.

5) Data Breach:

Risk: Unauthorized access to or theft of sensitive data.

Mitigation: Employ strong cybersecurity measures encryption, and regular security audits.

6) Scalability Challenges:

Risk: Inability to handle an increasing number of users and data.

Mitigation: Design the system with scalability in mind, using cloud-based solutions if necessary.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

Hardware specifications:

- Microsoft Server-enabled computers, preferably workstations
- **Ram** : Higher RAM, of about 4GB or above
- **Processor of frequency**: 1.5GHz or above

Software specifications:

- **Operating System** : Windows 8 and above
- **Languages** : Python
- **Tools used** : Jupyter Notebook

3.2.1 Python :

- Python is a scripting language that is high-level, interpreted, interactive, and object-oriented. Python is intended to be extremely readable. It commonly employs English terms rather than punctuation, and it has fewer syntactical structures than other languages.

Python Features

- Easy-to-learn - Python has a small number of keywords, a basic structure, and a well-defined syntax. This enables the pupil to swiftly learn the language.
- Easy-to-read - Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain - Python's source code is relatively simple to maintain.
- A broad standard library - The majority of Python's library is portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Interactive Mode - Python features an interactive mode that enables for interactive testing and debugging of code snippets.
- Portable - Python can operate on a broad range of hardware devices and has the same interface across them all.
- Extendable - The Python interpreter may be extended using low-level modules. These modules allow programmers to enhance or adapt their tools to make them more efficient.
- Databases - Python provides interfaces to all major commercial databases
- GUI Programming - Python can construct and port GUI programmes to numerous system calls, libraries, and windows systems, including Windows MFC, Macintosh, and Unix's X Window system.
- Scalable - Python provides a better structure and support for large programs than shell scripting.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

During the data pre-treatment stage, all databases are cleaned to remove any potentially erroneous examples, then converted and integrated depending on market group. The feature extractor extracts and generates handmade attributes that are intended to describe the segment of the market.

The goal of adaptive filtering modules is to improve accurate channels by assessing the utility of the characteristics and removing any unnecessary characteristics. Finally, we use the selected criteria to build our forecasting techniques that result in the finished product of the projected airline.

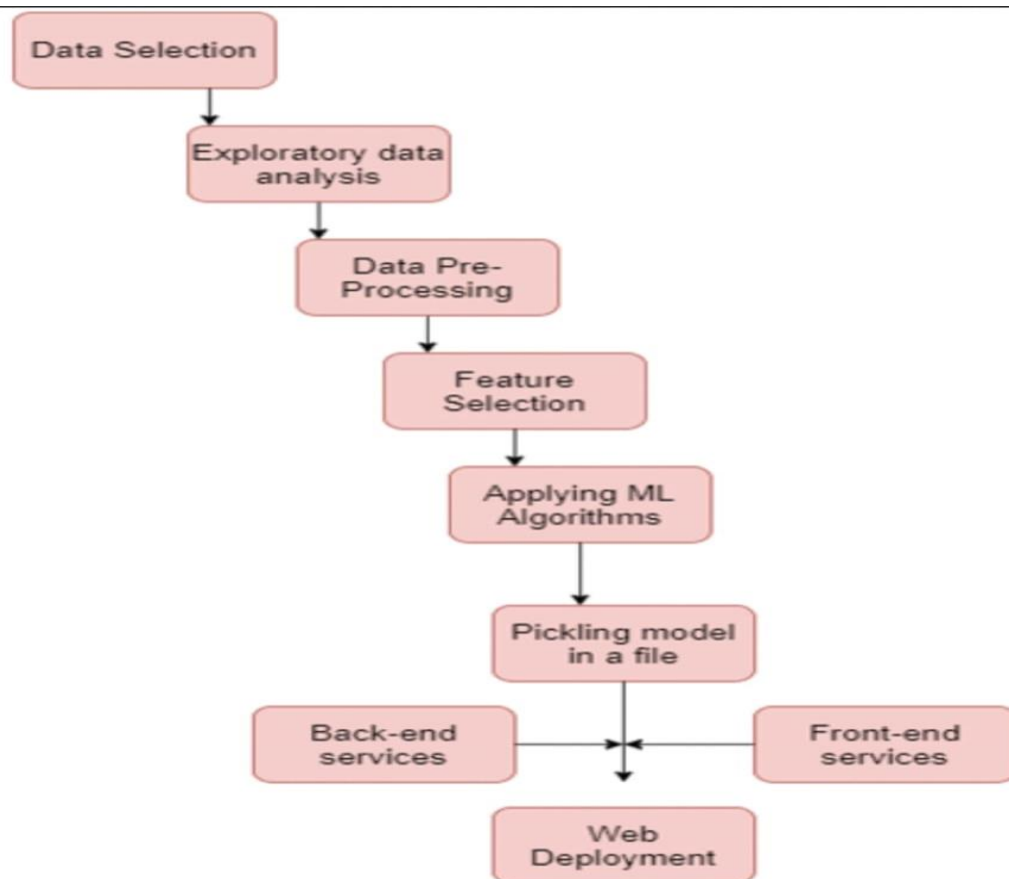


Fig 4.1 Flow chart.

4.2 SELECTED METHODOLOGY OR PROCESS MODEL

➤ ***Data Collection.***

- The python-script makes it simple to pre-process data and forecast results. In the preprocessing step, duplicate values are prevented. The focus of this Data set is on computing the plane pricing value.
- The data set contains the data with attributes such as
 - o Journey Date
 - o Departure
 - o Designation
 - o Arrival Time & Duration

➤ ***Automated Script to Collect Historical Data***

- For any prediction/classification problem, we need historical data to work with. In this project, past flight prices for each route collected on a daily basis is needed. Manually collecting data daily is not efficient and thus a python script was run on a remote server which collected prices daily at specific time.

➤ ***Cleaning & Preparing Data***

- After we have the data, we need to clean & prepare the data according to the model's requirements. In any machine learning problem, this is the step that is the most important and the most time consuming. We used various statistical techniques & logics and implemented them using built in R packages.

➤ **Analyzing & Building Models**

- Data preparation is followed by analyzing the data, uncovering hidden Trends and then applying various predictive & classification models on the train.

➤ **Modelling**

- We will now fit the image and forecast the results. We fit given data Into different regression analysis in order to compare their efficient choose.

➤ **Feature Engineering:**

- Here, we divide the characteristics and names first, then convert the hours to minutes. We are arranging the form of such a date of travel in our information in Date of Journey for easier preprocessing inside the model phase. Dep Time converts the departing date into clock time. Arrival Time is translated to minutes and hours by Arrival Time.

➤ **Feature Selection:**

- To enhance model performance, a features extraction approach is used to investigate the level of influence of each information on the prediction outcome. The price section has been removed since it is no longer useful.

➤ **Merging Models & Accuracy Calculation.**

- Having built various models, we have to test the models on our testing set and calculate the savings or loss done on each query put by the user. A statistic of the over-savings, Loss, and the mean saving per transaction are the measures used to calculate the accuracy of the model implemented.

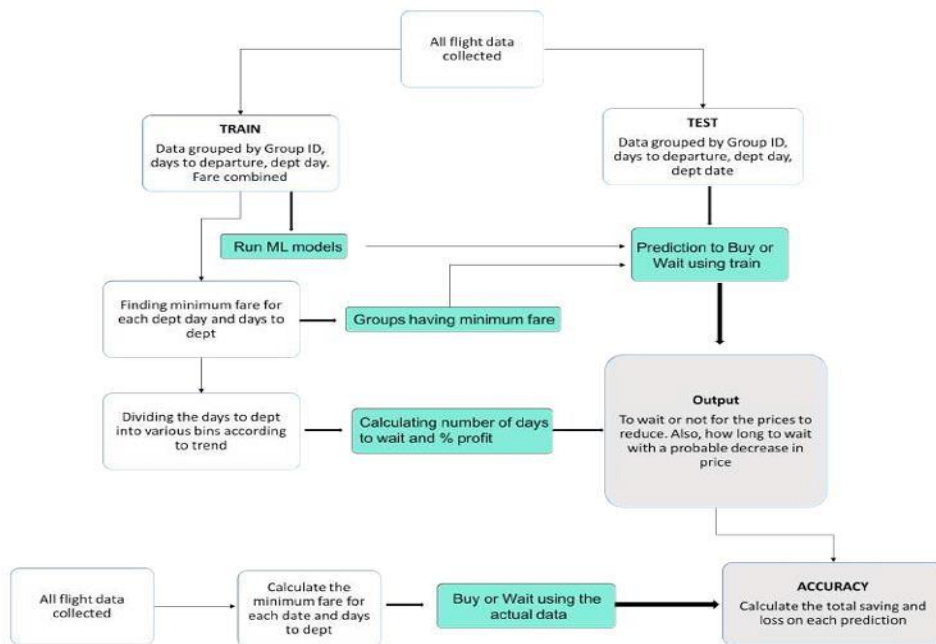


Figure 1 : Overview of the model

4.2 Overview Of model

4.3 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

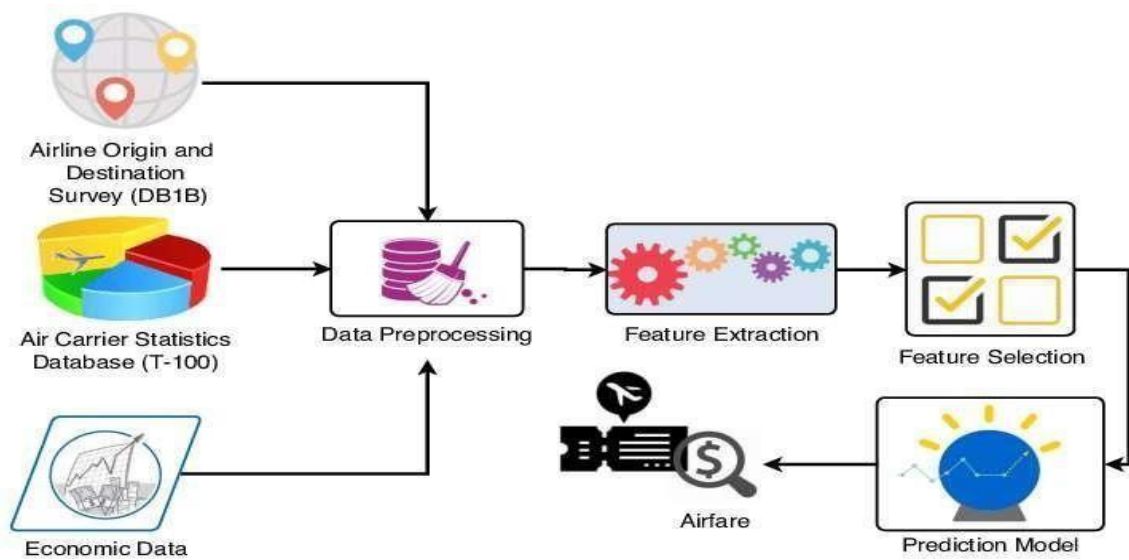


Fig 4.3: System Architecture

4.4 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

MACHINE LEARNING

- Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention.

DATASET

- Dataset is a collection of various types of data stored in a digital format. Data is the key component of any Machine Learning project. Datasets primarily consist of images, texts, audio, videos, numerical data points, etc., for solving various Artificial Intelligence challenges.

STEPS TO CREATE A DATASET

To create your dataset (and before doing data transformation), you should:

- Collect the raw data.
- Identify feature and label sources.
- Select a sampling strategy.
- Split the data.
- These steps depend a lot on how you've framed your ML problem.
- Use the self-check below to refresh your memory about problem framing and to check your assumptions about data collection.

PYTHON

- Python, when used in machine learning, offers developers of all skill sets exceptional versatility and power. Developers can use Python to develop a variety of applications because it integrates well with other software while its simple syntax makes it a good choice for coding algorithms and collaborating across teams. Python also has a huge number of libraries and frameworks that are very good for machine learning (such as Scikit-Learn), which handle basic machine learning algorithms.
- Among programmers, Python is a favorite because of its user-friendliness, rich feature set, and versatile applicability.
- Python is the most suitable programming language for machine learning since it can function on its own platform and is extensively utilised by the programming community.
- Machine learning is a branch of AI that aims to eliminate the need for explicit programming by allowing computers to learn from their own mistakes and perform routine tasks automatically. However, "artificial intelligence" (AI) encompasses a broader definition of "machine learning," which is the method through which computers are trained to recognize visual and auditory cues, and understand spoken Python, when used in machine learning, offers developers of all skill sets exceptional versatility and power.
- Developers can use Python to develop a variety of applications because it integrates well with other software while its simple syntax makes it a good choice for coding algorithms and collaborating across teams.
- Python also has a huge number of libraries and frameworks that are very good for machine learning (such as Scikit-Learn), which handle basic machine learning algorithms.

- The desire for intelligent solutions to real-world problems has necessitated the need to develop AI further in order to automate tasks that are arduous to program without AI. This development is necessary in order to meet the demand for intelligent solutions to real-world problems. Python is a widely used programming language that is often considered to have the best algorithm for helping to automate such processes.
- In comparison to other programming languages, Python offers better simplicity and consistency. In addition, the existence of an active Python community makes it simple for programmers to talk about ongoing projects and offer suggestions on how to improve the functionality of their programs.

ANACONDA

- Anaconda is an open-source package manager for Python and R. It is the most popular platform among data science professionals for running Python and R. There are over 300 libraries in data science, so having a robust distribution system for them is a must for any professional in this field. Anaconda simplifies package deployment and management. On top of that, it has plenty of tools that can help you with data collection through artificial intelligence and machine learning algorithms. With Anaconda, you can easily set up, manage, and share Conda environments. Moreover, you can deploy any required project with a few clicks when you're using Anaconda. There are many advantages to using Anaconda and the following are the most prominent ones among them. Anaconda is free and open-source. This means you can use it without spending any money. In the data science sector, Anaconda is an industry staple. It is open-source too, which has made it widely popular. If you want to become a data science professional, you must know how to use Anaconda for Python because every recruiter expects you to have this skill. It is a must-have for data science.

- It has more than 1500 Python and R data science packages, so you don't face any compatibility issues while collaborating with others. For example, suppose your colleague sends you a project which requires packages called A and B but you only have package A. Without having package B, you wouldn't be able to run the project. Anaconda mitigates the chances of such errors. You can easily collaborate on projects without worrying about any compatibility issues. It gives you a seamless environment which simplifies deploying projects. You can deploy any project with just a few clicks and commands while managing the rest. Anaconda has a thriving community of data scientists and machine learning professionals who use it regularly.
- If you encounter an issue, chances are, the community has already answered the same. On the other hand, you can also ask people in the community about the issues you face there, it's a very helpful community ready to help new learners. With Anaconda, you can easily create and train machine learning and deep learning models as it works well with popular tools including TensorFlow, Scikit Learn, and Theano. You can create visualizations by using Bokeh, Hollo views, Matplotlib, and Data shader while using Anaconda.

How to Use Anaconda for Python

Now that we have discussed all the basics in our Python Anaconda tutorial, let's discuss some fundamental commands you can use to start using this package manager.

Listing All Environments

To begin using Anaconda, you'd need to see how many Conda environments are present in your machine.

`Conda env list`

It will list all the available Conda environments in your machine.

Creating a New Environment

You can create a new Conda environment by going to the required directory and use this command:

```
conda create -n <your_environment_name>
```

You can replace `<your_environment_name>` with the name of your environment. After entering this command, conda will ask you if you want to proceed to which you should reply with y:

proceed ([y])/n)?

On the other hand, if you want to create an environment with a particular version of Python, you should use the following command:

```
conda create -n <your_environment_name> python=3.6
```

Similarly, if you want to create an environment with a particular package, you can use the following command:

```
conda create -n <your_environment_name> pack_name
```

Here, you can replace `pack_name` with the name of the package you want to use.

If you have a `.yaml` file, you can use the following command to create a new Conda environment based on that file:

```
conda env create -n <your_environment_name> -f <file_name>.yaml
```

We have also discussed how you can export an existing Conda environment to a `.yaml` file later in this article.

Activating an Environment

You can activate a Conda environment by using the following command:

```
conda activate <environment_name>
```

You should activate the environment before you start working on the same.

Also, replace the term <environment_name> with the environment name you want to activate. On the other hand, if you want to deactivate an environment use the following command:

```
conda deactivate
```

Installing Packages in an Environment

Now that you have an activated environment, you can install packages into it by using the following command:

```
conda install <pack_name>
```

Replace the term <pack_name> with the name of the package you want to install in your Conda environment while using this command.

Exporting an Environment Configuration

Suppose you want to share your project with someone else (colleague, friend, etc.). While you can share the directory on Github, it would have many Python packages, making the transfer process very challenging. Instead of that, you can create an environment configuration .yaml file and share it with that person. Now, they can create an environment like your one by using the .yaml file.

For exporting the environment to the .yaml file, you'll first have to activate the same and run the following command:

```
conda env export ><file_name>.yaml
```

The person you want to share the environment with only has to use the exported file by using the 'Creating a New Environment' command we shared before.

Removing a Package from an Environment

If you want to uninstall a package from a specific Conda environment, use the following command:

```
conda remove -n <env_name><package_name>
```

On the other hand, if you want to uninstall a package from an activated environment, you'd have to use the following command

```
conda remove <package_name>
```

Deleting an Environment

Sometimes, you don't need to add a new environment but remove one. In such cases, you must know how to delete a Conda environment, which you can do so by using the following command:

```
conda env remove --name <env_name>
```

The above command would delete the Conda environment right away.

JUPYTER NOTEBOOK

- This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.
- Project Jupyter is a project and community whose goal is to "develop open source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from Python in 2014 by Fernando Perez. Notebook documents are documents produced by the Jupyter Notebook App, which contain both computer code (e.g .python) and rich text elements (paragraph, equations, figures, links, etc)
- Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

.

4.5 PROJECT MANAGEMENT PLAN

1. *Project Objectives:*

- Develop a machine learning model/system for predicting flight prices accurately.
- Create a user-friendly interface for users to input travel preferences and receive predictions.
- Ensure scalability, reliability, and performance of the prediction system.

2. *Project Scope:*

- Data collection from various sources (flight APIs, historical databases, weather databases, etc.).
- Data preprocessing and feature engineering.
- Model development and training.
- Testing and validation.
- Deployment to a production environment.

3. *Project Team:*

- Data Scientists/Engineers: Responsible for data collection, preprocessing, model development, and testing.
- Software Developers: Responsible for building the user interface, integration, and deployment.
- Project Manager: Oversees the entire project, coordinates tasks, and manages resources.
- Quality Assurance/Testers: Conduct testing and validation of the system.
- UX/UI Designers: Design the user interface for optimal user experience.

4. *Project Timeline:*

- Define project milestones, deliverables, and timelines for each phase (e.g., data collection, model development, UI design, testing, deployment)
- Utilize project management tools like Gantt charts or Kanban boards to visualize and track progress.

5. Risk Management:

- Identify potential risks such as data quality issues, model overfitting, technical challenges, or changes in external factors (e.g., airline policies, economic conditions).
- Develop risk mitigation strategies, contingency plans, and escalation procedures.
- Regularly monitor and assess risks throughout the project lifecycle.

6. Communication Plan:

- Establish clear communication channels and protocols for team collaboration, meetings, and updates
- Hold regular status meetings to discuss progress, address issues, and align on priorities
- Use project management tools (e.g., Slack, Trello, Jira) for communication and task management.

7. Stakeholder Management:

- Identify key stakeholders such as project sponsors, end-users, and other relevant parties.
- Keep stakeholders informed about project progress, milestones, and any changes in scope or timelines.
- Solicit feedback and input from stakeholders to ensure alignment with project objectives.

8. Budget and Resources:

- Allocate resources (personnel, technology, infrastructure) based on project requirements and timelines
- Monitor budget utilization and resource allocation throughout the project.
- Adjust resources as needed to accommodate changes in project scope or requirements.

9. Documentation:

- Maintain comprehensive documentation for all project-related activities, including requirements, design, code, testing procedures, and user manuals.
- Ensure documentation is organized, accessible, and up-to-date to facilitate knowledge transfer and future maintenance.

10. Evaluation and Review:

- Conduct regular reviews and evaluations of project progress, milestones, and deliverables.
- Solicit feedback from team members, stakeholders, and end-users to identify areas for improvement.
- Adjust project plans and strategies based on lessons learned and feedback received.

CHAPTER 5

IMPLEMENTATION DETAILS

5.1. Development and Deployment Setup

1. *Data Collection:*

- Utilize flight APIs (such as Amadeus, Skyscanner, or FlightAware) to retrieve real-time and historical flight data.
- Scrape airline websites or other sources for additional data, including fares, schedules, and availability.
- Gather supplementary data such as weather forecasts, holidays, and economic indicators that may influence flight prices.

2. *Data Preprocessing:*

- Cleanse the collected data to handle missing values, outliers, and inconsistencies.
- Perform feature engineering to extract relevant features such as departure/arrival times, airline, route, seasonality, and demand indicators.
- Encode categorical variables, normalize numerical features, and apply transformations as needed for model compatibility.

3. *Model Development:*

- Experiment with various machine learning algorithms suitable for regression tasks, such as linear regression, decision trees, random forests, gradient boosting, or neural networks.
- Split the data into training, validation, and test sets for model evaluation.
- Train multiple models using different algorithms and hyperparameters to compare performance and select the best-performing model.

4. *User Interface Design:*

- Design an intuitive and user-friendly interface for users to input their travel preferences and receive flight price predictions.
- Implement features for users to specify departure/arrival locations, dates, preferred airlines, and other relevant criteria.
- Provide visualizations and summary statistics to help users interpret and understand the predicted prices.

5. *Testing and Validation:*

- Conduct unit tests to verify the functionality of individual components/modules, such as data preprocessing functions and model training routines.

- Perform integration tests to ensure proper communication and interaction between different modules of the system.
- Evaluate model performance using validation techniques such as k-fold cross-validation, assessing metrics like MAE, MSE, RMSE, and R-squared.

6. Deployment:

- Choose a deployment environment such as cloud platforms (AWS, Google Cloud, Azure) or on-premises servers.
- Containerize the application using Docker for easy deployment and scalability.
- Set up continuous integration/continuous deployment (CI/CD) pipelines for automated testing and deployment.
- Monitor the deployed system for performance, reliability, and security, using tools like Prometheus, Grafana, and ELK stack.

7. Maintenance and Updates:

- Establish processes for maintaining and updating the prediction system, including handling data updates, model retraining, and bug fixes.
- Monitor model performance over time and retrain models periodically to adapt to changing patterns and trends in flight prices.
- Incorporate user feedback and feature requests to enhance the user experience and improve prediction accuracy.

8. Documentation and Training:

- Document the implementation details, including data sources, preprocessing steps, model architectures, and user interface specifications.
- Provide user manuals, tutorials, and training materials for users and administrators to understand and utilize the system effectively.
- Foster a culture of knowledge sharing within the team to ensure continuity and sustainability of the project.

5.2 ALGORITHMS

5.2.1 *RANDOM FOREST*

Random Forest is a popular ensemble learning algorithm used in machine learning for both classification and regression tasks. It operates by constructing a multitude of decision trees during training and outputs the mode or mean of the individual predictions made by the trees (for classification and regression respectively).

The Random Forest algorithm works:

1. *Random Sampling:*

- It randomly selects subsets of the training data (with replacement), known as bootstrap samples, to train each decision tree. This technique is called bagging (Bootstrap Aggregating).

2. *Decision Tree Construction:*

- For each subset of data, a decision tree is constructed based on a random subset of features. This introduces randomness and reduces correlation among the trees, making the model less prone to overfitting.

3. *Voting (Classification) / Averaging (Regression):*

- For classification tasks, each tree "votes" for the class label, and the mode of these votes is considered the final prediction.
- For regression tasks, each tree produces a numerical prediction, and the average of these predictions is taken as the final output.

4. *Hyperparameters:*

- Random Forest has several hyperparameters that can be tuned to optimize its performance, including the number of trees (`n_estimators`), the maximum depth of each tree (`max_depth`), the minimum number of samples required to split an internal node (`min_samples_split`), etc.

5. Feature Importance:

- Random Forest provides a measure of feature importance based on how much each feature decreases the impurity across all trees. This can be useful for feature selection and understanding the importance of different features in making predictions.

6. Advantages:

- Random Forest tends to have high accuracy and generalizes well to unseen data.
- It is robust to overfitting, thanks to the ensemble approach and randomness introduced during training.
- It can handle large datasets with high dimensionality.

7. Disadvantages:

- Random Forest can be computationally expensive, especially for large numbers of trees and features.
- Interpretability can be challenging, as the model consists of multiple decision trees.
- It may not perform as well as other algorithms on tasks with highly imbalanced classes.

5.2.2 DECISION TREE

The Decision Tree algorithm is a popular and intuitive machine learning algorithm used for both classification and regression tasks. It works by recursively partitioning the input space into regions, with each partition corresponding to a decision node in the tree.

The Decision Tree algorithm works:

1. Decision Tree Construction:

- The algorithm starts with the entire dataset at the root node.
- It selects the best feature to split the dataset based on certain criteria (e.g., Gini impurity for classification, mean squared error for regression).
- The dataset is divided into subsets based on the chosen feature's values, creating child nodes.
- This process is recursively applied to each child node until a stopping criterion is

met (e.g., maximum tree depth, minimum number of samples per leaf).

- Decision Rule Learning:
- At each decision node, a decision rule is learned based on the selected feature and its threshold value. For example, "if feature A > 5, go left; otherwise, go right."
- This process results in a tree structure where each leaf node represents a class label (for classification) or a predicted value (for regression).

2. Classification:

- For classification tasks, the majority class label among the samples in a leaf node is assigned as the predicted class label for new instances falling into that leaf node.

3. Regression:

- For regression tasks, the mean or median of the target values in a leaf node is calculated and used as the predicted value for new instances falling into that leaf node.

4. Handling Categorical Features:

- Decision trees can handle both categorical and numerical features. For categorical features, the algorithm can use techniques like one-hot encoding or label encoding to represent them as numerical values.

5. Tree Pruning:

- Decision trees are prone to overfitting, especially when the tree grows too deep and captures noise in the training data. Pruning techniques, such as post-pruning (reducing the size of the tree after construction) or pre-pruning (stopping the tree construction process early), can help mitigate overfitting.

6. Advantages:

- Decision Trees are easy to interpret and visualize, making them useful for understanding the decision-making process.
- They can handle both numerical and categorical data.
- Decision Trees implicitly perform feature selection by selecting the most informative features for splitting.

7. Disadvantages:

- Decision Trees tend to be sensitive to small variations in the data and can easily overfit noisy datasets.
- They may not capture complex relationships in the data as effectively as other algorithms like Random Forests or Gradient Boosting Machines.

5.2.3 KNN ALGORITHM

The K-Nearest Neighbors (KNN) algorithm is a simple yet powerful supervised machine learning algorithm used for both classification and regression tasks. It is a non-parametric and lazy learning algorithm, meaning it does not make any assumptions about the underlying data distribution and does not learn a model during training. Instead, it stores all available training data and makes predictions based on the similarity of new data points to the existing data points.

The KNN algorithm works:

1. Training Phase:

- The training phase of the KNN algorithm simply involves storing the feature vectors and their corresponding labels (class labels for classification or target values for regression) in memory.

2. Prediction Phase:

- Given a new data point (instance) for which we want to make a prediction, the KNN algorithm calculates the distance between the new data point and all the data points in the training set. The most common distance metrics used are Euclidean distance, Manhattan distance, or Minkowski distance.
- It then identifies the K nearest neighbors (data points with the smallest distances) to the new data point.
- For classification tasks, the algorithm assigns the class label that is most frequent among the K nearest neighbors (the mode of their labels).
- For regression tasks, the algorithm computes the average (or weighted average) of the target values of the K nearest neighbors, which serves as the predicted value for the new data point.

3. Choosing the Value of K:

- The value of K, the number of nearest neighbors to consider, is a hyperparameter of the algorithm.
- It significantly influences the performance of the algorithm. A smaller value of K leads to more flexible decision boundaries (and potentially overfitting), while a larger value of K results in smoother decision boundaries (and potentially underfitting).
- The value of K is typically chosen using techniques like cross-validation or grid search.

4. Scaling Features:

- It is important to scale features before applying the KNN algorithm, as it is sensitive to the scale of the features. Normalizing or standardizing features to have a mean of 0 and a standard deviation of 1 can improve the performance of the algorithm.

5. Advantages:

- KNN is easy to understand and implement, making it a good choice for beginners and as a baseline algorithm.
- It is non-parametric, meaning it can handle complex decision boundaries and does not assume any specific distribution of the data.
- KNN can be used for both classification and regression tasks.

6. Disadvantages:

- KNN can be computationally expensive, especially for large datasets, as it requires calculating distances between the new data point and all the training data points.
- It is sensitive to the choice of the distance metric and the value of K.
- KNN does not learn a model during training, so prediction time can be slow for large datasets.

5.3 Testing Plan:

I. Unit Testing:

- Test individual components/modules to ensure they function correctly in isolation.
- Example: Test data collection functions to verify they retrieve the expected data from APIs or databases.

II. Integration Testing:

- Test the interaction between different modules to ensure they work together as expected.
- Example: Test the integration between the model development module and the data preprocessing module to verify that the correct data is fed into the models.

III. Model Testing:

- Test the trained models using validation techniques such as cross-validation to evaluate their performance and generalization ability.
- Example: Use k-fold cross-validation to assess the accuracy and robustness of the trained models.

IV. User Interface Testing:

- Test the user interface for usability, responsiveness, and accuracy of input/output.
- Example: Test user input validation to ensure users provide valid inputs for travel dates, destinations, etc.

V. End-to-End Testing:

- Test the entire system from data collection to user interface interaction to ensure it performs as expected in a real-world scenario.
- Example: Simulate user interactions with the system, including entering travel preferences and receiving predictions.

VI. Performance Testing:

- Test the performance of the system under different loads to ensure scalability and responsiveness.
- Example: Measure the response time of the system when multiple users access it simultaneously.

VII. User Acceptance Testing (UAT):

- Involve real users to test the system and gather feedback on usability, accuracy, and overall satisfaction.
- Example: Conduct surveys or interviews with users to collect feedback on their experience.

CHAPTER 6

RESULTS AND DISCUSSION

6.1 DATA ANALYSIS AND INTERPRETATION

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Airline	Date of Journey	Source	Destination	Route	Dep. Time	Arrival Time	Duration	Total Stops	Additional Info	Price								
2	IndiGo	24/03/2018	Bangalore	New Delhi	BLR → DEL	22:20	01:10	22 h 2h 50m	non-stop	No info	3897								
3	Air India	1/05/2018	Kolkata	Bangalore	CCU → IXB	05:50	13:15	7h 25m	2 stops	No info	7662								
4	Jet Airway	9/06/2018	Delhi	Cochin	DEL → LKC	09:25	04:25	10 h 19h	2 stops	No info	13882								
5	IndiGo	12/05/2018	Kolkata	Bangalore	CCU → N/A	18:05	23:30	5h 25m	1 stop	No info	6218								
6	IndiGo	01/03/2018	Bangalore	New Delhi	BLR → N/A	16:50	21:35	4h 45m	1 stop	No info	13302								
7	SpiceJet	24/06/2018	Kolkata	Bangalore	CCU → BLU	09:00	11:25	2h 25m	non-stop	No info	3873								
8	Jet Airway	12/03/2018	Bangalore	New Delhi	BLR → BO	18:55	10:25	13 h 15h 30m	1 stop	In-flight m	11087								
9	Jet Airway	01/03/2018	Bangalore	New Delhi	BLR → BO	08:00	05:05	02 h 21h 5m	1 stop	No info	22270								
10	Jet Airway	12/03/2018	Bangalore	New Delhi	BLR → BO	08:55	10:25	13 h 25h 30m	1 stop	In-flight m	11087								
11	Multiple c	27/05/2018	Delhi	Cochin	DEL → BO	11:25	19:15	7h 50m	1 stop	No info	8625								
12	Air India	1/06/2018	Delhi	Cochin	DEL → BLF	09:45	23:00	13h 15m	1 stop	No info	8907								
13	IndiGo	18/04/2018	Kolkata	Bangalore	CCU → BLU	20:20	22:55	2h 35m	non-stop	No info	4174								
14	Air India	24/06/2018	Chennai	Kolkata	MAA → CI	11:40	13:55	2h 15m	non-stop	No info	4667								
15	Jet Airway	9/05/2018	Kolkata	Bangalore	CCU → BC	21:10	09:20	10 h 12h 10m	1 stop	In-flight m	9663								
16	IndiGo	24/04/2018	Kolkata	Bangalore	CCU → BLU	17:15	19:50	2h 35m	non-stop	No info	4804								
17	Air India	3/03/2018	Delhi	Cochin	DEL → AV	16:40	19:15	04 h 26h 35m	2 stops	No info	14011								
18	SpiceJet	15/04/2018	Delhi	Cochin	DEL → PNW	08:45	13:15	4h 30m	1 stop	No info	5830								
19	Jet Airway	12/06/2018	Delhi	Cochin	DEL → BO	14:00	12:35	13 h 22h 35m	1 stop	In-flight m	10262								
20	Air India	12/06/2018	Delhi	Cochin	DEL → CCI	20:15	19:15	13 h 23h	2 stops	No info	13381								
21	Jet Airway	27/05/2018	Delhi	Cochin	DEL → BO	16:00	12:35	28 h 20h 35m	1 stop	In-flight m	12898								
22	Go Air	6/05/2018	Delhi	Cochin	DEL → BO	16:10	19:30	5h 30m	1 stop	No info	10405								

Fig 6.1.1 Train Data set

We have used the training dataset which contains a dataset of 10683 records of 11 columns.

Test Set [Read-Only] - Excel

Renuka Devi M

File Home Insert Page Layout Formulas Data Review View Add-ins Help Tell me what you want to do

Clipboard Font Alignment Number Styles Cells Editing Add-ins

A1 Airline

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info									
1	Jet Airway	6/06/2015	Delhi	Cochin	DEL → BO	17:30	04:25 07 J	10h 55m	1 stop	No info									
2	IndiGo	12/05/201	Kolkata	Banglore	CCU → M	06:20	10:20	4h	1 stop	No info									
3	Jet Airway	21/05/201	Delhi	Cochin	DEL → BO	19:15	19:00 22 J	23h 45m	1 stop	In-flight meal not included									
4	Multiple c	21/05/201	Delhi	Cochin	DEL → BO	08:00	21:00	13h	1 stop	No info									
5	Air Asia	24/06/201	Banglore	Delhi	BLR → DEI	23:55	02:45 25 J	2h 50m	non-stop	No info									
6	Jet Airway	12/06/201	Delhi	Cochin	DEL → BO	18:15	12:35 13 J	18h 20m	1 stop	In-flight meal not included									
7	Air India	12/03/201	Banglore	New Delhi	BLR → TRI	07:30	22:35	15h 5m	1 stop	No info									
8	IndiGo	1/05/2015	Kolkata	Banglore	CCU → HY	15:15	20:30	5h 15m	1 stop	No info									
9	IndiGo	15/03/201	Kolkata	Banglore	CCU → BLU	10:10	12:55	2h 45m	non-stop	No info									
10	Jet Airway	18/05/201	Kolkata	Banglore	CCU → BC	16:30	22:35	6h 5m	1 stop	No info									
11	Jet Airway	21/03/201	Delhi	Cochin	DEL → MA	13:55	18:50 22 J	28h 55m	2 stops	In-flight meal not included									
12	IndiGo	15/06/201	Delhi	Cochin	DEL → HY	06:50	16:10	9h 20m	1 stop	No info									
13	Multiple c	15/05/201	Delhi	Cochin	DEL → BO	09:00	19:15	10h 15m	1 stop	No info									
14	Jet Airway	12/03/201	Banglore	New Delhi	BLR → BO	05:45	10:25	4h 40m	1 stop	No info									
15	Jet Airway	3/06/2015	Delhi	Cochin	DEL → BO	19:15	12:35 04 J	17h 20m	1 stop	In-flight meal not included									
16	Jet Airway	06/03/201	Banglore	New Delhi	BLR → BO	21:25	08:15 07 J	10h 50m	1 stop	No info									
17	Multiple c	6/06/2015	Delhi	Cochin	DEL → HY	13:15	22:30	9h 15m	1 stop	No info									
18	Vistara	24/03/201	Kolkata	Banglore	CCU → DE	09:55	22:10	12h 15m	1 stop	No info									
19	Jet Airway	12/06/201	Delhi	Cochin	DEL → BO	19:15	04:25 13 J	9h 10m	1 stop	In-flight meal not included									
20	Jet Airway	12/03/201	Banglore	New Delhi	BLR → BO	22:55	08:15 13 J	9h 20m	1 stop	No info									
21	IndiGo	6/02/2015	Delhi	Cochin	DEL → BO	10:45	01:25 07 J	14h 50m	1 stop	No info									
22	IndiGo	6/02/2015	Delhi	Cochin	DEL → BO	10:45	01:25 07 J	14h 50m	1 stop	No info									

Ready Accessibility: Good to go

Type here to search

28°C 21:51 05-04-2024

Fig 6.1.2 Test Data Set

We have used the testing dataset which contains a dataset of 2671 records of 10 columns.

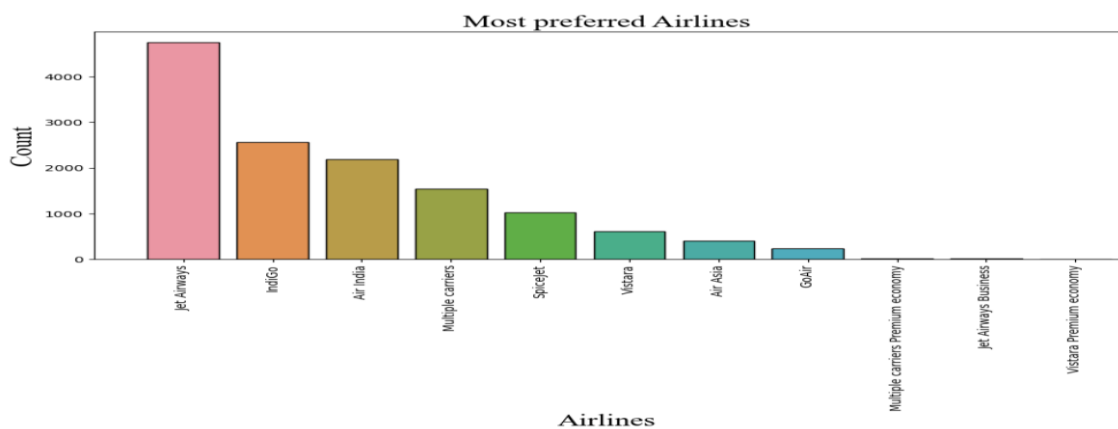


Fig 6.1.3 Most Preferred Airlines

Fig 6.1.3 explains the Airlines and the count of each flight. It will be useful while they are booking the flight according to the user's choice

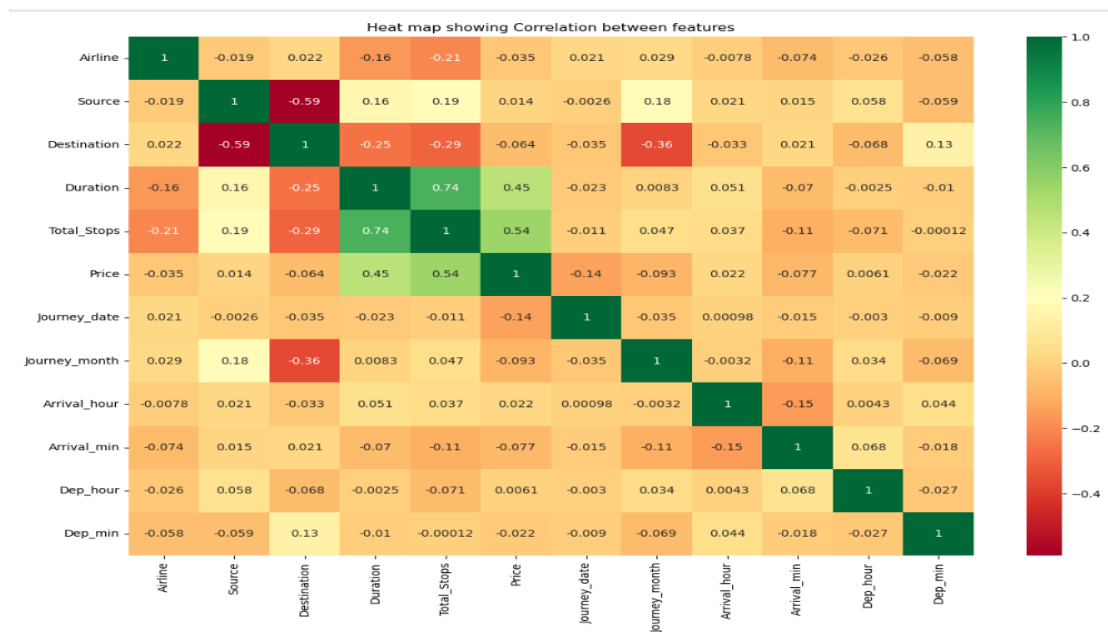


Fig 6.1.4 Heat Map

Fig 6.1.4 represents a Heat map, where the distinctive attributes of a matrix are visually presented through a color-coded graphical representation. A heat map proves to be a valuable instrument for illustrating the distribution of values across two dimensions within a matrix. This visualization technique improves the detection of patterns and provides a sense of depth in understanding the data.

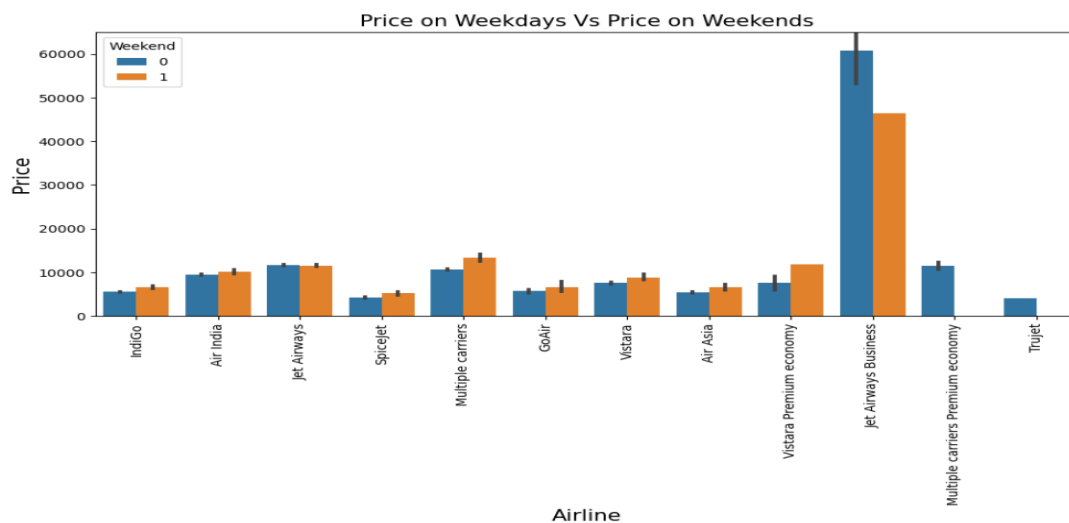


Fig 6.1.5 Price Differences

In Fig 6.1.5 , the presented fitting model showcases the variation in prices between weekdays and weekends. The data is transmitted to the SNS display(), offering insights into the differences observed in flight ticket prices between the two temporal categories

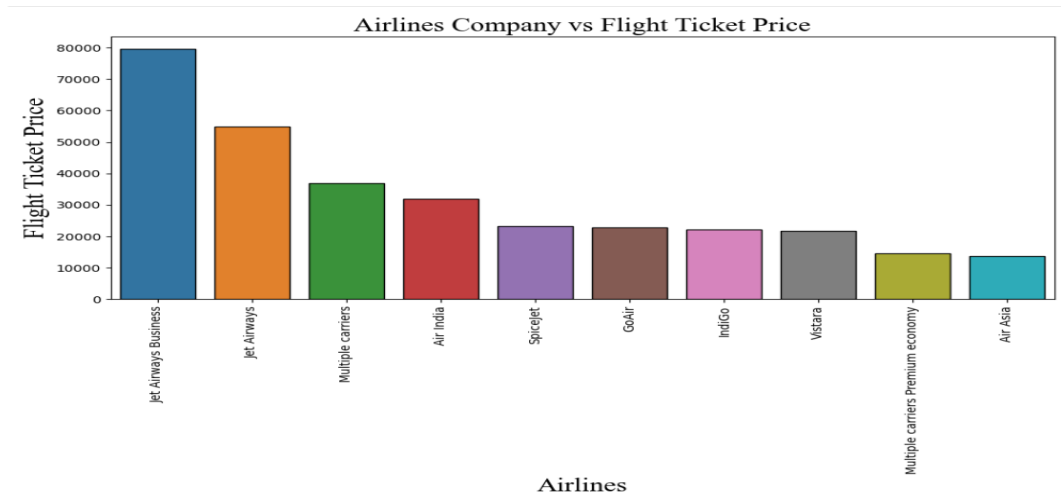


Fig 6.1.6 AirlinesCompany vsFlight Ticket Price

In Fig 6.1.6 , the showcased regression model illustrates a comparative analysis between different airlines and their respective flight ticket prices.

```

min_samples_split : [2, 5, 10, 15, 100],
'n_estimators': [100, 120, 150, 180, 200, 220, 250]),
random_state=0, verbose=2)

In [65]: # best parameter
rf_model.best_params_

Out[65]: {'n_estimators': 220,
'min_samples_split': 2,
'min_samples_leaf': 2,
'max_features': 'auto',
'max_depth': 15}

In [66]: #predicting the values
pred=rf_model.predict(x_test)
r2_score(y_test,pred)

Out[66]: 0.8230492038276901

In [67]: print('r2_score:',r2_score(y_test,pred))
print('MAE:', mean_absolute_error(y_test, pred))
print('MSE:', mean_squared_error(y_test, pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, pred)))

r2_score: 0.8230492038276901
MAE: 1137.2458437829803
MSE: 3686775.9413997084
RMSE: 1920.0978989102896
  
```

Fig 6.1.7 Accuracy Values

We achieved an accuracy of 82% for the Random Forest Algorithm after Training and

Testing.

6.2 FUTURE ENHANCEMENTS

Enhancing a flight price prediction project can involve several avenues of development. Here are some potential future enhancements you could consider:

1. **Advanced Machine Learning Models:** Incorporate more sophisticated machine learning models such as deep learning algorithms (e.g., LSTM, GRU, or even transformers like BERT) to capture more complex patterns and improve prediction accuracy.
2. **Ensemble Methods:** Utilize ensemble methods like Random Forest, Gradient Boosting, or stacking to combine predictions from multiple models, potentially improving overall performance.
3. **Feature Engineering:** Explore additional features that could influence flight prices, such as weather data, seasonal trends, holidays, or geopolitical events. Feature engineering can significantly impact model performance.
4. **Hyperparameter Tuning:** Conduct extensive hyperparameter tuning to optimize the performance of your models. Techniques like grid search, random search, or Bayesian optimization can help find the best set of hyperparameters for your models.
5. **Data Augmentation:** Augment your dataset to increase its size or diversity, which can help improve model generalization and robustness.
6. **Real-time Data Integration:** Implement mechanisms to fetch real-time data on flight prices, availability, and other relevant factors to make your predictions more up-to-date and accurate.
7. **User Interface Improvements:** Enhance the user interface of your application to make it more user-friendly and intuitive. This could involve better visualization of predictions, interactive features, or personalized recommendations.

8. **Performance Optimization:** Optimize your code and infrastructure for better scalability, speed, and efficiency, especially if dealing with large datasets or high traffic.
9. **Feedback Mechanism:** Implement a feedback loop where users can provide feedback on the accuracy of predictions, which can be used to continuously improve the models.
10. **Integration with Travel Services:** Partner with travel agencies or airlines to integrate booking functionalities directly into your application, providing users with a seamless experience from prediction to booking.
11. **Mobile App Development:** Develop a mobile application version of your project to reach users on mobile devices, offering convenience and accessibility.
12. **Subscription or Premium Features:** Consider offering subscription-based or premium features, such as advanced prediction algorithms, ad-free experience, or exclusive deals, to monetize your application.
13. **Community Building:** Foster a community around your project where users can share tips, experiences, and insights related to flight booking and travel.
14. **Ethical Considerations:** Ensure that your project adheres to ethical guidelines, especially regarding data privacy, fairness, and transparency.

6.3 PERFORMANCE ANALYSIS

1. Accuracy Metrics:

- Mean Absolute Error (MAE): This metric calculates the average absolute difference between predicted and actual prices. Lower MAE indicates better accuracy.
- Mean Squared Error (MSE): MSE measures the average squared difference between predicted and actual prices. Lower MSE signifies better accuracy, with outliers having a more significant impact.
- Root Mean Squared Error (RMSE): RMSE is the square root of MSE, providing an interpretable measure in the same units as the original data.

2. *Relative Metrics:*

- **Percentage Error:** This metric calculates the percentage difference between predicted and actual prices, providing insight into the relative accuracy of predictions.
- **Percentage Accuracy:** The percentage of predictions falling within a certain threshold of the actual price. For example, within $\pm 5\%$ or $\pm 10\%$ of the true price.

3. *Robustness Analysis:*

- **Cross-validation:** Perform cross-validation to assess model performance on different subsets of the data. Techniques like k-fold cross-validation help evaluate generalization performance and identify overfitting.
- **Outlier Analysis:** Evaluate model performance in the presence of outliers, which can significantly impact prediction accuracy. Robust models should exhibit resilience to outliers.

4. *Efficiency Metrics:*

- **Training Time:** Measure the time taken to train the prediction models. Faster training times are preferable, especially when dealing with large datasets or frequent model updates.
- **Inference Time:** Assess the time required to generate predictions for new data points. Low inference time is essential for real-time or near-real-time applications.

5. *Comparison with Baselines:*

- Compare the performance of your prediction models against simple baseline models (e.g., mean price, linear regression) to gauge the effectiveness of your approach.
- Benchmark against existing flight price prediction services or historical data to assess the improvement achieved by your models.

6. Visualizations:

- Visualize predicted versus actual prices over time to identify trends, patterns, and discrepancies.
- Plot residual distributions to examine model errors and identify areas for improvement.

7. Feedback Loop:

- Incorporate user feedback to continuously refine and improve the prediction models. Solicit feedback on prediction accuracy, usability, and relevance to iteratively enhance the performance of the system.

8. Scalability Analysis:

- Evaluate the scalability of the prediction system by measuring performance metrics under increasing data volumes, user loads, or computational resources.

CHAPTER 7

CONCLUSION

Traveling is one of the biggest reasons, and everyone wants to try new things in life. With its diversity, India offers one of the most fascinating cities in the world, as well as a wealth of fantastic options and one of the world's most religiously and ethnically diverse nations. Because of its cultural and geographical variety, India is one of the world's most promising countries.

SUMMARY

This aims to provide users with a tool to predict flight fares based on various parameters, allowing them to make informed decisions when booking air travel. The app utilizes machine learning algorithms trained on historical flight data to estimate future fares.

Recent years have seen a dramatic rise in air travel for a variety of reasons, including economic expansion and the emergence of low-cost airlines. Fares are very variable because Indian airlines employ a revenue management system to respond to market conditions in real time. The cost of a flight ticket varies based on duration of the flight, destination, route, arrival time, departure time as well as certain events like holidays or vacations. The main aim is to predict the correct flight fare using Machine Learning Techniques and Auto ML. Additionally, the properties of the provided dataset have been examined using a variety of visualisation techniques, including scatter plots, distribution plots, catplots, and ggplots. The results show that the Random Forest Regressor and Randomized Search CV techniques give highest accuracy with the fare prediction dataset.

These developments also raise concerns regarding the cost of airline tickets. When comparing the price of an airline ticket today to the previous day, it might be difficult to make an educated judgement. Tourists who wish to visit a new location in India should be aware of ticket prices in order to obtain the cheapest and most reliable ticket price that meets their requirements.

The Flight Fare Prediction System is a comprehensive solution aimed at accurately forecasting flight ticket prices, providing travelers with valuable insights for better planning and decision-making.

CONCLUSION

We discovered that this form of prediction-based service may be utilized to estimate the pricing of various products and services in the online world after applying this study. As an example, we computed the best airfare price and made it simple for clients to book tickets now or later. This type of technology can be utilized in e-commerce platforms to determine product prices based on market demand, not just for airfare prices. Because the price of every commodity fluctuates based on market demand. Aside from this, technology is also utilized in the stock market to manipulate stock values based on market demand. More research is being done in this area to increase prediction accuracy using various machine learning method. In conclusion, a flight price prediction project holds significant potential for assisting travelers in making informed decisions and securing the best deals for their journeys. By leveraging machine learning techniques, data analysis, and real-time information, such projects can offer valuable insights into the fluctuating nature of flight prices. However, it's important to acknowledge that predicting flight prices accurately is a challenging task due to the multitude of factors influencing them. Throughout the development and enhancement of such projects, several key considerations emerge:

1. **Accuracy:** Striving for accurate predictions remains paramount. Employing advanced machine learning models, refining feature engineering, and continuously updating datasets are essential for improving prediction accuracy.
2. **User Experience:** Prioritizing a seamless and intuitive user experience is crucial for ensuring the usability and adoption of the application. This involves designing clear interfaces, providing relevant information, and possibly integrating feedback mechanisms.
3. **Real-time Updates:** Incorporating real-time data updates allows users to access the most current information, empowering them to make timely decisions and potentially capitalize on price fluctuations.
4. **Ethical Considerations:** Upholding ethical standards, particularly regarding data privacy, fairness, and transparency, is non-negotiable. Users must trust that their data is handled responsibly and that predictions are made without bias.

REFERENCES

- [1] Boruah, Abhijit, et al. "A bayesian approach for flight fare prediction based on kalman filter." Progress in Advanced Computing and Intelligent Engineering. Springer, Singapore, 2019. 191-203.
- [2] Cheung, Jacob CH."Flight planning: node- based trajectory prediction and turbulence avoidance."Meteorological Applications :78-85,2018
- [3] Dey.S, Purkayastha.M.B., and Saikia .H.K., "Prediction of Flight Fare Using Machine Learning Techniques," in 2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), 2019.
- [4] Groves.W and Gini.M, An agent for optimizing airline ticket purchasing, 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), St. Paul, MN, May 06 - 10, 2013.
- [5] Horiguchi, Y., Baba, Y., Kashima, H., Suzuki, M., Kayahara, H., & Maeno, J. . Predicting fuel consumption and flight delays for low-cost airlines. In Twenty-Ninth IAAI Conference-2017.
- [6] Iqbal.S, M. Rehman.M, and Riaz.M, "Predicting Domestic Air Fare Using Machine Learning Techniques," in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019.
- [7] Janssen.T \"A linear quantile mixed regression model for prediction of airline ticket prices\",2022.
- [8] Jain.A.V , Raval.A.S , and Oza.R.K , \"Airfare price prediction based on reviews using machine learning techniques,\" vol. 9, issue 3, March 2020.
- [9] Kimbhaune, V., Donga, H., Trivedi, A., Mahajan, S., & Mahajan, V. Flight Fare Prediction System (No. 5542),2021

- [10] Supriya Rajankar, Neha Sakharkar, Omprakar Rajankar, "Predicting the price of a flight ticket with the use of Machine Learning algorithms", international journal of scientific & technology research, vol.8, December, 2019
- [11] Swayze, Neal C., et al. "Influence of UAS Flight Altitude and Speed on Above ground Biomass Prediction." Remote Sensing 14.9,2022:
- [12] Shah.K, Kedia.V and Jha.R.K, "Integrated Vendor-Managed Time Efficient Application to Production of Inventory Systems," 2021 6th International Conference on Inventive Computation Technologies(ICICT),2021,pp.275280,doi:10.1109/ICICT50816.2021.9358504.
- [13] Sharma, Garg, P. (Eds.) From Visual Surveillance to Internet of Things, Chapman and Hall/CRC, New York, 2020.
- [14] Tziridis.K Kalampokas.T Papakostas.G and Diamantaras.K "Airfare price prediction using machine learning techniques" in European Signal Processing Conference (EUSIPCO), DOI: 10.23919/EUSIPCO .2017..
- [15] Tianyi Wang, Samira Pouyanfar, Haiman Tian, Yudong Tao, Miguel Alonso Jr., Steven Luis and Shu-Ching Chen, "A Framework for Airfare Price Prediction: A Machine Learning Approach", 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI), September 9, 2019.
- [16] Vandehzad, Mashhood. "Efficient flight schedules with utilizing Machine Learning prediction algorithms" -2020
- [17] Vinod Kimbhaune, Harshil Donga, Ashutosh Trivedi, Sonam Mahajan and Viraj Mahajan research paper on flight fare prediction system,2021
- [18] William Groves and Maria Gini "An agent for optimizing airline ticket purchasing" in proceedings of the 2018 international conference on autonomous agents and multi-agent systems.
- [19] Wohlfarth, T. clemencon, S. Roueff "A Data mining approach to travel price forecasting" 10th international conference on machine learning Honolulu-2021.
- [20] Yunita.R.I , Lubis.B.F.S, and Elvandari.R, "Predicting Flight Ticket Price Using Decision Tree and Artificial Neural Network," in 2019 6th International Conference on Information Technology and Electrical Engineering (ICITEE), 2019.

APPENDIX

A. SOURCE CODE

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

import warnings

warnings.filterwarnings("ignore")

Train_data=pd.read_csv("Data_Train.csv")

Train_data.shape

Test_data=pd.read_csv("Test_set.csv")

Test_data.shape

flight_df=Train_data.append(Test_data,sort=False)

flight_df.head(10)

flight_df.shape

flight_df.describe()

flight_df.info()
```

```

flight_df.dtypes

plt.figure(figsize=(12,5))

sns.countplot(x="Airline", data=flight_df,order =
flight_df['Airline'].value_counts().index,ec = "black")

font_style={'family':'times new
roman','size':20,'color':'black'}

plt.title("Most preferred Airlines",fontdict=font_style)

plt.ylabel("Count",fontdict=font_style)

plt.xlabel("Airlines",fontdict=font_style)

plt.xticks(rotation= 90)

plt.xlim(-1,10.5)

plt.show()

airlines = flight_df.groupby('Airline').Price.max()

airlines_df=
airlines.to_frame().sort_values('Price',ascending
False)[0:10]

airlines_df

```

```

plt.subplots(figsize=(12,5))

sns.barplot(x=airlines_df.index,
y=airlines_df["Price"],ec = "black")

font_style={'family':'times new
roman','size':20,'color':'black'}

plt.title("Airlines Company vs Flight Ticket
Price",fontdict=font_style )

plt.ylabel("Flight Ticket Price", fontdict=font_style)

plt.xlabel("Airlines", fontdict=font_style)

plt.xticks(rotation=90)

plt.show()

days_df = flight_df[['Airline','Date_of_Journey',
'Price']].copy()

days_df.head()

days_df['Date_of_Journey'] =
pd.to_datetime(days_df['Date_of_Journey'],format='%
d/%m/%Y')

days_df['Weekday'] =
days_df['Date_of_Journey'].dt.day_name()

days_df['Weekend'] =

```

```

days_df['Weekday'].apply(lambda day: 1 if day ==
'Sunday' else 0)

days_df.head()

plt.subplots(figsize=(12,5))

sns.barplot(data=days_df,x='Airline',y='Price',hue='We
ekend')

plt.xlabel("Airline",size = 15)

plt.xticks(rotation = 90)

plt.ylabel("Price",size = 15)

plt.title("Price on Weekdays Vs Price on
Weekends",size = 15)

plt.legend(title='Weekend'.title())

plt.ylim(0,65000)

plt.show()

flight_df["Journey_date"]=flight_df["Date_of_Journey"].
str.split("/").str[0].astype(int)

flight_df["Journey_month"]=flight_df["Date_of_Journey
"].str.split("/").str[1].astype(int)

flight_df["Journey_year"]=flight_df["Date_of_Journey"].
str.split("/").str[2].astype(int)

```

```

flight_df.head()

flight_df=flight_df.drop(["Date_of_Journey"],axis=1)

flight_df["Journey_year"].value_counts()

flight_df=flight_df.drop(["Journey_year"],axis=1)

flight_df.head()

flight_df["Total_Stops"]=flight_df["Total_Stops"].str.split(" ").str[0]

flight_df["Total_Stops"]=flight_df["Total_Stops"].replace("non-stop","0")

flight_df.head()

# Arrival_Time

flight_df["Arrival_Time"]=flight_df["Arrival_Time"].str.split(" ").str[0]

flight_df["Arrival_hour"]=flight_df["Arrival_Time"].str.split(':').str[0].astype(int)

flight_df["Arrival_min"]=flight_df["Arrival_Time"].str.split(':').str[1].astype(int)

flight_df=flight_df.drop(["Arrival_Time"],axis=1)

```

```

# Dep_Time

flight_df["Dep_hour"]=flight_df["Dep_Time"].str.split(':').
str[0].astype(int)

flight_df["Dep_min"]=flight_df["Dep_Time"].str.split(':').s
tr[1].astype(int)

flight_df=flight_df.drop(["Dep_Time"],axis=1)

flight_df.head()

flight_df.drop(["Route", "Additional_Info"], axis = 1,
inplace = True)

flight_df["Duration_hr"]=flight_df["Duration"].str.split('
').str[0].str.split('h').str[0]

flight_df.head()

flight_df["Duration_min"]=flight_df["Duration"].str.split('
').str[1].str.split('m').str[0]

flight_df.head()

flight_df["Duration_min"].fillna("0",inplace=True)

flight_df["Duration_hr"].str.contains('h').sum()

flight_df["Duration_hr"].str.contains('m').sum()

flight_df["Duration_min"].str.contains('h').sum()

```

```

flight_df['Duration_min'].str.contains('m').sum()

flight_df[flight_df['Duration_hr'].str.contains('m')]

flight_df.loc[[6474, 2660], 'Duration_min'] = flight_df.loc[[6474, 2660], 'Duration_hr']

flight_df.loc[flight_df['Duration_min'] == '5m', 'Duration_min'] = 5

flight_df.head()

flight_df["Duration_hr"] = flight_df["Duration_hr"].replace("5m", "0")

flight_df['Duration_hr'].str.contains('m').sum()

flight_df[flight_df['Duration_hr'].str.contains('m')]

flight_df["Duration_hr"] = flight_df["Duration_hr"].astype(int)

flight_df["Duration_min"] = flight_df["Duration_min"].astype(int)

flight_df.head()

flight_df["Duration"] = (flight_df["Duration_hr"]*60) + flight_df["Duration_min"]

flight_df = flight_df.drop(['Duration_hr', 'Duration_min'], axis=1)

flight_df.head()

flight_df.head(6475)

flight_df.isnull().sum()

flight_df["Price"].fillna(flight_df["Price"].mean(), inplace=True)

flight_df["Total_Stops"].value_counts()

flight_df.dtypes

flight_df["Total_Stops"] = flight_df["Total_Stops"].fillna('1')

flight_df["Total_Stops"] = flight_df["Total_Stops"].astype(int)

flight_df.isnull().sum()

```

```

from sklearn.preprocessing import LabelEncoder

la=LabelEncoder()

for i in ["Airline","Source","Destination"]:

flight_df[i]=la.fit_transform(flight_df[i])

flight_df.head()

plt.figure(figsize=(15,10))

sns.heatmap(flight_df.corr(),annot=True,cmap='RdYlGn')

plt.title("Heat map showing Correlation between features")

plt.show()

train_df=flight_df[0:10683]

test_df=flight_df[10683:]

train_df.tail()

test_df.head()

x=train_df.drop(["Price"],axis=1)

y=train_df.loc[:,["Price"]].values

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state=0)

from sklearn.neighbors import KNeighborsRegressor

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error

```

```

def predict(algorithm):

    print("Algorithm:",algorithm)

    model=algorithm.fit(x_train,y_train)

    y_pred=model.predict(x_test)

    print("The Predicted Price values are:",y_pred)

    print('\n')

    r2=r2_score(y_test,y_pred)

    print("r2 score :",r2)

    #updating score in df

    #score=score.append({'Model':algorithm,'Accuracy':r2})

    print('MAE:',(mean_absolute_error(y_test,y_pred)))

    print('MSE:',(mean_squared_error(y_test,y_pred)))

    return(r2)

knn=predict(KNeighborsRegressor())

dt=predict(DecisionTreeRegressor())

rf=predict(RandomForestRegressor())

#storing the accuracies of these 3 models in a dataframe

score_df=pd.DataFrame({'Model': ['K-Nearest Neighbour',

    'Decision Tree','Random Forest'],

    'Accuracy': [ knn*100,dt*100, rf*100]})

score_df

from sklearn.model_selection import RandomizedSearchCV

random_search = {'n_estimators' : [100, 120, 150, 180, 200,220,250],

    'max_features':['auto','sqrt'],

```

```

'max_depth':[5,10,15,20],
'min_samples_split' : [2, 5, 10, 15, 100],

'min_samples_leaf' : [1, 2, 5, 10]}

rf_regressor=RandomForestRegressor()

rf_model=RandomizedSearchCV(estimator=rf_regressor,param_distributions=random_se
arch, cv=3,n_jobs=-1,verbose=2,random_state=0)

rf_model.fit(x_train,y_train)

# best parameter

rf_model.best_params_

#predicting the values

pred=rf_model.predict(x_test)

r2_score(y_test,pred)

print('r2_score:',r2_score(y_test,pred))

print('MAE:', mean_absolute_error(y_test, pred))

print('MSE:', mean_squared_error(y_test, pred))

print('RMSE:', np.sqrt(mean_squared_error(y_test, pred)))

```

B.SCREENSHOTS

The screenshot displays a Jupyter Notebook titled "Final Year Project" running on a local host. The notebook contains five code cells. The first cell imports necessary libraries: pandas, numpy, matplotlib.pyplot, seaborn, and warnings. The second cell loads "Data_Train.csv" and shows its shape as (10683, 11). The third cell loads "Test_set.csv" and shows its shape as (2671, 10). The fourth cell appends the test data to the train data and displays the first 10 rows of the combined dataset. The output shows a table with columns: Airline, Date_of_Journey, Source, Destination, Route, Dep_Time, Arrival_Time, Duration, Total_Stops, Additional_Info, and Price.

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")

In [3]: Train_data=pd.read_csv("Data_Train.csv")
Train_data.shape

Out[3]: (10683, 11)

In [4]: Test_data=pd.read_csv("Test_set.csv")
Test_data.shape

Out[4]: (2671, 10)

In [5]: flight_df=Train_data.append(Test_data,sort=False)
flight_df.head(10)

Out[5]:
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897.0
1	Air India	1/05/2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25m	2 stops	No info	7662.0
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882.0

Final Year Project Last Checkpoint: 19 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
In [5]: flight_df=train_data.append(Test_data,sort=False)
flight_df.head(10)
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897.0
1	Air India	1/05/2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25m	2 stops	No info	7662.0
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882.0
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h 25m	1 stop	No info	6218.0
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 45m	1 stop	No info	13302.0
5	SpiceJet	24/06/2019	Kolkata	Banglore	CCU ? BLR	09:00	11:25	2h 25m	non-stop	No info	3873.0
6	Jet Airways	12/03/2019	Banglore	New Delhi	BLR ? BOM ? DEL	18:55	10:25 13 Mar	15h 30m	1 stop	In-flight meal not included	11087.0
7	Jet Airways	01/03/2019	Banglore	New Delhi	BLR ? BOM ? DEL	08:00	05:05 02 Mar	21h 5m	1 stop	No info	22270.0
8	Jet Airways	12/03/2019	Banglore	New Delhi	BLR ? BOM ? DEL	08:55	10:25 13 Mar	25h 30m	1 stop	In-flight meal not included	11087.0
9	Multiple carriers	27/05/2019	Delhi	Cochin	DEL ? BOM ? COK	11:25	19:15	7h 50m	1 stop	No info	8625.0

```
In [6]: flight_df.shape
Out[6]: (13354, 11)
```

Final Year Project Last Checkpoint: 19 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
In [7]: flight_df.describe()
```

	Price
count	10683.000000
mean	9087.064121
std	4611.359167
min	1759.000000
25%	5277.000000
50%	8372.000000
75%	12373.000000
max	79512.000000

```
In [8]: flight_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13354 entries, 0 to 2670
Data columns (total 11 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Airline         13354 non-null  object
1   Date_of_Journey 13354 non-null  object
2   Source          13354 non-null  object
3   Destination     13354 non-null  object
4   Route          13353 non-null  object
5   Dep_Time        13354 non-null  object
```

Final Year Project - Jupyter Notebook

localhost:8888/notebooks/Final%20Year%20Project.ipynb

Final Year Project Last Checkpoint: 19 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3 (pykernel)

```

In [8]: flight_df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 13354 entries, 0 to 2670
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype  
---  -
 0   Airline              13354 non-null  object  
 1   Date_of_Journey      13354 non-null  object  
 2   Source               13354 non-null  object  
 3   Destination          13354 non-null  object  
 4   Route               13353 non-null  object  
 5   Dep_Time             13354 non-null  object  
 6   Arrival_Time         13354 non-null  object  
 7   Duration             13354 non-null  object  
 8   Total_Stops          13353 non-null  object  
 9   Additional_Info      13354 non-null  object  
10   Price               10683 non-null  float64  
dtypes: float64(1), object(10)
memory usage: 1.2+ MB

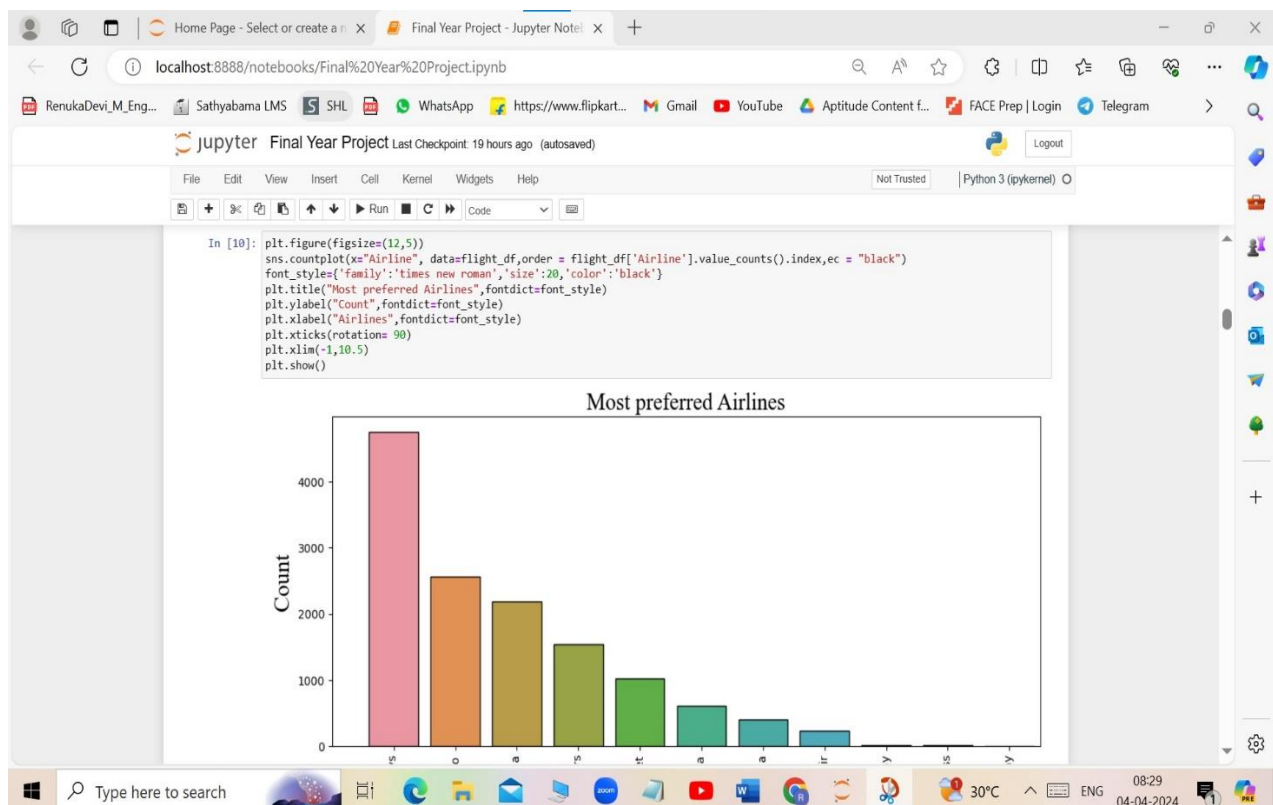
In [9]: flight_df.dtypes

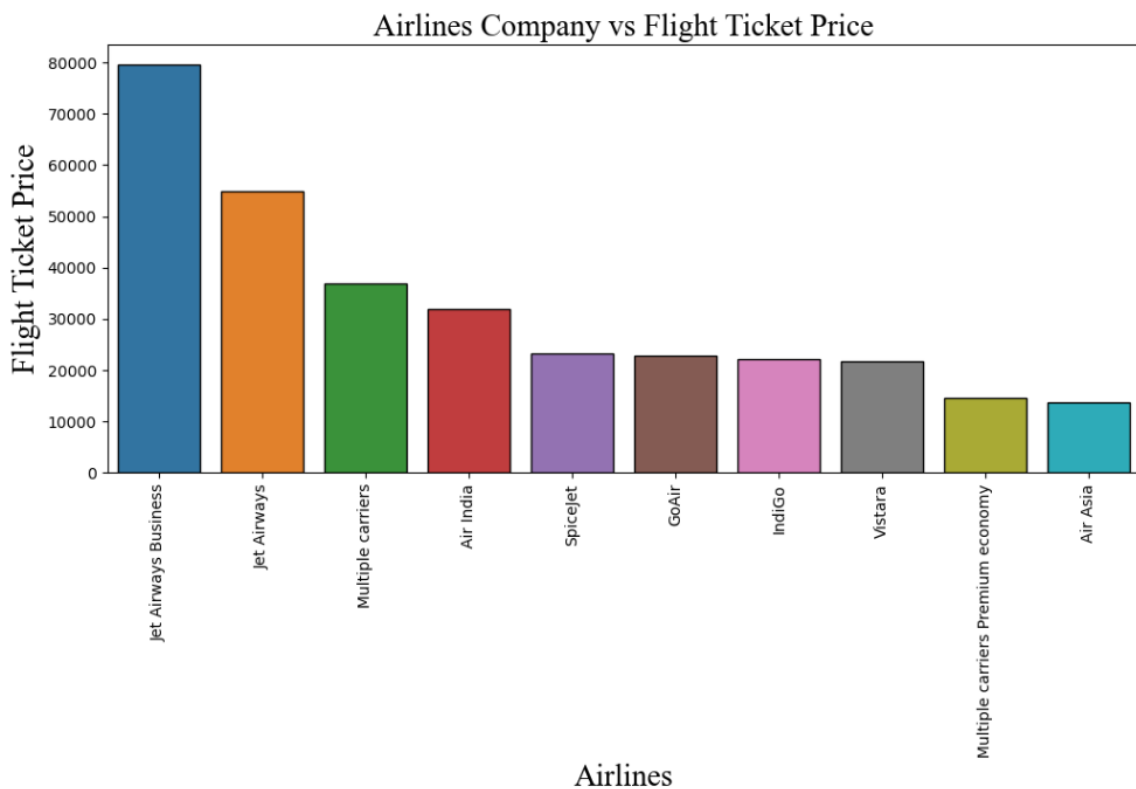
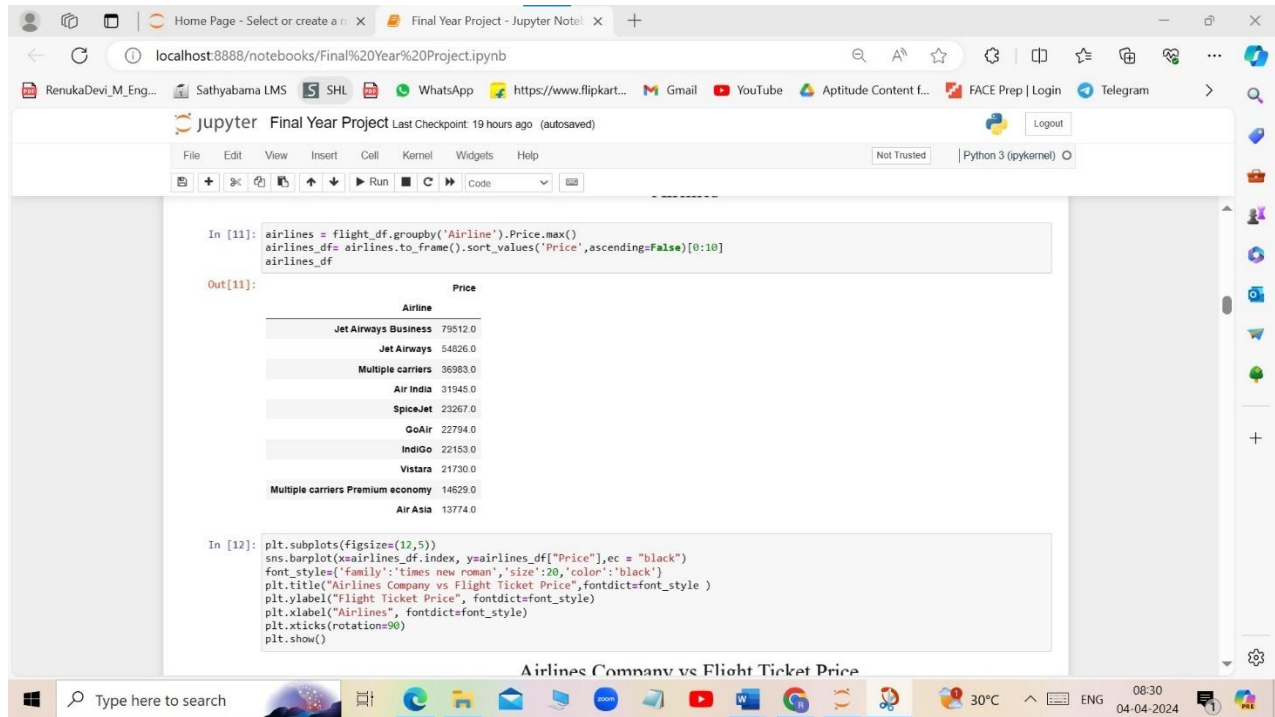
Out[9]: Airline              object
Date_of_Journey            object
Source                     object
Destination                 object
Route                      object
Dep_Time                   object
Arrival_Time               object
Duration                   object
Total_Stops                object
Additional_Info             object
Price                      float64
dtype: object

```

Type here to search

30°C 06:28 04-04-2024





Final Year Project - Jupyter Notebook

localhost:8888/notebooks/Final%20Year%20Project.ipynb

Final Year Project Last Checkpoint: 19 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
In [13]: days_df = flight_df[['Airline', 'Date_of_Journey', 'Price']].copy()
days_df.head()
```

```
Out[13]:
```

	Airline	Date_of_Journey	Price
0	IndiGo	24/03/2019	3897.0
1	Air India	1/05/2019	7662.0
2	Jet Airways	9/06/2019	13882.0
3	IndiGo	12/05/2019	6218.0
4	IndiGo	01/03/2019	13302.0

```
In [14]: days_df['Date_of_Journey'] = pd.to_datetime(days_df['Date_of_Journey'], format='%d/%m/%Y')
days_df['Weekday'] = days_df['Date_of_Journey'].dt.day_name()
days_df['Weekend'] = days_df['Weekday'].apply(lambda day: 1 if day == 'Sunday' else 0)
days_df.head()
```

```
Out[14]:
```

	Airline	Date_of_Journey	Price	Weekday	Weekend
0	IndiGo	2019-03-24	3897.0	Sunday	1
1	Air India	2019-05-01	7662.0	Wednesday	0
2	Jet Airways	2019-06-09	13882.0	Sunday	1
3	IndiGo	2019-05-12	6218.0	Sunday	1
4	IndiGo	2019-03-01	13302.0	Friday	0

Final Year Project - Jupyter Notebook

localhost:8888/notebooks/Final%20Year%20Project.ipynb

Final Year Project Last Checkpoint: 19 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
In [15]: plt.subplots(figsize=(10,5))
plt.bar(days_df[days_df['Airline']=='IndiGo'], y='Price', hue='Weekend')
plt.xlabel('Airline', size=15)
plt.xticks(rotation=90)
plt.ylabel('Price', size=15)
plt.title('Price on Weekdays vs Price on Weekends', size=15)
plt.legend(title='Weekend', titleloc='left')
plt.show()
```

Airline	Weekend (0)	Weekend (1)
IndiGo	3897.0	13302.0
Air India	7662.0	0.0
Jet Airways	13882.0	0.0
Spencer	0.0	0.0
Multiple carriers	0.0	0.0
GoAir	0.0	0.0
Vistara	0.0	0.0
Air Asia	0.0	0.0
Volare Premium economy	0.0	0.0
Jet Airways Business	0.0	0.0
Multiple carriers Premium economy	0.0	0.0
Thyair	0.0	0.0

Final Year Project - Jupyter Notebook

localhost:8888/notebooks/Final%20Year%20Project.ipynb

Final Year Project Last Checkpoint: 19 hours ago (autosaved)

Python 3 (ipykernel)

```
In [16]: flight_df["Journey_date"] = flight_df["Date_of_Journey"].str.split("/").str[0].astype(int)
flight_df["Journey_month"] = flight_df["Date_of_Journey"].str.split("/").str[1].astype(int)
flight_df["Journey_year"] = flight_df["Date_of_Journey"].str.split("/").str[2].astype(int)
```

```
In [17]: flight_df.head()
```

```
Out[17]:
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journey_date	Journey_month
0	IndiGo	24/03/2019	Bangalore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897.0	24	
1	Air India	1/05/2019	Kolkata	Bangalore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25m	2 stops	No info	7662.0	1	
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882.0	9	
3	IndiGo	12/05/2019	Kolkata	Bangalore	CCU ? NAG ? BLR	18:05	23:30	5h 25m	1 stop	No info	6218.0	12	
4	IndiGo	01/03/2019	Bangalore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 45m	1 stop	No info	13302.0	1	

Final Year Project - Jupyter Notebook

localhost:8888/notebooks/Final%20Year%20Project.ipynb

Final Year Project Last Checkpoint: 19 hours ago (autosaved)

Python 3 (ipykernel)

```
In [18]: flight_df = flight_df.drop(["Date_of_Journey"], axis=1)
```

```
In [19]: flight_df["Journey_year"].value_counts()
```

```
Out[19]:
```

```
2019    13354
Name: Journey_year, dtype: int64
```

```
In [20]: flight_df = flight_df.drop(["Journey_year"], axis=1)
flight_df.head()
```

```
Out[20]:
```

	Airline	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journey_date	Journey_month
0	IndiGo	Bangalore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897.0	24	3
1	Air India	Kolkata	Bangalore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25m	2 stops	No info	7662.0	1	5
2	Jet Airways	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882.0	9	6
3	IndiGo	Kolkata	Bangalore	CCU ? NAG ? BLR	18:05	23:30	5h 25m	1 stop	No info	6218.0	12	5
4	IndiGo	Bangalore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 45m	1 stop	No info	13302.0	1	3

```
In [21]: flight_df["Total_Stops"] = flight_df["Total_Stops"].str.split(" ").str[0]
flight_df["Total_Stops"] = flight_df["Total_Stops"].replace("non-stop", "0")
flight_df.head()
```

```
Out[21]:
```

	Airline	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journey_date	Journey_month
0	IndiGo	Bangalore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	0	No info	3897.0	24	3

Final Year Project - Jupyter Notebook

localhost:8888/notebooks/Final%20Year%20Project.ipynb

Final Year Project Last Checkpoint: 19 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3 (ipykernel)

Out[21]:

	Airline	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journey_date	Journey_month
0	IndiGo	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	0	No info	3897.0	24	3
1	Air India	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25m	2	No info	7662.0	1	5
2	Jet Airways	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19h	2	No info	13882.0	9	6
3	IndiGo	Kolkata	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h 25m	1	No info	6218.0	12	5
4	IndiGo	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 45m	1	No info	13302.0	1	3

In [22]:

```
# Arrival_Time
flight_df["Arrival_Time"] = flight_df["Arrival_Time"].str.split(" ").str[0]
flight_df["Arrival_hour"] = flight_df["Arrival_Time"].str.split(':').str[0].astype(int)
flight_df["Arrival_min"] = flight_df["Arrival_Time"].str.split(':').str[1].astype(int)
flight_df = flight_df.drop(["Arrival_Time"], axis=1)

# Dep_Time
flight_df["Dep_hour"] = flight_df["Dep_Time"].str.split(':').str[0].astype(int)
flight_df["Dep_min"] = flight_df["Dep_Time"].str.split(':').str[1].astype(int)
flight_df = flight_df.drop(["Dep_Time"], axis=1)

flight_df.head()
```

Out[22]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_date	Journey_month	Arrival_hour	Arrival_min	Dep_hour	Dep_min
0	IndiGo	Banglore	New Delhi	BLR ? DEL	2h 50m	0	No info	3897.0	24	3	1	10	22	
1	Air India	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	7h 25m	2	No info	7662.0	1	5	13	15	5	
2	Jet Airways	Delhi	Cochin	DEL ? LKO ? BOM ? COK	19h	2	No info	13882.0	9	6	4	25	9	
3	IndiGo	Kolkata	Banglore	CCU ? NAG ? BLR	5h 25m	1	No info	6218.0	12	5	23	30	18	
4	IndiGo	Banglore	New Delhi	BLR ? NAG ? DEL	4h 45m	1	No info	13302.0	1	3	21	35	16	

Final Year Project - Jupyter Notebook

localhost:8888/notebooks/Final%20Year%20Project.ipynb

Final Year Project Last Checkpoint: 19 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3 (ipykernel)

Out[22]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_date	Journey_month	Arrival_hour	Arrival_min	Dep_hour	Dep_min
0	IndiGo	Banglore	New Delhi	BLR ? DEL	2h 50m	0	No info	3897.0	24	3	1	10	22	
1	Air India	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	7h 25m	2	No info	7662.0	1	5	13	15	5	
2	Jet Airways	Delhi	Cochin	DEL ? LKO ? BOM ? COK	19h	2	No info	13882.0	9	6	4	25	9	
3	IndiGo	Kolkata	Banglore	CCU ? NAG ? BLR	5h 25m	1	No info	6218.0	12	5	23	30	18	
4	IndiGo	Banglore	New Delhi	BLR ? NAG ? DEL	4h 45m	1	No info	13302.0	1	3	21	35	16	

In [23]:

```
flight_df.drop(["Route", "Additional_Info"], axis = 1, inplace = True)
```

In [24]:

```
flight_df["Duration_hr"] = flight_df["Duration"].str.split(' ').str[0].str.split('h').str[0]
```

In [25]:

```
flight_df.head()
```

Out[25]:

	Airline	Source	Destination	Duration	Total_Stops	Price	Journey_date	Journey_month	Arrival_hour	Arrival_min	Dep_hour	Dep_min	Duration_hr
0	IndiGo	Banglore	New Delhi	2h 50m	0	3897.0	24	3	1	10	22		2
1	Air India	Kolkata	Banglore	7h 25m	2	7662.0	1	5	13	15	5		7
2	Jet Airways	Delhi	Cochin	19h	2	13882.0	9	6	4	25	9		19
3	IndiGo	Kolkata	Banglore	5h 25m	1	6218.0	12	5	23	30	18		5
4	IndiGo	Banglore	New Delhi	4h 45m	1	13302.0	1	3	21	35	16		4

Final Year Project - Jupyter Notebook

localhost:8888/notebooks/Final%20Year%20Project.ipynb

Final Year Project Last Checkpoint: 19 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3 (ipykernel)

Out[25]:

	Airline	Source	Destination	Duration	Total_Stops	Price	Journey_date	Journey_month	Arrival_hour	Arrival_min	Dep_hour	Dep_min	Duration_hr
0	IndiGo	Banglore	New Delhi	2h 50m	0	3897.0	24	3	1	10	22	20	2
1	Air India	Kolkata	Banglore	7h 25m	2	7662.0	1	5	13	15	5	50	7
2	Jet Airways	Delhi	Cochin	19h	2	13882.0	9	6	4	25	9	25	19
3	IndiGo	Kolkata	Banglore	5h 25m	1	6218.0	12	5	23	30	18	5	5
4	IndiGo	Banglore	New Delhi	4h 45m	1	13302.0	1	3	21	35	16	50	4

In [26]: flight_df["Duration_min"] = flight_df["Duration"].str.split(' ').str[1].str.split('m').str[0]

In [27]: flight_df.head()

Out[27]:

	Airline	Source	Destination	Duration	Total_Stops	Price	Journey_date	Journey_month	Arrival_hour	Arrival_min	Dep_hour	Dep_min	Duration_hr
0	IndiGo	Banglore	New Delhi	2h 50m	0	3897.0	24	3	1	10	22	20	2
1	Air India	Kolkata	Banglore	7h 25m	2	7662.0	1	5	13	15	5	50	7
2	Jet Airways	Delhi	Cochin	19h	2	13882.0	9	6	4	25	9	25	19
3	IndiGo	Kolkata	Banglore	5h 25m	1	6218.0	12	5	23	30	18	5	5
4	IndiGo	Banglore	New Delhi	4h 45m	1	13302.0	1	3	21	35	16	50	4

In [28]: flight_df["Duration_min"].fillna("0", inplace=True)

Final Year Project - Jupyter Notebook

localhost:8888/notebooks/Final%20Year%20Project.ipynb

Final Year Project Last Checkpoint: 19 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3 (ipykernel)

In [28]: flight_df["Duration_min"].fillna("0", inplace=True)

In [29]: flight_df["Duration_hr"].str.contains('h').sum()

Out[29]: 0

In [30]: flight_df["Duration_hr"].str.contains('m').sum()

Out[30]: 2

In [31]: flight_df["Duration_min"].str.contains('h').sum()

Out[31]: 0

In [32]: flight_df["Duration_min"].str.contains('m').sum()

Out[32]: 0

In [33]: flight_df[flight_df["Duration_hr"].str.contains('m')]

Out[33]:

	Airline	Source	Destination	Duration	Total_Stops	Price	Journey_date	Journey_month	Arrival_hour	Arrival_min	Dep_hour	Dep_min	Duration_hr
6474	Air India	Mumbai	Hyderabad	5m	2	17327.0	6	3	16	55	16	50	5m
2660	Air India	Mumbai	Hyderabad	5m	2	NaN	12	3	16	55	16	50	5m

Final Year Project - Jupyter Notebook

localhost:8888/notebooks/Final%20Year%20Project.ipynb

Final Year Project Last Checkpoint: 19 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
In [34]: flight_df.loc[[6474, 2660], 'Duration_min'] = flight_df.loc[[6474, 2660], 'Duration_hr']
flight_df.loc[flight_df['Duration_min'] == '5m', 'Duration_min'] = 5
```

```
In [35]: flight_df.head()
```

```
Out[35]:
```

	Airline	Source	Destination	Duration	Total_Stops	Price	Journey_date	Journey_month	Arrival_hour	Arrival_min	Dep_hour	Dep_min	Duration_hr	Duration_min
0	IndiGo	Banglore	New Delhi	2h 50m	0	3897.0	24	3	1	10	22	20	2	120
1	Air India	Kolkata	Banglore	7h 25m	2	7662.0	1	5	13	15	5	50	7	420
2	Jet Airways	Delhi	Cochin	19h	2	13882.0	9	6	4	25	9	25	19	1140
3	IndiGo	Kolkata	Banglore	5h 25m	1	6218.0	12	5	23	30	18	5	5	300
4	IndiGo	Banglore	New Delhi	4h 45m	1	13302.0	1	3	21	35	16	50	4	240

```
In [36]: flight_df["Duration_hr"] = flight_df["Duration_hr"].replace("5m", "0")
```

```
In [37]: flight_df["Duration_hr"].str.contains('m').sum()
```

```
Out[37]: 0
```

```
In [38]: flight_df[flight_df["Duration_hr"].str.contains('m')]
```

```
Out[38]:
```

	Airline	Source	Destination	Duration	Total_Stops	Price	Journey_date	Journey_month	Arrival_hour	Arrival_min	Dep_hour	Dep_min	Duration_hr	Duration_min
--	---------	--------	-------------	----------	-------------	-------	--------------	---------------	--------------	-------------	----------	---------	-------------	--------------

Final Year Project - Jupyter Notebook

localhost:8888/notebooks/Final%20Year%20Project.ipynb

Final Year Project Last Checkpoint: 19 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
In [39]: flight_df["Duration_hr"] = flight_df["Duration_hr"].astype(int)
```

```
In [40]: flight_df["Duration_min"] = flight_df["Duration_min"].astype(int)
```

```
In [41]: flight_df.head()
```

```
Out[41]:
```

	Airline	Source	Destination	Duration	Total_Stops	Price	Journey_date	Journey_month	Arrival_hour	Arrival_min	Dep_hour	Dep_min	Duration_hr	Duration_min
0	IndiGo	Banglore	New Delhi	2h 50m	0	3897.0	24	3	1	10	22	20	2	120
1	Air India	Kolkata	Banglore	7h 25m	2	7662.0	1	5	13	15	5	50	7	420
2	Jet Airways	Delhi	Cochin	19h	2	13882.0	9	6	4	25	9	25	19	1140
3	IndiGo	Kolkata	Banglore	5h 25m	1	6218.0	12	5	23	30	18	5	5	300
4	IndiGo	Banglore	New Delhi	4h 45m	1	13302.0	1	3	21	35	16	50	4	240

```
In [42]: flight_df["Duration"] = (flight_df["Duration_hr"]*60) + flight_df["Duration_min"]
flight_df=flight_df.drop(['Duration_hr', 'Duration_min'], axis=1)
flight_df.head()
```

```
Out[42]:
```

	Airline	Source	Destination	Duration	Total_Stops	Price	Journey_date	Journey_month	Arrival_hour	Arrival_min	Dep_hour	Dep_min	Duration
0	IndiGo	Banglore	New Delhi	170	0	3897.0	24	3	1	10	22	20	170
1	Air India	Kolkata	Banglore	445	2	7662.0	1	5	13	15	5	50	445
2	Jet Airways	Delhi	Cochin	1140	2	13882.0	9	6	4	25	9	25	1140

Final Year Project - Jupyter Notebook

localhost:8888/notebooks/Final%20Year%20Project.ipynb

Final Year Project Last Checkpoint: 19 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Out[42]:

	Airline	Source	Destination	Duration	Total_Stops	Price	Journey_date	Journey_month	Arrival_hour	Arrival_min	Dep_hour	Dep_min
0	IndiGo	Bangalore	New Delhi	170	0	3897.0	24	3	1	10	22	20
1	Air India	Kolkata	Bangalore	445	2	7662.0	1	5	13	15	5	50
2	Jet Airways	Delhi	Cochin	1140	2	13882.0	9	6	4	25	9	25
3	IndiGo	Kolkata	Bangalore	325	1	6218.0	12	5	23	30	18	5
4	IndiGo	Bangalore	New Delhi	285	1	13302.0	1	3	21	35	16	50

In [43]: flight_df.head(6475)

Out[43]:

	Airline	Source	Destination	Duration	Total_Stops	Price	Journey_date	Journey_month	Arrival_hour	Arrival_min	Dep_hour	Dep_min
0	IndiGo	Bangalore	New Delhi	170	0	3897.0	24	3	1	10	22	20
1	Air India	Kolkata	Bangalore	445	2	7662.0	1	5	13	15	5	50
2	Jet Airways	Delhi	Cochin	1140	2	13882.0	9	6	4	25	9	25
3	IndiGo	Kolkata	Bangalore	325	1	6218.0	12	5	23	30	18	5
4	IndiGo	Bangalore	New Delhi	285	1	13302.0	1	3	21	35	16	50
...	...	...	...	...	...	...	...	...	...	...	...	...
6470	Multiple carriers Premium economy	Delhi	Cochin	725	1	9845.0	21	3	21	0	8	55
6471	IndiGo	Bangalore	Delhi	165	0	4077.0	1	5	4	15	1	30
6472	Multiple carriers	Delhi	Cochin	690	1	9315.0	21	5	19	0	7	30
6473	Jet Airways	Kolkata	Bangalore	570	1	14781.0	9	5	23	35	14	5
6474	Air India	Mumbai	Hyderabad	5	2	17327.0	6	3	16	55	16	50

Final Year Project - Jupyter Notebook

localhost:8888/notebooks/Final%20Year%20Project.ipynb

Final Year Project Last Checkpoint: 20 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

6470	Multiple carriers Premium economy	Delhi	Cochin	725	1	9845.0	21	3	21	0	8	55
6471	IndiGo	Bangalore	Delhi	165	0	4077.0	1	5	4	15	1	30
6472	Multiple carriers	Delhi	Cochin	690	1	9315.0	21	5	19	0	7	30
6473	Jet Airways	Kolkata	Bangalore	570	1	14781.0	9	5	23	35	14	5
6474	Air India	Mumbai	Hyderabad	5	2	17327.0	6	3	16	55	16	50

6475 rows x 12 columns

In [44]: flight_df.isnull().sum()

Out[44]:

```
Airline      0
Source       0
Destination  0
Duration     0
Total_Stops  1
Price      2671
Journey_date 0
Journey_month 0
Arrival_hour 0
Arrival_min  0
Dep_hour     0
Dep_min      0
dtype: int64
```

In [45]: flight_df["Price"].fillna(flight_df["Price"].mean(), inplace=True)

Final Year Project - Jupyter Notebook

localhost:8888/notebooks/Final%20Year%20Project.ipynb

Final Year Project Last Checkpoint: 20 hours ago (autosaved)

Python 3 (ipykernel)

```

In [46]: flight_df["Total_Stops"].value_counts()
Out[46]: 1    7056
         0    4340
         2    1899
         3     56
         4      2
         Name: Total_Stops, dtype: int64

In [47]: flight_df.dtypes
Out[47]: Airline      object
         Source      object
         Destination  object
         Duration     int32
         Total_Stops  object
         Price        float64
         Journey_date int32
         Journey_month int32
         Arrival_hour int32
         Arrival_min  int32
         Dep_hour     int32
         Dep_min      int32
         dtype: object

In [48]: flight_df["Total_Stops"] = flight_df["Total_Stops"].fillna('1')

In [49]: flight_df["Total_Stops"] = flight_df["Total_Stops"].astype(int)

```

Final Year Project - Jupyter Notebook

localhost:8888/notebooks/Final%20Year%20Project.ipynb

Final Year Project Last Checkpoint: 20 hours ago (autosaved)

Python 3 (ipykernel)

```

In [50]: flight_df.isnull().sum()
Out[50]: Airline      0
         Source      0
         Destination  0
         Duration     0
         Total_Stops  0
         Price        0
         Journey_date 0
         Journey_month 0
         Arrival_hour 0
         Arrival_min  0
         Dep_hour     0
         Dep_min      0
         dtype: int64

In [51]: from sklearn.preprocessing import LabelEncoder
         la=LabelEncoder()
         for i in ["Airline","Source","Destination"]:
             flight_df[i]=la.fit_transform(flight_df[i])
         flight_df.head()
Out[51]:

```

	Airline	Source	Destination	Duration	Total_Stops	Price	Journey_date	Journey_month	Arrival_hour	Arrival_min	Dep_hour	Dep_min
0	3	0	5	170	0	3897.0	24	3	1	10	22	20
1	1	3	0	445	2	7662.0	1	5	13	15	5	50
2	4	2	1	1140	2	13882.0	9	6	4	25	9	25
3	3	3	0	325	1	6218.0	12	5	23	30	18	5

Final Year Project - Jupyter Notebook

localhost:8888/notebooks/Final%20Year%20Project.ipynb

jupyter Final Year Project Last Checkpoint: 20 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Out[51]:

	Airline	Source	Destination	Duration	Total_Stops	Price	Journey_date	Journey_month	Arrival_hour	Arrival_min	Dep_hour	Dep_min
0	3	0	5	170	0	3897.0	24	3	1	10	22	20
1	1	3	0	445	2	7662.0	1	5	13	15	5	50
2	4	2	1	1140	2	13882.0	9	6	4	25	9	25
3	3	3	0	325	1	6218.0	12	5	23	30	18	5
4	3	0	5	285	1	13302.0	1	3	21	35	16	50

In [52]:

```
plt.figure(figsize=(15,10))
sns.heatmap(flight_df.corr(),annot=True,cmap='RdYlGn')
plt.title("Heat map showing Correlation between features")
plt.show()
```

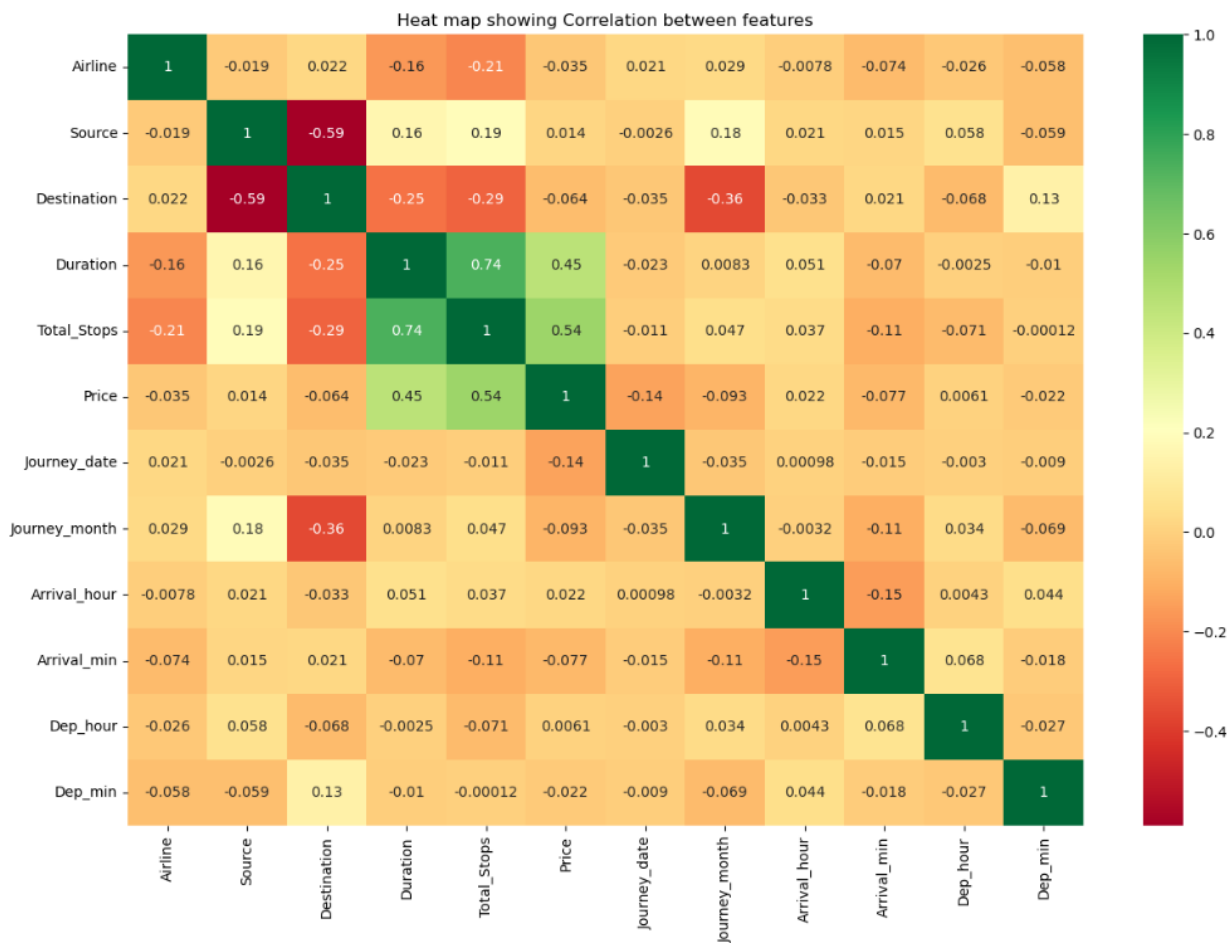
Heat map showing Correlation between features

Airline

Source

Destination

Duration



Final Year Project - Jupyter Notebook

localhost:8888/notebooks/Final%20Year%20Project.ipynb

Final Year Project Last Checkpoint: 20 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3 (ipykernel)

```

print('MAE:',(mean_absolute_error(y_test,y_pred)))
print('MSE:',(mean_squared_error(y_test,y_pred)))
return(r2)

In [60]: knn=predict(KNeighborsRegressor())

Algorithm: KNeighborsRegressor()
The Predicted Price values are: [[ 8399.4]
 [ 3843.4]
 [10933. ]
 ...
 [14808.8]
 [ 6443.2]
 [ 6902.4]]

r2 score : 0.5878415036033772
MAE: 1769.0793710220892
MSE: 8587336.488041932

In [61]: dt=predict(DecisionTreeRegressor())

Algorithm: DecisionTreeRegressor()
The Predicted Price values are: [ 6314. 3857. 10991. ... 14441. 7229. 5842.]

r2 score : 0.6694906806589163
MAE: 1401.9386122550854
MSE: 6886173.067955823

```

Type here to search

31°C 09:29 04-04-2024

Final Year Project - Jupyter Notebook

localhost:8888/notebooks/Final%20Year%20Project.ipynb

Final Year Project Last Checkpoint: 20 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3 (ipykernel)

```

In [62]: rf=predict(RandomForestRegressor())

Algorithm: RandomForestRegressor()
The Predicted Price values are: [ 6815.07      3840.62083333 11506.48333333 ... 14264.91261905
 6790.099      6338.46      ]

r2 score : 0.8016021009109884
MAE: 1186.5382896420315
MSE: 4133627.0703939092

In [63]: #storing the accuracies of these 3 models in a dataframe
score_df=pd.DataFrame({'Model': ['K-Nearest Neighbour',
                                'Decision Tree', 'Random Forest'],
                      'Accuracy': [ knn*100,dt*100, rf*100]})

score_df

Out[63]:
   Model Accuracy
0  K-Nearest Neighbour  58.784150
1    Decision Tree    66.949068
2    Random Forest    80.160210

In [64]: from sklearn.model_selection import RandomizedSearchCV
random_search = {'n_estimators': [100, 120, 150, 180, 200,220,250],
                 'max_features':['auto','sqrt'],
                 'max_depth':[5,10,15,20],
                 'min_samples_split': [2, 5, 10, 15, 100],

```

Type here to search

31°C 09:30 04-04-2024

Final Year Project - Jupyter Notebook

localhost:8888/notebooks/Final%20Year%20Project.ipynb

Final Year Project Last Checkpoint: 20 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Out[63]:

	Model	Accuracy
0	K-Nearest Neighbour	58.784150
1	Decision Tree	66.949068
2	Random Forest	80.160210

In [64]:

```
from sklearn.model_selection import RandomizedSearchCV
random_search = {'n_estimators': [100, 120, 150, 180, 200, 220, 250],
                  'max_features': ['auto', 'sqrt'],
                  'max_depth': [5, 10, 15, 20],
                  'min_samples_split': [2, 5, 10, 15, 100],
                  'min_samples_leaf': [1, 2, 5, 10]}
rf_regressor=RandomForestRegressor()
rf_model=RandomizedSearchCV(estimator=rf_regressor,param_distributions=random_search,
                             cv=3,n_jobs=-1,verbose=2,random_state=0)
rf_model.fit(x_train,y_train)
```

Fitting 3 folds for each of 10 candidates, totalling 30 fits

Out[64]:

```
RandomizedSearchCV(cv=3, estimator=RandomForestRegressor(), n_jobs=-1,
                   param_distributions={'max_depth': [5, 10, 15, 20],
                                       'max_features': ['auto', 'sqrt'],
                                       'min_samples_leaf': [1, 2, 5, 10],
                                       'min_samples_split': [2, 5, 10, 15,
                                                             100],
                                       'n_estimators': [100, 120, 150, 180,
                                                         200, 220, 250]},
                   random_state=0, verbose=2)
```

Final Year Project - Jupyter Notebook

localhost:8888/notebooks/Final%20Year%20Project.ipynb

Final Year Project Last Checkpoint: 20 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
min_samples_split: [2, 5, 10, 15,
                    100],
'n_estimators': [100, 120, 150, 180,
                 200, 220, 250]],
random_state=0, verbose=2)
```

In [65]:

```
# best parameter
rf_model.best_params_
```

Out[65]:

```
{'n_estimators': 220,
 'min_samples_split': 2,
 'min_samples_leaf': 2,
 'max_features': 'auto',
 'max_depth': 15}
```

In [66]:

```
#predicting the values
pred=rf_model.predict(x_test)
r2_score(y_test,pred)
```

Out[66]:

```
0.8230492038276901
```

In [67]:

```
print('r2_score:',r2_score(y_test,pred))
print('MAE:', mean_absolute_error(y_test, pred))
print('MSE:', mean_squared_error(y_test, pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, pred)))
```

r2_score: 0.8230492038276901
MAE: 1137.2458437829803
MSE: 3686775.9413997084
RMSE: 1920.097898102896

Harnessing Cognitive Algorithms for Flight Fare Detection

S.L. Jany Shabu^{a)}, R. Lakshmi Prasanna, M. Renuka Devi, J. Refonaa, M. Maheswari, Dhamodaran

Sathyabama Institute of Science and Technology, Chennai, India

^{a)}Corresponding author: janyshabu.cse@sathyabama.ac.in

Abstract. The main means of transportation for both long- and short-distance travel, in terms of convenience, is the airplane. However, flight costs are erratic, making it difficult for the average person to predict them. Flight costs vary depending on the time of day, with prices rising and falling depending on demand. The holidays and the unique festive season also cause it to change continuously. There are numerous factors that influence the cost of airline tickets. The utmost distributors retain data or information on multitudinous parameters, while purchasers have limited knowledge that makes it insolvable for them to prognosticate flight costs. To find the stylish time to buy tickets, we will probe a machine literacy model that considers variables like the departure position, the number of stops, the appearance time, the number of days until departure, and the time of day. Finding the factors that affect travel prices and the relationships between these factors and value changes is the aim of this research project.

Keywords — Flight fare, Random Forest, Machine-learning, Hyperparameter tuning.

INTRODUCTION

Travel is a major aspiration for many individuals, and the allure of exploring new facets of life is universal. India, with its incredible diversity, stands out as a captivating destination, boasting some of the world's most fascinating cities and serving as a melting pot of religions and ethnicities. The country's cultural and geographical richness positions it as one of the most promising nations globally.

However, the excitement of travel is often tempered by concerns about the fare of airline

tickets.

Given the dynamic nature of flight ticket prices, comparing them from one day to the next can be challenging, making it difficult for travelers to make informed decisions. Recognizing this gap, the concept of forecasting flight ticket prices has emerged to simplify the ticket-buying process. In order to meet this demand, we used machine learning techniques to analyze a dataset of 10,683 records with 13 columns that included information on both local and international flights in India in 2019.

Our approach involves meticulous data cleaning to eliminate null values and errors, beginning with the examination of the 'Duration' column to understand the distribution of passengers in each category.

Subsequently, we perform Exploratory Data Analysis (EDA) to gain insights into the dataset. Our machine learning model's output is mostly described by feature engineering, and managing categorical data becomes critical during EDA. In the case of nominal and ordinal categorical data, we use methods like One Hot Encoding and Label Encoding, respectively.

One Hot Encoding is applied to the 'Airline' and 'Source' columns to compare the effect of various airlines and suppliers on ticket costs. We ensure data separation into training and test datasets to prevent data leakage during the machine learning process. Pre-processing involves importing and cleaning the data, paving the way for predicting ticket prices. Feature selection becomes crucial, guiding us to identify which features have the most significant connection to the intended variable ('Price'). We employ the Random Forest algorithm for our prediction model, dividing the data using scikit-learn. To enhance model performance, we delve into hyperparameter tuning through techniques like Randomized Search CV and Grid Search CV. The study explores three prominent design and learning methodologies, addressing challenges encountered in the project. Section 1 of the documentation presents the context through a literature review, and Section 2 delineates the suggested system design. The subsequent sections detail the architectural algorithm and procedure, offering a comprehensive overview of our approach.

LITERATURE SURVEY

Tiyani Wang [1] proposed using DB1B and T-100 datasets coupled with economic data to estimate price depending on marketing strategy. The framework gives a summary of its main elements, beginning with the data preparation phase, which involves cleaning datasets, excluding erroneous sample data, and merging data according to market categories. Then, each market segment's unique features are produced via the feature extraction module.

P.H.K. Tissera [2] presented the study's findings in the form of a web application developed using the hybrid web application development platform React Native. The program integrates two APIs, one written in Node.js and the other in Python Flask. Th. Kalampokas(4) and K. Tziridis(3) investigate the intricate methods airline companies employ to set their own prices for tickets. These tactics consider many commercial, sociological, economical, and marketing aspects that add together to affect the total cost of the travel. Due to the complexity of airline pricing mechanisms, customers often find it challenging to secure the best deal as prices frequently fluctuate.

G.A. Papakostas [5] highlighted recent strategies that focus on predicting the optimal moment for consumers to purchase airline tickets. These methods are based on sophisticated prediction models from the computational intelligence study field of machine learning.

A linear quantile hybrid regressor model was created by Janssen [6], who showed it could accurately estimate airline ticket prices many days in advance of travel. A study on the forecast of airline ticket prices was carried out by Ren, Yang, and Yuan [7]. They discovered that the models that they predicted worked well: LR (76.06% accuracy), NB (72.06% accuracy), SR (75.84% accuracy), and SVM (81.6% accuracy for two bins).

MODULAR DESIGN

➤ **Automated Script to Collect Historical Data**

- To tackle any prediction or classification task, we require access to past data. For this project, historical flight costs for every route gathered every day are required. Because it is inefficient to manually collect data every day, A remote server was used to run a Python script that collected prices at a certain time each day.

➤ **Cleaning and Preparing Data**

- Once the data is obtained, it has to be cleaned and ready according to the model's instructions. The most important and time-consuming phase of every machine learning assignment is this one. We incorporated a variety of statistical approaches and logic using pre-built R packages.

➤ **Analyzing and Building Models**

- After preparing the data, it is analyzed to find hidden trends and then the train is subjected to a variety of prediction and classification models.

➤ **Modelling**

- Next, we'll fit the image and project the outcome. In order to assess their effective decisions, we fit the provided data into several regression analyses.

➤ **Feature Engineering**

- In this case, we first split the attributes and names before converting the hours to minutes. For simpler preprocessing during the model phase, we are organizing the form of such a date of trip in our info in Date of Journey. The departure date is converted to clock time using Dep Time. Arrival Time converts from hours and minutes to minutes.

➤ **Feature Selection**

- A features extraction strategy is utilized to look into how much each piece of information influences the prediction outcome in order to improve model performance. Since the pricing part is no longer relevant, it has been deleted.

➤ **Data Collection**

- The Python script simplifies the pre-processing of data and outcomes forecasting. It prevents duplicate values in the preprocessing stage. This data collection is mostly concerned with calculating the value of aircraft fare.
- The data set contains the data with attributes such as
- Journey Date
- Departure
- Designation
- Arrival Time & Duration

➤ **Merging Models and Accuracy Calculation**

- Having erected colorful models, we've to test the models on testing set and calculate the savings or loss done on each query put by the stoner. A statistic of the over-savings, Loss, and the mean saving per sale are the measures used to calculate the delicacy of the model enforced.

PROPOSED SYSTEM

The proposed system made use of the Random Forest Algorithm, a versatile machine-learning method appropriate for both regression and classification applications. Based on the concepts of ensemble learning, it performs very well when working with big, highly dimensional datasets. This method may be used to determine the cheapest airline ticket even with large datasets. Using arbitrary subsets of the dataset, the approach builds many decision trees, resulting in a para estimator with extra-trees regressors. This ensemble method uses aggregation to reduce overfitting and improve prediction accuracy.

To visualize the relationships within the dataset, we utilized a heatmap - a graphical representation using colors to depict matrix values' concentration between two dimensions. This improves pattern detection and provides a sense of depth to the data.

In the hyperparameter tuning phase, we opted for a Randomized Search CV, resulting in a notable improvement in our score from 79% to 82%.

The suggested system's architecture is shown in Figure 1 A phase of data preparation, which involves removing data to improve efficiency, comes after data gathering. To generalize the training data to fresh, unknown data for predictions, we build a model before testing and training. The subsequent step involves training and testing the model to assess its accuracy. The dataset is divided into two sets - 81% for training and 19% for testing. After successful testing, the best-fitting model is determined to achieve. Finally, using machine learning algorithms, optimal accuracy. The system predicts the most cost-effective flight ticket price.

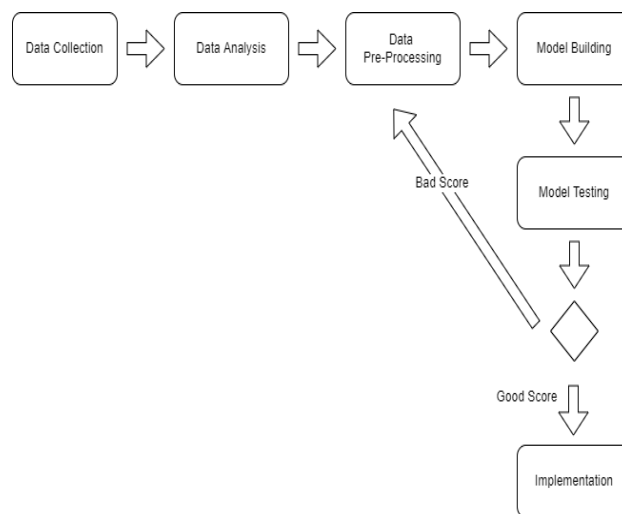


Fig. 1. Architecture of Proposed System

ALGORITHM

The steps in the functioning technique are as follows:

Step 1: From the training data set, select R observations at random.

Step 2: Produce bracket trees for the named sample points.

Step 3: To display the system's categorization trees, choose a number A.

Step 4: You'll repeat Steps 1 and 2.

Step 5: Give the samples to the class that scored the highest after determining the predictions for each decision tree's current data elements.

The following are the stages involved in building the Random Forest Algorithm:

1. Preparing the data
2. Using the Random Forest Algorithm on the training dataset
3. Projecting the study's outcome
4. Confusion Matrix Construction
5. Examining the potential outcome

Random Forest is an aggregate strategy that combines several decision trees with a process known as Bootstrap and Aggregation, or bagging, to handle both regression and classification issues.

The random forest training approach trains three learners using the conventional bootstrap aggregation mechanism, sometimes referred to as bagging. Using a training set $Z = z_1, \dots, z_n$ and responses $Y = y_1, \dots, y_n$, After selecting a random sample and replacing the training set (V times), trees are fitted based on the selected sample: Regarding $v = 1, \dots, V$:

Examine n distinct training situations from Z, Y with substitutions; mark them Z_v, Y_v .

2. Apply Z_v and Y_v to train a bracket or retrogression tree. After training, you may prognosticate values for unknown samples by adding up the prognostications from all of the retrogression trees on z_a

$$\hat{f} = \frac{1}{V} \sum_{v=1}^V f(\bar{z}) \quad (1)$$

In case of classification trees, the majority vote is utilized as an alternative.

Likewise, the query of the vaticination may be measured using the standard divagation of the prognostications made from each individual retrogression tree.

$$\sigma = \sqrt{\sum_{v=1}^V (f_v(\bar{Z}) - \bar{f})^2 / V - 1} \quad (2)$$

RESULTS



Fig. 2. Heat Map

Figure 2 represents a Heat map, where the distinctive attributes of a matrix are visually presented through a color- coded graphical representation. A heat map proves to be a valuable instrument for illustrating the distribution of values across two dimensions within a matrix. This visualization technique improves the detection of patterns and provides a sense of depth in understanding the data.

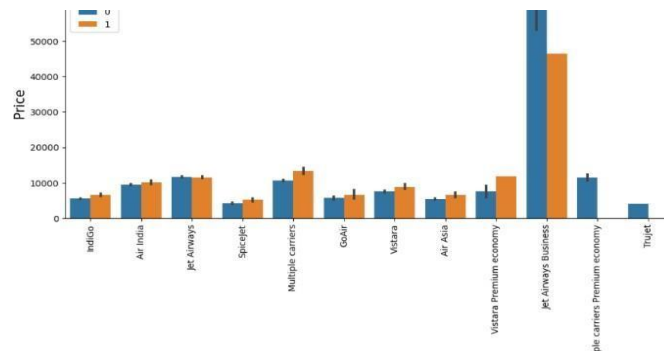


Fig. 3. Price Differences

In Figure 3, the presented fitting model showcases the variation in prices between weekdays and weekends. The data is transmitted to the `SNS display()`, offering insights into the differences observed in flight ticket prices between the two temporal categories.

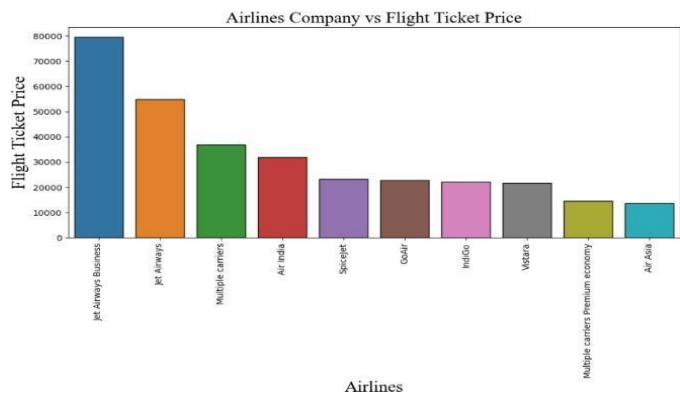


Fig. 4. Airlines Company vs Flight Ticket Price

In Figure 4, the showcased regression model illustrates a comparative analysis between different airlines and their respective flight ticket prices.

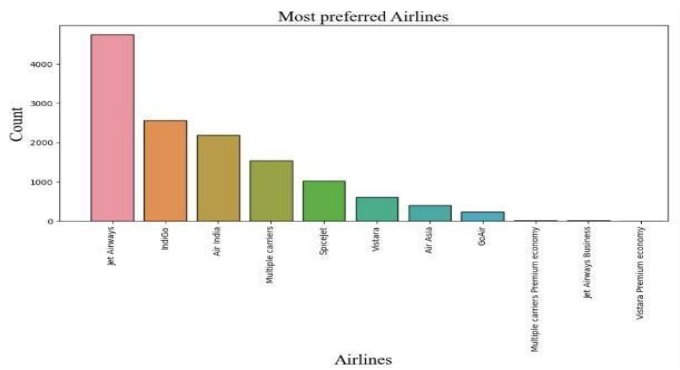


Fig. 5. Most Preferred Airlines

Figure 5 explains the Airlines and the count of each flight. It will be useful while they are booking the flight according to the user's choice

CONCLUSION

In the study, we concentrated on using machine learning algorithms to find most affordable ticket pricing. The literature analysis covered an overview of airline pricing plans that include periodic variations grounded on both internal and external factors, as well as ticket expectation and demand protuberance algorithms. We elucidated the dynamic nature of ticket costs determined through interactions between customers and airlines. Upon importing the dataset, we conducted exploratory data analysis to anticipate outcomes. Recognizing that machine learning deals with numerical data, we converted all text data types accordingly. Irrelevant columns were removed, and categorical data was appropriately handled. To ensure data integrity, we divided the dataset into training and test sets, preventing any potential data leakage. For Retrogression and Bracket tasks, ensemble literacy, specifically the Random Forest Regressor, was employed. The data underwent preprocessing, and hyperparameter tuning was executed to enhance the model's performance. Utilizing Randomized Search CV for hyperparameter tuning, we achieved a significant improvement in our score, elevating it from 79 to 81 percent.

REFERENCES

- [1] Boruah, Abhijit, et al., "A Bayesian approach for flight fare prediction based on Kalman filter," *Progress in Advanced Computing and Intelligent Engineering*. Springer, Singapore, 2019; 191-203.
- [2] Cheung, and C.H. Jacob "Flight planning: node-based trajectory prediction and turbulence avoidance," *Meteorological Applications*, 2018; 25.1: 78-85.
- [3] S. Dey, M.B. Purkayastha, and H.K. Saikia, "Prediction of Flight Fare Using Machine Learning Techniques," in *2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, 2019.
- [4] W. Groves, and M. Gini, "An agent for optimizing airline ticket purchasing," *12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021)*, St. Paul, MN, May 06 -10, 2013; 1341-1342.
- [5] Y. Horiguchi, Y. Baba, H. Kashima, M. Suzuki, H. Kayahara, and J. Maeno, "Predicting fuel consumption and flight delays for low-cost airlines," In *Twenty-Ninth IAAI Conference*, February 2017.
- [6] S.M. Iqbal, M. Rehman, and M. Riaz, "Predicting Domestic Air Fare Using Machine Learning Techniques," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.
- [7] T. Janssen, "A linear quantile mixed regression model for prediction of airline ticket prices," 2022.
- [8] A.V. Jain, A.S. Raval, and R.K. Oza, "Airfare price prediction based on reviews using machine learning techniques," *March 2020*; 9(3).
- [9] V. Kimbhaune, H. Donga, A. Trivedi, S. Mahajan, and V. Mahajan, *Flight Fare Prediction System*, 2021; 5542.

- [10] L. Sharma, (Ed.) Towards Smart World, Chapman and Hall/CRC, New York, 2021. doi: 10.1201/9781003056751.
- [11] Swarali M. Pathak, and Archana K. Chaudhari "Comparison of Machine Learning Algorithms for House Price Prediction using Real-Time Data," International Journal of Engineering Research & Technology (IJERT). December 2021; 10(12).
- [12] J. Santos Dominguez-Mencherro, Javier Rivera, and Emilio Torres Manzanera, "Optimal purchase timing in the airline market," 2019.
- [13] Supriya Rajankar, Neha Sakharkar, and Omprakar Rajankar, "Predicting the price of a flight ticket with the use of Machine Learning algorithms," International Journal of Scientific & Technology Research, December 2019. 8.
- [14] Swayze, and C. Neal, et al. "Influence of UAS Flight Altitude and Speed on above ground Biomass Prediction," Remote Sensing, 2022; 14.9.
- [15] K. Shah, V. Kedia, and R.K. Jha, "Integrated Vendor-Managed Time Efficient Application to Production of Inventory Systems," 2021 6th International Conference on Inventive Computation Technologies (ICICT), 2021; 275280, doi:10.1109/ICICT50816. 2021.9358504.
- [16] Sharma, and P. Garg, (Eds.) From Visual Surveillance to Internet of Things, Chapman and Hall/CRC, New York, 2020.
- [17] K. Tziridis, T. Kalampokas, G. Papakostas, and K. Diamantaras, "Airfare price prediction using machine learning techniques," in European Signal Processing Conference (EUSIPCO), DOI: 10.23919/EUSIPCO .2017.8081365L.
- [18] Tianyi Wang, Samira Pouyanfar, Haiman Tian, Yudong Tao, and Miguel Alonso Jr., Steven Luis, and Shu-Ching Chen, "A Framework for Airfare Price Prediction: A Machine Learning Approach," 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI), September 9, 2019.
- [19] Vandehzad, and Mashhood, "Efficient flight schedules with utilizing Machine Learning prediction algorithms," 2020.
- [20] Vinod Kimbhaune, Harshil Donga, Ashutosh Trivedi, Sonam Mahajan, and Viraj Mahajan, "Research paper on flight fare prediction system," 2021.
- [21] William Groves and Maria Gini, "An agent for optimizing airline ticket purchasing," In Proceedings of the 2018 international conference on autonomous agents and multi-agent systems.
- [22] Wohlfarth, T. Clemencon, and S. Roueff, "A Data mining approach to travel price forecasting," 10th international conference on machine learning Honolulu, 2021.
- [23] R.I. Yunita, B.F.S. Lubis, and R. Elvandari, "Predicting Flight Ticket Price Using Decision Tree and Artificial Neural Network," in 2019 6th International Conference on Information Technology and Electrical Engineering (ICITEE), 2019.
- [24] Zhang, Qianqian, Yuan He, and Xiaojun Jing. "Mask Neural Network for Predicting Flight Ticket Price," Signal and Information Processing, Networking, and Computers. Springer, Singapore, 2020.