A Project Report

On

FLIGHT TICKET PRICE PRIDICTION USING MACHINE LEARNING

Submitted partial fulfillment for the award of the degree of

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

K. SANGEETHA	**214E1A0577**
J. ROHITH	**214E1A0574**
M. PHANI KUMAR	**224E5A0509**
K. VAMSI KRISHNA	**214E1A05A5**
G. SARATH BABU	**214E1A0578**

Under the esteemed guidance of

Mrs. B. Himabindu, M. Tech

Assistant Professor

Dept. of C.S.E

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SIDDARTHA INSTITUTE OF SCIENCE AND TECHNOLOGY
(AUTONOMOUS)

(Approved by A.I.C.T.E., New Delhi Affiliated to J.N.T.U. Anantapur, Ananthapuramu.)

Siddharth Nagar, Narayanavanam Road, Puttur– 517 583, Chittoor District

2024-2025

SIDDARTHA INSTITUTE OF SCIENCE AND TECHNOLOGY (AUTONOMOUS)

(Approved by A.I.C.T.E., New Delhi Affiliated to J.N.T.U. Anantapur, Ananthapuramu.)

Siddharth Nagar, Narayanavanam Road, Puttur-517583

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Certificate

This is to certify that the Project is entitled

"FLIGHT TICKET PRICE PREDICTION USING MACHINE LEARNING"

that is being submitted by

K. SANGEETHA	**214E1A0577**
J. ROHIT	**214E1A0574**
M. PHANI KUAMR	**224E5A0509**
K. VAMSI KRISHNA	**214E1A05A5**
G. SARATH BABU	**214E1A0578**

in partial fulfillment of the requirements for the award of **BACHELOR OF TECHNOLOGY** in **Computer Science and Engineering** to **JNTUA, Ananthapuramu**. This project work or part thereof has not been submitted to any other University or Institute for the award of any degree.

Guide *Head of the Department*

Submitted for the University Examination held on ________________

INTERNAL EXAMINER EXTERNAL EXAMINER

ACKNOWLEDGEMENT

An endeavour of a long period can be successful only with the advice of many well-wishers. We take this opportunity to express our deep gratitude and appreciation to all those who encouraged us for successful completion of the project work.

We wish to express my sincere thanks to **Dr. K. ASHOK RAJU**., **Ph.D**., ***Chairman*** of Siddartha Institute of Science and Technology, Puttur, for providing ample facilities to complete the project work.

Our special thanks to **Dr. M. JANARDHANA RAJU, ME., Ph.D**., **Principal,** Siddartha Institute of Science and Technology, Puttur, for his wonderful support during the progress of project work.

We are very much thankful to **Dr. M. A. MANIVASAGAM, M.E, Ph.D., Professor and Head of the Department,** *Computer Science and Engineering,* Siddartha Institute of Science and Technology, Puttur, for his valuable suggestions in completion of the project work

We owe our deep Gratitude to our project guide **Mrs. B. HIMABINDU MTech**., Associate Professor, Department of Computer Science and Engineering, Siddartha Institute of Science and technology, Puttur., who took keep interest on our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system.

Finally, we would like to express sincere thanks to our project coordinator **V. HANUMANTHA RAO, M.Tech., Associate Professor** and **I.D.C (Internal Departmental Committee) members, faculty members of C.S.E** and Lab technicians, one and all who have helped us to complete the project work successfully.

As a gesture of respect towards our **Family Members, friends** and their support we whole heartedly dedicate this work.

TABLE OF CONTENTS

Chapter No.	Title	Page No.

ABSTRACT

In the dynamic and highly competitive airline industry, accurate flight price prediction is paramount for both travelers seeking cost-effective options and airlines optimizing revenue management. This study presents a comprehensive analysis of three distinct machine learning algorithms – Decision Trees, Random Forest, and Logistic Regression – in the context of flight price prediction. Firstly, Decision Trees, known for their simplicity and interpretability, are examined. We explore their capacity to capture pricing patterns and their limitations in handling complex relationships within the dataset. Next, Random Forest, an ensemble method, is investigated to assess its ability to improve prediction accuracy by combining multiple decision trees. Finally, Logistic Regression, a widely used classification algorithm, is adapted for flight price prediction, highlighting its strengths in handling binary outcomes and potential benefits when applied to ticket pricing. The study utilizes real-world flight pricing data and evaluates the performance of each algorithm in terms of r2 score, and means squared Error mean absolute Error. Additionally, feature importance analysis is conducted to uncover the key factors influencing flight prices. By comparing the three machine learning approaches, this research aims to provide valuable insights into the strengths and weaknesses of each method in the context of flight price prediction. These findings will help stakeholders, including travellers and airlines, make more informed decisions and enhance pricing strategies in the aviation industry.

Keywords: Decision Tree, Random Forest, Logistic Regression. Lasso, MLP

LIST OF FIGURES

LIST OF TABLES

CHAPTER -1

INTRODUCTION

1.1 Domain Description

Machine Learning (ML) is a subfield of artificial intelligence (AI) that focuses on the development of algorithms and statistical models that enable computers to perform tasks without being explicitly programmed. Instead, these algorithms learn from data, identify patterns, and make predictions or decisions. Machine learning encompasses various techniques, including supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, and deep learning.

Types Of Machine Learning

- **Supervised Learning**

  In supervised learning, the algorithm learns from labeled data, meaning the input data is paired with the correct output. The goal is to learn a mapping from input to output so that it can predict the correct output for new, unseen inputs. Supervised learning can be further divided into two subcategories:

  - Classification
  - Regression

- **Unsupervised Learning**

  In unsupervised learning, the algorithm learns from unlabeled data, meaning there is no explicit output provided. The goal is to discover hidden patterns or structures in the data. Unsupervised learning can be further divided into several types:

  - Clustering
  - Dimensionality Reduction
  - Association Rule Learning

- **Reinforcement Learning**

  In reinforcement learning, the algorithm learns through interaction with an environment. It learns to make decisions by taking actions and receiving feedback in the form of rewards or penalties. The goal is to learn a policy that maximizes the cumulative reward over time. Reinforcement learning is often used in scenarios such as game playing (e.g., AlphaGo), robotics, and autonomous vehicle control.

Features of Machine Learning:

- **Data-driven:** Machine learning algorithms rely on data to learn patterns and make predictions or decisions.
- **Adaptability**: ML models can adapt and improve their performance over time as they are exposed to more data.
- **Automation:** ML enables automation of tasks, reducing the need for manual intervention.
- **Scalability:** ML models can scale to handle large datasets and complex problems.
- **Generalization**: ML models aim to generalize patterns learned from training data to make predictions on unseen data.
- **Machine**: learning techniques to find applications across various domains, including finance, healthcare, marketing, e-commerce, and more.

Needs for Machine Learning:

- **Data:** Machine learning algorithms require large amounts of data to train effectively.
- **Computational Resources:** ML models often require significant computational resources, especially for training complex models like deep learning neural networks.
- **Feature Engineering:** The process of selecting and transforming features from raw data is crucial for the performance of machine learning algorithms.
- **Evaluation Metrics**: Proper evaluation metrics are needed to assess the performance of ML models accurately.

- **Healthcare:** ML is applied in diagnosing diseases, personalized medicine, drug discovery, and analyzing medical images.
- **Finance:** ML techniques are used in fraud detection, credit scoring, algorithmic trading, and risk management.
- **Recommendation Systems:** ML algorithms power recommendation engines in e-commerce platforms, streaming services, and social media platforms to personalize content for users.
- **Autonomous Vehicles:** ML plays a vital role in enabling self-driving cars by processing sensor data and making real-time driving decisions.
- **Marketing and Advertising:** ML is used for customer segmentation, personalized marketing campaigns, click-through rate prediction, and ad targeting.
- **Robotics:** ML algorithms are used in robotic control, object manipulation, and task planning in various industrial and service robotics applications.
- **Environmental Monitoring:** ML techniques are applied in analyzing environmental data for tasks such as climate modeling, pollution monitoring, and wildlife conservation.

1.2 About Project

In the competitive airline industry, accurate flight price prediction is crucial for both travelers and airlines. It helps travelers make informed decisions and enables airlines to optimize revenue management. This study analyzes three machine learning algorithms—Decision Trees, Random Forest, and Logistic Regression—for flight price prediction. Decision Trees are examined for their simplicity and interpretability but are limited in handling complex relationships within pricing data. Random Forest, an ensemble of Decision Trees, aims to improve accuracy by combining multiple tree outputs. Logistic Regression, typically used for binary classification, is adapted to flight price prediction due to its effectiveness in handling pricing decisions. The study uses real-world flight pricing data and evaluates performance using metrics like R^2 score, mean squared error, and mean absolute error. The findings aim to guide both travelers and airlines in making better pricing decisions.

CHAPTER 2

LITERATURE SURVEY

[1] "A Hybrid Machine Learning Approach for Airfare Prediction in the Era of Big Data" Authors: Hamed Valizadegan, et al. Published Year: 2020

In this study, the authors propose a hybrid machine learning approach, which combines multiple machine learning techniques to enhance the accuracy of airfare prediction. The era of big data in the aviation sector has introduced a wealth of variables and factors that influence ticket prices, making traditional prediction methods less effective. The hybrid approach likely incorporates various algorithms such as decision trees, regression models, and ensemble techniques like random forests or gradient boosting. It leverages the strengths of these algorithms to capture intricate pricing patterns, account for nonlinear relationships, and handle the complexity of the data. This research is expected to contribute significantly to the field of airfare prediction by addressing the unique challenges posed by big data and offering more reliable pricing forecasts, ultimately benefiting both travelers and airlines in making more informed decisions and optimizing revenue management strategies.

[2] Paper Title: "Airline Ticket Price Prediction Using Machine Learning Techniques" Authors: Mohammed E. Tawfeeq, et al. Published Year: 2020.

Airline ticket price prediction using machine learning techniques has gained significant importance in the airline industry, benefiting both travelers and airlines alike. This predictive modeling leverages advanced algorithms to forecast airfare costs accurately, aiding travelers in making informed decisions and enabling airlines to optimize their pricing strategies. Machine learning models applied to this context typically use historical flight data, including factors such as departure and arrival locations, flight times, airline carriers, booking dates, and seat availability. These models can capture complex relationships within the data, allowing them to make predictions based on real-time inputs. Key machine learning techniques employed for airline ticket price prediction include regression algorithms like Linear Regression and Random Forest, as well as more advanced methods such as Gradient Boosting and Neural Networks. These techniques analyze historical price trends and various influencing factors to generate price estimates. By accurately predicting ticket prices, airlines can maximize revenue by adjusting fares dynamically, while travelers can secure cost-effective deals by booking flights at optimal times. This application of machine learning contributes to a more efficient and competitive airline industry, ultimately benefiting both passengers and carriers.

[3] Paper Title: "A Machine Learning-Based Approach for Airfare Prediction: A Comparative Analysis" Authors: Syed Muhammad Usama, et al. Published Year: 2021.

The paper titled "A Machine Learning-Based Approach for Airfare Prediction: A Comparative Analysis" presents a detailed exploration of machine learning techniques applied to the domain of airfare prediction. Published in 2021, the study conducts a comparative analysis of various machine learning algorithms to assess their effectiveness in predicting airfare prices accurately.

The authors systematically evaluate and compare different machine learning models, considering factors such as predictive accuracy, computational efficiency, and scalability. This comprehensive analysis helps in identifying the most suitable algorithms for airfare prediction. The paper likely discusses the utilization of real-world airfare data, feature engineering techniques, and performance evaluation metrics such as Mean Absolute Error (MAE) or Root Mean Square Error (RMSE) to measure prediction accuracy. Furthermore, it may delve into the importance of feature selection and preprocessing steps in enhancing model performance. Such research is valuable for both travelers and airlines, as it aids in optimizing pricing strategies and assists travelers in making informed decisions about when to book flights for the best value. Additionally, the study may shed light on the practical implications and challenges associated with implementing machine learning models in the dynamic airline industry.

[4] "Airfare Prediction Using Machine Learning Algorithms: A Case Study of Domestic Flights in the United States" Authors: Nizar El Hachemi, et al. Published Year: 2020.

is a critical area of research in the aviation and travel industry. This field employs various machine learning techniques to forecast airfare prices accurately, benefiting both airlines and travelers. Machine learning algorithms, including Decision Trees, Random Forest, Linear Regression, and Neural Networks, are applied to historical flight data, which typically includes factors like departure and arrival locations, dates, times, airline carriers, and booking classes. These algorithms analyze these data points to identify patterns, trends, and correlations that influence airfare fluctuations.

The predictive models generated through these algorithms can offer valuable insights. For airlines, they aid in optimizing pricing strategies, ensuring competitive fares while maximizing revenue. For travelers, they provide a means to make informed decisions, such as choosing the best time to book a flight or selecting alternative airports or dates to secure cost-effective tickets. Moreover, ongoing advancements in machine learning and access to large datasets enable the development of more accurate and dynamic pricing models, making airfare prediction a continually evolving and indispensable field within the airline industry.

[5] **"Flight Price Prediction Using Machine Learning Techniques and Data Mining" Authors: Nirmala Sivakumar, et al. Published Year: 2020**.

This research focuses on using machine learning and data mining to predict flight ticket prices, aiming to improve airline pricing strategies and help travelers make informed decisions. Machine learning techniques, such as regression models, decision trees, random forests, and neural networks, analyze historical pricing data, considering factors like booking time, routes, seasonality, and economic variables. Data mining uncovers patterns and insights from large datasets. The goal is to optimize airline revenue management by adjusting prices based on market conditions, while also providing travelers with cost-effective options. This approach enhances both airline operations and the travel experience.

CHAPTER 3

SYSTEM ANALYSIS

3.1 Problem Statement:

The problem at hand is the inability of existing flight ticket pricing systems to accurately predict and adapt to the constantly changing market dynamics in the airline industry. Current pricing models often lack the sophistication needed to consider various influential factors, resulting in suboptimal pricing for both airlines and travelers. Consequently, there is a pressing need for the development of more robust and precise machine learning-based models that can effectively predict flight prices, thereby benefiting both airlines seeking revenue optimization and passengers in search of cost-effective travel options.

3.2 Problem Description:

The problem at hand is the inability of existing flight ticket pricing systems to accurately predict and adapt to the constantly changing market dynamics in the airline industry. Current pricing models often lack the sophistication needed to consider various influential factors, resulting in suboptimal pricing for both airlines and travelers. Consequently, there is a pressing need for the development of more robust and precise machine learning-based models that can effectively predict flight prices, thereby benefiting both airlines seeking revenue optimization and passengers in search of cost-effective travel options.

3.3 Existing System:

In the realm of flight ticket pricing, existing Logistic Regression models are primarily used for binary classification tasks, such as determining whether a customer will purchase a ticket or not. These models rely on historical data and specific features to estimate the likelihood of a customer making a booking. However, Logistic Regression may not fully capture the nuances of continuous price prediction, as it's more suited for categorical outcomes. More sophisticated machine learning techniques are sought after to enhance flight price forecasting precision.

3.4 Disadvantages of Existing System:

1. **Limited Complexity**: Logistic Regression lacks the complexity to model intricate patterns in price fluctuations, leading to oversimplified predictions.
2. **Noisy Data Handling**: It struggles to handle noisy or incomplete data, reducing accuracy in the presence of data imperfections.
3. **Nonlinear Relationships**: Inability to capture nonlinear relationships between input features and flight prices, limiting its predictive power.
4. **Assumes Independence:** Logistic Regression assumes feature independence, which may not hold in the dynamic world of flight pricing.
5. **Outliers Impact:** Sensitivity to outliers can distort predictions, making it less robust in scenarios with extreme price variations.

3.5 Proposed System

The proposed system aims to enhance flight price prediction in the dynamic airline industry through a comprehensive analysis of three machine learning algorithms: Decision Trees, Random Forest. By evaluating their performance using real-world flight pricing data based on metrics like r2 score, mean squared error, and mean absolute error, this research will

provide valuable insights into each method's strengths and weaknesses. Additionally, feature importance analysis will uncover key pricing factors. These findings will empower stakeholders, including travelers and airlines, to make more informed decisions and improve pricing strategies in the aviation industry.

3.6 Advantages of Proposed System

1 **Enhanced Predictive Score:** Machine learning algorithms improve flight price prediction accuracy, benefiting both travelers and airlines.

2 **Data-Driven Insights**: Real-world data analysis provides actionable insights for informed decision-making in the aviation sector.

3 **Optimized Revenue Management**: Airlines can fine-tune pricing strategies, maximizing profitability through data-driven approaches.

4 **Cost-Effective Travel**: Travelers can access cost-effective flight options, saving money on their journeys.

5 **Competitive Edge**: Airlines gain a competitive edge by offering well-priced tickets based on accurate predictions, attracting more customers.

- **Workflow of Proposed System**

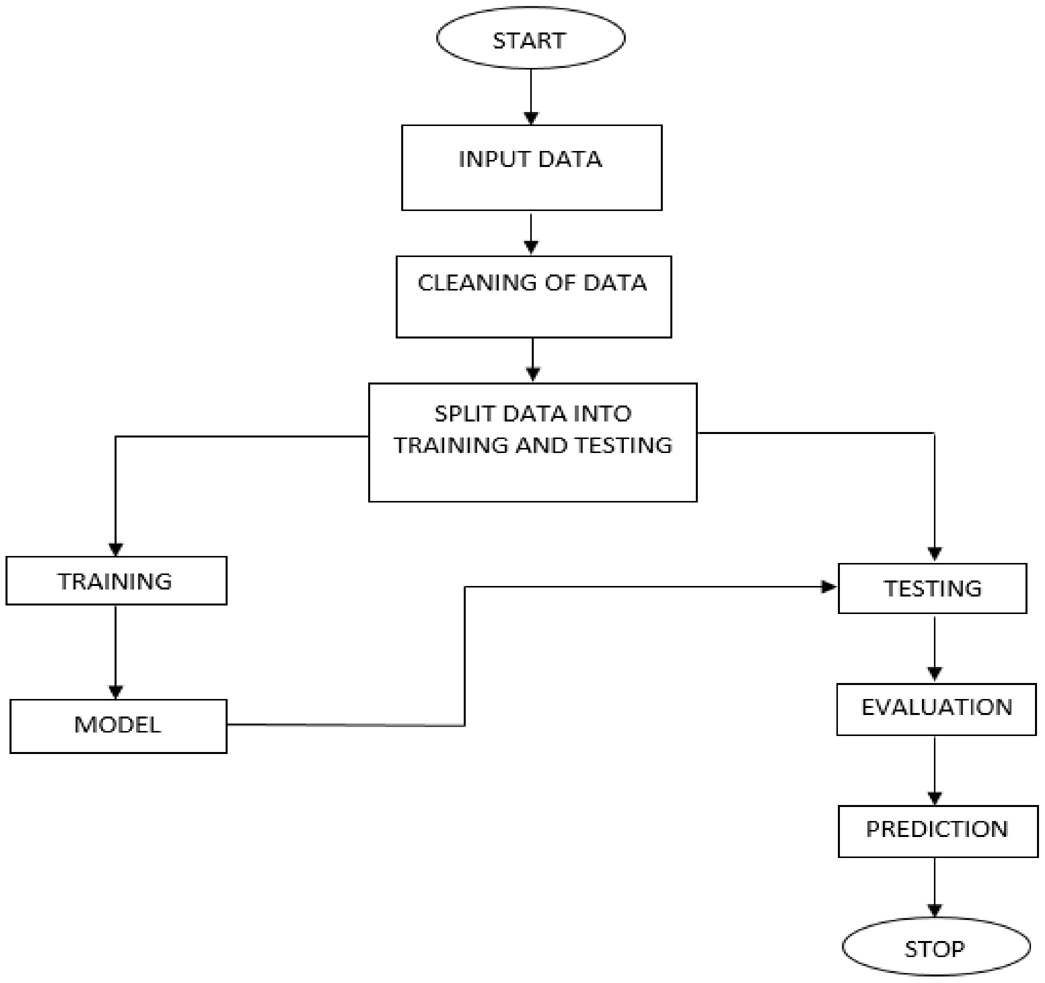


Figure 3.1 *Workflow Diagram*

CHAPTER -4

SYSTEM REQUIREMENTS

4.1 Hardware Requirements

➢ Processor	- I7/Intel Processor
➢ Hard Disk	- 160GB
➢ Keyboard	- Standard Windows Keyboard
➢ Mouse	- Two or Three Button Mouse
➢ Monitor	- SVGA
➢ RAM	- 8GB

4.2 Software Requirements

Operating System	: Windows 11
Server-side Script	: HTML, CSS, Bootstrap & JS
Programming Language	: Python
Libraries	: Django, Pandas, Mysql.connector, Os, Smtplib, Numpy
IDE/Workbench	: PyCharm
Technology	: Python 3.6+

4.3 Feasibility Study

The feasibility of the project is analyzed in this phase and the business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- Economic feasibility

- Technical feasibility
- Social feasibility

Economic Feasibility: This study is carried out to check the economic impact that the system will have on the organization. The amount of funds that the company can pour into the research and development of the system is limited. The expenditure must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only customized products had to be purchased.

Technical Feasibility: This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

Social Feasibility: The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 5

SYSTEM DEVELOPMENT

5.1 System Design

Introduction of Input Design:

In an information system, input is the raw data that is processed to produce output. During the input design, the developers must consider input devices such as PC, MICR, OMR, etc.

Therefore, the quality of system input determines the quality of system output. Well-designed input forms and screens have following properties −

- It should serve specific purpose effectively such as storing, recording, and retrieving the information.
- It ensures proper completion with accuracy.
- It should be easy to fill and straightforward.
- It should focus on user's attention, consistency, and simplicity.

Objectives for Input Design:

The objectives of input design are −

- To design data entry and input procedures
- To reduce input volume
- To design input data records, data entry screens, user interface screens, etc.
- To use validation checks and develop effective input controls.

Output Design:

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed and consider the necessary output controls and prototype report layouts.

Objectives of Output Design:

The objectives of input design are:

- Developing output design that serves the intended purpose and eliminates the production of unwanted output.
- To develop the output design that meets the end user's requirements.
- To deliver the appropriate quantity of output.
- To form the output in appropriate format and direct it to the right person.
- To make the output available on time for making good decisions.

5.1.1 System Architecture

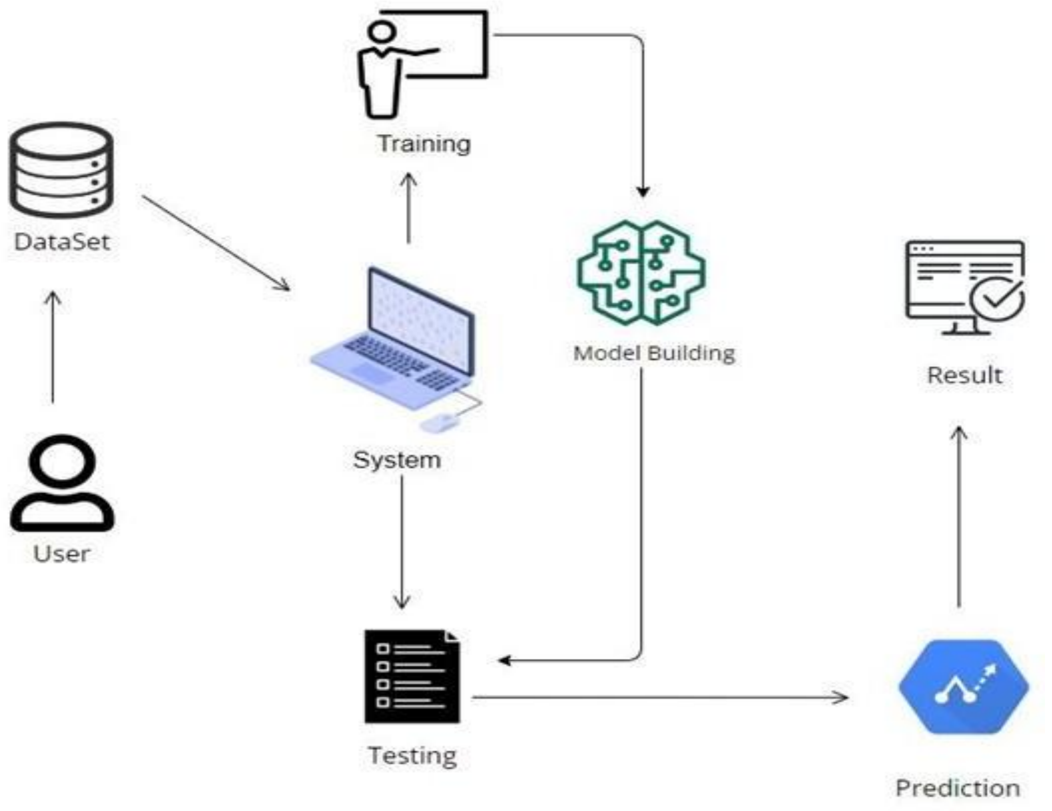


Figure 5.1 *Architecture*

5.1.2 Modules

5.1.2.1 User:

5.1.2.2 Register:

- Users can register for the Flight price Prediction web application here.
- Login:
- After registering, the user can access his portal.
- View Data:
- View data after preprocessing (cleaned dataset)
- Input:
- User will give the input values.
- Result History:

After giving the inputs, the model will predict the result which it was set according to performance, it will predict that the flight transform price.

Take Dataset:

- The dataset for the flight data is collected from the kaggle website (kaggle.com). The size of the overall dataset is 248 KB (2,53,952 bytes)

Pre-processing:

- In preprocessing first of all we will check whether there is any Nan values.
- If any Nan values is present, we will fill the Nan values with different fillna techniques like bfill, ffill, mode, and mean.
- Here we used the ffill (front fill) technique on our project.

Training the data:

- Irrespective of the algorithm we select the training is the same for every algorithm**.**

Given a dataset we split the data into two parts training and testing, the reason behind doing this is to test our model/algorithm performance just like the exams for a student the testing is also exam for the model.

We can split data into anything we want but it is just good practice to split the data such that the training has more data than the testing data, we generally split the data.

And for training and testing there are two variables X and Y in each of them, the X is the features that we use to predict the Y target and same for the testing also.

Then we call the .fit ( ) method on any given algorithm which takes two parameters i.e., X and Y for calculating the math and after that when we call the .predict ( ) giving our testing X as parameter and checking it with the accuracy score giving the testing Y and predicted X as the two parameters will get our accuracy score and same steps , these are just checking for how good our model performed on a given dataset

5.1.3 Algorithms

Decision Tree:

A Decision Tree is a fundamental and widely used machine learning algorithm that belongs to the category of supervised learning methods. It is a versatile tool employed for both classification and regression tasks, offering a structured and intuitive approach to decision-making based on data.

The fundamental concept behind a Decision Tree is to create a tree-like structure where each internal node represents a feature or attribute, each branch signifies a decision or rule, and each leaf node represents the outcome or classification result. The process of constructing a Decision Tree involves selecting the most informative feature at each step to split the data into subsets that are as homogenous as possible in terms of the target variable. This split-and-segment process continues recursively until a predefined stopping criterion is met, such as a maximum tree depth or a minimum number of samples per leaf. Decision Trees are appreciated for their simplicity and interpretability. They provide a clear visualization of the decision-making process, making it easy to understand how input features influence the final prediction. Moreover, Decision Trees can handle both categorical and numerical data, making them applicable to a wide range of problems.

However, Decision Trees are prone to overfitting, especially when the tree becomes excessively deep and complex. To mitigate this, methods like Random

Forests, which combine multiple Decision Trees, are often used.

Random Forest:

A random forest is a machine learning technique that's used to solve regression and classification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems.

A random forest algorithm consists of many decision trees. The 'forest' generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms.

The random forest algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or means of the output from various trees. Increasing the number of trees increases the precision of the outcome. A random forest eradicates the limitations of a decision tree algorithm. It reduces the overfitting of datasets and increases precision. It generates predictions without requiring many configurations in packages (like Scikit-learn).

Features of a Random Forest Algorithm:

- It's more accurate than the decision tree algorithm.
- It provides an effective way of handling missing data.
- It can produce a reasonable prediction without hyper-parameter tuning.
- It solves the issue of over-fitting in decision trees.

Decision trees are the building blocks of a random forest algorithm. A decision tree is a decision support technique that forms a tree-like structure. An overview of decision trees will help us understand how random forest algorithms work.

A decision tree consists of three components: decision nodes, leaf nodes, and a root node. A decision tree algorithm divides a training dataset into branches, which further segregate into other branches. This sequence continues until a leaf node is attained. The leaf node cannot be segregated further.

The nodes in the decision tree represent attributes that are used for predicting the outcome. Decision nodes provide a link to the leaves. The following diagram shows the three types of nodes in a decision tree.

Entropy is a metric for calculating uncertainty. Information gain is a measure of how uncertainty in the target variable is reduced, given a set of independent variables.

Information gain is used in the training of decision trees.
Applying decision trees in random forest:

The main difference between the decision tree algorithm and the random forest algorithm is that establishing root nodes and segregating nodes is done randomly in the latter. The random forest employs the bagging method to generate the required prediction.

Bagging involves using different samples of data (training data) rather than just one sample. A training dataset comprises observations and features that are used for making predictions. The decision trees produce different outputs, depending on the training data fed to the random forest algorithm. These outputs will be ranked, and the highest will be selected as the final output.

Our first example can still be used to explain how random forests work. Instead of having a single decision tree, the random forest will have many decision trees. Let's assume we have only four decision trees. In this case, the training data comprising the phone's observations and features will be divided into four root nodes.

The root nodes could represent four features that could influence the customer's choice (price, internal storage, camera, and RAM). The random forest will split the nodes by selecting features randomly. The final prediction will be selected based on the outcome of the four trees.

The outcome chosen by most decision trees will be the final choice. If three trees predict buying, and one tree predicts not buying, then the final prediction will be buying. In this case, it's predicted that the customer will buy the phone.

Logistic Regression:

Logistic regression is a popular statistical method used in regression projects, particularly when dealing with binary or categorical outcomes. Unlike traditional linear regression, which aims to predict continuous numeric values, logistic regression is designed to model the probability of a binary event occurring. It is widely employed in various fields, such as healthcare, finance, marketing, and social sciences, where understanding and predicting binary outcomes are crucial.

The fundamental concept behind logistic regression lies in the use of the logistic function, also known as the sigmoid function, to transform the linear equation into a range between 0 and 1. This transformation allows us to interpret the output as a probability value. The model estimates the probability that an instance belongs to a specific class based on its input futures.

During the model training process, the algorithm optimizes the coefficients (weights) for each feature to minimize the error between predicted probabilities and actual outcomes in the training data. This process is typically accomplished using optimization techniques like gradient descent.

Interpreting the coefficients in logistic regression is essential to understand the influence of each feature on the probability of the target class occurrence. Positive coefficients indicate that an increase in the corresponding feature raises the probability of the event happening, while negative coefficients suggest the opposite. In regression projects, logistic regression can be used for tasks such as predicting customer churn, determining the likelihood of a disease diagnosis, fraud detection, and sentiment analysis. However, it's crucial to ensure data preprocessing, feature engineering, and model evaluation are appropriately carried out to achieve reliable and accurate results. Additionally, considering other advanced models like support vector machines or decision trees might be beneficial depending on the complexity of the problem at hand.

LASSO:

Lasso regression, short for "Least Absolute Shrinkage and Selection Operator," is a popular linear regression technique used in statistics and machine learning. It addresses the issue of multicollinearity and feature selection by adding a regularization term to the linear regression equation. The key idea behind Lasso is to penalize the absolute values of the regression coefficients, encouraging some of them to be exactly zero. This property makes Lasso useful for feature selection, helping to identify the most relevant predictors in a dataset while simultaneously reducing overfitting.

The strength of the regularization term is controlled by a hyperparameter, often denoted as lambda (λ). By adjusting λ, you can strike a balance between model simplicity and predictive accuracy. Lasso regression is particularly valuable when dealing with high-dimensional datasets, as it effectively simplifies the model by eliminating less important features. Its versatility and interpretability have made it a valuable tool in various fields, including economics, biology, and data science.

MLP

Multilayer Perceptron (MLP) regression is a type of artificial neural network used for supervised learning tasks, particularly in regression problems. It consists of multiple layers of interconnected nodes or neurons, where each neuron is connected to the neurons in the adjacent layers. In MLP regression, the input layer receives the features of the dataset, and these features are passed through one or more hidden layers before reaching the output layer. The goal of MLP regression is to learn a mapping from the input data to a continuous target variable. This is achieved through a training process that involves adjusting the weights and biases of the network to minimize the difference between the predicted values and the actual target values.

MLP regression is a versatile and powerful tool for modeling complex, non-linear relationships in data, making it a valuable technique in various fields, including finance, healthcare, and engineering. It is often used in combination with optimization algorithms like gradient descent and activation functions such as ReLU to enhance its learning capabilities. Hyperparameter tuning, regularization techniques, and cross-validation are crucial in fine-tuning MLP regression models to ensure optimal performance and avoid overfitting.

5.1.4 UML Diagrams

5.1.4.1 Introduction to UML

The Unified Modelling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modelling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

Following are the fundamental concepts of the object-oriented world −

- Objects − Objects represent an entity and the basic building block.
- Class − Class is the blueprint of an object.
- Abstraction − Abstraction represents the behaviour of a real world entity.
- Encapsulation − Encapsulation is the mechanism of binding the data together

and hiding them from the outside world.

- Inheritance − Inheritance is the mechanism of making new classes from existing ones.
- Polymorphism − It defines the mechanism to exists in different forms. In this project, basic UML diagrams have been explained.

1. Class Diagram
2. Use Case Diagram
3. Sequence Diagram
4. Activity Diagram
5. Deployment Diagram

5.1.4.2 Use Case Diagram:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

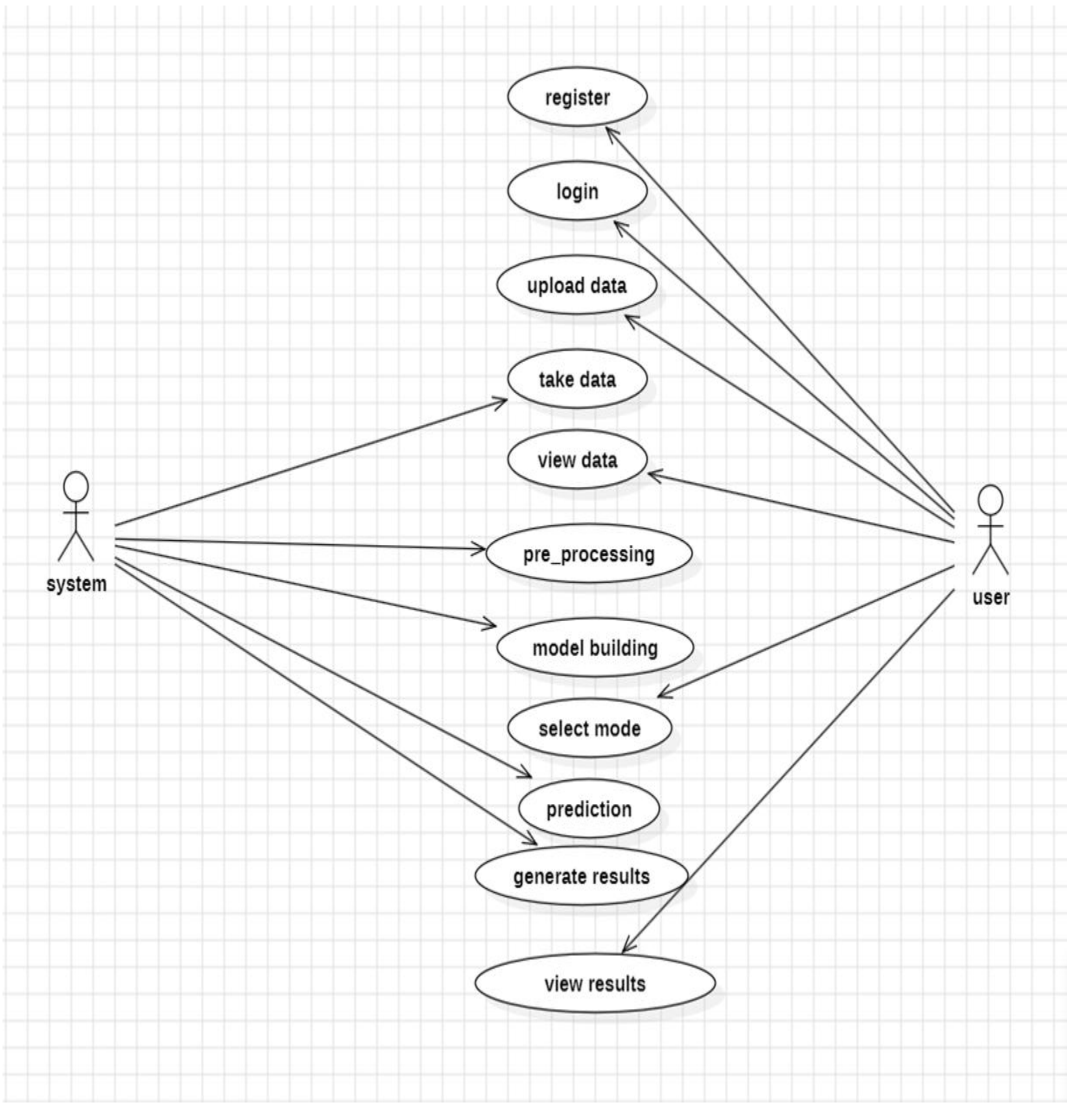


Figure 5.2 *Use Case Diagrams*

5.1.4.3 Class Diagram:

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

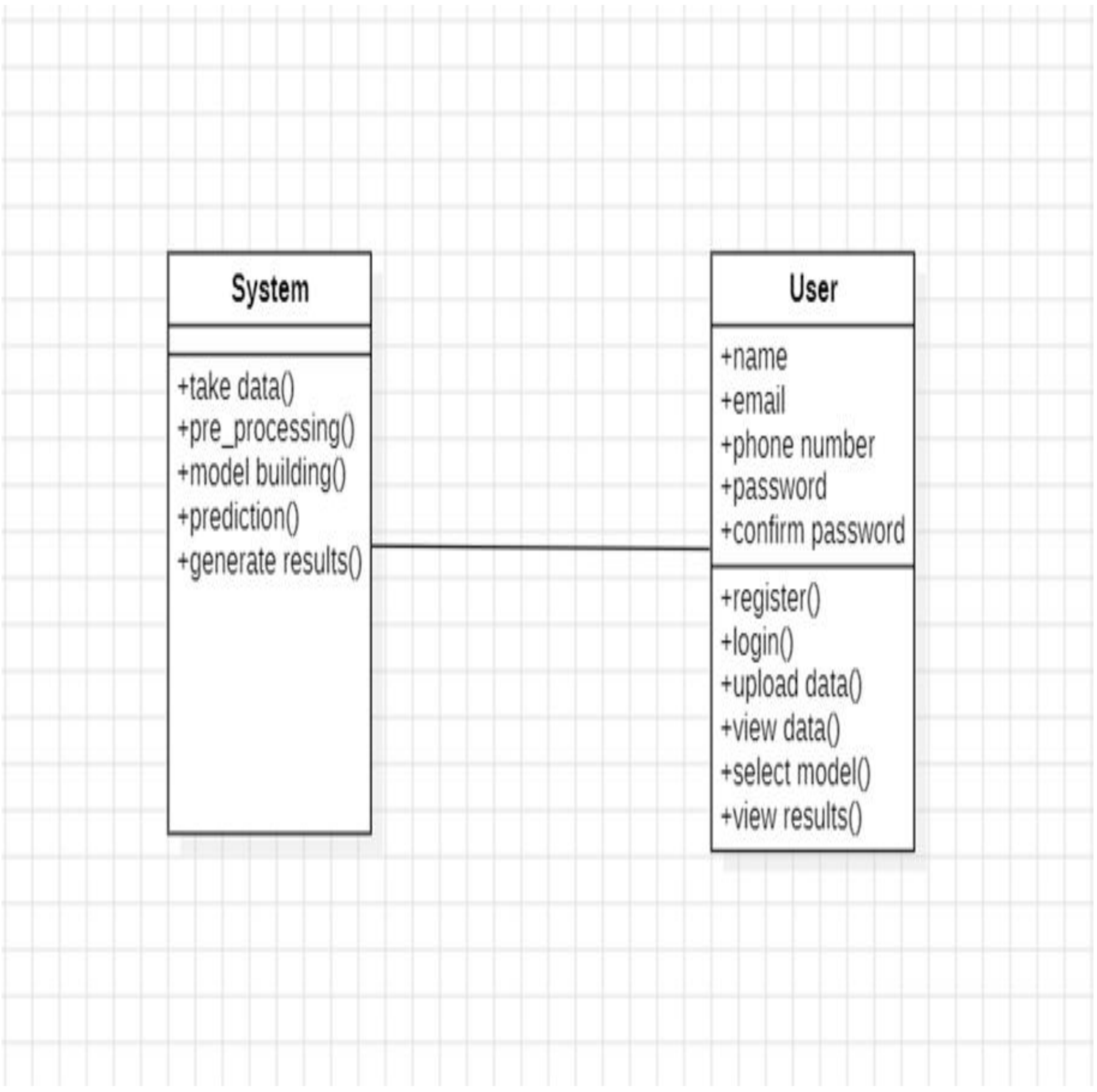


Figure 5.3 *Class Diagram*

5.1.4.4 Sequence Diagram:

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is the construction of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

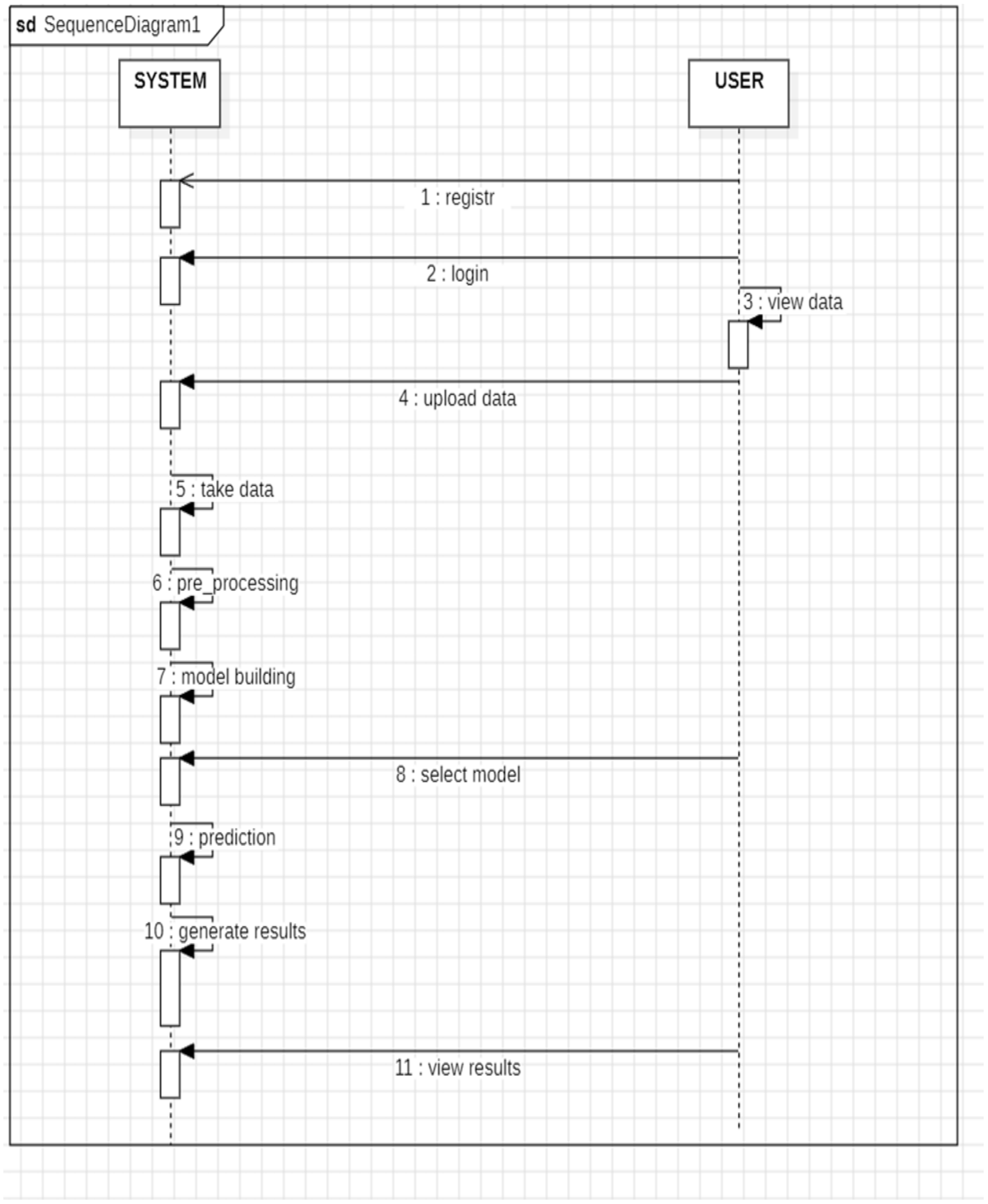


Figure 5.4 *Sequence Diagram*

5.1.4.5 Collaboration Diagram:

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization.

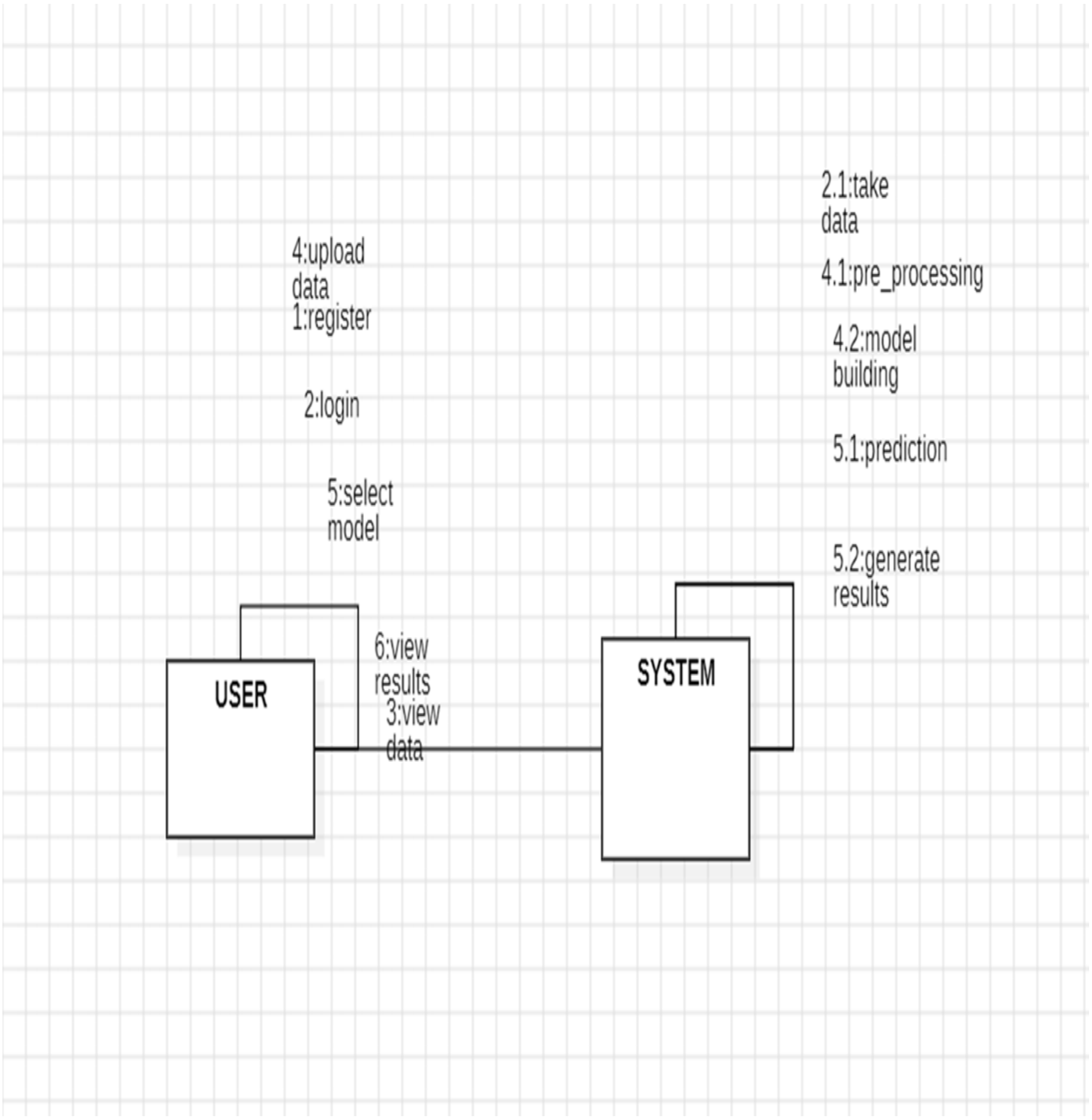


Figure 5.5 *Collaboration Diagram*

5.1.4.6 Deployment Diagram

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware's used to deploy the application.

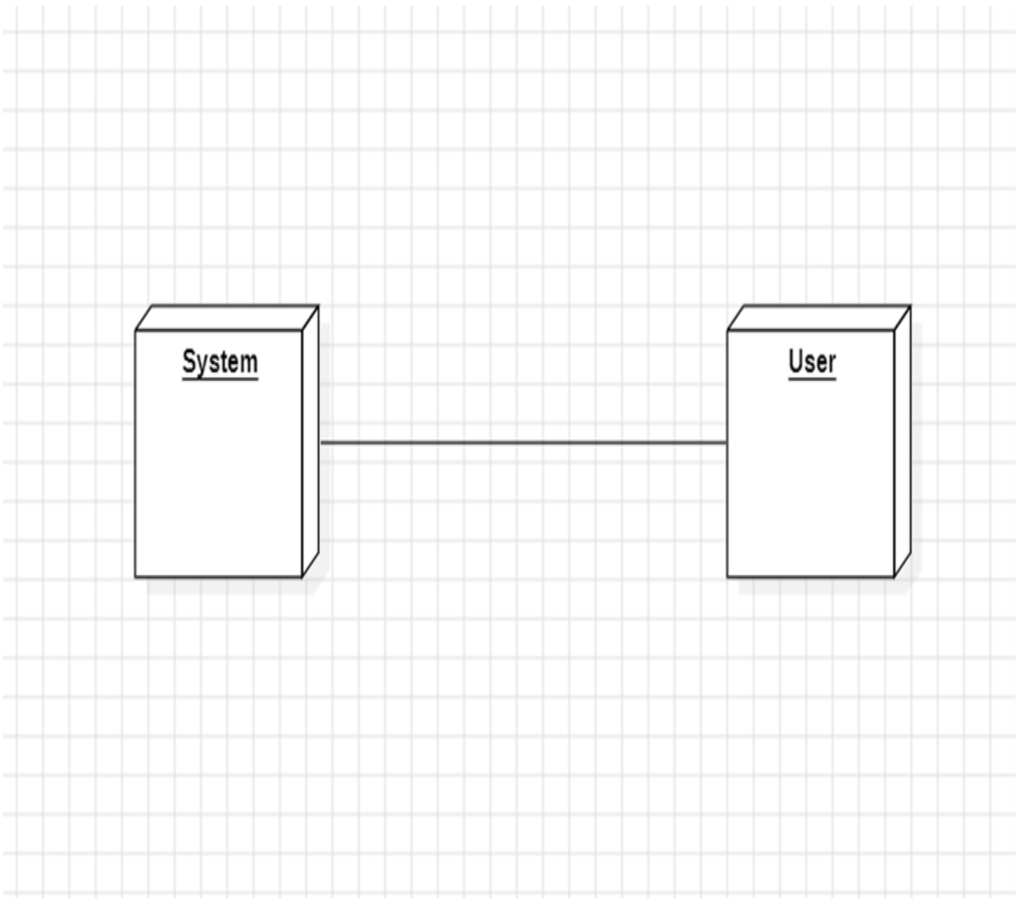


Figure 5.6 *Deployment Diagram*

5.1.4.7 Activity Diagram:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

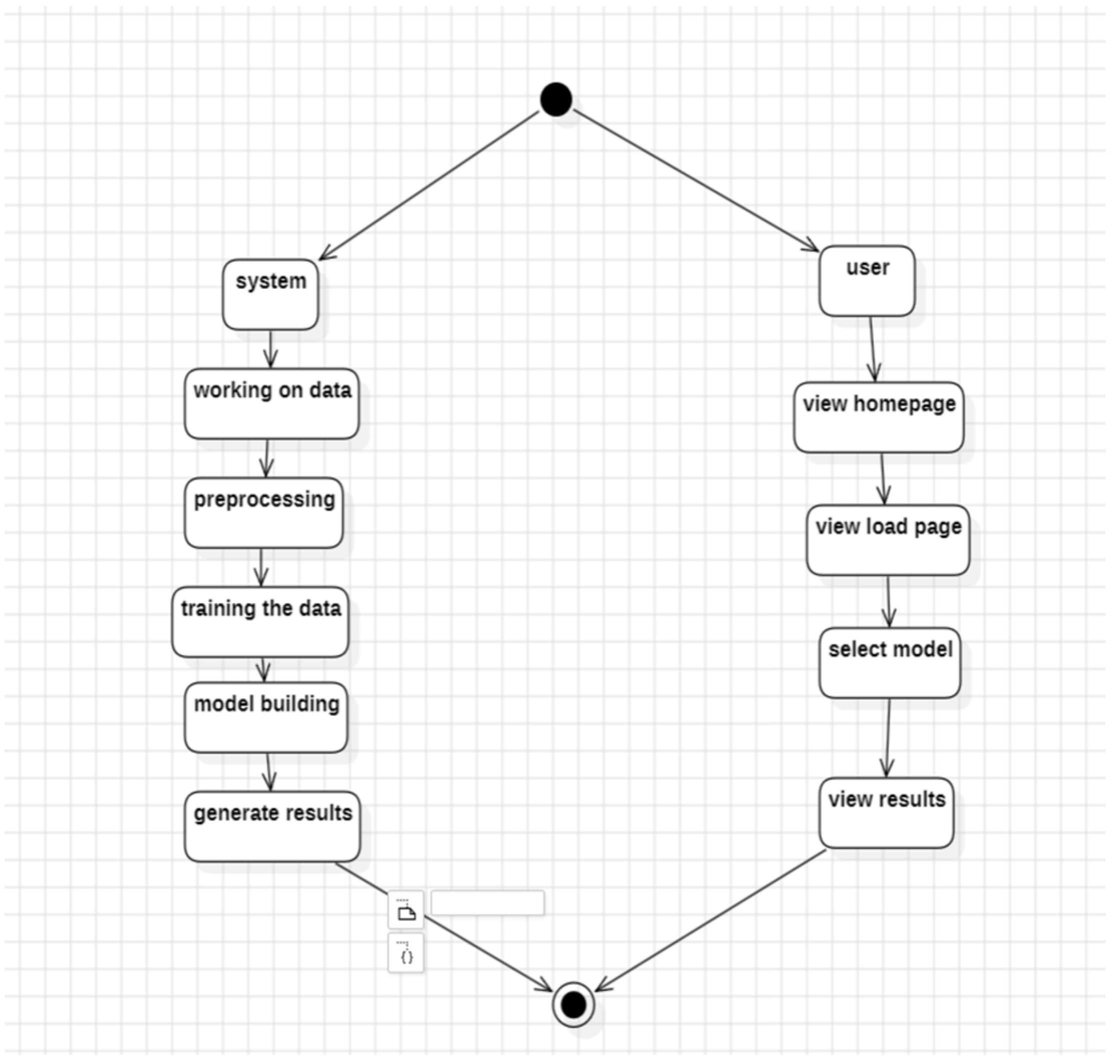


Figure 5.7 *Activity Diagram*

5.1.4.8 Component Diagram:

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the required function is covered by planned development.

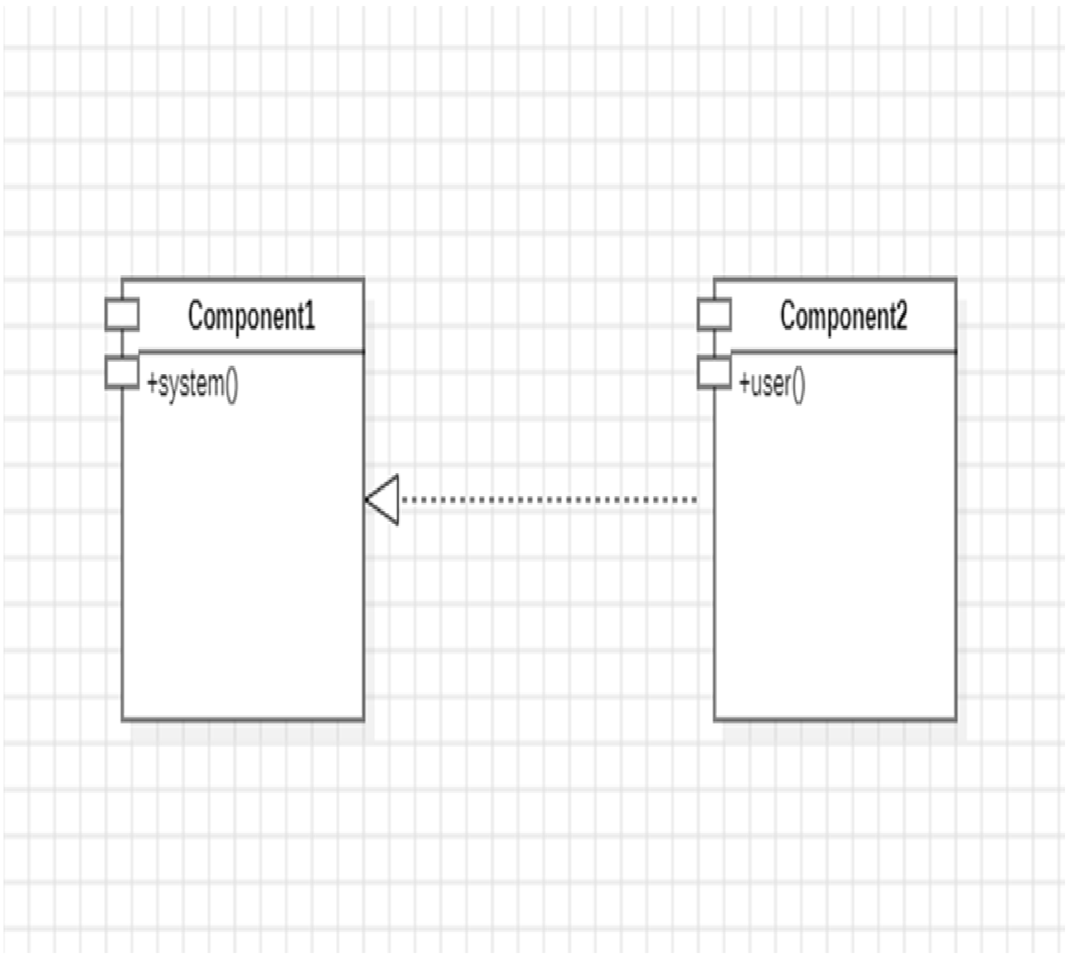


Figure 5.8 *Component Diagram*

5.2 System Implementation

5.2.1 Software Description:

Python:

Python is a very popular general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python is dynamically typed and garbage-collected programming language. It was created by Guido van Rossum during 1985-1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). Today, Python is very high in demand and all the major companies are looking for great Python Programmers to develop websites, software components, and applications or to work with Data Science, AI, and ML technologies. When we are developing this tutorial in 2022, there is a high shortage of Python Programmers whereas market demands more number of Python Programmers due to its application in Machine Learning, Artificial Intelligence etc.

What Is Script?

Up to this point, I have concentrated on the interactive programming capability of Python. This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

Scripts are reusable

Basically, a script is a text file containing the statements that comprise a Python program. Once you have created the script, you can execute it over and over without having to retype it each time.

Scripts are editable

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor. Then you can execute each of the individual versions. In this way, it is easy to create different programs with a minimum amount of typing.

You will need a text editor

Just about any text editor will suffice for creating Python script files.

You can use Microsoft Notepad, Microsoft WordPad, Microsoft Word, or just about any word processor if you want to.

Difference between a script and a program

Script:

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, whereas the applications they control are traditionally compiled to native machine code.

Program:

The program has an executable form that the computer can use directly to execute the instructions.

The same program in its human-readable source code form, from which executable programs are derived (e.g., compiled)

Python

What is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don't know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to "big name" languages. Hopefully I can explain it for you.

Python concepts

If you're not interested in the how and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it's a great one to start programming with.

- Open-source general-purpose language.

- Object Oriented, Procedural, Functional
- Easy to interface with C/ObjC/Java/Fortran
- Easy-is to interface with C++ (via SWIG)
- Great interactive environment

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactical constructions than other languages.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Possum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol68, Smalltalk, and UNIX shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Possum still holds a vital role in directing its progress.

Python Features

Python's features include

- Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- Easy-to-read − Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain − Python's source code is easy-to-maintain.
- A broad standard library − Python's bulk of the library is very portable and cross platform compatible on UNIX, Windows, and Macintosh.
- Interactive Mode − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- Extendable − you can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- Databases − Python provides interfaces to all major commercial databases.
- GUI Programming − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- Scalable − Python provides a better structure and support for large programs than shell scripting.
- Apart from the above-mentioned features, Python has a big list of good features, few are listed below −
- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to bytecode for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Dynamic vs. Static

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of "thing" each data value is.

- For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a "float" type.
- This tells the compiler that the only data that can be used for that variable must be a floating-point number, i.e. a number with a decimal point.
- If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.
- Python, however, doesn't require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating-point number) you need in your program.
- With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating-point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).
- If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.
- With Python, it doesn't matter. You simply give it whatever number you want, and Python will take care of manipulating it as needed. It even works for derived values.
- For example, say you are dividing two numbers. One is a floating-point number, and one is an integer. Python realizes that it's more accurate to keep track of decimals, so it automatically calculates the result as a floating-point number

Variables

- Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.
- Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables**.**

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them. Python has five standard data types

- NumPy
- String
- List
- Tuple
- Dictionary

Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them

Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of several values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as read-only lists.

Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

Different modes in python

Python has two basic modes: normal and interactive.

The normal mode is the mode where the scripted and finished .pie files are run in the Python interpreter.

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

5.2.2 Hardware Description:

RAM:

Random Access Memory is volatile. That means data is retained in RAM as long as the computer is on, but it is lost when the computer is turned off. When the computer is rebooted, the OS and other files are reloaded into RAM, usually from an HDD or SSD. Because of its volatility, RAM can't store permanent data. RAM can be compared to a person's short-term memory, and a hard disk drive to a person's long-term memory. Short-term memory is focused on immediate work, but it can only keep a limited number of facts in view at any one time. When a person's short-term memory fills up, it can be refreshed with facts stored in the brain's long-term memory.

A computer also works this way. If RAM fills up, the computer's processor must repeatedly go to the hard disk to overlay the old data in RAM with new data. This process slows the computer's operation.

Hard disk

Hard disk, also called hard disk drive or hard drive.A computer's hard drive is a device consisting of several hard disks, read/write heads, a drive motor to spin the disks, and a small amount of circuitry, all sealed in a metal case to protect the disks from dust. In addition to referring to the disks themselves, the term hard disk is also used to refer to the whole of a computer's internal data storage. Beginning in the early 21st century, some personal computers and laptops were produced that used solid-state drives (SSDs) that relied on flash memory chips instead of hard disks to store information

5.2.3 Database Description

A database is an organized collection of structured data that is stored and managed in a way that allows for efficient retrieval, manipulation, and management of information. Here's a more detailed description of what a database is and its key components:

- Data: A database stores data, which can be in various forms such as text, numbers, images, audio, or video. This data is typically organized into tables or collections, with each table or collection containing records (rows) and fields (columns).
- Tables: Tables are the fundamental data structures in a database. They consist of rows (records or tuples) and columns (fields or attributes). Each row represents a unique instance or entity, while columns store specific pieces of information about that entity.
- Schema: A schema is the logical structure or blueprint that defines how the data in a database is organized. It specifies the tables, fields, data types, relationships between tables, and other constraints or rules that govern the data.
- Database Management System (DBMS): A DBMS is the software that manages and controls access to the database. It provides interfaces for creating, modifying, and querying the data, as well as implementing security measures, ensuring data integrity, and facilitating backup and recovery operations.
- Query Language: To interact with the database and perform operations like retrieving, inserting, updating, or deleting data, a query language is used. The most widely used query language is SQL (Structured Query Language), but there are also other languages like NoSQL for non-relational databases.
- Data Integrity: Databases enforce data integrity rules to maintain the accuracy and consistency of the data. These rules include constraints like primary keys (unique identifiers), foreign keys (referential integrity), and other validation rules.
- Backup and Recovery: DBMS systems typically include features for creating backups of the data, as well as mechanisms for recovering data in case of system failures, errors, or disasters.
- Data Models: Databases can be based on different data models, such as relational (SQL), hierarchical, network, object-oriented, or NoSQL (key-value, document, column-family, graph). The data model determines how the data is structured and organized.

5.2.4 Sample Code

```
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
df = pd.read_csv('flight_data1.csv')
from matplotlib import pyplot as plt
#plot the graph to check wether there are any missing value present
missing = pd.DataFrame((df.isnull().sum())*100/df.shape[0]).reset_index()
plt.figure(figsize=(16,5))
ax = sns.pointplot(x='index',y=0,data=missing)
plt.xticks(rotation =90,fontsize =7)
plt.title("Percentage of Missing values")
plt.ylabel("PERCENTAGE")
plt.show()
# Drop non-numeric columns
numeric_df = df.select_dtypes(include=['number'])
# Calculate the correlation matrix
corr_matrix = numeric_df.corr()
# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
# Get the numerical columns of the DataFrame
numerical_columns = df.select_dtypes(include=['float64', 'int64']).columns
# Create a boxplot for each numerical column
for column in numerical_columns[:5]:
    plt.figure()  # Create a new figure for each boxplot
    sns.boxplot(data=df[column])
    plt.title(f"Boxplot of {column}")
    plt.xlabel(column)
# Display the boxplots
plt.show()
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Source', hue='Destination')
plt.title('Comparison of Source and Destination')
plt.xlabel('Source')
plt.ylabel('Count')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Source', hue='Total Stops')
plt.title('Comparison of Source and Destination')
plt.xlabel('Source')
plt.ylabel('Count')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()
# Get the numerical columns of the DataFrame
numerical_columns = df.select_dtypes(include=['float64', 'int64']).columns
# Create a kdplot for each numerical column
for column in numerical_columns[:5]:
    plt.figure()  # Create a new figure for each boxplot
    sns.kdeplot(data=df[column])
    plt.title(f"Boxplot of {column}")
    plt.xlabel(column)
# Display the boxplots
plt.show()
# Convert 'Departure Date' and 'Arrival Date' to datetime
```

```
df['Departure Date'] = pd.to_datetime(df['Departure Date'], format='%Y-%m-%d')

df['Arrival Date'] = pd.to_datetime(df['Arrival Date'], format='%Y-%m-%d')

# Convert dates to integer format (number of days since '1970-01-01')

df['Departure Date'] = (df['Departure Date'] - pd.Timestamp("1970-01-01")) //
pd.Timedelta('1D')

df['Arrival Date'] = (df['Arrival Date'] - pd.Timestamp("1970-01-01")) //
pd.Timedelta('1D')

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

for i in col:

   df[i] = le.fit_transform(df[i])

from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error

from sklearn.tree import DecisionTreeRegressor

de = DecisionTreeRegressor(random_state=1)

de.fit(x_train,y_train)

de_pred = de.predict(x_test)

r2_de = r2_score (de_pred, y_test)

mae_de = mean_absolute_error(y_test, de_pred)

mse_de = mean_squared_error(y_test, de_pred)

Adj_r2 = r2_de-r2_de*2

print (f'r2 score = {r2_de}')

print (f'MAE = {mae_de}')

print (f'MSE = {mse_de}')

print (f'Adj_r2 = {Adj_r2}')

import pickle

with open ('DT.pkl', 'wb') as fp:

   pickle.dump(de, fp)

def adjusted_r2_score (y_true, y_pred, n, p):

   r2 = r2_score (y_true, y_pred)

   adjusted_r2 = 1 - ((1 - r2) * (n - 1)) / (n - p - 1)
```

```
    return adjusted_r2
# Number of data points (n) and number of features (p)
n = x_test.shape[0]
p = x_test.shape[1]
# Calculate Adjusted R² score
adjusted_r2 = adjusted_r2_score (y_test, de_pred, n, p)
adjusted_r2
from sklearn.tree import DecisionTreeRegressor
de = DecisionTreeRegressor()
de.fit(x_train,y_train)
de_pred = de.predict(x_train)
r2_de = r2_score (y_train, de_pred)
mae_de = mean_absolute_error(y_train, de_pred)
mse_de = mean_squared_error(y_train, de_pred)
Adj_r2 = r2_de-r2_de*2
print (f'r2 score = {r2_de}')
print (f'MAE = {mae_de}')
print (f'MSE = {mse_de}')
print (f'Adj_r2 = {Adj_r2}')
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor()
rf.fit(x_train,y_train)
rf_pred = rf.predict(x_train)
r2_rf = r2_score (y_train, rf_pred)
mae_rf = mean_absolute_error(y_train, rf_pred)
mse_rf = mean_squared_error(y_train, rf_pred)
Adj_r2 = r2_rf-r2_rf*2
print (f'r2 score = {r2_rf}')
print (f'MAE = {mae_rf}')
print (f'MSE = {mse_rf}')

print (f'Adj_r2 = {Adj_r2}')

from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor()

rf.fit(x_train,y_train)

rf_pred = rf.predict(x_test)

r2_rf = r2_score (y_test, rf_pred)

mae_rf = mean_absolute_error(y_test, rf_pred)

mse_rf = mean_squared_error(y_test, rf_pred)

Adj_r2 = r2_rf-r2_rf*2

print (f'r2 score = {r2_rf}')

print (f'MAE = {mae_rf}')

print (f'MSE = {mse_rf}')

print (f'Adj_r2 = {Adj_r2}')

## Prediction 934

inp = [[2, 0, 1, 0, 2, 0, 1, 18, 5, 0.941103, 0, 0]]

result = rf.predict(inp)

result

# 379

inp = [[1, 2, 0, 1, 2, 2, 1, 17, 5, 0.960137, 1, 0]]

result = rf.predict(inp)

result

from sklearn.linear_model import LinearRegression

lr = LinearRegression()

lr.fit(x_train,y_train)

lr_pred = lr.predict(x_train)

r2_lr = r2_score (y_train, lr_pred)

mae_lr = mean_absolute_error(y_train, lr_pred)

mse_lr = mean_squared_error(y_train, lr_pred)
```

```
Adj_r2 = r2_lr-r2_lr*2
print (f'r2 score = {r2_lr}')
print (f'MAE = {mae_lr}')
print (f'MSE = {mse_lr}')
print (f'Adj_r2 = {Adj_r2}')
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(x_train,y_train)
lr_pred = lr.predict(x_test)
r2_lr = r2_score (y_test, lr_pred)
mae_lr = mean_absolute_error(y_test, lr_pred)
mse_lr = mean_squared_error(y_test, lr_pred)
Adj_r2 = r2_lr-r2_lr*2
print (f'r2 score = {r2_lr}')
print (f'MAE = {mae_lr}')
print (f'MSE = {mse_lr}')
print (f'Adj_r2 = {Adj_r2}')
## Prediction 934
inp = [[2, 0, 1, 0, 2, 0, 1, 18, 5, 0.941103, 0, 0]]
result = lr.predict(inp)
result
# 379
inp = [[1, 2, 0, 1, 2, 2, 1, 17, 5, 0.960137, 1, 0]]
result = lr.predict(inp)
result
from sklearn.linear_model import Lasso
lasso = Lasso ()
lasso.fit(x_train,y_train)
lasso_pred = lasso.predict(x_train)
```

```
r2_lasso = r2_score (y_train, lasso_pred)

mae_lasso = mean_absolute_error(y_train, lasso_pred)

mse_lasso = mean_squared_error(y_train, lasso_pred)

Adj_r2 = r2_lasso-r2_lasso*2

print (f'r2 score = {r2_lasso}')

print (f'MAE = {mae_lasso}')

print (f'MSE = {mse_lasso}')

print (f'Adj_r2 = {Adj_r2}')

from sklearn.linear_model import Lasso

lasso = Lasso ()

lasso.fit(x_train,y_train)

lasso_pred = lasso.predict(x_test)

r2_lasso = r2_score (y_test, lasso_pred)

mae_lasso = mean_absolute_error(y_test, lasso_pred)

mse_lasso = mean_squared_error(y_test, lasso_pred)

Adj_r2 = r2_lasso-r2_lasso*2

print (f'r2 score = {r2_lasso}')

print (f'MAE = {mae_lasso}')

print (f'MSE = {mse_lasso}')

print (f'Adj_r2 = {Adj_r2}')

## Prediction 934

inp = [[2, 0, 1, 0, 2, 0, 1, 18, 5, 0.941103, 0, 0]]

result = lasso.predict(inp)

result

# 379

inp = [[1, 2, 0, 1, 2, 2, 1, 17, 5, 0.960137, 1, 0]]

result = lasso.predict(inp)

result

from sklearn.neural_network import MLPRegressor
```

```
mlp = MLPRegressor()
mlp.fit(x_train,y_train)
mlp_pred = mlp.predict(x_train)
r2_mlp = r2_score (y_train, mlp_pred)
mae_mlp = mean_absolute_error(y_train, mlp_pred)
mse_mlp = mean_squared_error(y_train, mlp_pred)
Adj_r2 = r2_mlp-r2_mlp*2
print (f'r2 score = {r2_mlp}')
print (f'MAE = {mae_mlp}')
print (f'MSE = {mse_mlp}')
print (f'Adj_r2 = {Adj_r2}')
from sklearn.neural_network import MLPRegressor
mlp = MLPRegressor()
mlp.fit(x_train,y_train)
mlp_pred = mlp.predict(x_test)
r2_mlp = r2_score (y_test, mlp_pred)
mae_mlp = mean_absolute_error(y_test, mlp_pred)
mse_mlp = mean_squared_error(y_test, mlp_pred)
Adj_r2 = r2_mlp-r2_mlp*2
print (f'r2 score = {r2_mlp}')
print (f'MAE = {mae_mlp}')
print (f'MSE = {mse_mlp}')
print (f'Adj_r2 = {Adj_r2}')
## Prediction 934
inp = [[2, 0, 1, 0, 2, 0, 1, 18, 5, 0.941103, 0, 0]]
result = mlp.predict(inp)
result
# 379
inp = [[1, 2, 0, 1, 2, 2, 1, 17, 5, 0.960137, 1, 0]]
```

```
result = mlp.predict(inp)
result
```

5.3 System Testing:

5.3.1 Software Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

5.3.2 Types of Tests

5.3.2.1 Unit Testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. it is done after the completion of an individual unit before integration. This is structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

5.3.2.2 Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is even driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered. **Acceptance Testing:** User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

5.3.2.3 Functional Testing:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified, and the effective value of current tests is determined.

5.3.2.4 White Box Testing:

White Box Testing is a test in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

5.3.2.5 Black Box Testing:

Black Box Testing tests the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, like most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a test in which the software under test is treated, as a black box. you cannot "see" into it. The test provides input and responds to outputs without considering how the software works.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page

5.3.3 Test Cases:

Table 5.1 *Test Cases*

Input	**Output**	**Result**
Input	Tested for different models given by users on the different model.	Success
Decision Tree	Tested for different input given by the user on different models are created using different algorithms and data.	Success
Prediction	Prediction will be performed using the different models built from the algorithms.	Success

- **Testcase Model Building:**

Table 5.2 *Test Case Model Building*

S.NO	**Test cases**	**I/O**	**Expected O/T**	**Actual O/T**	**P/F**
1	Register	Enter email, password, Username	Registration successful	Registration successfully completed	P
2	Register	Enter email, password, mobile number, gender, address	Registration successful	User Email already existed	F
3	Login	Enter valid email	Login to the Userhome page successfully	Login to the Userhome page successfully	P
4	Login	Enter invalid email	Login to the Userhome page successfully	Invalid Email	F
5	User can give a Input	Proper Input.	Algorithm can Predict a required input and generate result on Flight price	Result Generated Successfully	P
6	User can give a Input	In proper Input	Algorithm cannot Predict a specific input and	Invalid Input	F

CHAPTER- 6

RESULTS

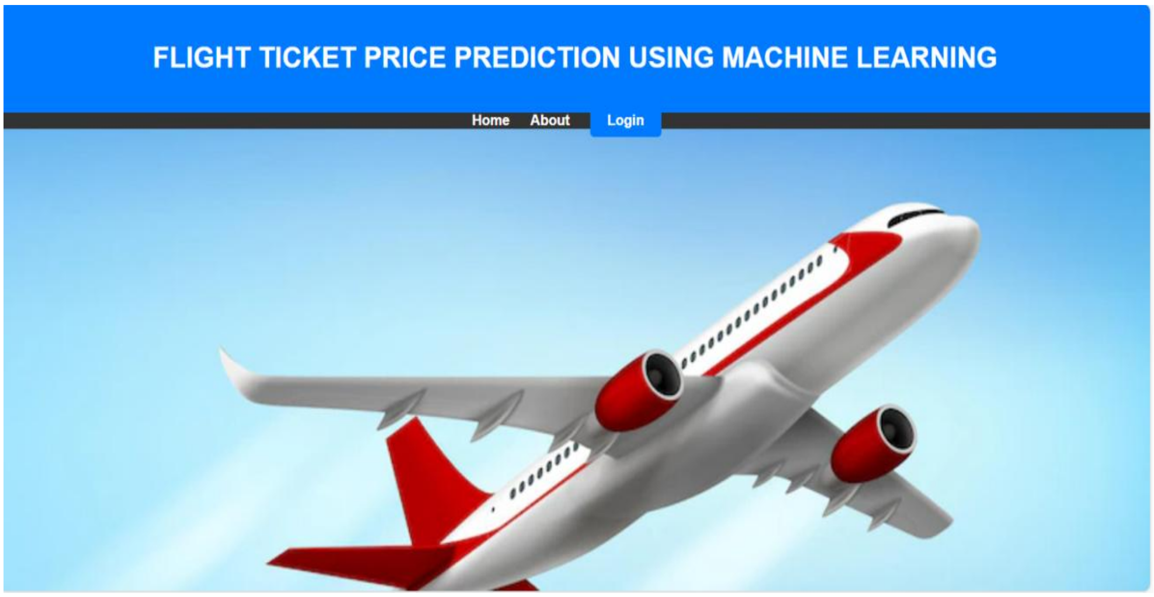
6.1 Execution Procedure

The Execution procedure is as follows:

1. Make sure Visual Studio IDE is already installed in our system.
2. Install the python Latest Version for Windows.
3. Create a Project.
4. Build an application.
5. install required packages
6. After that run the application in Visual Studio IDE.
7. It will generate a link and click it.
8. It will automatically run the application in your web.
9. And click the login button.
10. In login section you have to click the new user option.
11. And register with your mail and set a password.
12. After you can login our application.
13. And you can click train button and that section you can train the algorithms.
14. After you can click the prediction button and the necessary detail.
15. After clicking the submit button and it will show the price of the flight ticket.

6.2 Screenshots:

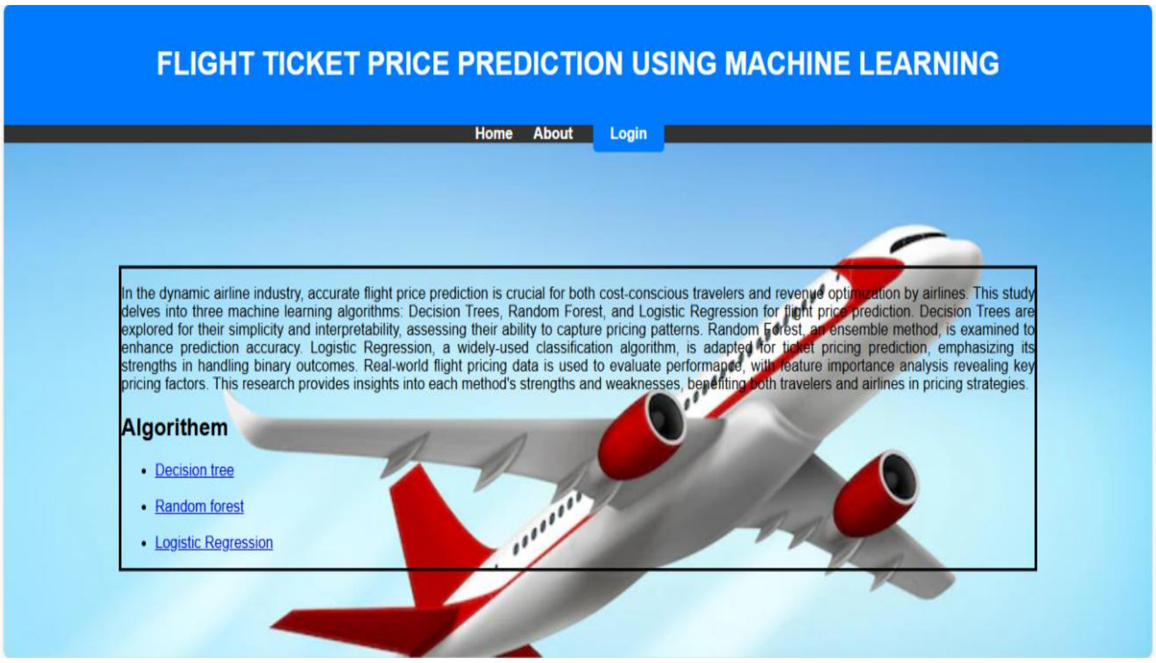
Home Page:
Here user view the home page of Flight price Prediction web application.



Description: *Home Page of Flight Ticket Price Prediction web application*

Figure 6.1 *Home Page*

About:
Here we can read about our project.



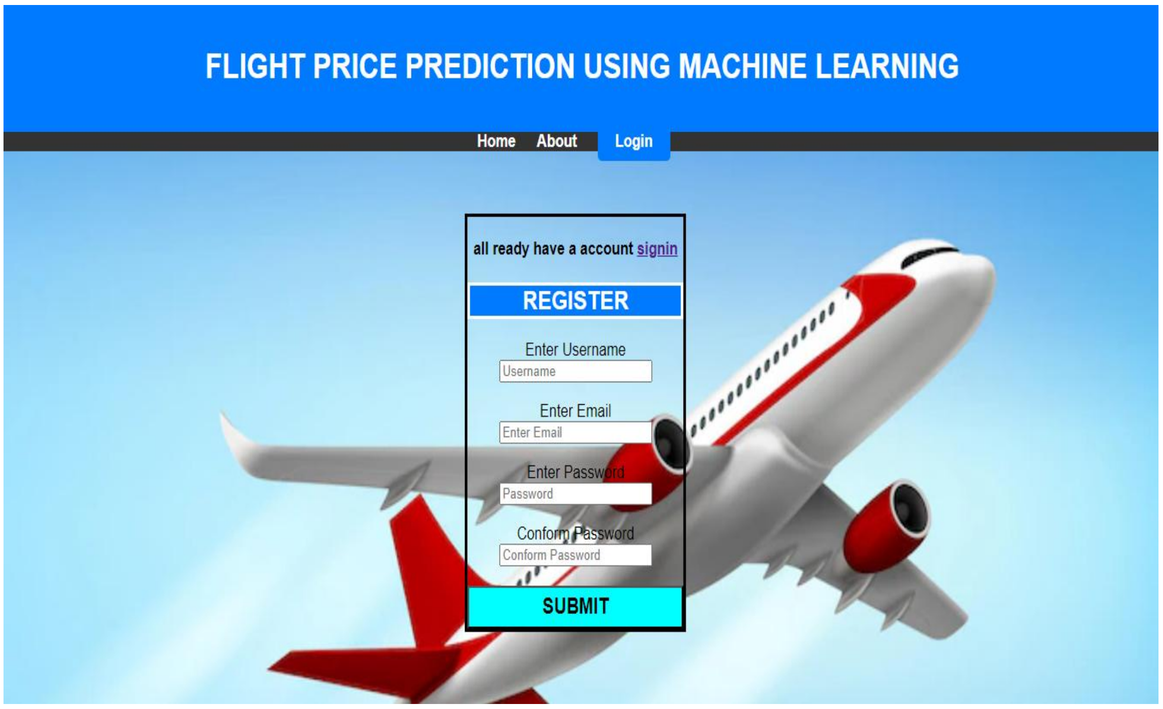
In the dynamic airline industry, accurate flight price prediction is crucial for both cost-conscious travelers and revenue optimization by airlines. This study delves into three machine learning algorithms: Decision Trees, Random Forest, and Logistic Regression for flight price prediction. Decision Trees are explored for their simplicity and interpretability, assessing their ability to capture pricing patterns. Random Forest, an ensemble method, is examined to enhance prediction accuracy. Logistic Regression, a widely-used classification algorithm, is adapted for ticket pricing prediction, emphasizing its strengths in handling binary outcomes. Real-world flight pricing data is used to evaluate performance, with feature importance analysis revealing key pricing factors. This research provides insights into each method's strengths and weaknesses, benefiting both travelers and airlines in pricing strategies.

Algorithem

- Decision tree
- Random forest
- Logistic Regression

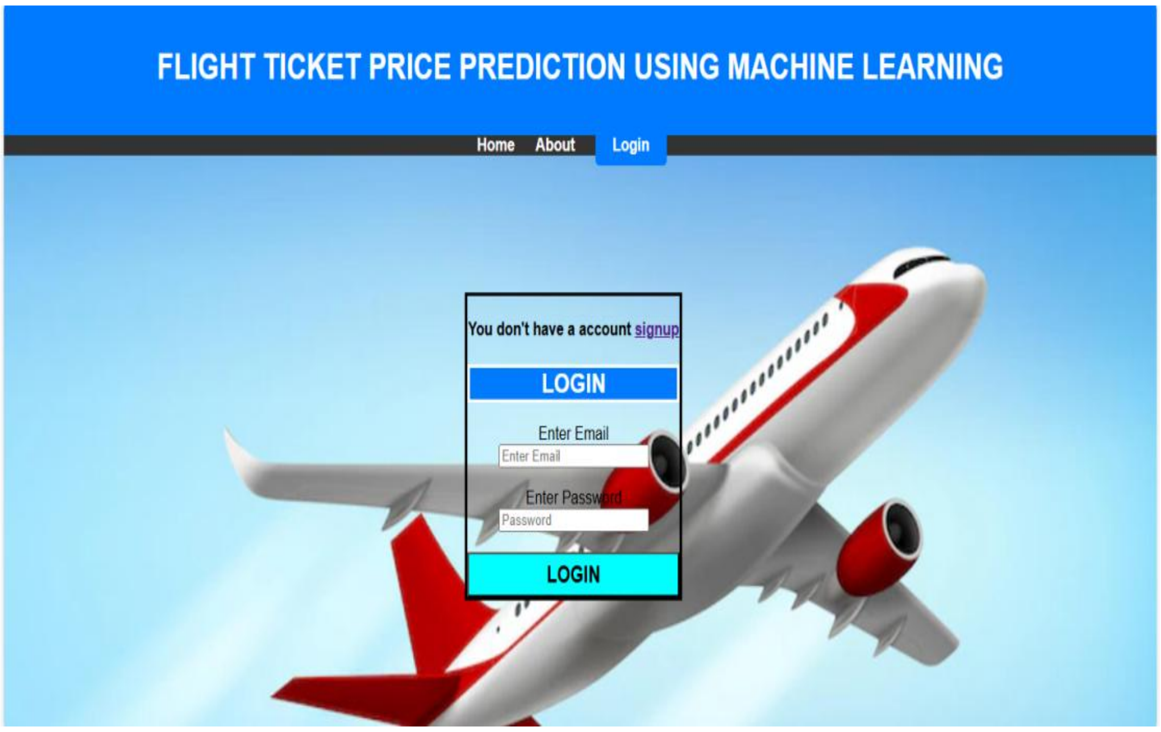
Description: *Here We can read about our project*

Figure 6.2 *About the Project*



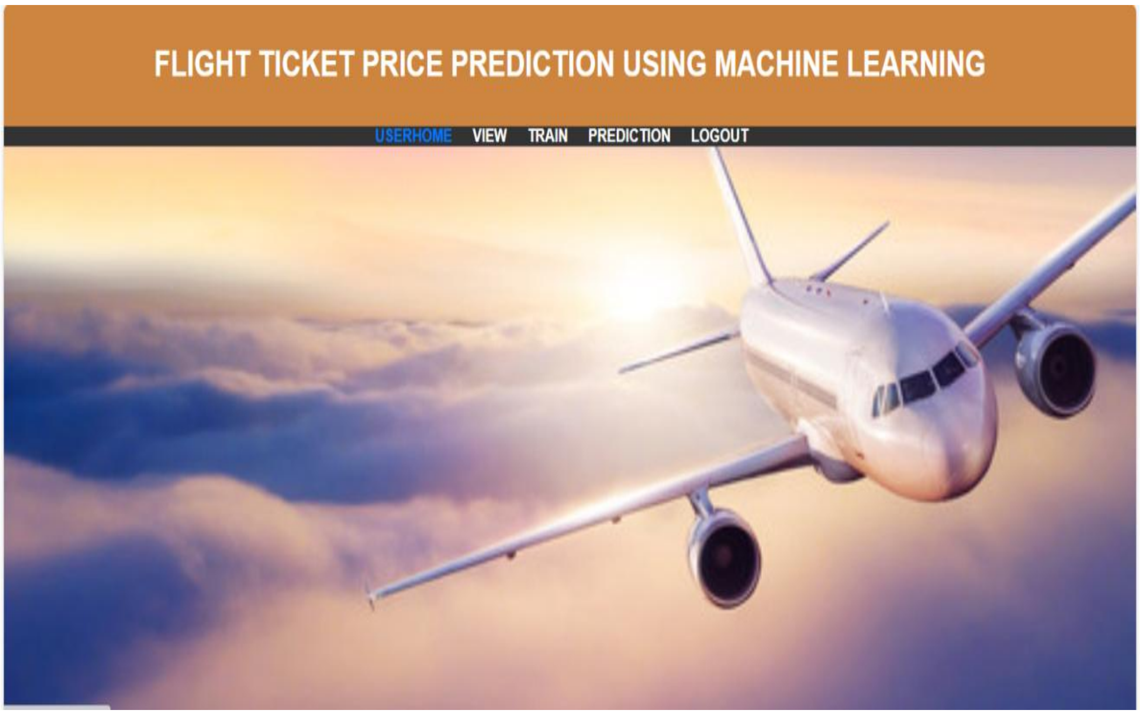
Description: *In This Page Users Need to Register By entering Credentials*

Figure 6.3 *Registration Page*


FLIGHT TICKET PRICE PREDICTION USING MACHINE LEARNING
Home
About
Login
You don't have a account signup
LOGIN
Enter Email
Enter Email
Enter Password
Password
LOGIN

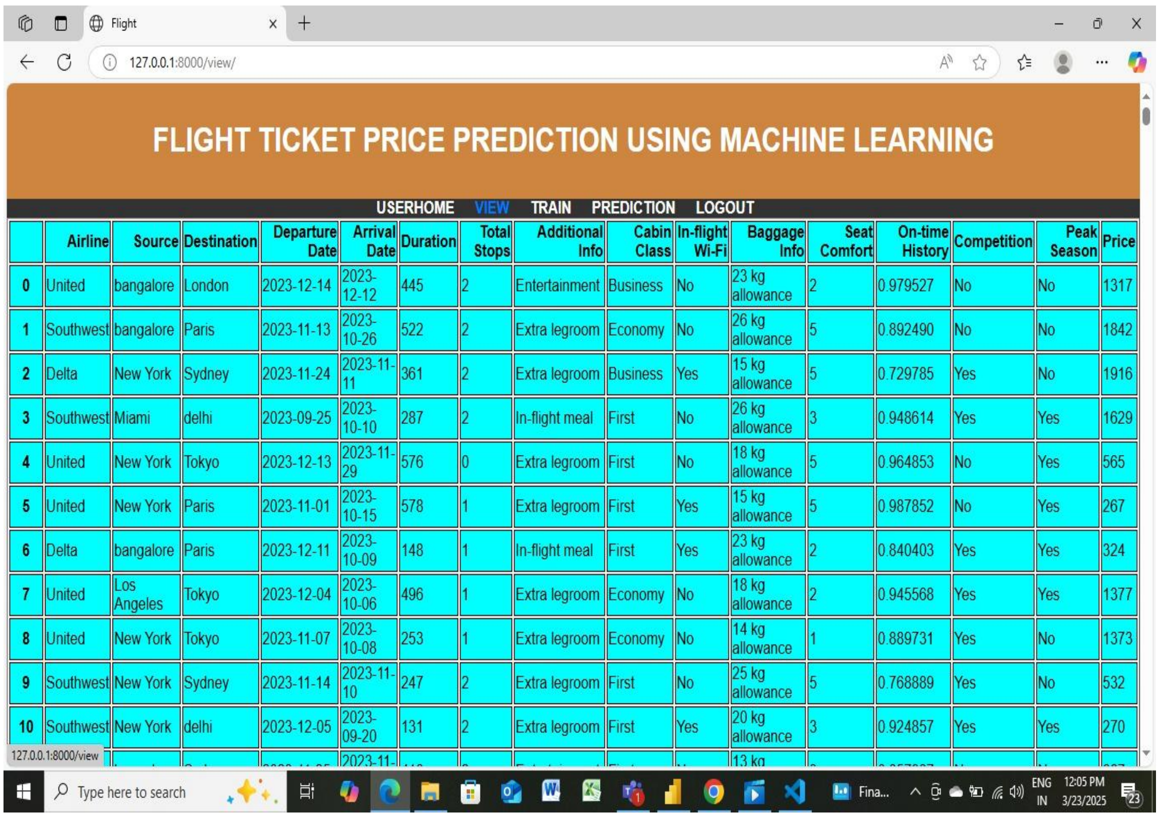
Description: *In this login page the User have to enter the credentials and login*

Figure 6.4 *Login Page*



Description: *After logging in the user will view the next page of login*

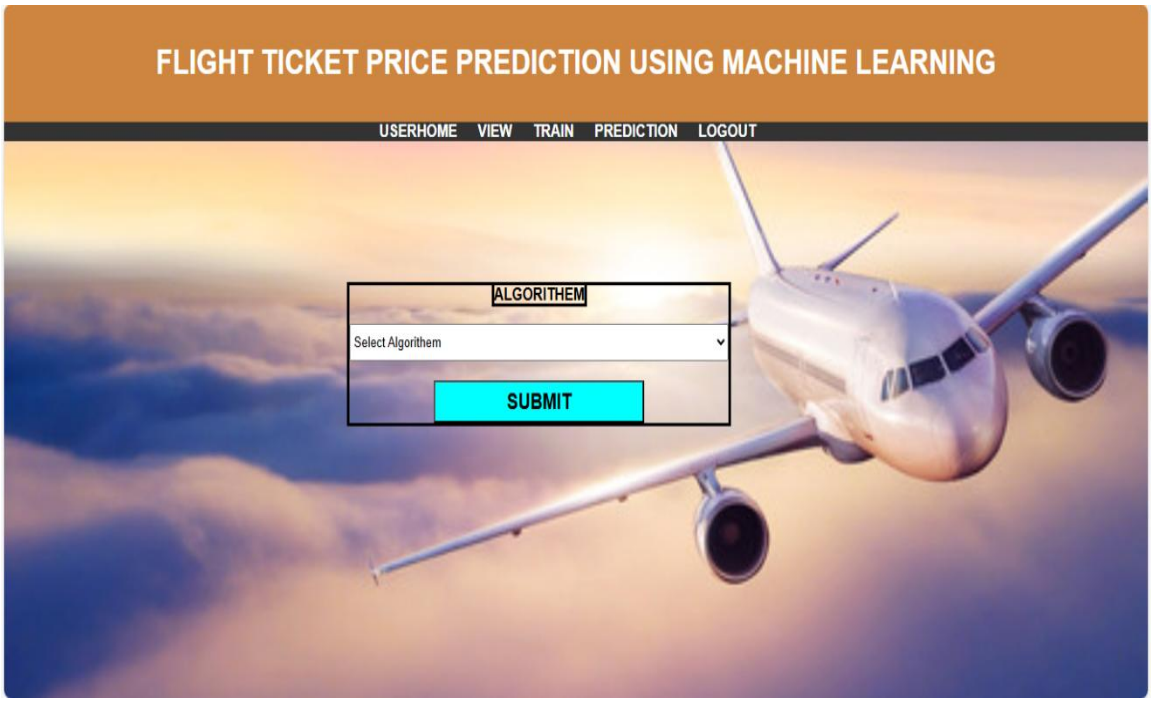
Figure 6.5 *User Home Page*



	Airline	Source	Destination	Departure Date	Arrival Date	Duration	Total Stops	Additional Info	Cabin Class	In-flight Wi-Fi	Baggage Info	Seat Comfort	On-time History	Competition	Peak Season	Price
0	United	bangalore	London	2023-12-14	2023-12-12	445	2	Entertainment	Business	No	23 kg allowance	2	0.979527	No	No	1317
1	Southwest	bangalore	Paris	2023-11-13	2023-10-26	522	2	Extra legroom	Economy	No	26 kg allowance	5	0.892490	No	No	1842
2	Delta	New York	Sydney	2023-11-24	2023-11-11	361	2	Extra legroom	Business	Yes	15 kg allowance	5	0.729785	Yes	No	1916
3	Southwest	Miami	delhi	2023-09-25	2023-10-10	287	2	In-flight meal	First	No	26 kg allowance	3	0.948614	Yes	Yes	1629
4	United	New York	Tokyo	2023-12-13	2023-11-29	576	0	Extra legroom	First	No	18 kg allowance	5	0.964853	No	Yes	565
5	United	New York	Paris	2023-11-01	2023-10-15	578	1	Extra legroom	First	Yes	15 kg allowance	5	0.987852	No	Yes	267
6	Delta	bangalore	Paris	2023-12-11	2023-10-09	148	1	In-flight meal	First	Yes	23 kg allowance	2	0.840403	Yes	Yes	324
7	United	Los Angeles	Tokyo	2023-12-04	2023-10-06	496	1	Extra legroom	Economy	No	18 kg allowance	2	0.945568	Yes	Yes	1377
8	United	New York	Tokyo	2023-11-07	2023-10-08	253	1	Extra legroom	Economy	No	14 kg allowance	1	0.889731	Yes	No	1373
9	Southwest	New York	Sydney	2023-11-14	2023-11-10	247	2	Extra legroom	First	No	25 kg allowance	5	0.768889	Yes	No	532
10	Southwest	New York	delhi	2023-12-05	2023-09-20	131	2	Extra legroom	First	Yes	20 kg allowance	3	0.924857	Yes	Yes	270

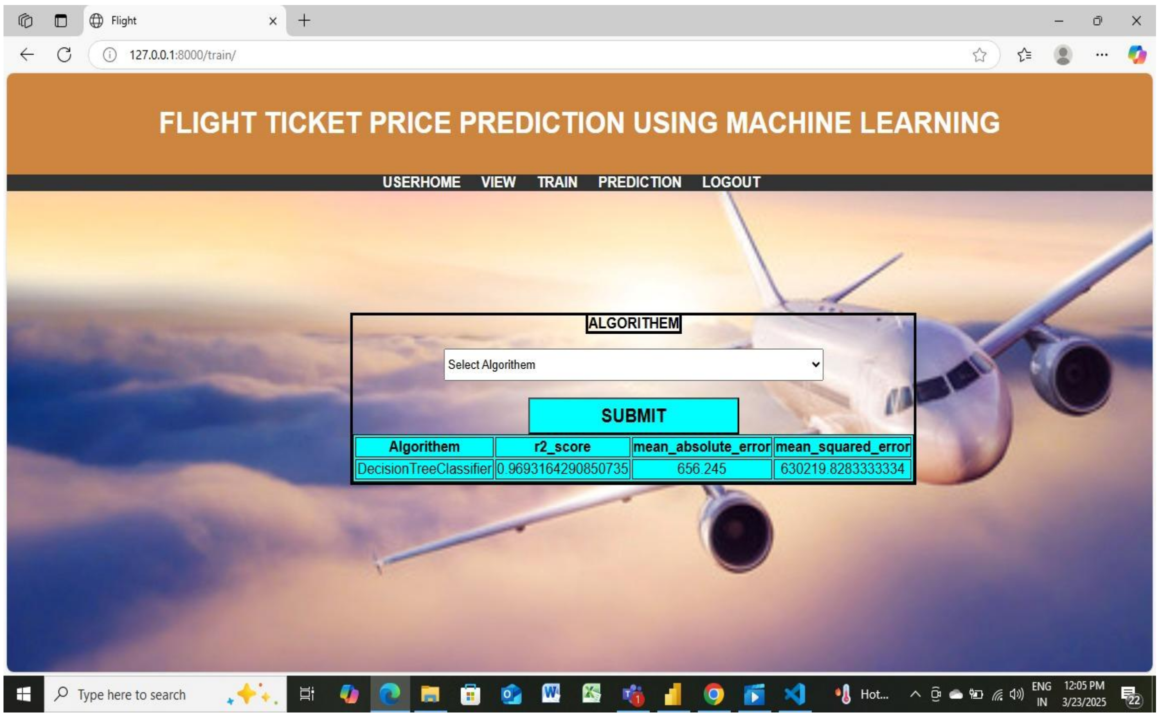
Description: *Here we can see the uploaded data set*

Figure 6.6 *View of the Dataset*



Description: *In Train option we can train the different ML Algorithms*

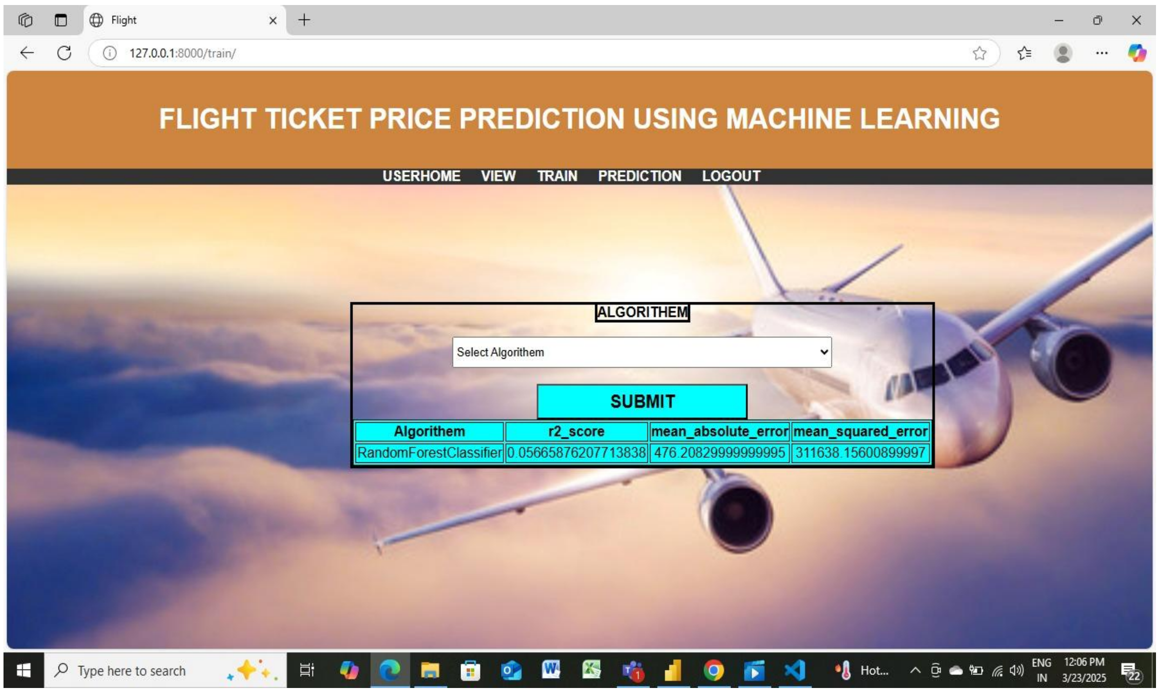
Figure 6.7 *Model Training*



Algorithem	r2_score	mean_absolute_error	mean_squared_error
DecisionTreeClassifier	0.9693164290850735	656.245	630219.8283333334

Description: *We have to select the Decision Tree Algorithm and click submit it will train*

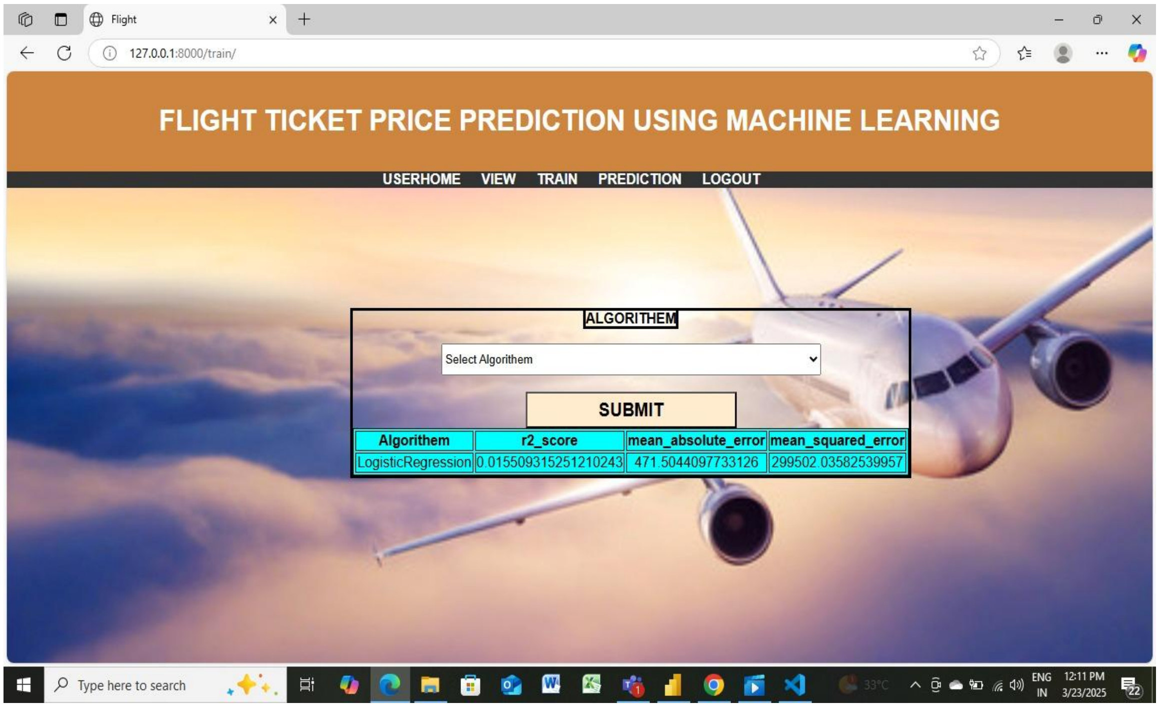
Figure 6.8 *To Train Decision Tree*


Flight
127.0.0.1:8000/train/
FLIGHT TICKET PRICE PREDICTION USING MACHINE LEARNING
USERHOME VIEW TRAIN PREDICTION LOGOUT
ALGORITHEM
Select Algorithem
SUBMIT

Algorithem	r2_score	mean_absolute_error	mean_squared_error
RandomForestClassifier	0.05665876207713838	476.20829999999995	311638.15600899997

Description: *We have to select the Random Forest Algorithm and click submit it will train*

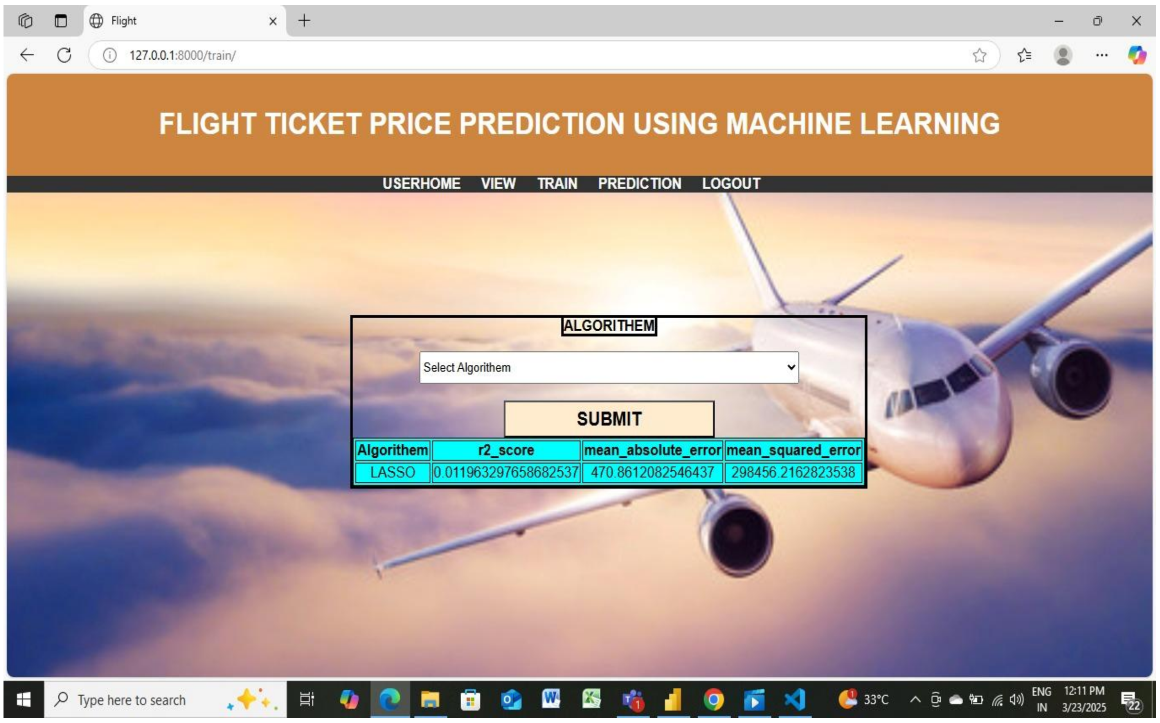
Figure 6.9 *To Train Random Forest*


Flight
127.0.0.1:8000/train/
FLIGHT TICKET PRICE PREDICTION USING MACHINE LEARNING
USERHOME VIEW TRAIN PREDICTION LOGOUT
ALGORITHEM
Select Algorithem
SUBMIT

Algorithem	r2_score	mean_absolute_error	mean_squared_error
LogisticRegression	0.015509315251210243	471.5044097733126	299502.03582539957

Description: *We have to select the Logistic Regression Algorithm and click submit it will train*

Figure 6.10 *To Train Logistic Regression*

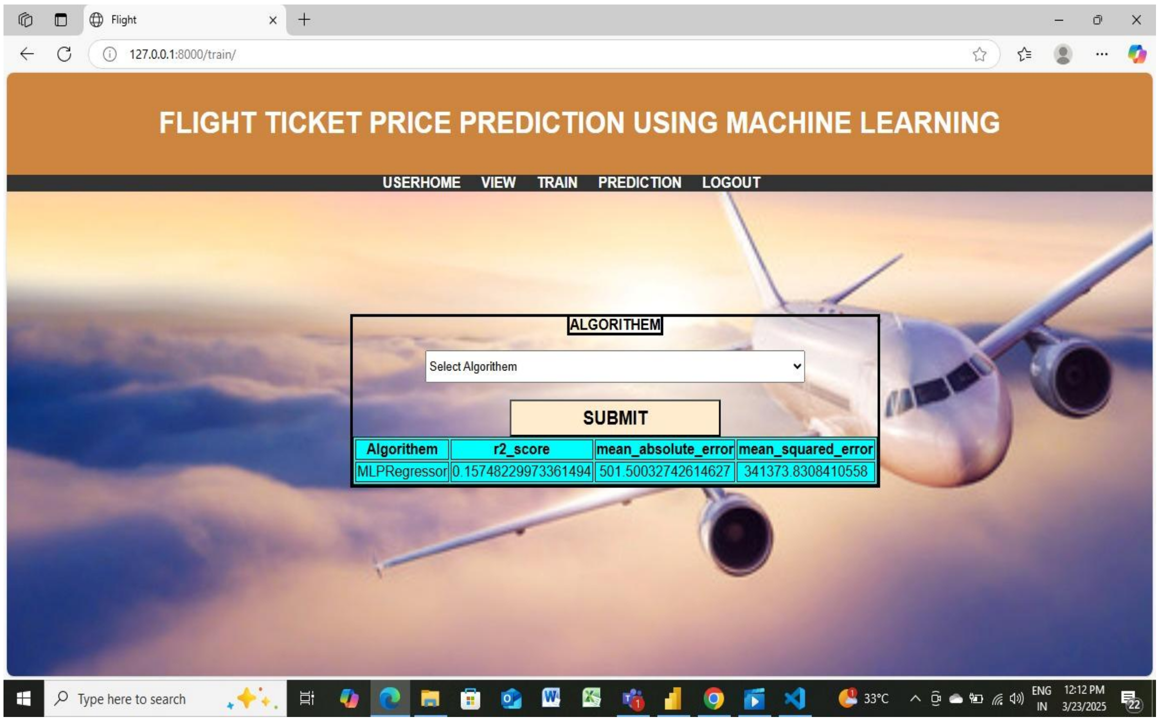


Flight
127.0.0.1:8000/train/
FLIGHT TICKET PRICE PREDICTION USING MACHINE LEARNING
USERHOME VIEW TRAIN PREDICTION LOGOUT
ALGORITHEM
Select Algorithem
SUBMIT
Type here to search

Algorithem	r2_score	mean_absolute_error	mean_squared_error
LASSO	0.011963297658682537	470.8612082546437	298456.2162823538

Description: *We have to select the LASSO Algorithm and click submit it will train*

Figure 6.11 *To Train LASSO*



Flight
127.0.0.1:8000/train/
FLIGHT TICKET PRICE PREDICTION USING MACHINE LEARNING
USERHOME VIEW TRAIN PREDICTION LOGOUT
ALGORITHEM
Select Algorithem
SUBMIT
Type here to search

Algorithem	r2_score	mean_absolute_error	mean_squared_error
MLPRegressor	0.15748229973361494	501.50032742614627	341373.8308410558

Description: *We have to select the MLP Regression Algorithm and click submit it will train*

Figure 6.12 *To Train MLP Regression*

I. Evaluation Metrics

- R^2
- Mean Squared Error (MSE)
- Mean Absolute Error (MAE)

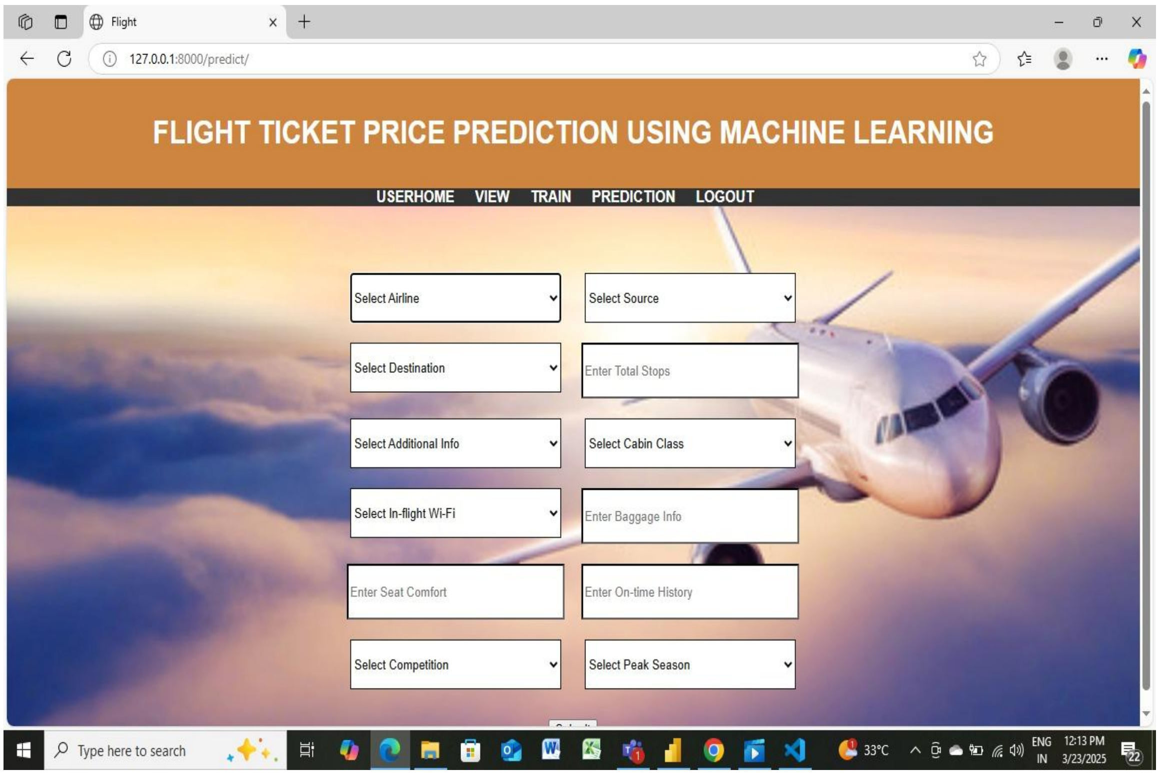
II. Model Comparison

Table 6.1 *Accuracy of Algorithms*

Algorithm	R^2	MSE	MAE
Decision tree	0.96	656.24	630219.82
Random Forest	0.056	476.20	311638.15
Logistic Regression	0.015	471.50	299502.03
LASSO	0.011	470.86	298456.21
MLP	0.15	501.50	341373.83

Discussion:

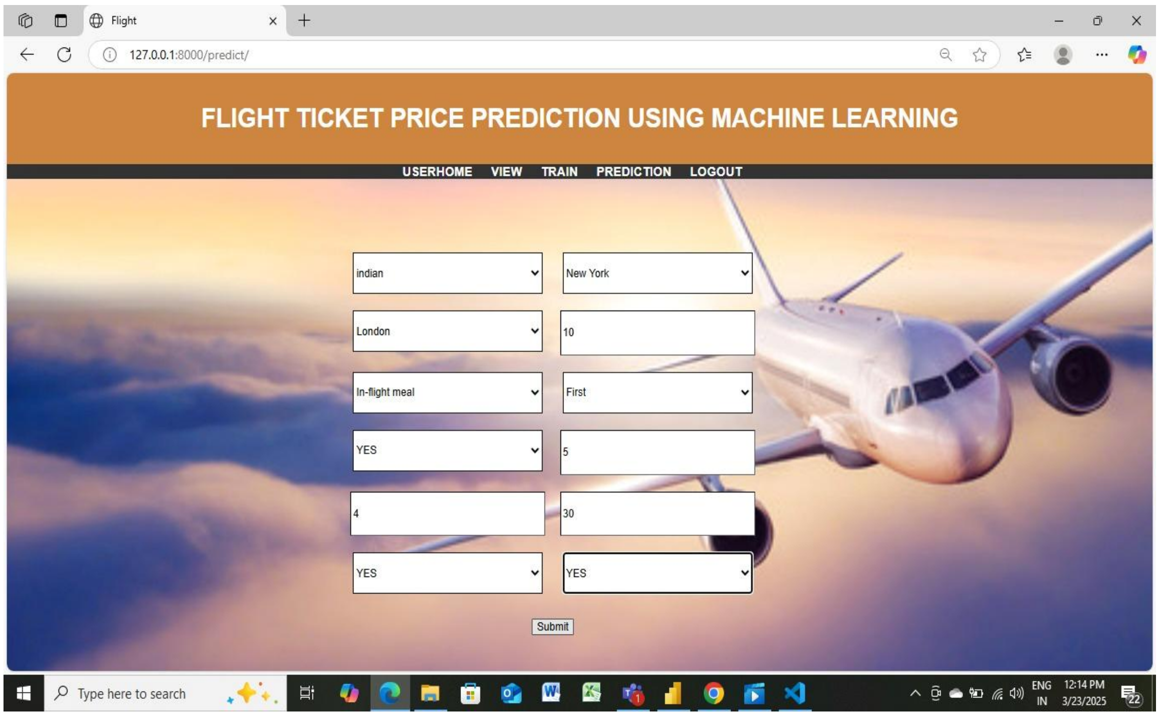
- The performance of five machine learning algorithms was analyzed using R2, MSE, and MAE metrics. The Decision Tree achieved the best results, with a high R2 score (0.96) and reasonable error rates (MSE: 656.24, MAE: 630,219.82), indicating strong predictive capability.
- The Random Forest showed a low R2 (0.056) but had lower error rates (MSE: 476.20, MAE: 311,638.15) compared to the Decision Tree, suggesting potential overfitting or the need for better tuning.
- Logistic Regression and LASSO had minimal R2 scores (0.015 and 0.011) but slightly better MAE values, indicating poor performance in capturing complex patterns. These models are more suited for simpler, linear relationships.
- The Multilayer Perceptron (MLP) achieved an R2 score of 0.15, with higher error rates (MSE: 501.50, MAE: 341,373.83) but showed some ability to model non-linear relationships.



Select Airline
Select Source
Select Destination
Enter Total Stops
Select Additional Info
Select Cabin Class
Select In-flight Wi-Fi
Enter Baggage Info
Enter Seat Comfort
Enter On-time History
Select Competition
Select Peak Season

Description: *This page shows the detection result of flight ticket price Prediction*

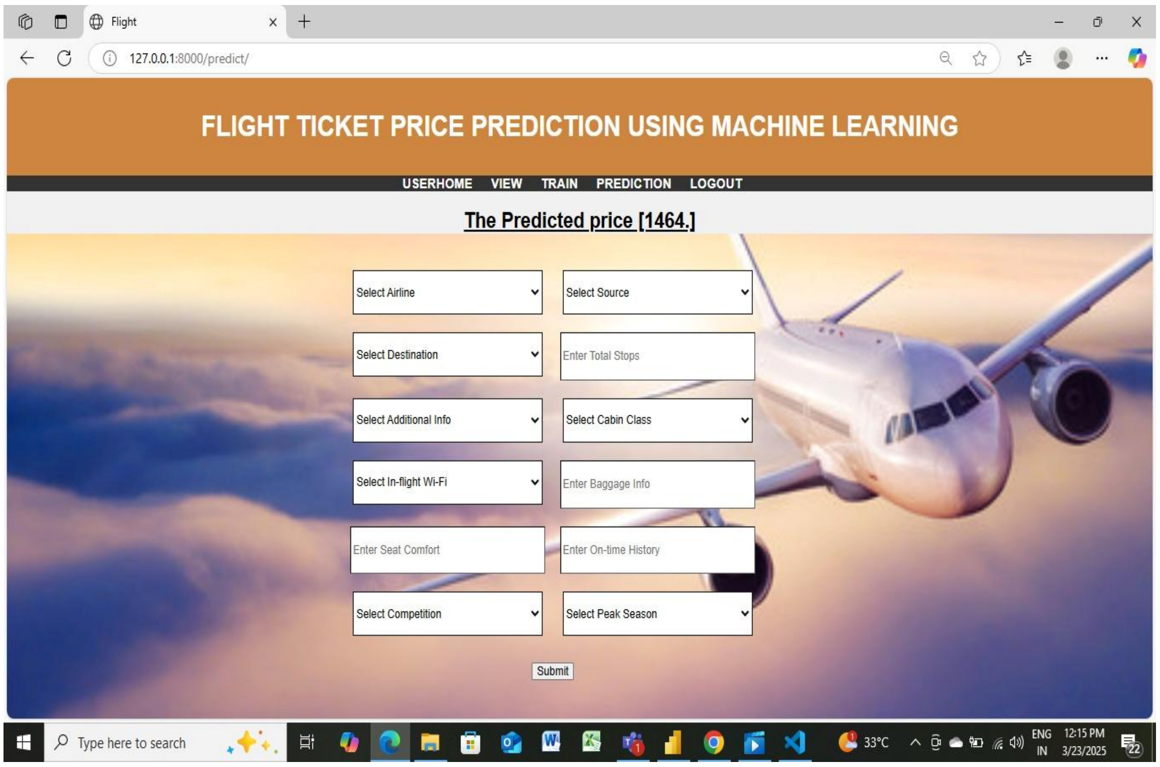
Figure 6.13 *Prediction Page*



indian	New York
London	10
In-flight meal	First
YES	5
4	30
YES	YES

Description: *We have to select the options to predict the flight ticket price*

Figure 6.14 *To Select the Options*



The Predicted price [1464.]

Description: *After selecting options and clicking submit button it will show the predicted price*

Figure 6.15 *Predict the Price*

CHAPTER -7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 Conclusion:

This study has undertaken a rigorous examination of three distinct machine learning algorithms - Decision Trees, Random Forest, and Logistic Regression - in the domain of flight price prediction within the dynamic airline industry. Each algorithm was scrutinized for its unique capabilities and limitations. Decision Trees, prized for their simplicity and interpretability, provide valuable insights into pricing patterns but struggle to handle complex relationships within the pricing dataset. Random Forest, an ensemble method, emerged as a promising approach by leveraging the collective wisdom of multiple decision trees to enhance prediction accuracy. Logistic Regression, adapted for flight price prediction, showcased its strength in handling binary outcomes, aligning well with the nature of ticket pricing decisions. Through performance evaluations, including metrics like r2 score, mean squared error, and mean absolute error, coupled with feature importance analysis, this research has illuminated the factors significantly influencing flight prices. These findings offer valuable guidance to stakeholders in the aviation industry, enabling them to make data-driven decisions, refine pricing strategies, and ultimately enhance the travel experience for both passengers and airlines in this highly competitive sector.

7.2 Future Enhancement

Future improvements in flight price prediction can enhance machine learning models' accuracy and usefulness. Integrating real-time data feeds could help models adapt to changing market conditions, weather, and events. Using advanced natural language processing (NLP) techniques to analyze customer reviews and sentiment could provide insights into how perceptions impact pricing. Additionally, hybrid models combining different machine learning algorithms, such as ensemble models with deep learning, could improve prediction accuracy. Collaboration between airlines and data scientists to share anonymized data could further enhance the effectiveness of these models in the competitive airline industry.

REFERENCES

[1] S Netessine and R. Shumsky, ‘‘Introduction to the theory and practice of yield management,’’ INFORMS Trans. Educ., vol. 3, no. 1, pp. 34–44, Sep. 2002.

[2] W. S. McCulloch and W. Pitts, A logical calculus of the ideas immanent in nervous activity,’ Bull. Math. Biophys., vol. 5, no. 4, pp. 115–133, Dec. 1943.

[3] F. Rosenblatt, ‘‘The perceptron: A probabilistic model for information storage and organization in the brain,’’ Psychol. Rev., vol. 65, no. 6, pp. 386–408, 1958.

[4] B. E. Boser, I. M. Guyon, and V. N. Vapnik, ‘‘A training algorithm for optimal margin classifiers,’’ in Proc. 5th Annu. workshop Comput. Learn. theory, Jul. 1992, pp. 144–152.

[5] E. Fix and J. L. Hodges, Discriminatory analysis. Nonparametric discrimination: Consistency properties,’ Int. Stat. Rev./Revue Internationale de Statistique, vol. 57, no. 3, p. 238, Dec. 1989.

[6] R. E. Schapire, ‘‘The strength of weak learnability,’’ Mach. Learn., vol. 5, no. 2, pp. 197–227, Jun. 1990.

[7] K. Fukushima, ‘‘Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,’’ Biol. Cybern., vol. 36, no. 4, pp. 193–202, Apr. 1980.

[8] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, ‘‘Gradient-based learning applied to document recognition,’’ Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[9] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T.

Graepel, and D. Hassabis, ‘‘Mastering the game of go with deep neural networks and tree search,’’ Nature, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, ‘‘Generative adversarial networks,’’ Commun. ACM, vol. 63, no. 11, pp. 139–144, Oct. 2020.

[11] P. W. Shor, ‘‘Algorithms for quantum computation: Discrete logarithms and factoring,’’ in Proc. 35th Annu. Symp. Found. Comput. Sci., 1994, pp. 124–134.

[12] L. K. Grover, A framework for fast quantum mechanical algorithms,’ in Proc. 30th Annu. ACM Symp. Theory Comput. (STOC), 1998, pp. 53–62. S