A

PROJECT REPORT ON

FLIGHT FARE PREDICTION SYSTEM USING MACHINE LEARNING ALGORITHMS

SUBMITTED BY
SONY SINHA

UNDER GUIDANCE OF
Prof. Mr. Mohammad Ali

At

iNeuron Intelligence Private Limited

TO

SAVITRIBAI PHULE PUNE UNIVERSITY

In

Partial Fulfilment of Master of Business Administration in

Information Technology (MBA-IT)

AKI's

Poona Institute of Management Sciences & Entrepreneurship

K. B. Hidayatullah Road, Camp, PUNE-41101

Academic Year 2022-2023

ACKNOWLEDGEMENTS

I find great pleasure in expressing my deep sense of gratitude towards Code Clause Software company at Pune, my mentors, my co-interns, and all those who have made helped me to complete this Project successfully.

I am greatly thankful to our Director (In Charge), Dr. Porinita Banerjee, Poona Institute of Management Sciences and Entrepreneurship, Pune.

I would like to thanks Prof. Mohammad Ali, my Project Guide who provided fantastic project coordination, excellent ideas and suggestions and an unending patience as we worked together to produce an excellent thesis work.

I am greatly thankful to Staff of Poona Institute of Management Sciences and Entrepreneurship, Pune for helping through the entire Course.

I further want to extend my thanks to everyone who has been helpful.

I am also thanking the IT department, student colleagues, laboratory assistants, and all who stood with me in my entire course at PIMSE & wished me well.

DECLARATION

I Sony Sinha hereby declare that I have completed this project entitled

as "**FLIGHT FARE PREDICTION USING MACHINE LEARNING ALGORITHM**" for the academic year 2022-2023

The information submitted is true and original to the best of my knowledge.

Date:

Place: (Signature of the Student)

INDEX

CHAPTER-1

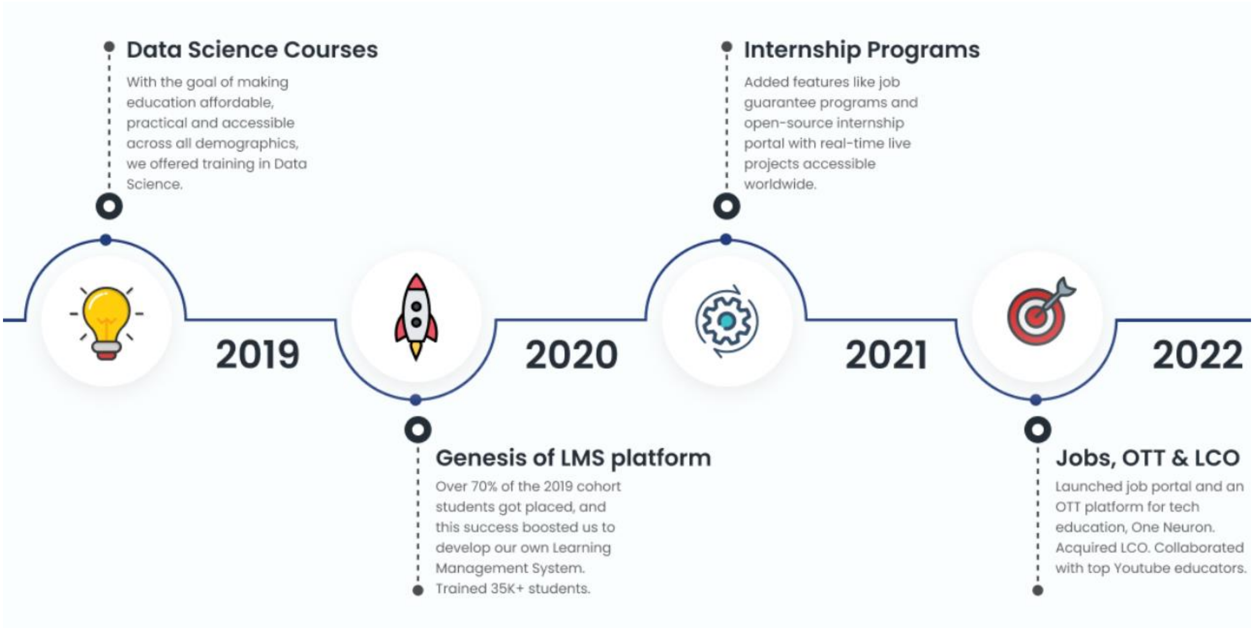
INTRODUCTION

INTRODUCTION

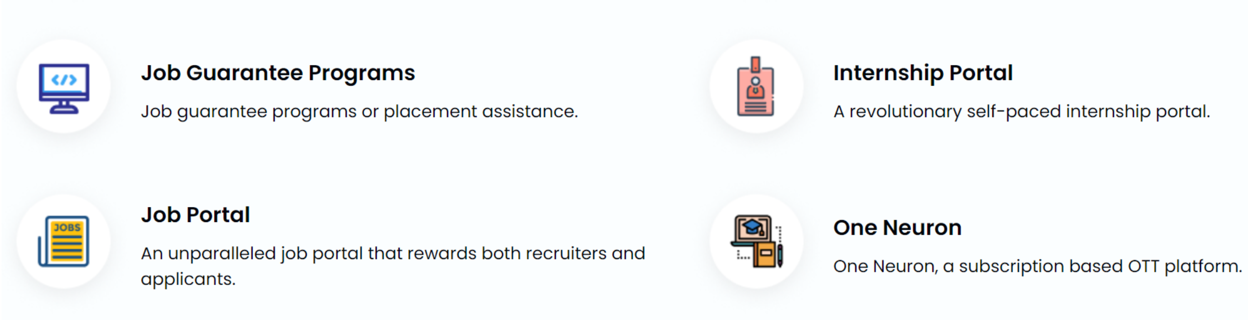
1.1 Company Profile

iNeuron started as a product development company, then launched its ed-tech division. We provide 360-degree solutions from learning to internship to finding a job, and the first ever educational OTT platform to upgrade your skill test.

Company Timelines



Company Services



Mission: Our goal is to make education and experiential skills affordable and accessible to everyone regardless of their disparate economic and educational backgrounds. We empower students to make demands unlike any other platform or institute because curiosity cannot be contained. Learning cannot be boxed in a book. So, let us step ahead and 'build together'.

DEPLOYMENT LINK:

https://flight-fare-pridiction3.onrender.com/predict

1.2 EXISTING SYSTEM AND NEED FOR SYSTEM

As domestic air travel is getting more and more popular these days in India with various air ticket booking channels coming up online, travelers are trying to understand how these airline companies make decisions regarding ticket prices over time. Nowadays, airline corporations are using complex strategies and methods to assign airfare prices in a dynamic fashion. These strategies are taking into consideration several financial, marketing, commercial and social factors are closely connected with the ultimate airfare prices. Due to the high complexity of the pricing models applied by the airlines, it is very difficult for a customer to purchase an air ticket at the lowest price, since the price changes dynamically. For this reason, several techniques ready to provide the proper time to the customer to buy an air ticket by predicting the airfare price, are proposed recently.

Most of those methods are making use of sophisticated prediction models from the computational intelligence research field known as Machine Learning (ML). In this machine learning in python project there is only one module namely, User. User can login with valid credentials in order to access the web application. A travelers can access this module to get the future price prediction of individual airlines. The prediction will help a traveler to decide a specific airline as per his/her budget. Single entries of current or previous data can be made. This training set is used to train the algorithm for accurate predictions.

1.3 SCOPE OF SYSTEM

The scope of the project is to form an application that predicts the price of the flight price tag by taking bound input from the user like date of journey, aboard location and destination and number of stops.

The work is projected to require the desired input of user from the created interface and method all the provided information to satisfy the wants of the machine learning model and at last show the output in the form of text message the generated predicted flight fare price.

The application will even predict the price of price tag considering whether is it a weekday, season, or alternative social reasons, however, considering from the angle of business, if we tend to method such information and predict the price of the discounted price tag it will bring some loss to the airlines company so therefore, this technique is not thought-about.

There are not any hardware needs needed for usage this application, the user should have interactive device that has access to the web and user should have the fundamental understanding of providing the input. Thus, the application has user friendly interface, which can be operated and managed by anyone having basic knowledge to input details as required by user

1.4 OPERATING ENVIRONMENT - HARDWARE AND SOFTWARE SPECIFICATIONS

1. **HARDWARE SPECIFICATIONS**	
Client Side	
RAM	Minimum 8GB and above
HARD DISK	Minimum 500 GB and above
PROCESSOR	Intel I3 and above
KEYBOARD	
Server Side	
RAM	16 GB and above
HARD DISK	512 GB and above
PROCESSOR	Intel I3 processor and above

2. **SOFTWARE SPECIFICATION**	
Client Side	
Operating System	Windows 10 and above
Web Browser	Google Chrome/Internet Explorer version 8.0 or higher
Server Side	
Operating System	Windows 10 and above
Database Used	Cassandra
Language Used	Python Programming Language, HTML, CSS
Editor Used for Development	Jupyter Notebook, PyCharm
Tools and Technology	Machine Learning, Flask, postman, render, GitHub
Libraries used	Pandas, Numpy, Seaborn, matplotlib, Sklearn, LinearRegression, train_test_split, RandomForestRegressor, DecisionTreeRegressor, RandomizedSearchCV

1.5 DETAIL DESCRIPTION OF TECHNOLOGY USED

The project solution proposed to take the required input of user from the created interface and process all the provided data to meet the requirements of the machine learning model and finally display the output saying so and so amount is the predicted cost.

The Technology used in the project is Data Science (Machine Learning Algorithms) to predict the fare flight based on the historical data.

There are no hardware requirements required for using this application, the user must have an interactive device which has access to the internet and must have the basic understanding of providing the input. And for the backend part the server must run all the software that is required for the processing the provided data and to display the results.

- Python 3.9 is used as the programming language and frame works like Numpy, pandas, Sklearn and other modules for building the model.
- PyCharm and Jupyter Notebook are used as IDE.
- For visualizations seaborn and parts of matplotlib are being used.
- For data collection Cassandra database is being used.
- Front end development is done using HTML/CSS.
- Flask is used for both data and backend deployment.
- GitHub is used for version control.

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation via the off-side rule. Python is dynamically typed and garbage-collected.

Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience.

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development.

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require tools or libraries.

Cassandra is a free and open-source, distributed, wide-column store, NoSQL database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure.

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML.

The Hypertext Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It defines the meaning and structure of web content. It is often assisted by technologies such as Cascading Style Sheets and scripting languages.

Render makes the deploying the application as easy as pushing the code to source control. We need to connect the GitHub or GitLab account to the Render account, Render will automatically build and deploy the services with every push.

GitHub, Inc. is a platform and cloud-based service for software development and version control using Git, allowing developers to store and manage their code.

Random Forest is a method of artificial intelligence, which is an integrated learning method that uses bagging algorithm combined with many decision trees for classification, regression, or other tasks. Random forest is a forest composed of many decision trees, and the decision trees are independent of each other. Random forest, proposed by Leo Breiman, uses a bootstrap resampling technique to randomly extract k samples randomly from the original training sample set N to generate a new set of training samples. Then generate k classification trees to form random forest, and the classification result of new data depends on the classification tree voting.

CHAPTER 2

PROPOSED SYSTEM

2.1 PROPOSED METHODOLOGY

The framework involves data cleaning, pre-processing, and feature selection, the LinearRegression, train_test_split, Linear Regression, Lasso Regression, ElasticNet regression, Ridge Regression, RandomForestRegressor, DecisionTreeRegressor, RandomizedSearchCV.

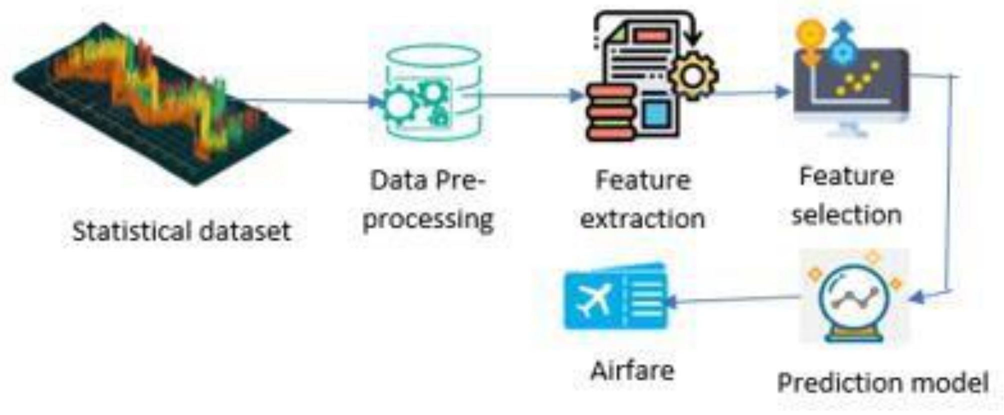


Fig. 1: Depicts the proposed framework for Flight Fare price prediction

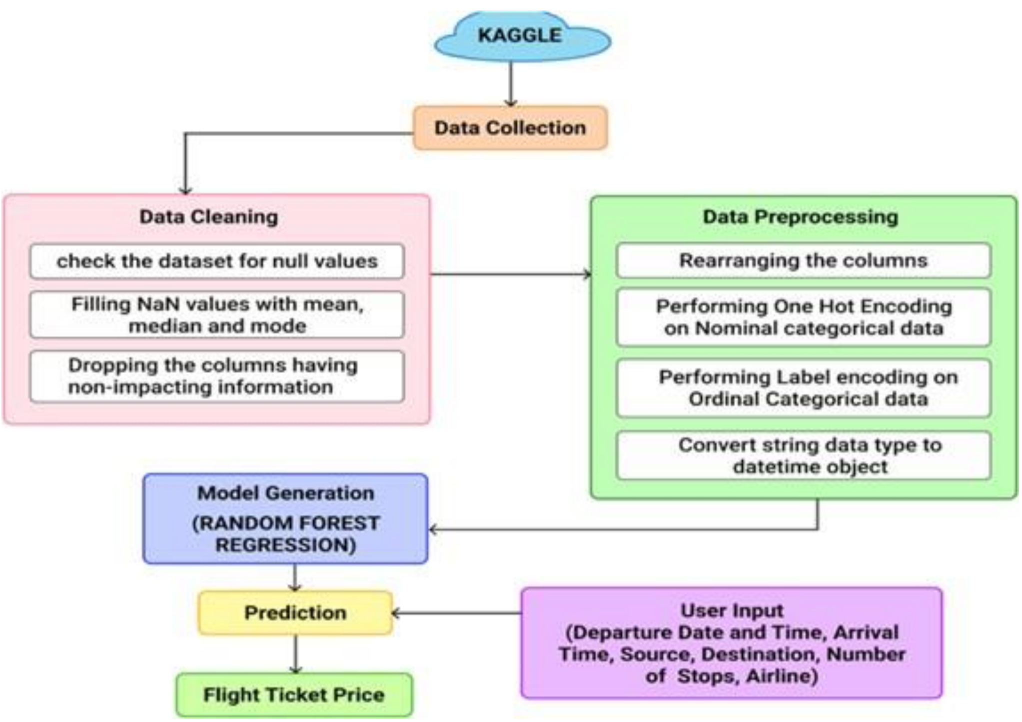


Fig. 2: Depicts the Block Diagram Methodology of the proposed system

Forecasting the price of an airline ticket is a very challenging task because many factors depend on the price of an airline ticket. Many researchers used various machine learning algorithms to obtain a model with higher prediction accuracy from the ticket price. Researchers use various regression models such as support vector machines (SVMs), Linear regression (LR), decision tree, random forests, etc. to predict the exact price of a flight.

There are various steps involved in building an ML model, starting with importing a dataset and cleaning the data. All null values and duplicate values are removed from the dataset. Then the data is encoded by conversion of some variables into a certain format. Converts categorical data to numeric data. After the dataset is processed, feature selection is performed. Properties or variables that are not so important are removed from the dataset. Exploratory data analysis is performed to provide insight and identify important features using the Extra Tress Regressor. Feature Engineering is performed to reduce computational cost and sometimes to improve accuracy. This is done using a correlation matrix. Then the data is split into train and test data, where the train data is used to train the models. The test data is used to check the accuracy of the models. Then it decides on the optimal features and parameters performing hyper tuning. After models are trained, their accuracy is checked using their R-squared value. This study is using Python3 to implement machine learning algorithms to create a model that will make predictions with high precision. Various python libraries are imported to perform these actions.

2.2 OBJECTIVE OF THE PROJECT

The primary objective of a Flight Fare Prediction System is to provide travelers with accurate and insightful predictions of flight ticket prices for a specific route and travel date. This system aims to assist users in making informed decisions about their travel plans by offering estimates of future flight fares. The main objectives of a Flight Fare Prediction System include:

Assist Travel Planning: The system aims to help travelers plan their trips more effectively by providing them with estimated flight prices well in advance. This allows users to explore various options and make informed decisions about when to book their flights.

Cost Savings: By offering fare predictions, the system enables users to identify potentially lower fare periods, helping them save money on their travel expenses.

Enhance User Experience: The system aims to improve the overall travel experience by providing users with a tool that simplifies the process of researching and comparing flight prices.

Time Efficiency: Travelers can quickly access fare estimates without the need to manually search multiple websites or wait for promotions. This saves users time and effort in finding the best flight deals.

Data-Driven Decisions: The system empowers users to make data-driven decisions by presenting historical and real-time flight data, trends, and patterns that influence fare fluctuations.

Flexibility in Planning: By offering fare estimates for different dates and times, the system allows users to adjust their travel plans based on potential fare variations.

Transparency: The system promotes transparency by helping users understand the factors affecting fare prices and the rationale behind the predictions.

Personalization: Users can receive fare predictions tailored to their preferences, such as specific airlines, travel classes, and routes.

Informative Insights: The system provides insights into how flight prices change over time, enabling users to anticipate fare trends and plan accordingly.

Competitive Advantage: For airlines and travel companies, implementing a fare prediction system can offer a competitive edge by attracting users who value cost-effective and well-informed travel choices.

Continuous Improvement: As users engage with the system, their interactions and feedback contribute to the continuous improvement of the prediction model, enhancing its accuracy over time.

Educational Value: The system can educate users about the factors that influence flight fares, helping them become more savvy travelers.

The Flight Fare Prediction System aims to provide users with a valuable tool that simplifies the process of finding and comparing flight prices, enhances their travel planning experience, and empowers them to make cost-effective decisions based on data-driven insights.

2.4 USER REQUIREMENTS

User requirements for a Flight Fare Prediction System are the specific needs and expectations of the users that the system must address in order to provide a valuable and effective experience. These requirements guide the design, development, and functionality of the system. Here are some user requirements for a Flight Fare Prediction System:

Accurate Fare Predictions:

Users expect the system to provide accurate and reliable predictions of flight fares based on historical data and relevant factors.

User-Friendly Interface:
The system should have an intuitive and easy-to-navigate interface that allows users to input travel details and retrieve fare predictions effortlessly.

Customizable Search:
Users should be able to customize their search based on departure and destination cities, travel dates, and other preferences.

Multiple Options:
The system should present users with multiple flight options and fare estimates for different airlines, times, and travel classes.

Flexible Date Exploration:
Users should be able to explore fare predictions for a range of dates to find the most cost-effective travel options.

Real-Time Data:
Users expect the system to provide up-to-date and real-time information on flight prices, considering current market conditions.

Clear Presentation of Results:
The system should display fare predictions in a clear and understandable manner, allowing users to compare options easily.

Advanced Filters:
Users should have the option to apply filters such as direct flights, layovers, specific airlines, and preferred travel classes.

Historical Data Insights:
Users might want to access historical fare data and trends to make informed decisions about travel timing.

Mobile Accessibility:
The system should be accessible and user-friendly on both desktop and mobile devices for users on the go.

Privacy and Security:
Users expect their personal data and search history to be handled securely and in accordance with privacy regulations.

Feedback Mechanism:
Users should be able to provide feedback on fare predictions and the system's performance, helping to improve its accuracy.

Educational Resources:
The system could provide explanations of fare prediction factors, helping users better understand the dynamics of flight pricing.
Integration with Booking:

Users might appreciate the option to seamlessly transition from fare predictions to booking flights through integration with booking platforms.

Availability and Reliability:
Users expect the system to be available 24/7 and to provide consistent and reliable fare predictions.

Responsive Customer Support:
Users should have access to responsive and helpful customer support for inquiries and assistance.

Understanding and addressing these user requirements plays a crucial role in designing and developing a Flight Fare Prediction System that meets users' needs and provides a valuable service.

CHAPTER: 3

ANALYSIS AND DESIGN

3.1 METHODOLOGY DESIGN

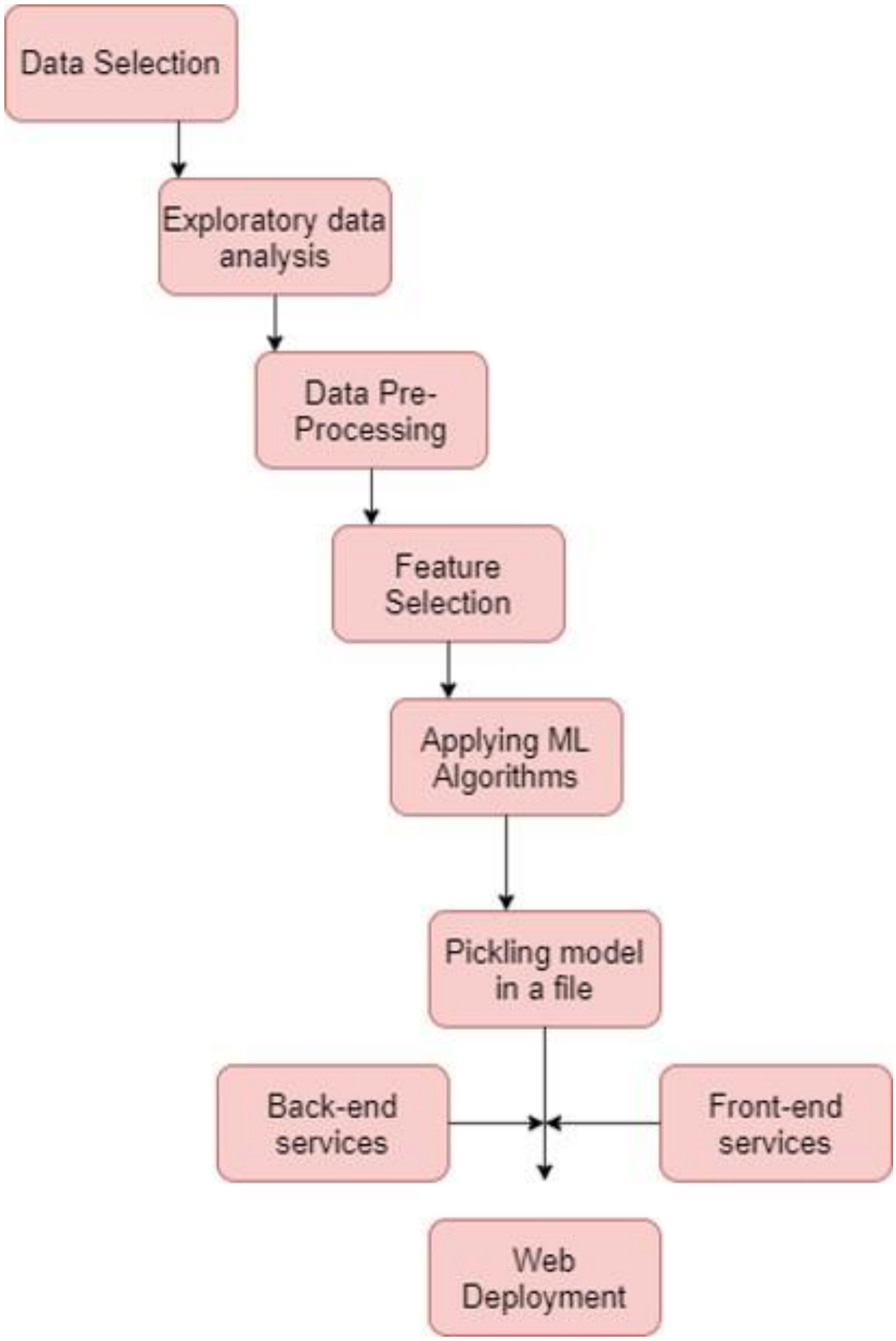
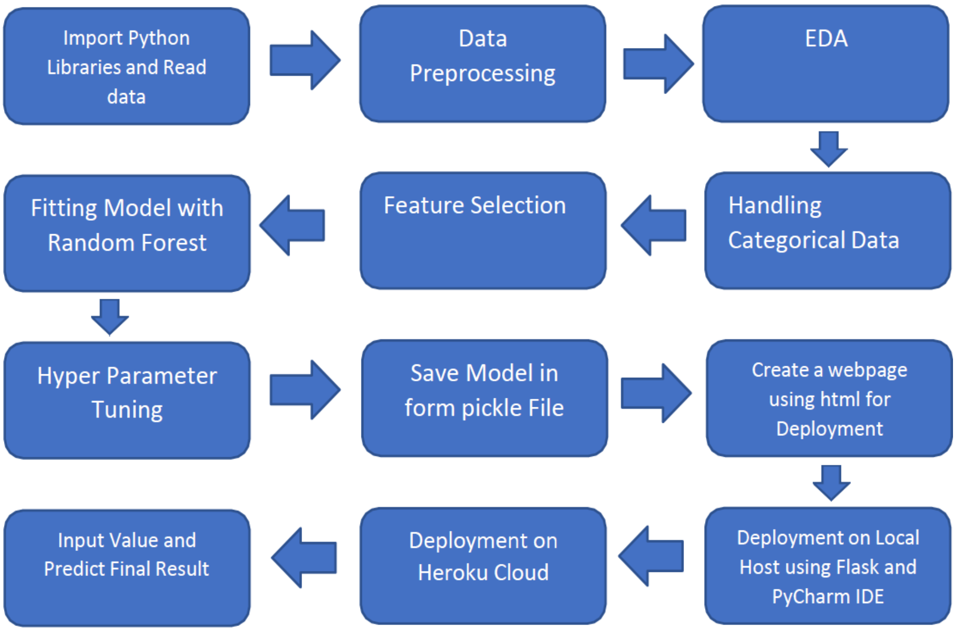


Fig 3: System methodology of Flight Fare Prediction

3.2 ARCHITECTURE DIAGRAM



3.2 ER DIAGRAM

Relationships:

Each Flight has one Departure Date (Many-to-One between Flight and Departure).

Each Flight has one Arrival Date (Many-to-One between Flight and Arrival).

Each Flight has one Departure City (Many-to-One between Flight and DepartureCity).

Each Flight has one Arrival City (Many-to-One between Flight and ArrivalCity).

One Airline can operate Many Flights (One-to-Many between Airline and Flight).

Each Flight can have Many Stoppages (One-to-Many between Flight and Stoppage).

This ER diagram captures the entities and relationships related to departure date, arrival date, airline name, departure city name, arrival city name, and the number of stoppages for a flight fare prediction system.

The Entity-Relationship diagram and approach is easy to understand, powerful to model real-world problems and readily translated into a database schema.

An ER diagram will help to make a system work well, how the software works behind the scenes and where all the data that goes in and out of the system will be stored.

3.2 ER DIAGRAM

```
+--------------+        +-------------+        +--------------+
|   Departure  |        |    Arrival  |        |    Airline   |
+--------------+        +-------------+        +--------------+
| Departure_ID |1      *| Arrival_ID  |        | Airline_ID   |
| Date         |--------| Date        |1      *| Name         |
|              |        |             |--------| ...          |
+--------------+        +-------------+        +--------------+
                                  |   1
                                  |
                                  |
+--------------+        +-------------+        +--------------+
| DepartureCity|        | ArrivalCity |        |  Flight      |
+--------------+        +-------------+        +--------------+
| City_ID      |1      *| City_ID     |        | Flight_ID    |
| Name         |--------| Name        |1      *| Departure    |
|              |        |             |--------| Arrival      |
+--------------+        +-------------+        | Departure_DT |
                                               | Arrival_DT   |
                                               | Airline_ID   |
                                               | ...          |
                                               +--------------+
                                                     |   1
                                                     |
                                                     |*
                                               +--------------+
                                               | Stoppage     |
                                               +--------------+
                                               | Stoppage_ID  |
                                               | Flight_ID    |
                                               | Number       |
                                               +--------------+
```

Entities and Attributes:

Departure: Departure_ID (Primary Key)

Date Arrival: Arrival_ID (Primary Key)

Date DepartureCity:

City_ID (Primary Key)

Name ArrivalCity:

City_ID (Primary Key)

Name Airline:

Airline_ID (Primary Key)

Flight: Flight_ID (Primary Key)

Departure (Foreign Key referencing Departure)

Arrival (Foreign Key referencing Arrival)

Departure_DT , Arrival_DT

Airline_ID (Foreign Key referencing Airline)

Other attributes related to the flight (e.g., flight number).

Stoppage_ID (Primary Key)

Flight_ID (Foreign Key referencing Flight)

Number (Number of stoppages for the flight)

3.3 CLASS DIAGRAM

Flight:

Attributes: flightNumber, ,departureCity,, arrivalCity, departureDate,

FlightDatabase:
Methods: searchFlights()
Contains a list of Flight objects and provides methods to interact with flight data, like searching, adding, removing flights, and retrieving a flight by its number.
.
FareCalculator:
Methods: calculateFare(): Provides a method to calculate the final fare.

FarePrediction:
Attributes: predictionModel.
Relationships: Uses a PredictionModel class, is trained by a ModelTrainer class.
Represents the prediction component for fare prediction. It contains a prediction model and can use this model to predict fares.

PredictionModel:

Methods: trainModel().
Represents the machine learning model used for fare prediction. It can be trained using the trainModel method.

ModelTrainer:
Methods: trainModel().
Represents the component responsible for training the prediction model. It is associated with the FarePrediction class and provides the trainModel method.

These classes and relationships together form the core structure of a Flight Fare Prediction System. Passengers are associated with flights, flights are managed by the FlightDatabase, and fare prediction is facilitated through the FarePrediction class using a trained PredictionModel. The ModelTrainer class is responsible for training the prediction model used by the FarePrediction component.

```
---------------------        ---------------------
|      Flight       |        |   FarePrediction  |
---------------------        ---------------------
| - flightNumber:   |        | - predictionModel:|
|   String          |        |     PredictionModel|
| - departureCity:  |        |                   |
|   String          |        | + trainModel():   |
| - arrivalCity:    |        |   void            |
|   String          |        | + predictFare():  |
| - departureDate:  |        |   double          |
|   Date            |        |                   |
| - passengers:     |        |                   |
|   List<Passenger> |        ---------------------
| - aircraft:       |
|   Aircraft        |
---------------------          |               |
        |                      | has a         | is a
        |                      |               |
        v                      v               v
--------------------      -------------    ---------------
|   FlightDatabase  |     | Prediction |   |  ModelTrainer |
--------------------      -------------    ---------------
| + searchFlights(): |        |
|   List<Flight>     |<>-------|
| + addFlight():     |        |
| + removeFlight():  |        |
| + getFlightByNumber|        |
|   String           |        |
--------------------          |
                              |
---------------------         |
|     Passenger     |         |
---------------------         |
| - passengerName:  |         |
|   String          |         |
| - age: int        |         |
| - email: String   |         |
| - bookingReference:|        |
|   String          |         |
---------------------         |
        |                     |
        | takes               |
        |                     |
        v                     |
---------------------         |
|   FareCalculator  |         |
---------------------         |
| - baseFare: double |        |
| - taxPercentage:  |         |
|   int             |         |
| + calculateFare(): |       |
|   double          |        |
---------------------        |
             |               |
           -----------------
           |  FarePrediction  |
           -----------------
           | - predictionModel|
           |   PredictionModel|
           | + trainModel():  |
           |                  |
           | + predictFare(): |
           |   double         |
            -----------------
```

3.4 USE CASE DIAGRAM

```
          User
           |
           v
+-------------------------+
| Flight Fare Prediction  |
|       System            |
+-------------------------+
           |
           v
+-------------------------+
|    Use Case:            |
|    Predict Fare         |
|                         |
| Description:            |
| - User requests         |
|   fare prediction       |
|                         |
| Actors:                 |
| - User                  |
|                         |
| Pre-conditions:         |
| - User selects a        |
|   flight route          |
|                         |
| Post-conditions:        |
| - Fare estimate         |
|   displayed to user     |
|                         |
| Main Flow:              |
| 1. User selects flight  |
|    route                |
| 2. System prompts for   |
|    date of travel       |
| 3. User provides date   |
| 4. System calculates    |
|    fare estimate        |
| 5. System displays      |
|    fare estimate to     |
|    user                 |
+-------------------------+
```

Actors:

User: The primary actor who interacts with the Flight Fare Prediction System. The user wants to predict flight fares for a specific route and travel date.

Use Cases:

Predict Fare: This is the main use case of the system. It represents the process of predicting the fare for a selected flight route and travel date. It involves interactions between the user and the system.

Predict Fare: This use case involves the following steps:
The user selects a flight route for which they want to predict the fare.
The system prompts the user to provide the date of travel.
The user provides the date of travel.
The system calculates the fare estimate based on the selected route and travel date.

The system displays the calculated fare estimate to the user.
Pre-conditions:

User selects a flight route: Before initiating the fare prediction process, the user must choose a specific flight route for which they want to predict the fare.
Post-conditions:

Fare estimate displayed to user: After the system calculates the fare estimate, it displays the estimated fare amount to the user.
Main Flow:

This describes the sequence of interactions between the user and the system for the "Predict Fare" use case.
The user initiates the fare prediction process by selecting a flight route.
The system prompts the user to provide the date of travel.
The user enters the date of travel.
The system performs the fare calculation based on the selected route and travel date.
The system presents the calculated fare estimate to the user.

3.5 FLOWCHART

Start

User initiates the fare prediction process.

User Input

User selects departure and destination cities.

User provides travel dates and preferred options (e.g., direct flights, travel class).

Data Retrieval:

System collects historical flight data, including prices, routes, and other relevant factors.

External data sources may be accessed for real-time information.

Prediction Algorithm:

The system processes the data using the prediction algorithm.

Algorithm considers historical trends, demand, seasonal variations, and other variables.

Generate Predictions:

System generates fare predictions based on the algorithm's calculations.

Display Options:

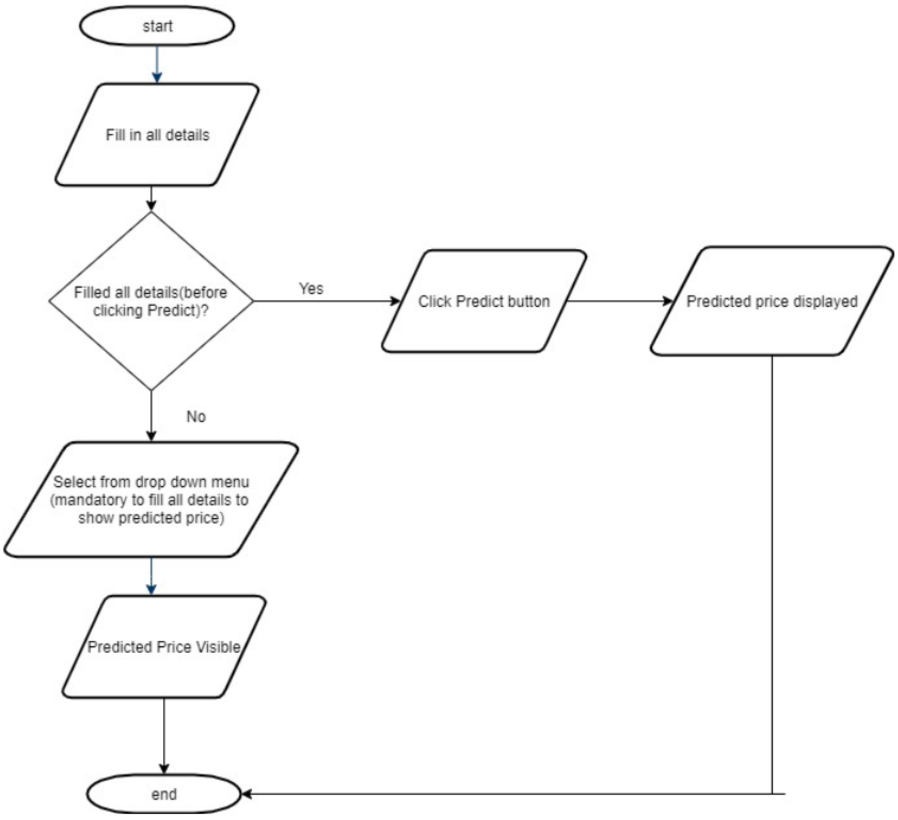
Predicted fares for different dates and times are displayed to the user.

Users can explore various options and compare predictions.

User Interaction:

User can modify search parameters and re-run predictions if needed.

User complete the fare prediction process.



3.6 ACTIVITY DIAGRAMS

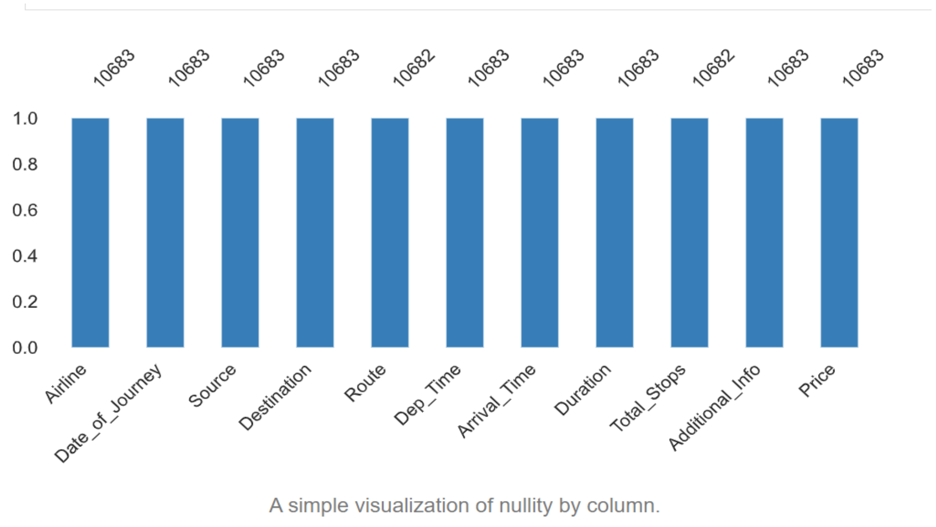


Fig 4: Count plot of each data attribute of dataset

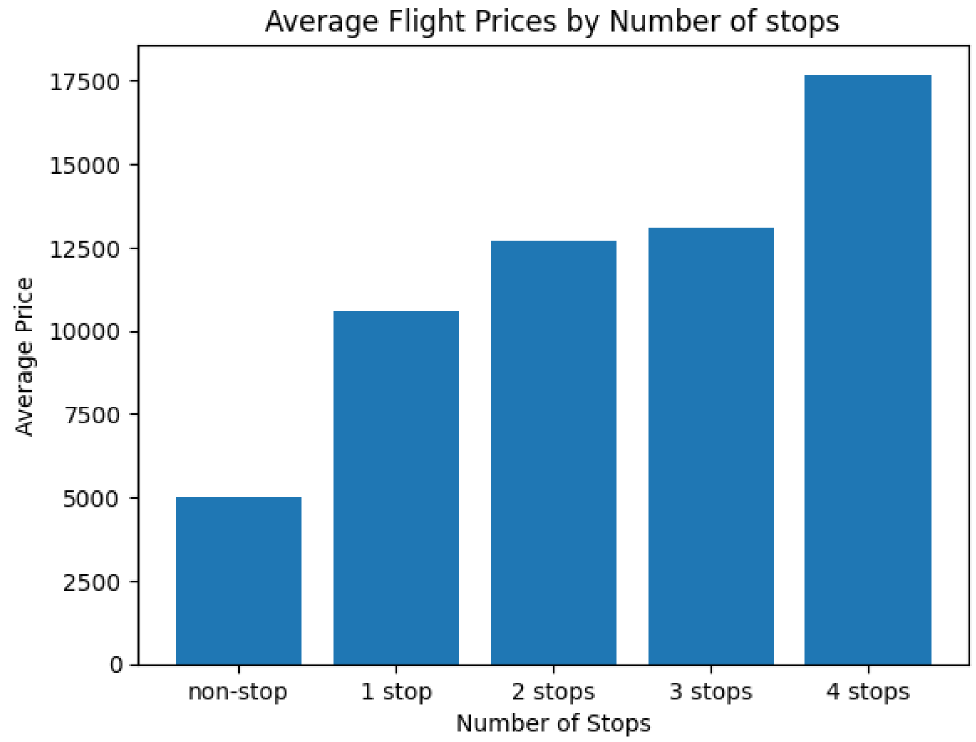


Fig 5: Histogram of Average Flight Prices vs Number of Stops

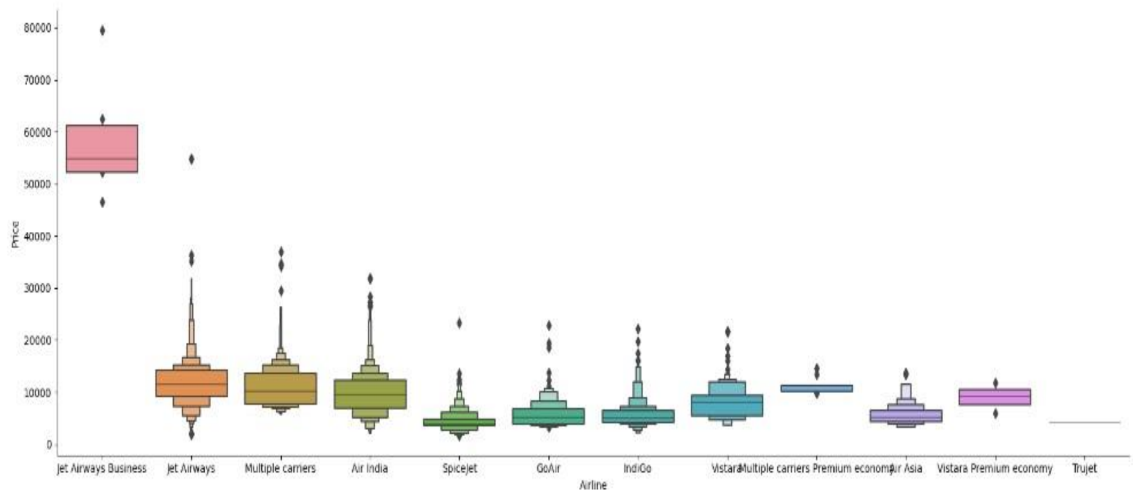


Fig 6: Cat plot Airline vs Price

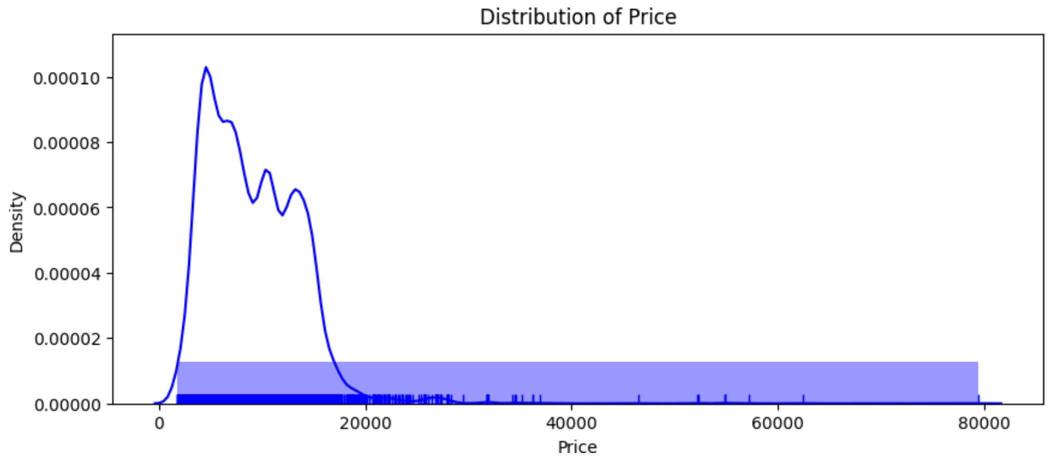


Fig 7: Graphical Distribution of Price

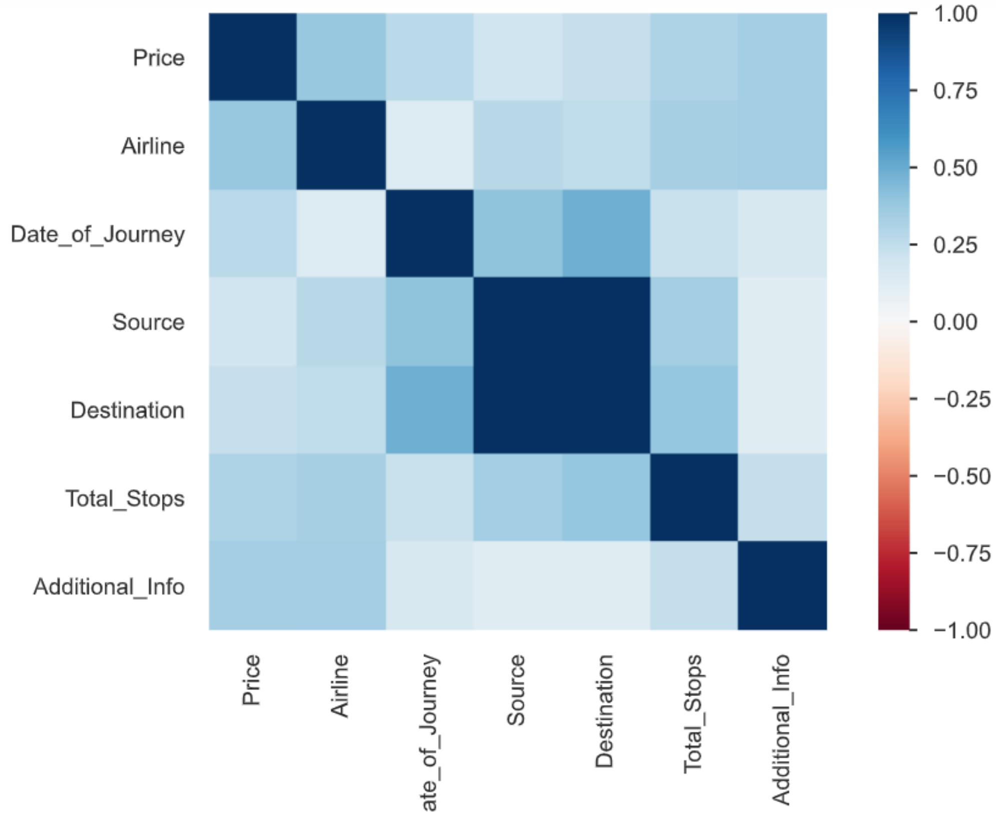
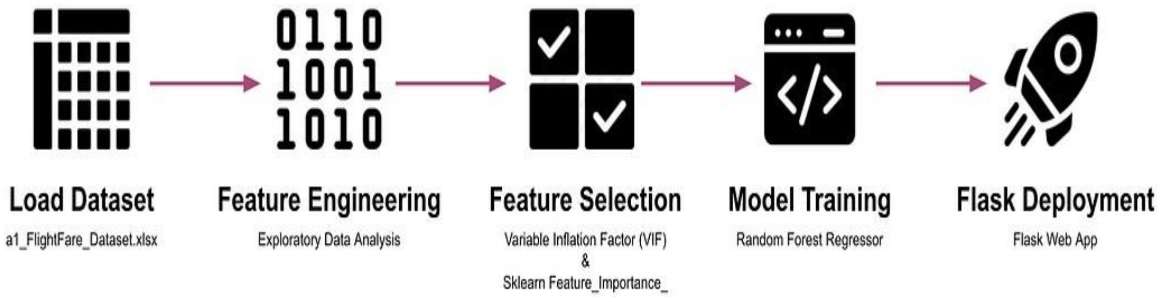


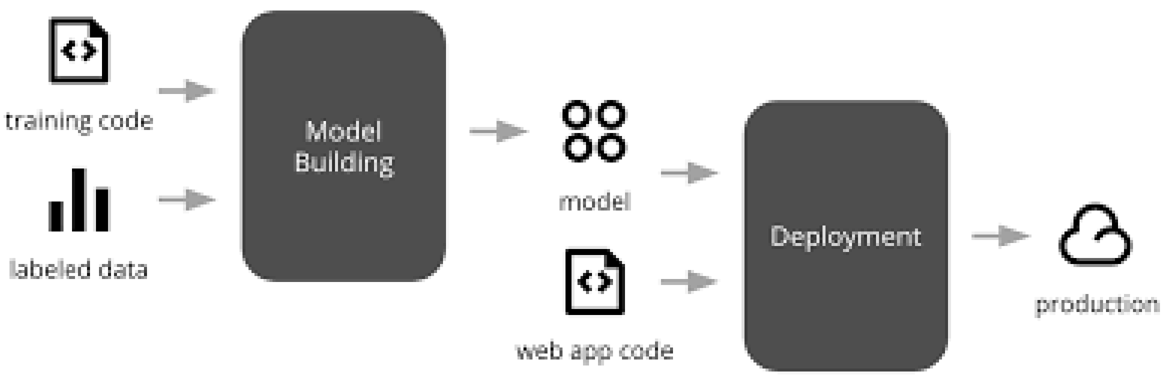
Fig 8: Heatmap of data attributes in dataset.

3.7 SEQUENCE DIAGRAM PROJECT FLOW

STAGE 1: Building Fare prediction Machine Learning Model



STAGE 2: Building a Web App and deploy it in live environment



3.8 User Interface Design (Screens)

Fig 9: Fetching input for Departure Date

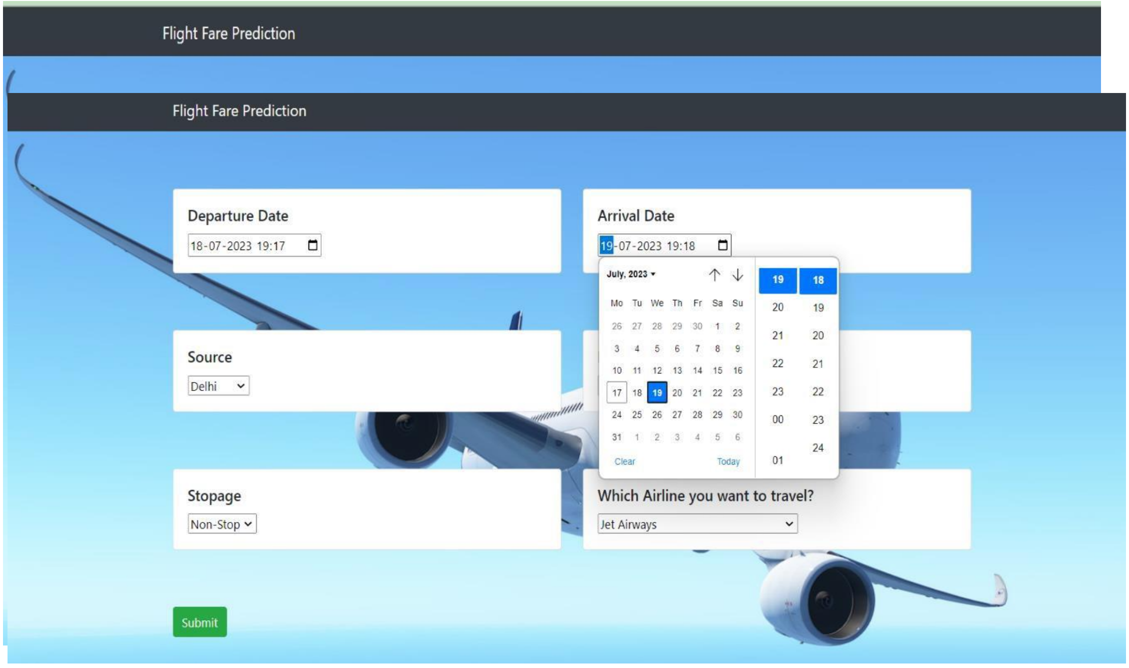


Fig 10: Fetching input for Arrival Date

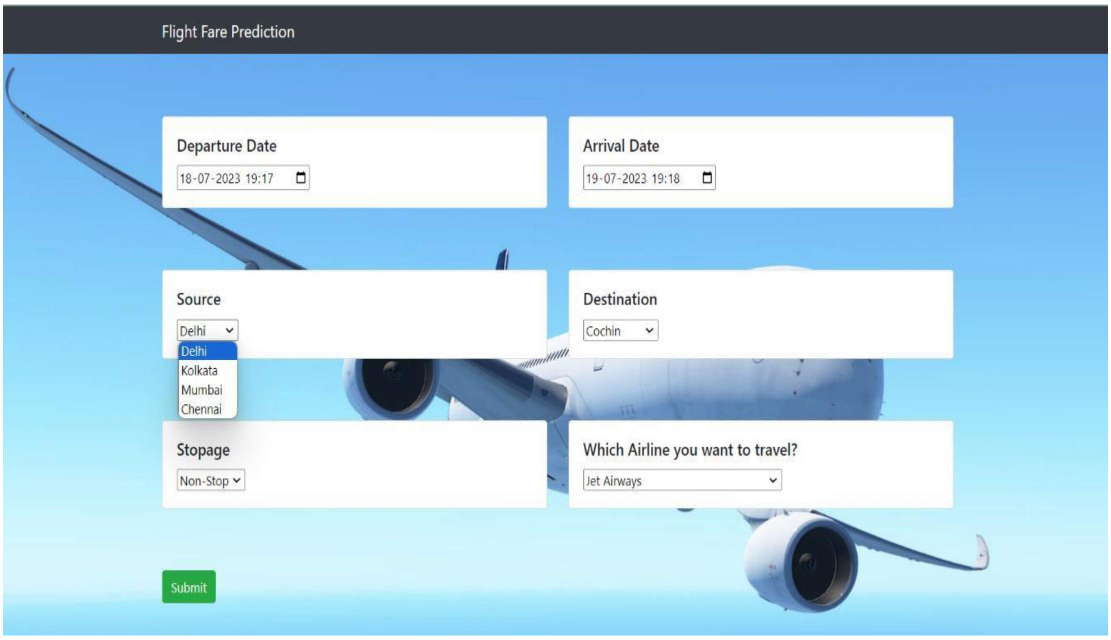


Fig 11: Fetching input for Source City

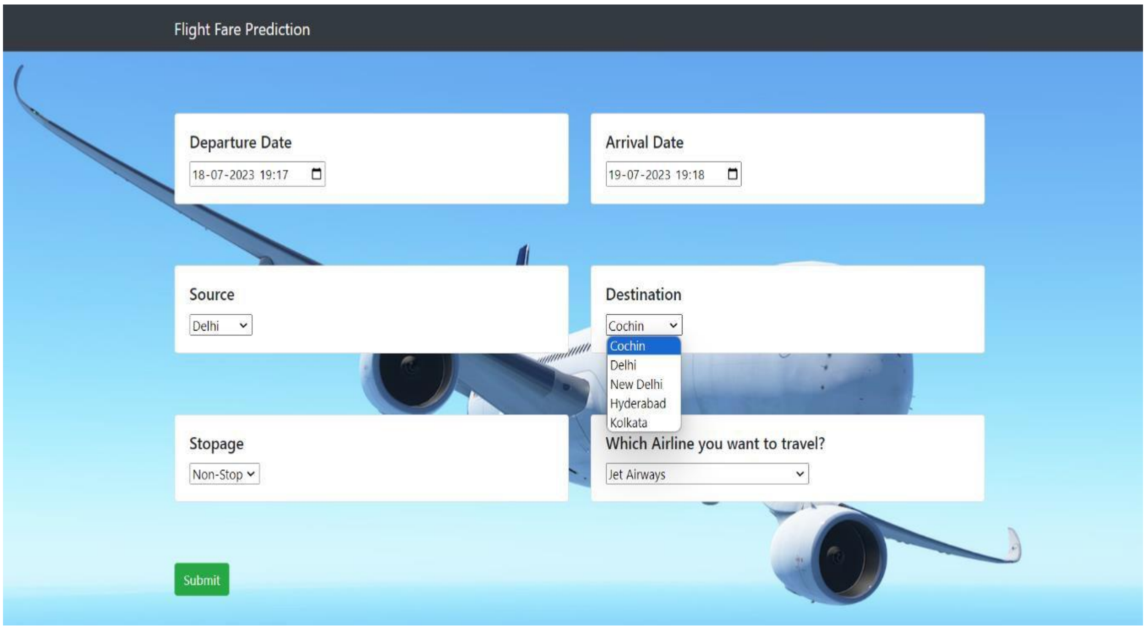


Fig 12: Fetching input for Destination City

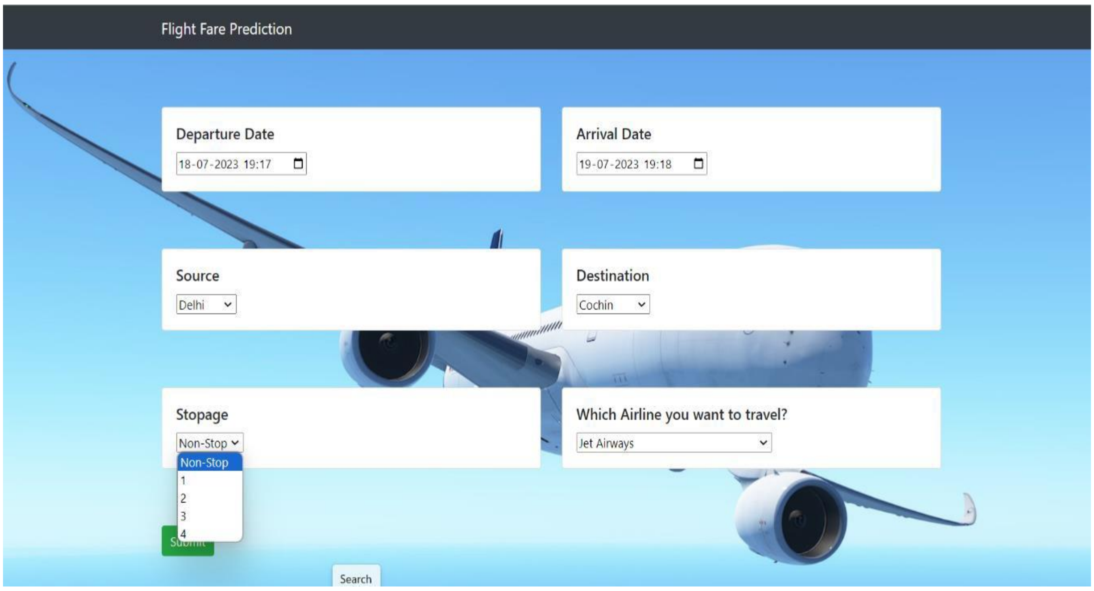


Fig 13: Fetching input for Stoppage Count

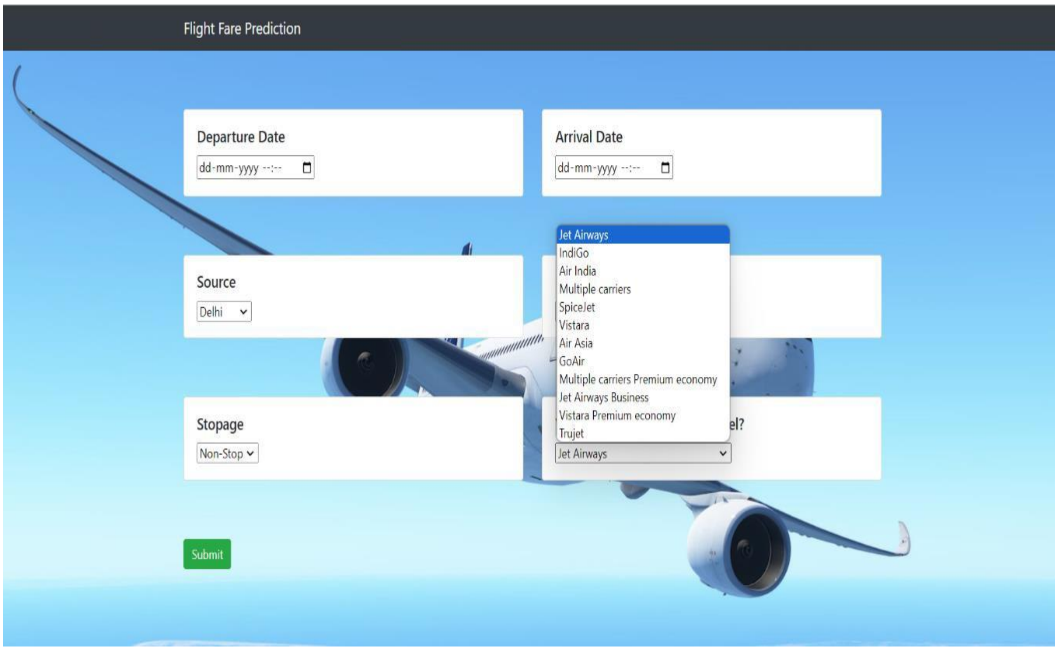


Fig 14: Fetching input for Airlines

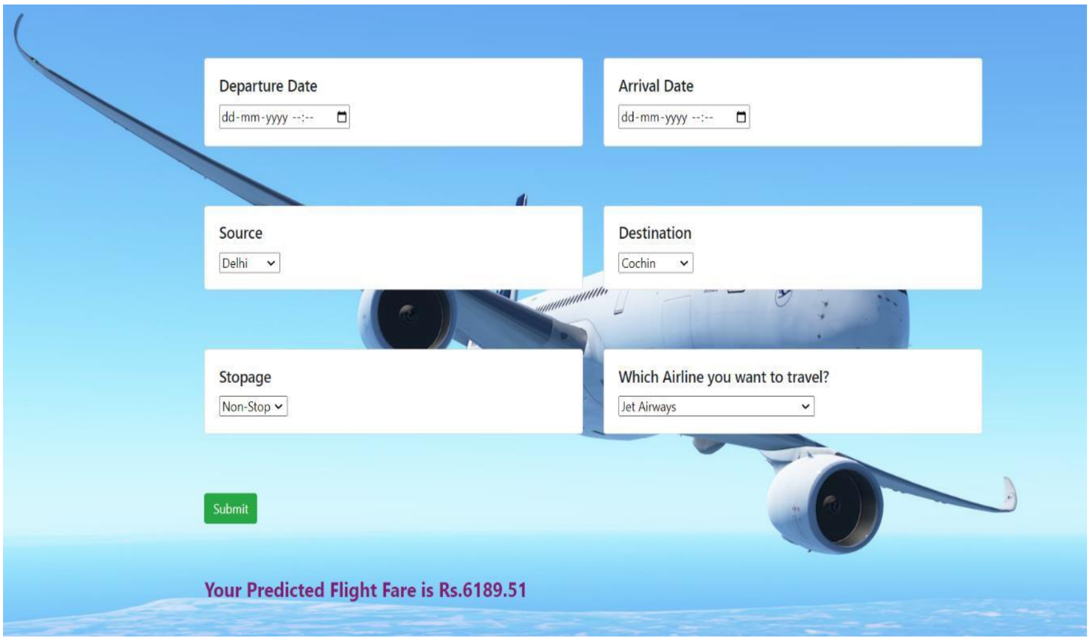


Fig 15: Displaying the output Predicted Flight Fare

3.9 DATA DICTIONARY

Data Collection: https://www.kaggle.com/datasets/nikhilmittal/flight-fare-prediction-mh

There are two datasets here: the training set and the test set

Attribute Nomenclature and Information

Range Index: 10683 entries, 0 to10682 Data columns (total 11 columns):

---- ---------- ----

#	Column	Non-Null Count Dtype
0	Airline	10683 non-null object
1	Date_of_Journey	10683 non-null object
2	Source	10683 non-null object
3	Destination	10683 non-null object
4	Route	10682 non-null object
5	Dep_Time	10683 non-null object
6	Arrival Time	10683 non-null object
7	Duration	10683 non-null object
8	Total Stops	10682 non-null object
9	Price	10683 non null int64

dtypes: int64(1),

DATA DESCRIPTION

There are about 10000 records of flight information such as airlines, data of journey, source, destination, departure time, arrival time, duration, total stops, additional information, and price. A glance of the dataset is shown below.

1	Airline	e_of_Jour	Source	Destinatior	Route	Dep_Time	rrival_Tim	Duration	Total_Stop	ditional_Ir	Price
2	IndiGo	24/03/201	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 N	2h 50m	non-stop	No info	3897
3	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR	05:50	13:15	7h 25m	2 stops	No info	7662
4	Jet Airway	9/06/2019	Delhi	Cochin	DEL → LKC	09:25	04:25 10 Ju	19h	2 stops	No info	13882
5	IndiGo	12/05/201	Kolkata	Banglore	CCU → NA	18:05	23:30	5h 25m	1 stop	No info	6218
6	IndiGo	01/03/201	Banglore	New Delhi	BLR → NA	16:50	21:35	4h 45m	1 stop	No info	13302
7	SpiceJet	24/06/201	Kolkata	Banglore	CCU → BLI	09:00	11:25	2h 25m	non-stop	No info	3873
8	Jet Airway	12/03/201	Banglore	New Delhi	BLR → BOI	18:55	10:25 13 N	15h 30m	1 stop	In-flight m	11087
9	Jet Airway	01/03/201	Banglore	New Delhi	BLR → BOI	08:00	05:05 02 N	21h 5m	1 stop	No info	22270
10	Jet Airway	12/03/201	Banglore	New Delhi	BLR → BOI	08:55	10:25 13 N	25h 30m	1 stop	In-flight m	11087
11	Multiple c	27/05/201	Delhi	Cochin	DEL → BOI	11:25	19:15	7h 50m	1 stop	No info	8625
12	Air India	1/06/2019	Delhi	Cochin	DEL → BLR	09:45	23:00	13h 15m	1 stop	No info	8907
13	IndiGo	18/04/201	Kolkata	Banglore	CCU → BLI	20:20	22:55	2h 35m	non-stop	No info	4174
14	Air India	24/06/201	Chennai	Kolkata	MAA → CC	11:40	13:55	2h 15m	non-stop	No info	4667
15	Jet Airway	9/05/2019	Kolkata	Banglore	CCU → BO	21:10	09:20 10 N	12h 10m	1 stop	In-flight m	9663
16	IndiGo	24/04/201	Kolkata	Banglore	CCU → BLI	17:15	19:50	2h 35m	non-stop	No info	4804
17	Air India	3/03/2019	Delhi	Cochin	DEL → AM	16:40	19:15 04 N	26h 35m	2 stops	No info	14011
18	SpiceJet	15/04/201	Delhi	Cochin	DEL → PN	08:45	13:15	4h 30m	1 stop	No info	5830
19	Jet Airway	12/06/201	Delhi	Cochin	DEL → BOI	14:00	12:35 13 Ju	22h 35m	1 stop	In-flight m	10262

3.10 TEST PROCEDURES AND IMPLEMENTATION

Implementation

We implemented machine learning lifecycle for this project to create a basic web application that will predict flight prices using a machine learning algorithm with historical flight data using python libraries such as Pandas, NumPy, Matplotlib, seaborn and Sklearn.

Fig 16: Steps of Machine Learning Life Cycle

DATA PREPROCESSING

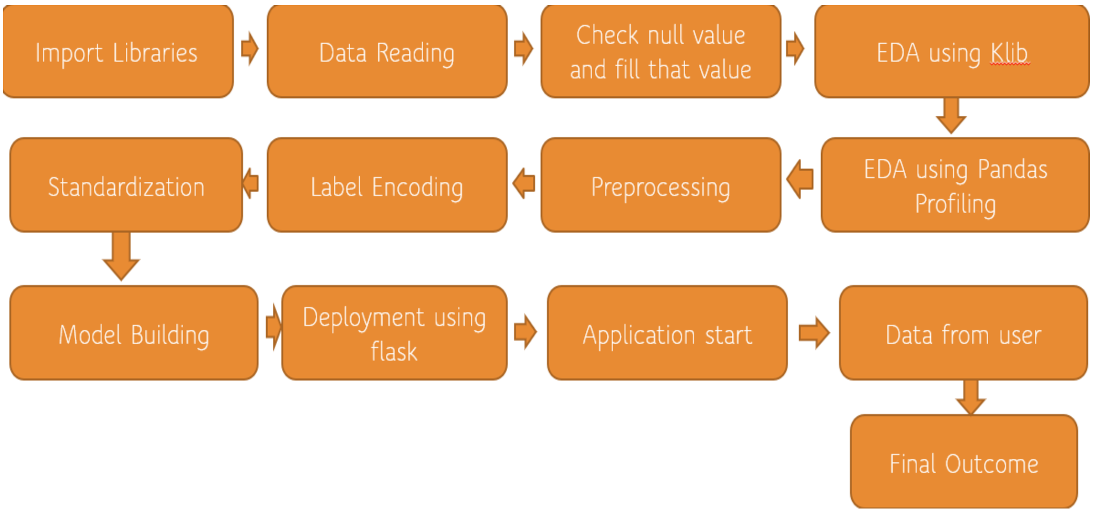
There is removal of duplicate and null values followed by data exploration to visualize statistical analysis by bar chart, frequency chart, cat plot and heatmap. Correlation of data attributes is checked and feature reduction in dataset is employed.

MODELLING

The pre-processed data is then visualized and all the required insights are being drawn. Although from the drawn insights, the data is randomly spread but still modelling is performed with different machine learning algorithms to make sure we cover all the possibilities. And finally, as expected random forest regression performed well and further hyperparameter tuning is done to increase the model's accuracy.

UI INTEGRATION

Both CSS and HTML files are being created and are being integrated with the created machine learning model. All the required files are then integrated to the main2.py file and tested locally.



Data from User

The data from the user is retrieved from the created HTML web page.

Data Validation

The data provided by the user is then being processed by app.py file and validated. The validated data is then sent for the prediction.

Rendering Result

The data sent for the prediction is then rendered to the web page.

3.12 TESTING

Testing a Flight Fare Prediction is crucial to ensure its accuracy, reliability and user-friendliness.

TEST PLAN

A test plan for a flight fare prediction system outlines the strategies, procedures, and conditions that will be tested to ensure the accuracy, reliability, and functionality of the system.

The purpose of this test plan is to outline the testing approach for the flight fare prediction system. The goal is to ensure that the system accurately predicts flight fares based on historical data and real-time data.

This test plan covers the testing of the flight fare prediction system's core functionality, accuracy of predictions, user interface and integration with external data sources.

The objectives of the test plan are:
1. Validate the accuracy of flight fare predictions.
2. Verify the system's responsiveness and reliability
3. Ensure the user interface provides a seamless experience for users.

Test Strategy

Testing Levels

1. Unit testing for individual components.
2. Integration system for system modules.
3. Acceptance testing for overall functionality

TESTING TYPES
1. Functional Testing
2. Regression Testing
3. Performance Testing
4. User Interface Testing

TEST CASES

TESTING

Testing a Flight Fare Prediction is crucial to ensure its accuracy, reliability, and user-friendliness.

1. **Unit Testing**

   Predicting Algorithms: Test each prediction algorithm to verify its accuracy.
   Data Sources: Ensure data retrieval from external sources like historical data.
   Calculation Modules: Test modules for calculating fares based on factors like distance, date of journey, airlines, and number of stoppages.
   Error Handling: Test how the system handles invalid inputs, missing

data

1. Test Case Description

Verify whether the user interface is accessible to the user

Pre-Requisites

User Interface URL should be defined.

Expected Results

User Interface URL should be accessible to the user.

2. Test Case Description

Verify whether the user interface loads completely for the user when the URL is accessed.

Pre-Requisites

User interface is accessible, User Interface is deployed

Expected Results

The user interface should load completely for the user when the URL is accessed.

3. Test Case Description

Verify whether the user is able to edit all input fields.

Pre-Requisites

User interface is accessible

Expected Results

The user should be able to edit all input fields.

2. Integration Testing

Module Integration: Test the integration of individual modules, including prediction algorithms, data retrieval and UI components.

Data Flow: Verify that data flows correctly between different components and modules

API Integration: Testing the integration and responses.

3.Functional Testing

Basic Fare Prediction: Test is the system ability to predict fares accurately for common and well-known routes.

Multi-City Journeys: Test fare predictions with layovers and multiple segments.

Last Minute Booking: Test fare predictions for bookings made very close to departure date.

Peak and Off-Peak seasons: Verify the system accuracy in predicting fare fluctuations during different peak and Off-Peak seasons.

4. **Regression Testing**

   Changes: Perform regression tests whenever there are updates or changes to the prediction algorithms or data sources to ensure existing functionality is not affected.

   New Features: Test existing test cases to ensure new features or enhancements.

   Ensuring new changes do not adversely affect existing functionality.

5. **Performance Testing**

   Load Testing: Test the system response under different user loads to ensure it can handle simultaneous requests.

   Scalability: Access how well the system scales when the number of users or data volume increases.
   Response Time: Measure the time it takes for the system to provide predictions, ensuring it meets acceptable performance standards.

6. **User Interface (UI) Testing**:

   Usability: Evaluate the user-friendliness of the interface by observing users interacting with it.
   Compatibility: Test the system on various browsers, devices, and screen sizes to ensure consistent user experience.
   Input Validation: Verify that user inputs are properly validated and errors are displayed appropriately.
   Visualization: Check if fare predictions are displayed clearly and comprehensively.

7. **Acceptance Testing**:

   User Testing: Invite real users to interact with the system and provide feedback on its accuracy and usability.
   Business Requirements: Ensure the system meets the original business requirements and objectives.
   User Stories: Validate that the system fulfills user stories related to accurate fare prediction.

8. **User Acceptance Testing (UAT):**
   End-User Testing: Engage actual users to perform testing in a real-world environment to ensure the system meets their needs.

This test plan outlines the testing approach for the flight fare prediction

system.
By executing the defined test cases and closely monitoring the results, we aim to ensure the accuracy, reliability, functionality of system, delivering a high-quality product to users.

TEST SCENARIOS

Test Scenarios for a flight fare prediction system help outline specific situations and conditions that need to be tested to ensure the accuracy and reliability of the prediction model.

1. Basic Fare Prediction

   Scenario: Predict the fare for a direct flight between two well-known cities.
   Expected Outcome: The predicted fare should be within an acceptable range of the actual fare for the given route.

2. Multi-City Journey

   Scenario: Predict the fare for a multi-city journey with layovers.
   Expected Outcome: The system should accurately calculate the combined fare for each segment of journey, considering layovers and different flight segments.

3. Last Minute Working

   Scenario: Predict the fare for a flight booked very close to the departure date.
   Expected Outcome: The system should account for the increased fare due to Last minute booking and provide an accurate prediction.

4. Peak Season Prediction

   Scenario: Predict the fare for a flight during a peak travel season.
   Expected Outcome: The system should consider seasonal fare variations and provide an accurate prediction for peak travel times.

5. Off-Peak Season Prediction

   Scenario: Predict the fare for a flight during an Off-peak travel season.
   Expected Outcome: The system should consider seasonal fare variations and provide an accurate prediction for off-peak travel times.

6. Variable Departure Times

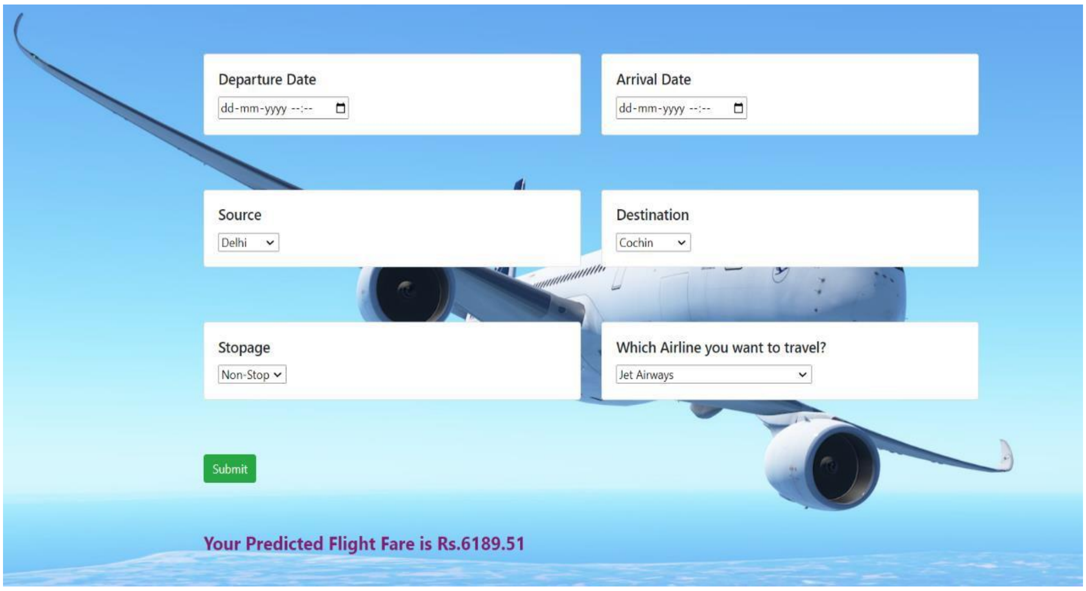
   Scenario: Predict fares for the same route on different departure times of the same day.
   Expected Outcome: The system should consider time-of-day fare variations and provide accurate predictions.

7. Cross-check with Historical Data

Scenario: Compare predicted fares with historical data for the same routes and dates.
Expected Outcome: Predicted fares should align with historical fare trends and actual past fares.

3.13 WEB INTERFACE

The user-friendly web page which provides single interface for inputs and outputs to user



USER INPUT

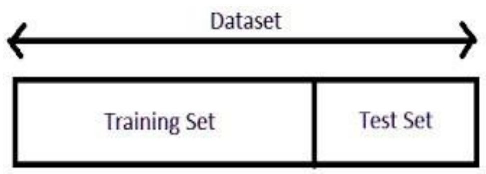
Whenever the user hits URL, they first see the user input page here they must provide the information like:
Every user input has its own dropdown where the user can select their input.
After providing the required input and pressing the submit button, the page refreshes and displays the output.

RESULT PAGE

After the user hits the submit button the page gets refreshed and the results are being displayed in the highlighted area in the above frame.
The user can refill all the inputs in same page and get the results in the same Way.

Exploring machine Learning Algorithms

Model Training: The data is divided into train and test data; the test data size is 0.33 and the random state is 44.



Stratified Sampling: Splitting dataset into random train and test subsets: X_train, X_test, y_train, y_test
test_size: represent the proportion of the dataset to include in the test split.
random_state: Controls the shuffling applied to the data before applying the split.

Stratify: Data is split in a stratified fashion, using this as the class labels.

Picking the Best Model: The model with the best accuracy and less error is best model for Flight Fare Prediction System.

The machine learning algorithms used in the project are:

1. Linear Regression
2. Ridge Regression
3. Lasso Regression
4. ElasticNet Regression
5. Decision Tree Regressor
6. Random Forest Regressor
7. Random Forest Regressor with hyperparameter tunning

3.14 ANALYSIS

1. Linear Regression Machine Learning Algorithm

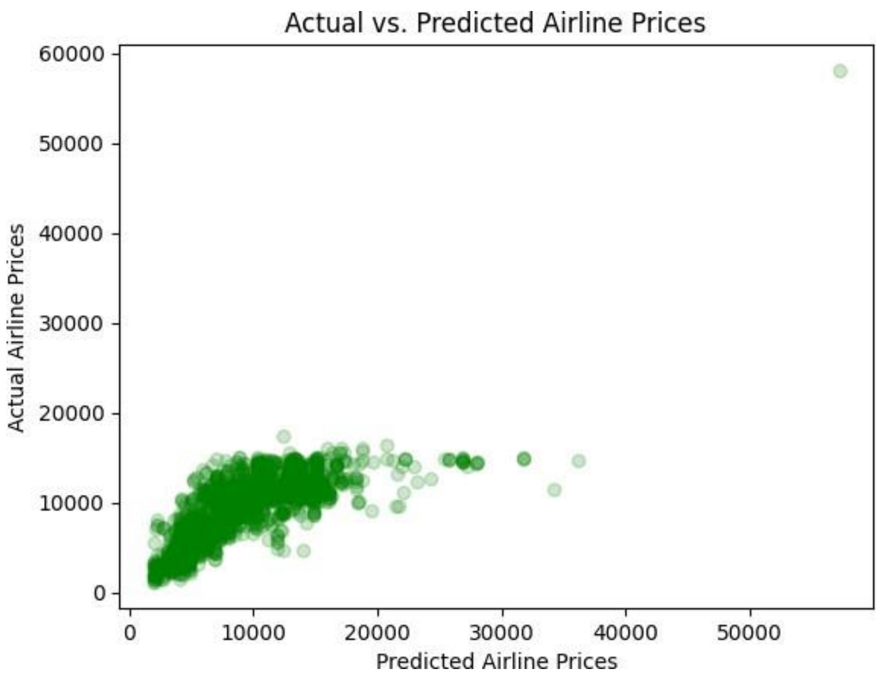


Fig 16: Scatterplot of Actual Airline prices vs Predicted Airline prices

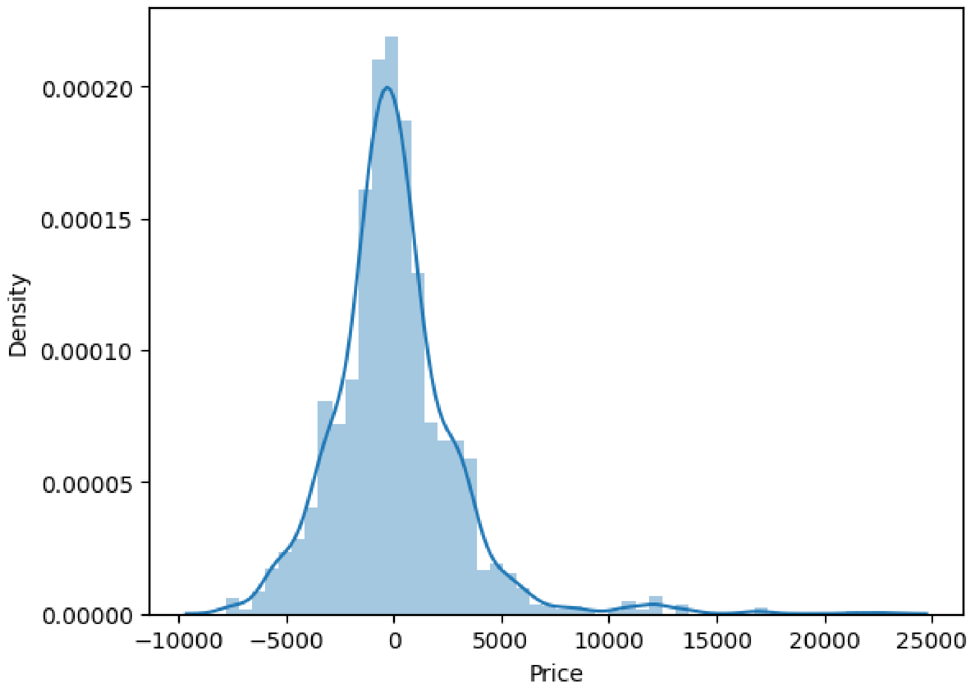


Fig 17: Showing Normal distribution of Price with Test Data

2. RIDGE REGRESSION

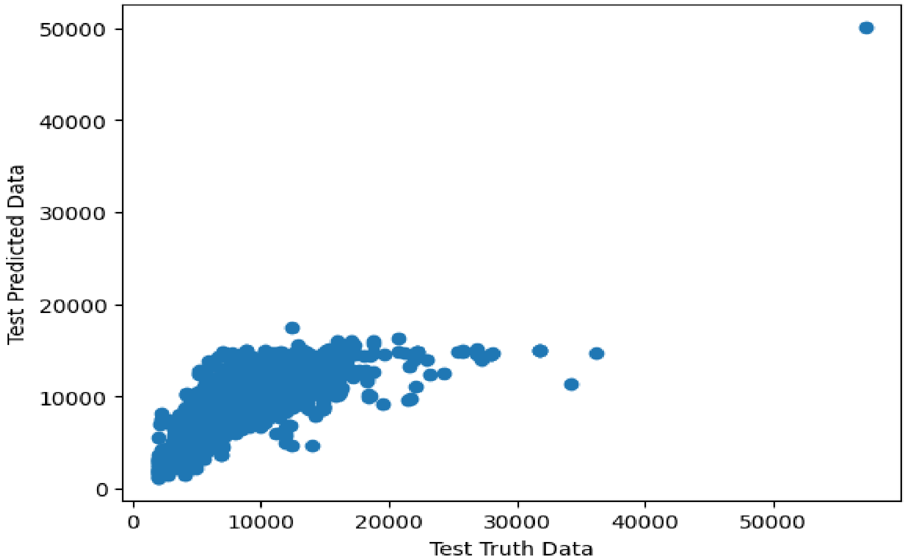


Fig 18: Scatterplot of Actual Airline prices vs Predicted Airline prices

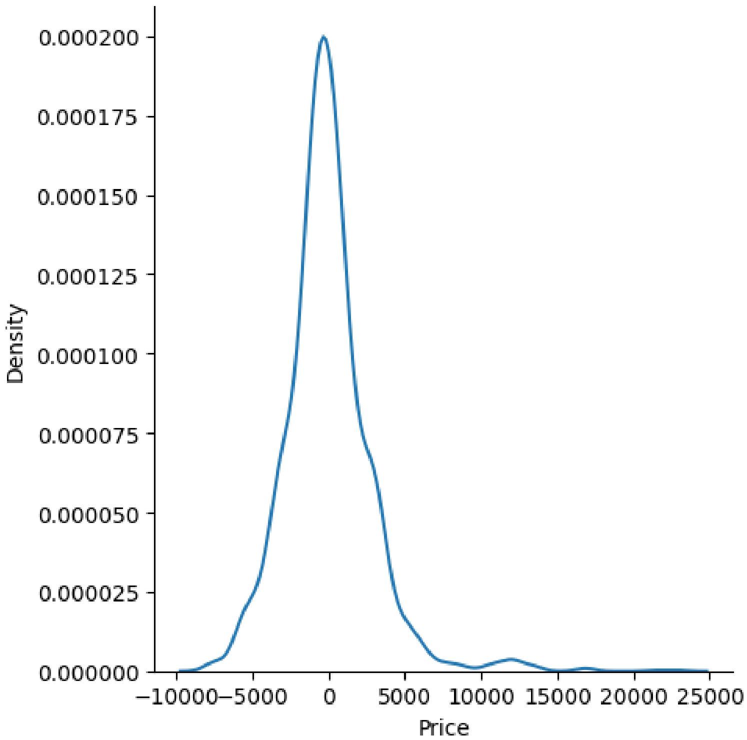


Fig 19: Showing Distplot of Price Vs with Test Data

3. LASSO REGRESSION

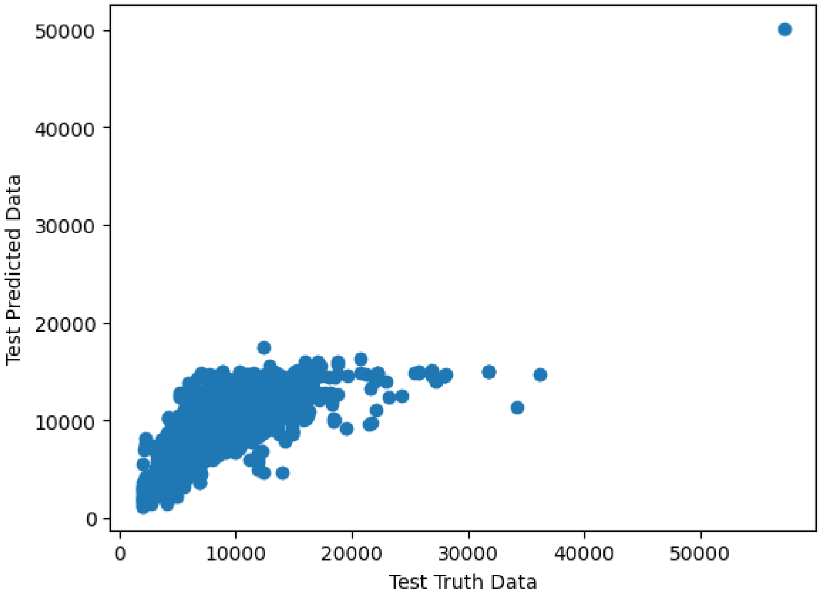


Fig 20: Scatterplot of Actual Airline prices vs Predicted Airline prices

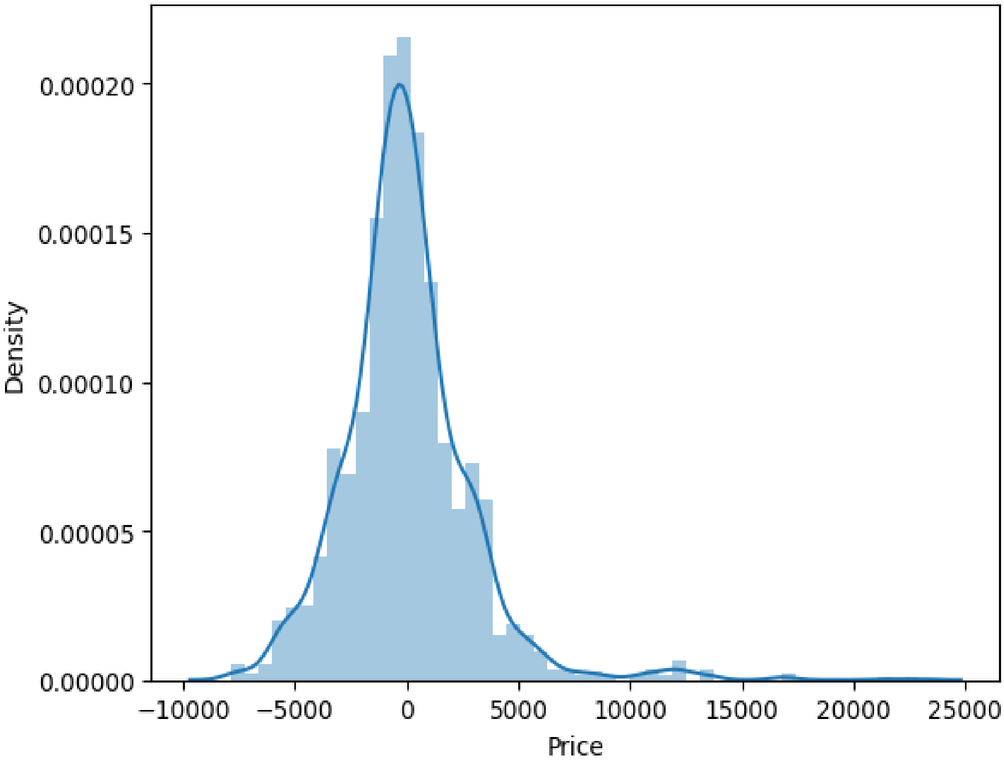


Fig 21: Showing Distplot of Price Vs with Test Data

4. ELASTICNET REGRESSION

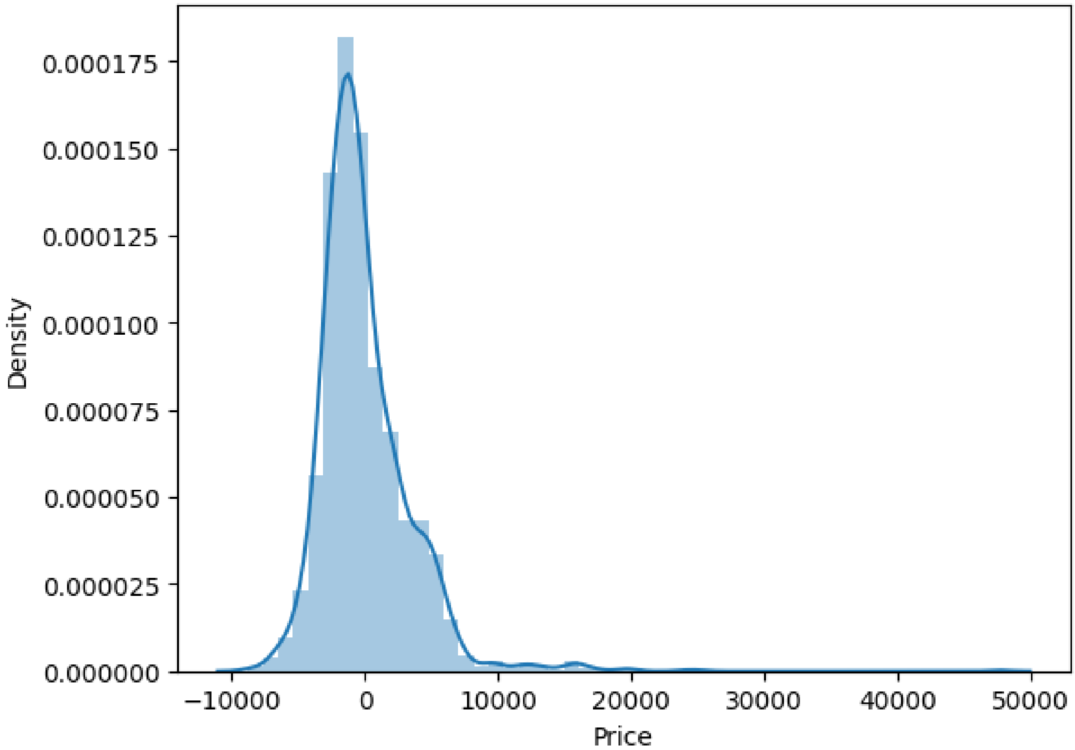


Fig 23: Showing Distplot of Price Vs with Test Data

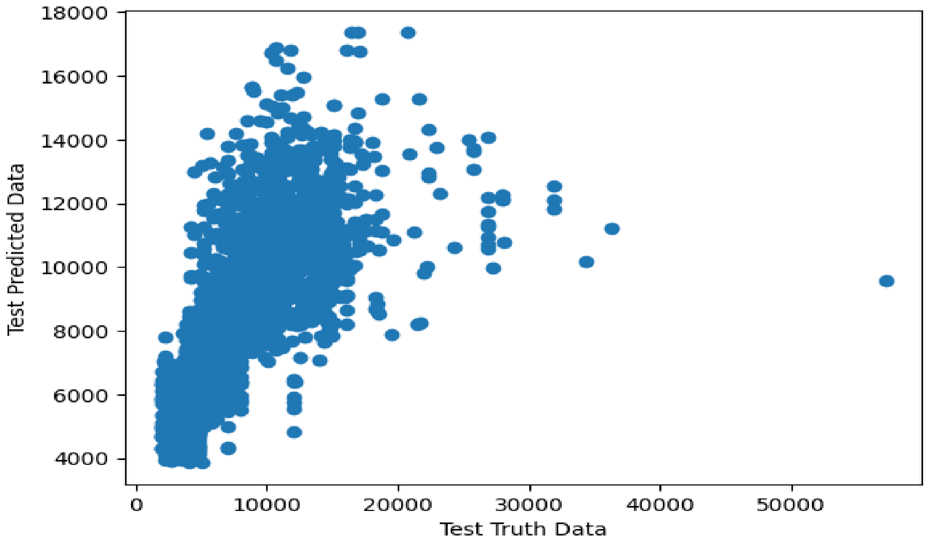


Fig 24: Scatterplot of Actual Airline prices vs Predicted Airline prices

5. DECISION TREE REGRESSOR

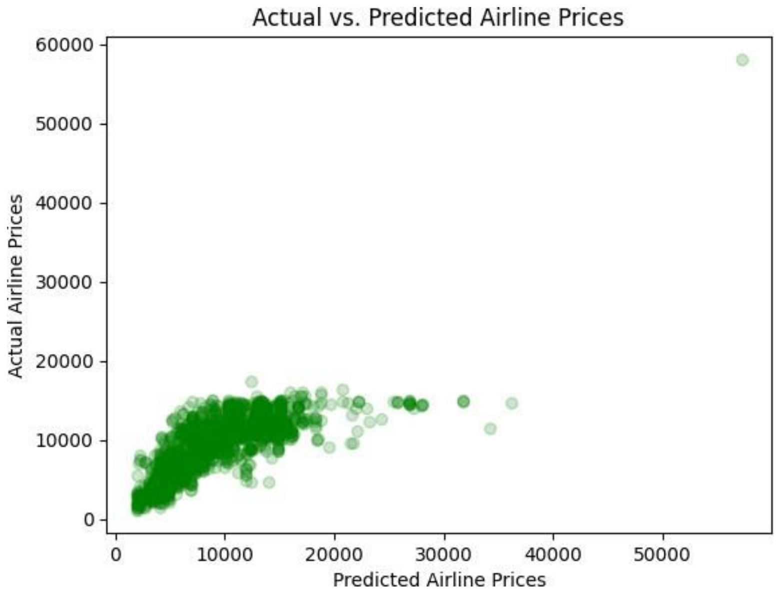


Fig 25: Scatterplot of Actual Airline prices vs Predicted Airline prices

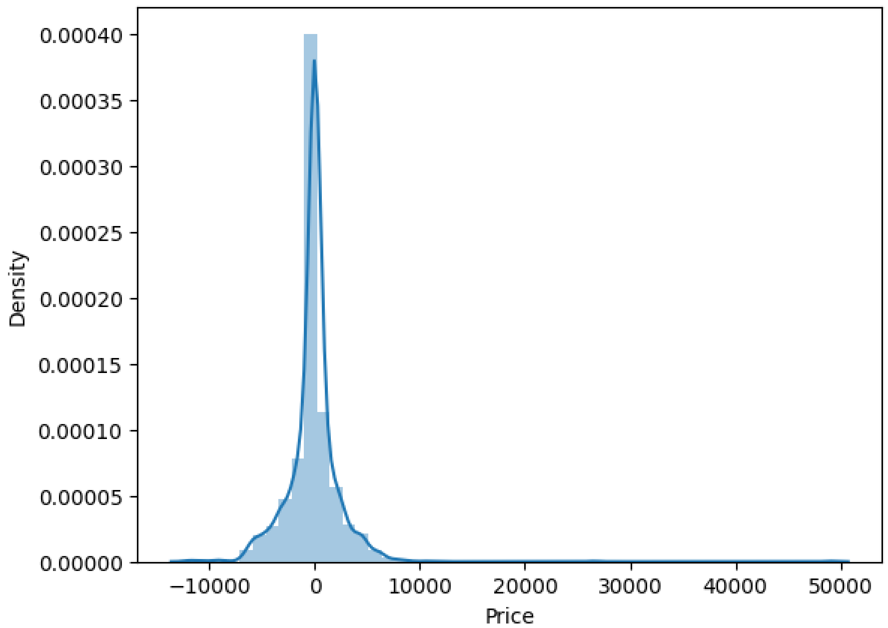


Fig 26: Showing Distplot of Price Vs with Test Data

6. RANDOM FOREST REGRESSOR

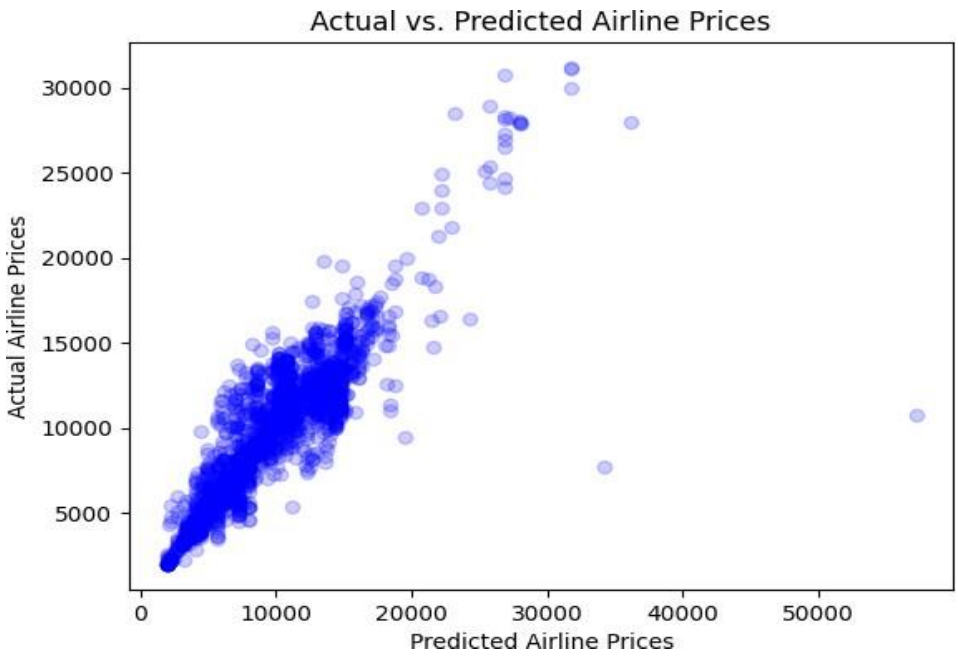


Fig 27: Scatterplot of Actual Airline prices vs Predicted Airline prices

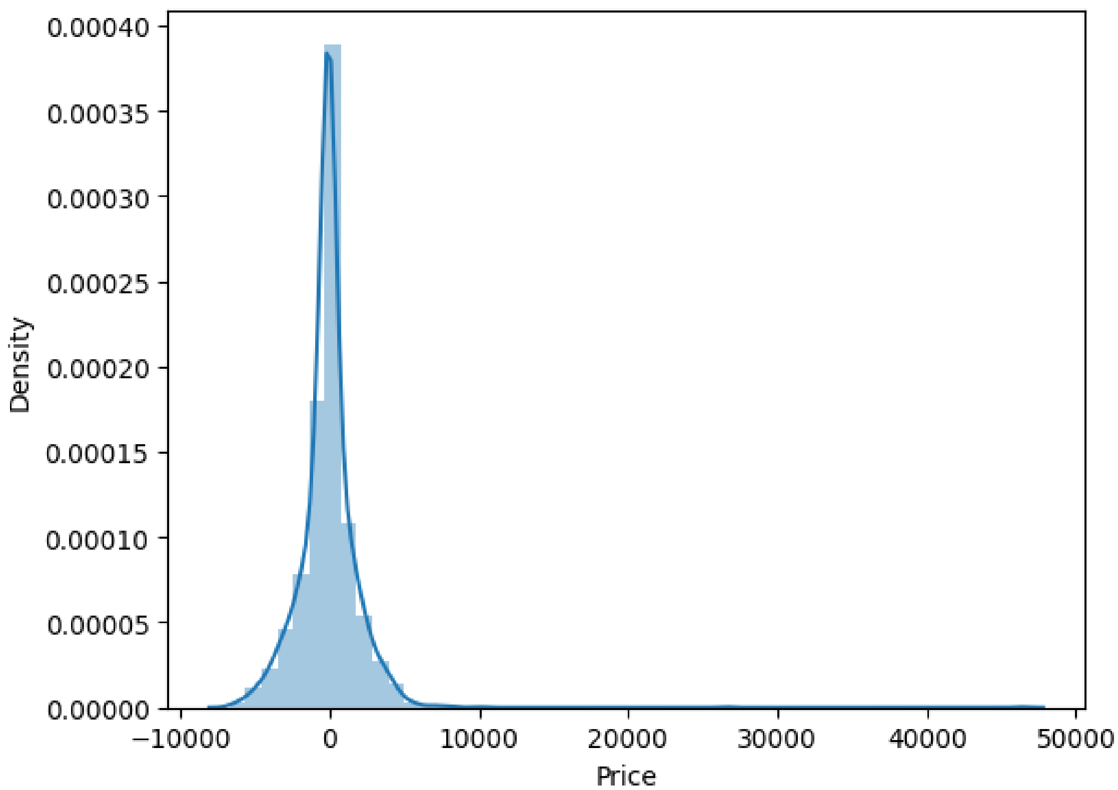


Fig 28: Showing Distplot of Price Vs with Test Data

7. RANDOM FOREST REGRESSOR WITH HYPERPARAMETER TUNNING

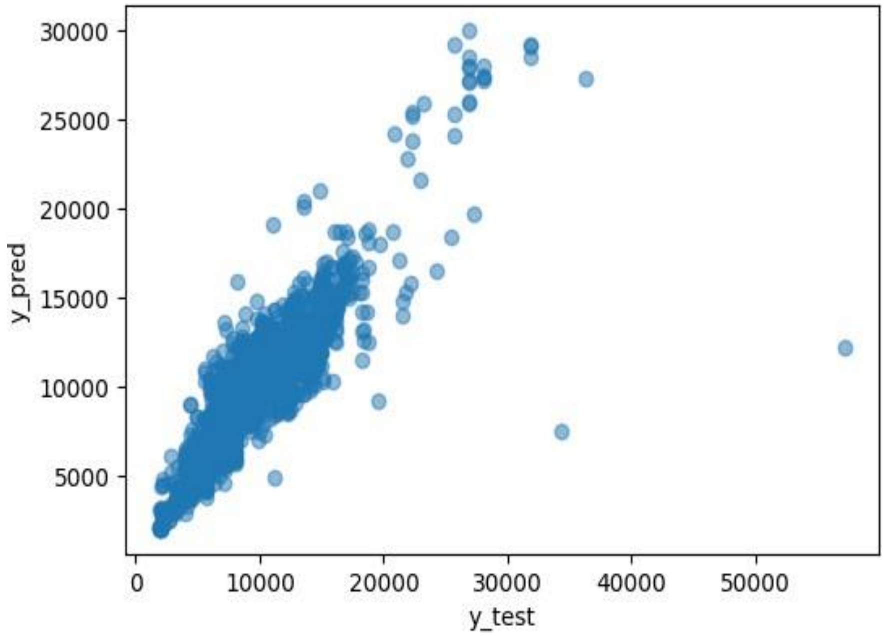


Fig 29: Scatterplot of Actual Airline prices vs Predicted Airline prices

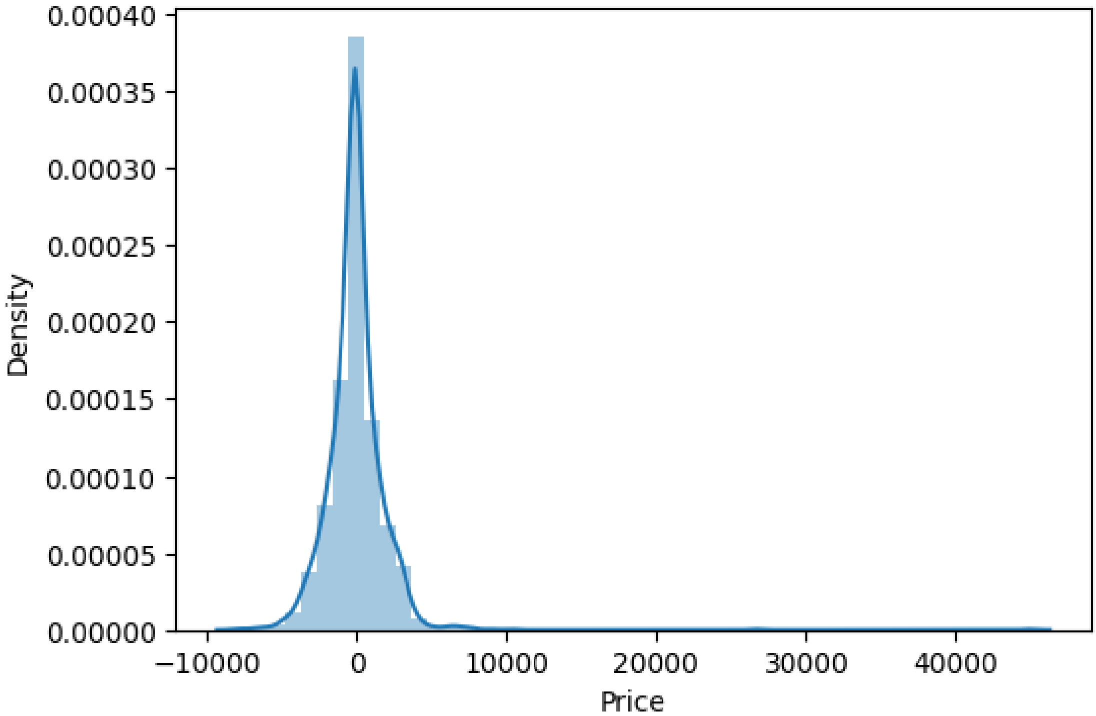


Fig 30: Showing Distplot of Price Vs with Test Data

3.15 PERFORMANCE METRICS

The accuracy of machine learning models trained by various algorithms will be c compared using performance metrics, which are statistical models. Regression metrics will be implemented for error measurement functions from each model using the Sklearn. metrics module.

The following metrics will be examined to determine each model's error rate:

MAE (Mean Absolute Error) The mean of the absolute difference between the expected and actual numbers is effectively added to determine the mean absolute error.

MAE = 1/n [Σ (y-ý)]

The expected output values are y' and the actual output values are y.
There are n total data points.

The model will perform better the lower the MAE number is.

MSE (Mean Square Error)

The root mean square error exponentiates the difference of the true a predicted output values before summing them instead using an absolute value. MSE = 1/n [Σ (y-ý) 2]

y=actual output values ý=predicted output values n = Total number of data points.

MSE penalizes large errors when we square the errors. Less the MSE value, the better the model performance.

RMSE (root mean square error)

RMSE is measured by taking the square root of the mean squared difference between forecast and actual value.

RMSE = √1/n [Σ (y-ý) 2]

The expected output values are y' and the actual output values are y.
There are n total data points.

The higher the performance of a model, the RMSE value is less.

R2 (Coefficient of determination)

It will help to understand how well the independent variable modified with a deviation in your model.

R2 = 1 −Σ(ý-y̅)2 / Σ(y-y̅)2

The R-squared value lies between 0 and 1.

The closer its value is for one, the better your model is compared to the model values

8.16 RESULTS

The accuracy (R Square) of different models employed:

```
R Square value of Linear regression Model : 61.918 %
R Square value of ElasticNet regression Model : 43.304 %
R Square value of Lasso regression Model : 61.883 %
R Square value of Ridge regression Model : 61.883 %
R Square value of Decision Tree regressor Model : 71.125 %
R Square value of Random Forest Regressor Model Before Hyperparameter Tunning : 79.491 %
R Square value of Random Forest Regressor Model After Hyperparameter Tunning : 80.962 %
```

	Predicted Price	Actual Price
0	16865.690000	16655
1	5984.370000	4959
2	8929.430000	9187
3	3641.370000	3858
4	15099.211833	12898
...	...	...
2132	12370.180000	7408
2133	4974.910000	4622
2134	6714.790000	7452
2135	12945.315810	8824
2136	12933.479500	14151

2137 rows × 2 columns

Table 1: Displaying Actual Price Vs Predicted Price

The accuracy of different models employed with Training and Test Data Score

	Model Name	R Square Value	Adjusted R square value	Training Data Score	Testing Data Score
0	Linear Regression	0.619176	0.613934	0.623828	0.619176
1	Decision Tree Regressor	0.711249	0.707274	0.968510	0.711249
2	Ridge regression	0.618826	0.613934	0.622117	0.618826
3	Lasso regression	0.618826	0.613934	0.623676	0.619353
4	ElasticNet regression	0.433044	0.613934	0.432794	0.433044
5	Random Forest Regressor Model Before Hyperpara...	0.794913	0.792091	0.951900	0.794913

	Model Name	R Square Value	Adjusted R square value	Mean Absolute Error	Mean Square Error	Root Mean Square Error
0	Linear Regression	0.619176	0.613934	1972.542785	8.211358e+06	2865.546801
1	Ridge regression	0.618826	0.613934	1976.832473	8.218905e+06	2866.863299
2	Lasso regression	0.618826	0.613934	1973.714353	8.207522e+06	2864.877369
3	ElasticNet regression	0.433044	0.613934	2427.132958	1.222475e+07	3496.390317
4	Decision Tree Regressor	0.711249	0.707274	1352.316760	6.226072e+06	2495.209721
5	Random Forest Regressor Model Before Hyperpara...	0.794913	0.792091	1199.062952	4.422087e+06	2102.875884
6	Random Forest Regressor Model After Hyperparam...	0.809619	0.806999	1190.732568	4.104998e+06	2026.079380

```
## Conclusion
# The Random Forest Regressor Model After Hyperparameter Tunning is the best model for Flight Fare Prediction
# The Random Forest Regressor model has best R Square value with less Errors(MAE,MSE,RMSE)
# There is minimal difference between r squares of Random Forest Regressor Model before and after hyperparameter tunning
```

Table 2: Displays the accuracy and Performance Metrics of different machine learning models.

FINDINGS

1. The Random Forest Regressor model after hyperparameter tunning is the best model for Flight Fare Prediction Application.
2. The Random Forest Regressor model has best r square value as compared to other machine learning algorithms.
3. The Random Forest Regressor model has less value of errors (Mean Square Error, Mean Absolute Error, Root mean Square Error)

CHAPTER: 4

USER MANUAL

4.1 USER MANUAL

A user manual for software application is the technical documentation framed to instruct users to provide information to use the application.

The user manual contains list of data attributes needed to fetch into the model by the user interface through websites or applications by the patients for the prediction of kidney chronic disease or renal disorders.

The user manual has precise readable instructions in Hindi and English as well as pictorial representation of how to enter each data attribute and detailed wise instructions to how to use the application.

The user manual will have few contact numbers of Customer contact support, which customers can use if case of disruption or system failure.

Flight Fare Prediction System User Manual will have following information.

Overview of the Flight Fare Prediction System
Purpose and benefits of using the system
Getting Started

System access (website, mobile app, etc.)
Navigating the System

Interface overview
Menu options and their functions
How to search for flights and routes
Predicting Flight Fares

Step-by-step guide to using the fare prediction feature
Selecting departure and destination cities
Choosing travel dates and times
Understanding fare estimates
Interpreting Results

Explaining fare prediction results
Factors influencing fare estimates
Disclaimers about prediction accuracy

Providing accurate travel details
Understanding how predictions are calculated
Frequently Asked Questions

Addressing common user queries
Troubleshooting

Common issues and solutions
How to contact customer support
Privacy and Data Security

Explanation of data handling and user privacy
Data encryption and protection measures
Terms of Use and Disclaimers

Legal information and user agreement
Limitations of the system's predictions
Glossary
Definitions of technical terms and concepts
Appendices
Sample screenshots and visuals
References to external resources
The user manual should be written in clear and user-friendly language, providing step-by-step instructions and helpful visuals whenever possible. It should aim to guide users through the entire process of using the Flight Fare Prediction System, from accessing the platform to interpreting the prediction results.

4.2 USER TEST RESULTS

The user-friendly user interface will provide the predicted Flight Fare depending upon the date of journey, arrival date, the source and destination city distance, the number of stops and the Airlines chosen by traveler for aviation.

Usability: Evaluate the system's ease of use and user-friendliness. User test results might indicate if participants found the system intuitive to navigate or if they faced difficulties in using certain features.

Accuracy of Predictions: Assess how accurately the system's fare predictions aligned with participants' expectations and actual prices, they've encountered.

Speed and Responsiveness: Determine if the system's response time met user expectations, especially when retrieving fare predictions or loading search results.

Error Identification: Identify any error messages, glitches, or technical issues that users encountered during testing.

User Satisfaction: Gauge participants' overall satisfaction with the system, including whether they found the predictions helpful and valuable.

Suggestions for Improvement: Collect participants' suggestions for enhancing the system's features, interface, and overall functionality.

User Preferences: Understand user preferences regarding filters, sorting options, and presentation of fare predictions.

User Behavior: Gather insights into how users explore different flight options, utilize advanced features, and make decisions based on the predictions.

Common Pain Points: Identify any recurring pain points or confusing aspects that users encountered during their interaction

Comparison with Competitors: If applicable, assess how users perceive your system in comparison to similar tools or competitors.

4.3 PROGRAM SPECIFICATIONS

The Program specification document contains an overview of the concept for the software product, tools for production, list of program features, user requirements, and overall design of the program code.

Utility of Application: The prediction of flight fare by use of application will provide the travelers get the fare prediction handy using which it is easy to decide the airlines and saving time in searching / deciding for airlines.

Flight Fare prediction is a classic time series forecasting problem that finds trends in past observations to make future predictions. Many popular flight booking websites today, including Google Flights, showcase intelligent insights on flight fare trends to help user decide what is the right time to book a flight ticket.

User Interface:

Web-based or mobile application interface for users to interact with the system.
Fare Prediction Algorithm
Development of a predictive algorithm that considers historical flight data,
Demand trends, seasonal variations, and other factors.
Continuous learning and refinement of the prediction model.
Ability for users to search for flight routes based on departure and destination cities.
Calendar interface for selecting travel dates.
Options to filter and sort results by criteria such as airlines, departure times

Performance Optimization:
Efficient handling of large datasets and quick response times for fare prediction.

Scalability to accommodate a growing number of users.

User Experience Design
Intuitive and user-friendly interface design for easy navigation and interaction.
Responsive design to ensure a consistent experience across different devices.

Testing and Quality Assurance:

Comprehensive testing, including unit testing, integration testing, and user acceptance

Testing.
Documentation

User manual and help documentation for guiding users through the system's features and

functionalities.

Deployment and Maintenance:

Selection of hosting environment (cloud-based, on-premises, etc.).

Regular updates, bug fixes, and maintenance to ensure system reliability.

CHAPTER: 5

DRAWBACKS AND LIMITATIONS

5.1 DRAWBACKS AND LIMITATIONS

A Flight Fare Prediction System can be a valuable tool for both travellers and airlines, but like any technology, it comes with certain limitations.

Here are some potential limitations of a Flight Fare Prediction System:

Data Accuracy and Availability: The accuracy of fare predictions heavily relies on the quality and availability of historical flight data. If the system lacks comprehensive, up-to-date, or accurate data, its predictions may be unreliable.

Dynamic Pricing Complexity: Airlines often use complex dynamic pricing models that consider numerous factors such as demand, seat availability, time to departure, competition, and more. These models can be difficult to accurately replicate, leading to discrepancies between predicted fares and actual prices.

Unforeseen Events: Flight prices can be affected by unforeseen events such as weather disruptions, geopolitical issues, and other emergencies. These events may not be adequately captured by historical data and can lead to inaccurate predictions.

User-Specific Factors: Individual user behaviours and preferences are not always accounted for in fare prediction models. For instance, a user's browsing history, loyalty program status, and booking patterns might influence the fares they see.

Limited Scope: The system might not cover all airlines, routes, or travel classes. This limitation can result in incomplete fare predictions and might not provide a comprehensive view of the available options.

Modelling Assumptions: Any predictive model relies on certain assumptions. If these assumptions do not hold true in a real-world scenario, the predictions may be less accurate.

Market Competition and Strategy: Airlines adjust their pricing strategies based on competitive dynamics and market conditions. These strategies can change frequently and might not be accurately captured by the prediction system.

Currency Fluctuations: Flight prices can vary due to currency exchange rate fluctuations, which might not be factored into the prediction model.

User Interaction: User input, such as the selected travel dates or departure times, can significantly influence fare predictions. Inaccurate user input can lead to incorrect predictions.

Legal and Regulatory Factors: Flight prices can be influenced by various legal and regulatory factors, such as taxes and fees, which might not be fully accounted for in the prediction model.

Limited Historical Data: New routes or emerging airlines might have limited historical data available, making it challenging to accurately predict fares for these cases.

Despite these limitations, a Flight Fare Prediction System can still offer valuable insights and assistance to travellers, helping them make more informed decisions when booking flights. However, users should be aware of these potential limitations and use the predictions as a guideline rather than a definitive source of pricing information.

CHAPTER: 6

ENHANCEMENTS

6.1 ENHANCEMENTS

1. To employ payment gateway into the application.
2. Different travel routes can be incorporated into the application.
3. To advertise the website and make practical use of the application.
4. Users may want to receive notifications or alerts when fare prices for their chosen route and dates change significantly

CHAPTER: 7

CONCLUSION

7.1 CONCLUSION

The Flight Fare Prediction application uses Random Forest model and its accuracy is improved by doing hyperparameter Tunning.
The model can be trained by using ‘Random Forest Model’ to forecast pricing flights with an r2 score of 80 percent after hyperparameter tunning.
The value of errors (mean square error, mean absolute error and root mean square error value drops) suggesting the best fit model.

In the Random Forest ensemble learning technique, the training model employs several different learning algorithms, and the separate outputs are then combined to produce the final anticipated outcome. The random forest belongs to the bagging category of ensemble learning, where a random number of elements and records are chosen and given to the model group. In essence, decision trees are used as a group of models in random forest. The average value of the anticipated values, if they are thought to be the output of the random forest model, can be calculated from the predictions produced by decision trees.

The flight fare prediction will predict the worth supported by the trained knowledge set within the machine learning intelligence. Thus, the user will recognize the approximate value for his or her journey and can plan the exact time to book the tickets and save their money.

CHAPTER: 8

BIBLIOGRAPGHY ANEXURE

BIBLIOGRAPGHY ANEXURES

1. Rutuja Konde, Rutuja Somvanshi, Pratiksha Khaire, Prachi Zende, Kamlesh Patil, "Airfare Price Prediction System", International Research Journal of Engineering and Technology (IRJET), Volume: 09 Issue: 05, May 2022.

2. Mrs. Sowjanya, Saud Ikram, Rayyan Khan, Shaik Haseeb Jawad, Saadul arifeen, "Prediction of Airfare Prices Using Machine Learning," International Journal of Mechanical Engineering, Vol.7 No.6, ISSN: 0974-5823, June, 2022.

3. Pavithra Maria K, Anitha K L, "Flight Price Prediction for Users by Machine Learning Techniques", International Advanced Research Journal in Science, Engineering and Technology, Vol.8, Issue 3, DOI: 10.17148/IARJSET.2021.8321, March 2021.

4. Prof. Ms. Archana Dirgule, Shubham Agarwal, Ram Agrawal, Neha Singh, Kiran Adsul, "Flight Fare Prediction using Random Forest Algorithm," International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), Volume 2, Issue 3, ISSN (Online) 2581-9429, May 2022.

5. Neel Bhosale, Pranav Gole, Hrutuja Handore, Priti Lakde, Gajanan Arsalwad, "Flight Fare Prediction Using Machine Learning," International Journal for Research in Applied Science & Engineering Technology (IJRASET), ISSN: 2321-9653, Volume 10, Issue V, May 20