CHAPTER-1
INTRODUCTION

1.1 INTRODUCTION

A go-around occurs when the flight crew makes the decision not to continue an approach or a landing, and follows procedures to conduct another approach or to divert to another airport. Go-around decision can be made by either flight crew members, and can be executed at any point from the final approach prefix point to wheels touching down on the runway (but prior to activation of brakes, spoilers, or thrust reversers). In addition to unstable approaches, traffic, blocked runway, or adverse weather conditions are other reasons for a go around. Despite a clear policy and training on go-around policies in most airlines, operational data show that flight crew decision-making process in deciding for a go-around could be influenced by many other factors.

These include fatigue, flight schedule pressure, time pressure, excessive a head-down work, incorrect anticipation of aircraft deselection, visual illusions, organizational policy/culture, inadequate training or practice, excessive confidence in the ability to stabilize approach, and Crew Resource Management issues. It is for these reasons that on-board real time performance monitoring and alerting systems that could assist the flight crew with the landing/go-around decision are needed.

Under the hypothesis that a hard-landing (HL) occurrence has precursors and, thus, it can be predicted, this project presents a cockpit deployable machine learning system to predict hard landings considering the aircraft dynamics and configuration.

Using a variety of qualitative and quantitative methods researching and stimulating change in the life world through the common development of solutions (either in the form of processes, institutional arrangements or technical solutions).

This changed and broadened role provides the opportunity to be part of a process of integrated knowledge generation and to be able to identify and describe problems that occur during the process. This gained knowledge offers a way to re-approach these problems better.

1.2 PROBLEM STATEMENT

Current systems for monitoring flight safety focus primarily on post-landing analysis or on the maintenance of general flight parameters, but they do not offer real-time predictive capabilities that can alert pilots or ground crews about the likelihood of a hard landing during the approach phase. As a result, there is a gap in providing timely and actionable insights that could help prevent these incidents. The lack of real-time prediction systems contributes to the challenge of minimizing the frequency of hard landings and their associated risks. Given the complexity of flight dynamics and the variability in landing conditions, existing systems fail to detect the early warning signs that may indicate an impending hard landing.

Hard landings are a significant safety concern in commercial aviation, resulting in potential damage to aircraft, injuries to passengers and crew, and increased maintenance costs. Current methods for predicting hard landings rely heavily on manual analysis of flight data, which can be time-consuming, prone to errors, and may not accurately capture complex patterns. Access to high-quality, relevant, and comprehensive flight data, including sensor readings, weather conditions, and pilot inputs. The complexity of flight dynamics accurately modeling the complex interactions between aircraft, pilots, and environmental factors that contribute to hard landings.

1.3 OBJECTIVES

By training the system with historical flight data, including both successful and hard landing instances, the aim is to enable the model to learn the patterns and conditions associated with hard landings. The system should be capable of making accurate, real-time predictions and providing timely alerts to pilots, helping them adjust their approach if necessary.

Additionally, the system must be designed to continuously learn from new flight data, improving its prediction capabilities over time and adapting to various flight conditions and aircraft types.

Another key objective is to seamlessly integrate the prediction system into existing flight control and monitoring systems, ensuring it enhances the decision-making process without adding complexity to the flight crew's workflow.

1.4 SCOPE OF THE PROJECT

This project will involve collecting and processing large volumes of flight data, both from historical flight records and real-time data feeds from commercial flights, to train machine

learning models capable of identifying patterns associated with hard landings.

The system will need to be designed to handle complex datasets and accurately predict landing risks in real-time.

Additionally, the machine learning model will continuously improve over time by learning from new flight data, ensuring its adaptability to changing flight conditions, aircraft types, and environmental factors.

A significant aspect of the project will be ensuring seamless integration with existing flight control systems and the efficient delivery of real-time predictions and alerts to flight crews, without adding undue complexity to their decision-making processes.

CHAPTER-2

LITERATURE REVIEW

Literature Review is the most important step in software development process. Before developing the tool, it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, to determine which operating system and language can be used for developing tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system, the above is taken into account for developing the proposed system.

2.1 MACHINE LEARNING

Machine learning (ML) is a field of study in artificial intelligence concerned with the development and study of statistical algorithms that can learn from data and generalize to unseen data, and thus perform tasks without explicit instructions.[1] Within a subdiscipline in machine learning, advances in the field of deep learning have allowed neural networks, a class of statistical algorithms, to surpass many previous machine learning approaches in performance. ML finds application in many fields, including natural language processing, computer vision, speech recognition, email filtering, agriculture, and medicine. The application of ML to business problems is known as predictive analytics.

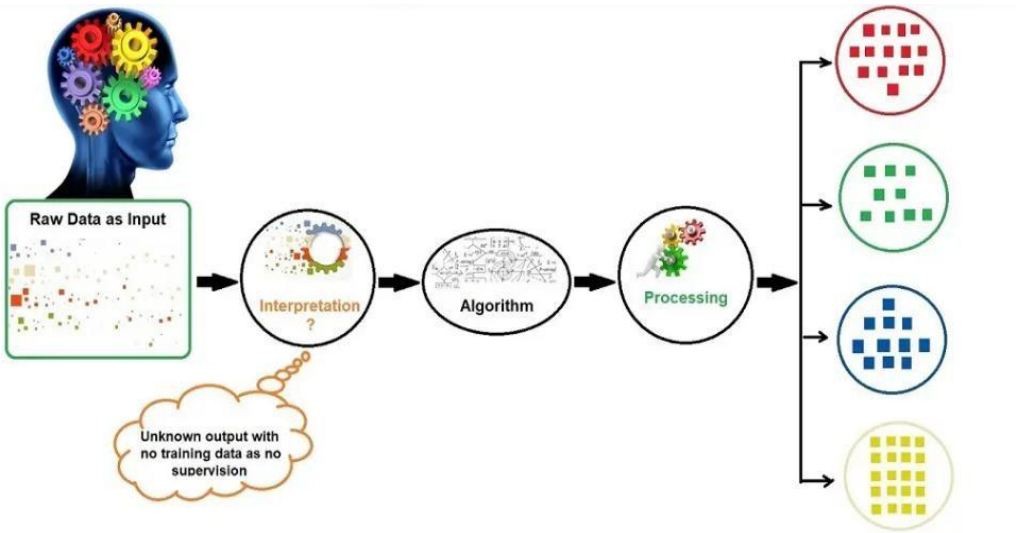

Fig.2.1: Machine learning

Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed. With the help of sample historical data, which is known as training data, machine learning algorithms build a mathematical model that helps in making predictions or decisions without being explicitly programmed.Machine learning brings computer science and statistics together for creating predictive models. Machine learning constructs or uses the algorithms that learn from historical data. The more we will provide the information, the higher will be the performance. A machine has the ability to learn if it can improve its performance by gaining more data.

How Does Machine Learning Work:

A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately. Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data and predict the output. Machine learning has changed our way of thinking about the problem.

2.1.1 Features of Machine Learning

- Machine learning uses data to detect various patterns in a given dataset.
- It can learn from past data and improve automatically.
- It is a data-driven technology.
- Machine learning is much similar to data mining as it also deals with the huge amount of the data.

2.1.2 Need for Machine Learning

The need for machine learning is increasing day by day. The reason behind the need for machine learning is that it is capable of doing tasks that are too complex for a person to implement directly. As a human, we have some limitations as we cannot access the huge amount of data manually, so for this, we need some computer systems and here comes the machine learning to make things easy for us. We can train machine learning algorithms by providing them the huge amount of data and let them explore the data, construct the models, and predict the required output automatically. The performance of the machine learning

algorithm depends on the amount of data, and it can be determined by the cost function. With the help of machine learning, we can save both time and money. The importance of machine learning can be easily understood by its use's cases, Currently, machine learning is used in self-driving cars, cyber fraud detection, face recognition, and friend suggestion by Facebook, etc. Various top companies such as Netflix and Amazon have built machine learning models that are using a vast amount of data to analyse the user interest and recommend product accordingly.

Following are some key points which show the importance of Machine Learning:

- Rapid increment in the production of data
- Solving complex problems, which are difficult for a human
- Decision making in various sector including finance
- Finding hidden patterns and extracting useful information from data

At a broad level, machine learning can be classified into three types:

- Supervised learning
- Unsupervised learning
- Reinforcement learning

2.1.3 TYPES OF MACHINE LEARNING

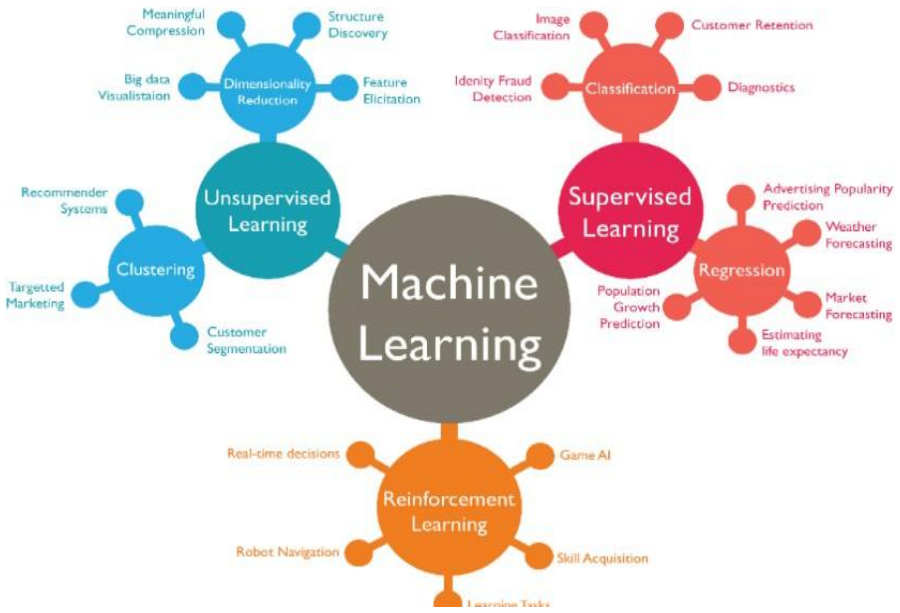

Fig.2.1.3: Types of Machine learning (ML)

The main types of machine learning model are:

Supervised Machine Learning:

Supervised learning is defined as when a model gets trained on a "Labelled Dataset". Labelled datasets have both input and output parameters. In Supervised Learning algorithms learn to map points between inputs and correct outputs. It has both training and validation datasets labelled.

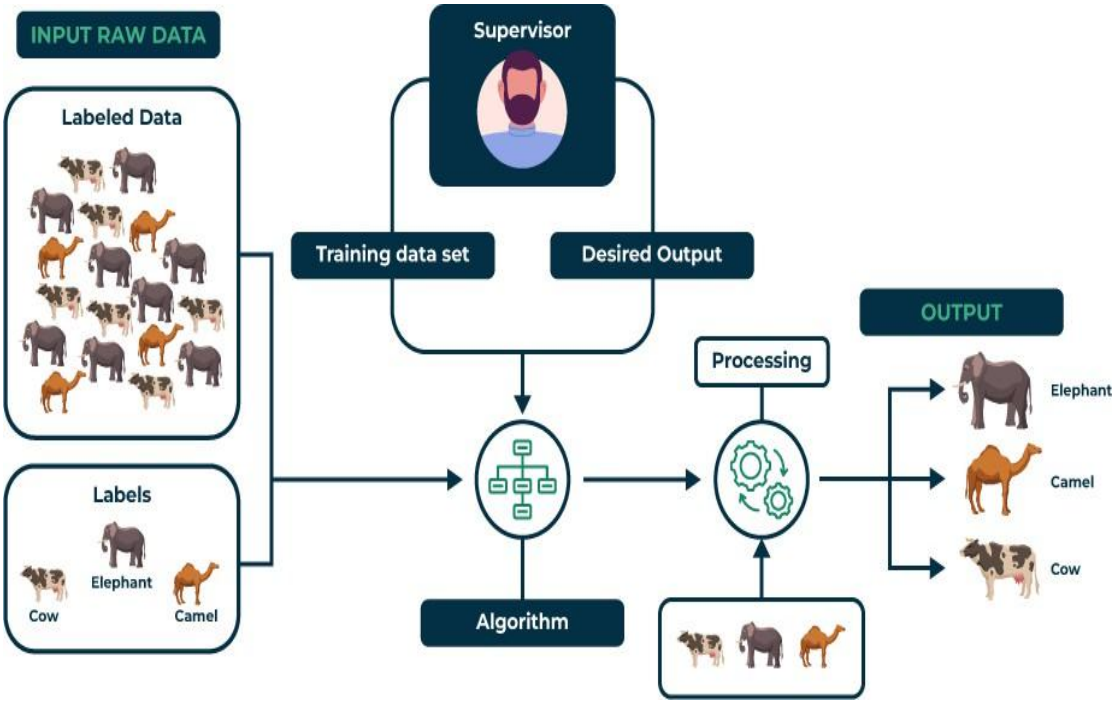

Fig.2.1.1.1: Supervised Machine Learning (ML)

Steps Involved in Supervised Learning:

- First Determine the type of training dataset
- Collect/Gather the labelled training data.
- Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

Advantages of Supervised Learning:

- With the help of supervised learning, the model can predict the output on the basis of prior experiences.
- In supervised learning, we can have an exact idea about the classes of objects.
- Supervised learning model helps us to solve various real-world problems such as fraud detection, spam filtering, etc.

Disadvantages of supervised Learning:

- Supervised learning models are not suitable for handling the complex tasks.
- Supervised learning cannot predict the correct output if the test data is different from the training dataset.
- Training required lots of computation times.
- In supervised learning, we need enough knowledge about the classes of object.

Unsupervised Machine Learning: Unsupervised Unsupervised learning is a type of machine learning technique in which an algorithm discovers patterns and relationships using unlabelled data. Unlike supervised learning, unsupervised learning doesn't involve providing the algorithm with labelled target outputs. The primary goal of Unsupervised learning is often to discover hidden patterns, similarities, or clusters within the data, which can then be used for various purposes, such as data exploration, visualization, dimensionality reduction, and more.

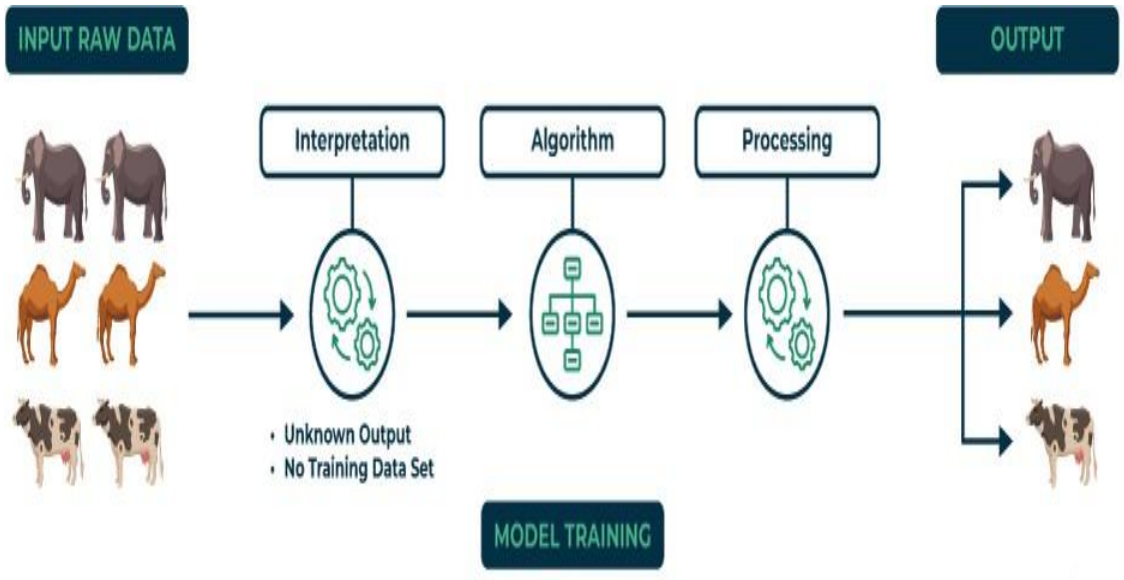

Fig.2.1.1.2: Unsupervised Machine Learning (ML)

Example: Suppose the unsupervised learning algorithm is given an input dataset containing images of different types of cats and dogs. The algorithm is never trained upon the given dataset, which means it does not have any idea about the features of the dataset. The task of the unsupervised learning algorithm is to identify the image features on their own. Unsupervised learning algorithm will perform this task by clustering the image dataset into the groups according to similarities between images.

Types of Unsupervised Learning Algorithm:

The unsupervised learning algorithm can be further categorized into two types of problems:

- **Clustering:** Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group. Cluster analysis finds the commonalities between the data objects and categorizes them as per the presence and absence of those commonalities.
- **Association:** An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. It determines the set of items that occurs together in the dataset. Association rule makes marketing strategy more effective. Such as people who buy X item (suppose a bread) are also tend to purchase Y (Butter/Jam) item. A typical example of Association rule is Market Basket Analysis.

Advantages of Unsupervised Learning:

- Unsupervised learning is used for more complex tasks as compared to supervised learning because, in unsupervised learning, we don't have labelled input data.
- Unsupervised learning is preferable as it is easy to get unlabelled data in comparison to labelled data.

Disadvantages of Unsupervised Learning:

- Unsupervised learning is intrinsically more difficult than supervised learning as it does not have corresponding output.
- The result of the unsupervised learning algorithm might be less accurate as input data is not labelled, and algorithms do not know the exact output in advance.

Reinforcement Machine Learning: Reinforcement machine learning algorithm is a learning method that interacts with the environment by producing actions and discovering errors. Trial, error, and delay are the most relevant characteristics of reinforcement learning. In this technique, the model keeps on increasing its performance using Reward Feedback to

learn the behaviour or pattern. These algorithms are specific to a particular problem e.g. Google Self Driving car, AlphaGo where a bot competes with humans and even itself to get better and better performers in Go Game. Each time we feed in data, they learn and add the data to their knowledge which is training data. So, the more it learns the better it gets trained and hence experienced.

2.1.2. Applications of ML

1. **Self-driving cars:** Machine learning plays a significant role in self-driving cars. Tesla, the most popular car manufacturing company is working on self-driving car. It is using unsupervised learning method to train the car models to detect people and objects while driving.

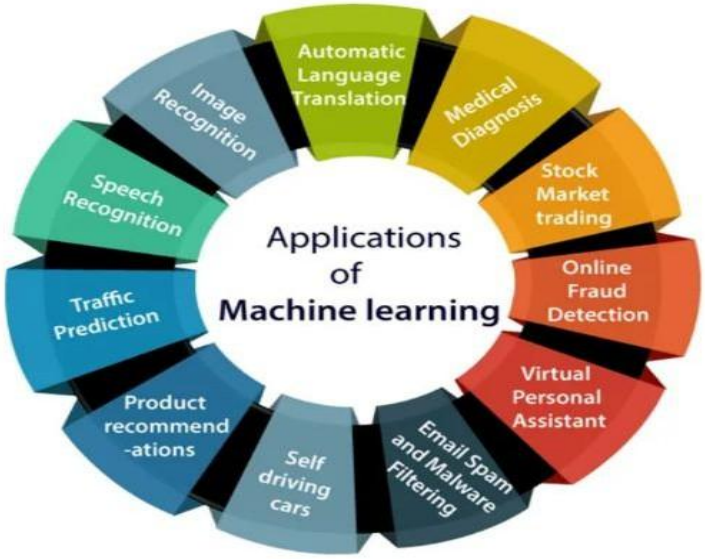

Fig.2.1.2: Applications of ML

2. **Speech recognition:** Speech recognition is a process of converting voice instructions into text, and it is also known as “Speech to text”. Machine learning algorithms are widely used by various applications of speech recognition. Google assistant, Alexa and many more are using speech recognition technology to follow the voice instructions. While using Google, we get an option of "Search by voice," it comes under speech recognition, and it's a popular application of machine learning.

Speech recognition is a process of converting voice instructions into text, and it is also known as "Speech to text", or "Computer speech recognition." At present, machine learning algorithms are widely used by various applications of speech recognition. Google assistant, Siri, Cortana, and Alexa are using speech recognition technology to follow the voice instructions.

3. **Online fraud detection:** Machine learning is making our online transaction safe and secure by detecting fraud transaction, there may be various ways that a fraud transaction can take place such as fake accounts, fake ids, and steal money in the middle of a transaction.
4. **Health care:** In medical science, machine learning is used for diseases diagnosis. Medical technology is growing very fast and able to build 3D models that can predict the exact position of lesions in the brain.
5. I**mage recognition:** Image recognition is one of the most common applications of machine learning. It is used to identify objects, persons, places etc.
6. **Traffic prediction:**

   If we want to visit a new place, we take help of Google Maps, which shows us the correct path with the shortest route and predicts the traffic conditions. It predicts the traffic conditions such as whether traffic is cleared, slow-moving, or heavily congested with the help of two ways:

   - Real Time location of the vehicle form Google Map app and sensors
   - Average time has taken on past days at the same time.
7. **Automatic Language Translation:** Nowadays, if we visit a new place and we are not aware of the language then it is not a problem at all, as for this also machine learning helps us by converting the text into our known languages.

   Google's GNMT (Google Neural Machine Translation) provide this feature, which is a Neural Machine Learning that translates the text into our familiar language, and it called as automatic translation. The technology behind the automatic translation is a sequence-to-sequence learning algorithm, which is used with image recognition and translates the text from one language to another language.

2.2 WEB APPLICATIONS

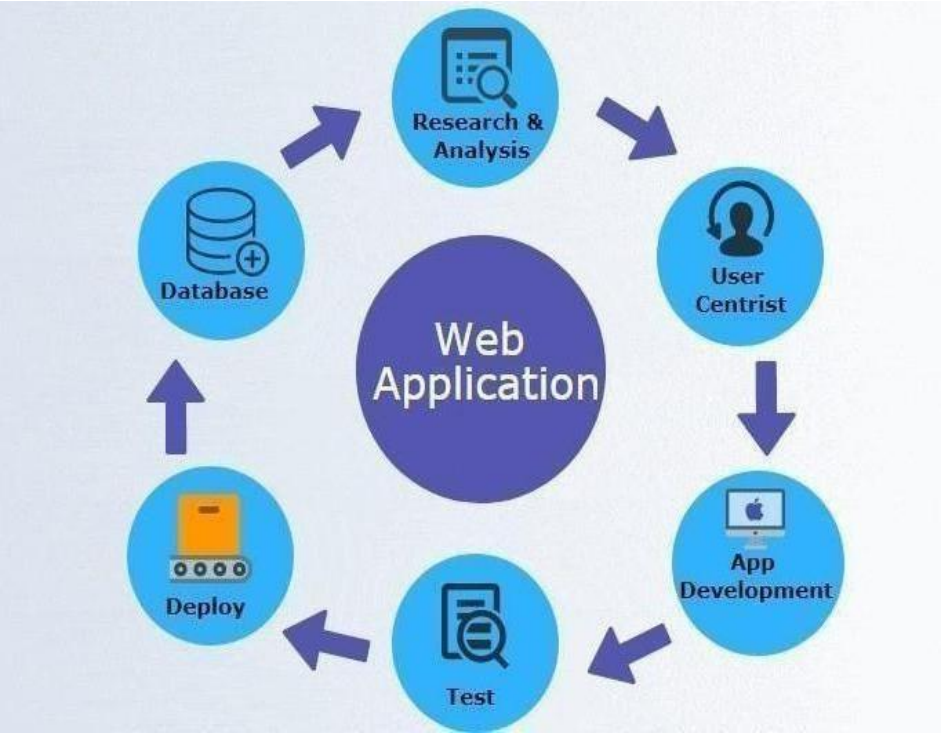

Fig.2.2: Web Application

Web applications, often abbreviated as web apps, are software applications that run on web servers and are accessed by users through web browsers over the Internet or an intranet. These applications leverage web technologies such as HTML, CSS, and JavaScript to provide dynamic and interactive user experiences. Web apps can range from simple websites with static content to complex applications with advanced functionality, including e-commerce platforms, social media networks, online banking systems, and productivity tools.

Accessibility: Web applications are accessible from any device with a web browser and an internet connection, making them platform-independent and allowing users to access them from desktops, laptops, tablets, and smartphones.

Client-Server Architecture: Web apps typically follow a client-server architecture, where the client (web browser) sends requests to the server, which processes these requests, performs necessary operations, and returns responses to the client.

Interactivity: Web apps can provide rich user interactions through client-side

scripting languages like JavaScript, enabling features such as form validation, real-time updates, and interactive user interfaces.

Dynamic Content: Unlike static websites, web applications generate dynamic content based on user input, database queries, and external APIs. This dynamic nature allows web apps to personalize content, retrieve real-time data, and respond to user actions in real- time.

State Management: Web applications can maintain state across multiple user interactions, enabling features like user sessions, shopping carts, and authentication mechanisms to provide personalized experiences and secure access to resources.

Scalability and Flexibility: Web applications can scale horizontally by adding more servers or leveraging cloud infrastructure to accommodate increasing user traffic and workload demands. They can also be easily updated and maintained without requiring users to install software updates.

Security: Security is a critical aspect of web applications, encompassing measures such as encryption, authentication, authorization, input validation, and protection against common vulnerabilities like cross-site scripting (XSS) and SQL injection. Web applications play a crucial role in modern digital experiences, empowering businesses, organizations, and individuals to deliver services, communicate, collaborate, and conduct transactions efficiently and effectively over the web.

2.2.1 Web Application Development

Web application development refers to the process of creating dynamic and interactive software applications that run on web servers and are accessed by users through web browsers over the Internet or an intranet. It involves several stages, including planning, designing, coding, testing, and deployment. Here's an overview of the web application development process:

Requirement Analysis: The first step in web application development is to gather and analyze requirements from stakeholders to understand the objectives, functionalities, and features of the application.

Planning and Architecture Design: Based on the requirements, developers create a plan and design the architecture of the web application, including database schema, user interface layout, and technology stack selection.

Frontend Development: Frontend development involves creating the user interface (UI) and user experience (UX) of the web application using technologies such as HTML, CSS, and JavaScript. Frameworks and libraries like React.js, Angular, or Vue.js are often used to build responsive and interactive interfaces.

Backend Development: Backend development involves building the server-side logic, database interactions, and application functionality using programming languages such as Python, PHP, Java, or Node.js. Frameworks like Django, Express.js can be used to streamline backend development tasks.

Database Design and Implementation: Developers design and implement the database structure, including tables, relationships, and data storage mechanisms using relational databases like MySQL or Oracle, or NoSQL databases like MongoDB or Redis, based on the application's requirements.

Integration of Third-party APIs: Web applications often integrate third-party APIs for functionalities such as payment processing, authentication, social media integration, mapping, or data analytics.

Testing: Testing is a critical phase in web application development to ensure that the application functions correctly, is user-friendly, and is secure. It involves unit testing, integration testing, performance testing, security testing, and user acceptance testing (UAT).

Deployment: Once the web application is tested and ready for production, it is deployed to a web server or cloud platform such as AWS, Azure, or Google Cloud Platform. Deployment involves configuring servers, setting up databases, and deploying application code using tools like Docker, Kubernetes, or traditional server configurations.

Maintenance and Updates: After deployment, web applications require ongoing maintenance, monitoring, and updates to ensure they remain functional, secure, and up- to-date with changing requirements and technology advancements.

Web application development is a complex and iterative process that requires collaboration between developers, designers, testers, and stakeholders to deliver high- quality applications that meet user needs and business objectives.

2.3. Deep learning-based approach for civil aircraft hazard identification and prediction

Safety is an eternal issue in the civil aviation transportation. Once a civil aviation accident occurs, it will cause great casualties and economic losses. In order to ensure the civil aviation safety, the hazard identification and prediction of civil aircraft should be effectively and accurately realized. The civil aircraft uses Aircraft Communications Addressing and Reporting System (ACARS) to interact with the ground during flight. The data generated by ACARS has a simple structure and strong timeliness. In view of the advantages of ACARS data, a hazard identification and prediction method based on support vector machine optimized by particle swarm optimization (PSO-SVM) and long short-term memory (LSTM) neural networks which uses ACARS report as analysis data is proposed.

2.4. Aircraft electronic board fault detection based on infrared thermal imaging and integrated SVM

In view of the current status of aircraft electronic board gradually tends to be small and highly integrated, an infrared thermal imaging circuit board fault diagnosis system based on integrated support vector machine LIBSVM is designed and implemented. The two circuit boards (microprocessor PWB assembly A3) mounted on the Airbus A330 auxiliary power unit (APU) are monitored at the same time. The collected temperature data are used to match the temperature rising process through classification and recognition system. The experimental results show that the diagnosis system can be more accurately and quickly complete the fault location while saving time and cost, and easy to grasp.

2.5 An ensemble machine and deep learning model for risk prediction in aviation systems

The ubiquitous availability of big data and computational resources has accelerated the development and adoption of artificial intelligence methods in many application domains albeit with many challenges. This paper proposes a novel safety assurance model for aviation systems. The model exploits the ensemble of machine learning and deep learning algorithms to improve the prediction of risks and risks' severity in aviation systems. The diversity score of base classifiers is used to fine-tune the ensemble performance. The model

is trained and evaluated using twelve years of data from the Aviation Safety Reporting System (ASRS) database.

2.6 Deep Learning for Real-Time Flight Anomaly Detection

This paper discusses how deep learning techniques, particularly recurrent neural networks (RNNs) and convolutional neural networks (CNNs), can be leveraged to detect anomalies in aircraft landing phases. The study utilizes large datasets from flight data recorders to train models that identify deviations from safe landing trajectories. The proposed system aims to assist pilots by providing real-time alerts for potential hard landings, reducing the risk of structural damage and passenger discomfort

2.7 Machine Learning Applications for Predicting Hard Landings in Aviation

This study explores the use of machine learning models to predict hard landings based on flight parameters such as descent rate, airspeed, wind shear, and aircraft weight. The authors analyze historical flight data and employ algorithms like Random Forest and Support Vector Machines (SVM) to classify landings as "hard" or "normal." The results demonstrate the potential of predictive analytics in improving aviation safety by providing early warnings to pilots and flight management system.

2.8 RESEARCH PAPERS REFERRED

Ref: 1

Title: Statistical Summary of Commercial Jet Airplane Accidents.

Author: Boeing Commercial Airplanes, Aviation Saf., Seattle, WA, USA, 2018.

In [1], This paper presents a cockpit-deployable machine learning system designed to assist flight crews in making go-around decisions by predicting hard landing events. The proposed hybrid approach utilizes features modelling temporal dependencies of aircraft variables as inputs to a neural network. Evaluations based on a dataset of 58,177 commercial flights demonstrate an average sensitivity of 85% and specificity of 74% at the go-around point, indicating the system's potential effectiveness in real-time applications.

Ref: 2

Title: Developing standardized FDM-based indicators.

Author: Ur. Aviation Saf. Plan 2012-2015, Cologne, Germany, 2016.

In [2], This study explores how recent advancements in flight data analysis contribute to predictive safety systems in aviation. Sophisticated techniques identify threats early, improving safety protocols for passengers. The authors advocate for continued research and collaboration to further enhance aviation safety.

Ref: 3

Title: Advisory circular ac no: 91-79a mitigating the risks of a runway overrun upon landing.

Author: Federal Aviation Admin., Washington, DC, USA, 2016.

In [3], Park, and Kim (2023) propose the use of real time flight simulation data for predictive analytics in aviation. This data offers a more comprehensive understanding of flight dynamics compared to historical data. The study discusses how this data can be analysed using machine learning to predict and prevent potential flight issues, aiming to improve prediction accuracy.

Ref: 4

Title: Why and when to perform a go-around maneuver.

Author: M. Coker and L. S. Pilot, Boeing Edge, vol. 2014, pp. 5–11, 2014.

In [4], This study by Gupta, Singh, and Reddy (2022) explores the use of deep learning to improve flight safety during the critical approach phase. Highlighting the challenges of this stage, the authors propose models that analyses flight data in real-time to predict and mitigate potential hard landing scenarios.

Ref: 5

Title: Go-around decision making and execution project: Final report to flight safety foundation Flight Saf. Found.

Author: T. Blajev and W. Curtis, Alexandria, VA, USA, Mar. 2017.

In [5], This paper describes a cockpit-deployable machine learning system to support flight crew go around decision-making based on the prediction of hard landing events. The hybrid approach uses features modelling temporal dependencies of aircraft variables as inputs to a neural network.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

A Hard Landing (HL) is a phenomenon in which the airplane has an excessive impact on the ground at the moment of landing. This impact is directly related to the vertical (or normal) acceleration; therefore, HL can be defined as flights where the vertical acceleration exceeds the limited value of the aircraft type during the landing phase. A threshold on such normal acceleration (Airbus uses vertical acceleration > 2G at Touch Down, TD) triggers maintenance requirement, so that can be considered as a criterion for HL detection.

Under the former definition of HL, existing approaches for HL prediction can be split into two groups: those based on a classifier to discriminate flights with normal acceleration at TD above a given threshold from other flights and those based on a regressor that predicts the normal acceleration with the aim of using this predicted value as the HL detector. These manual methods rely heavily on the skill and judgment of the flight crew, but they do not have the advantage of real-time data analysis or predictive modeling to help guide decisions.

Furthermore, some commercial flight systems utilize basic sensors to monitor factors such as descent rate, speed, and altitude during landing, but these systems typically focus on ensuring that the aircraft is within safe operating limits rather than predicting the likelihood of a hard landing.

3.1.1 DISADVANTAGES OF EXISTING SYSTEM

- An existing system not implemented Sources of errors and capability for go-around recommendation.
- An existing system not implemented hybrid approach for hard landing prediction that uses features modeling temporal dependencies of aircraft variables as inputs to a neural network.
- Most current systems do not incorporate machine learning or predictive analytics to forecast hard landings based on flight data.
- They primarily monitor flight parameters like speed, altitude, and descent rate without offering insights into potential landing risks.

- Limited Data Availability: High-quality aviation data is often restricted due to security and privacy concerns.
- Imbalanced Data: Hard landings are rare compared to normal landings, leading to imbalanced datasets, which can affect model performance.
- Here are some of the information about the accidents that are happened during the aviation the flights. The accidents are categorized into two types **fatal** and **non-fatal.**
- A fatal injury is defined as one which results in death, whereas a nonfatal injury is defined as one which results in at least 4 days absence from work.

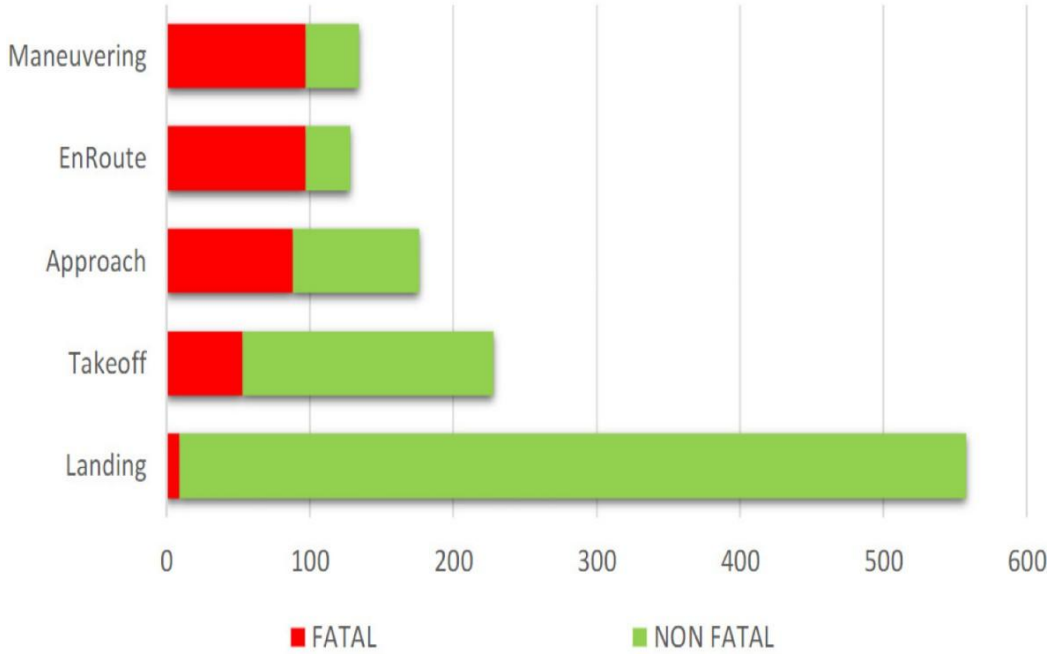

Fig:3.1.1 Fatal and Non-Fatal Accidents

Landing: Has the highest number of non-fatal accidents but the lowest number of fatal accidents.

Maneuvering: Has a relatively high proportion of fatal accidents compared to non-fatal ones.

Takeoff: Has a significant number of non-fatal accidents and a noticeable number of fatal accidents.

En-Route: Approach show a similar pattern, with more non-fatal accidents than fatal ones.

3.2 PROPOSED SYSTEM

This project presents an analysis of approaches for early prediction of hard-landing events in commercial flights. Unlike previous works, experiments are designed to analyze to what extend methods can be deployable in the cockpit as go around recommendation systems. With this final goal, we contribute to the following aspects:

The Contributions of this work are follows:

1) Hybrid model with optimized net architecture.

Propose a hybrid approach that uses features modeling temporal dependencies of aircraft variables as input to a neural network with an optimized architecture. In order to avoid any bias caused by a lack of convergence of complex models (like LSTM), we use a standard network and model potential temporal dependencies associated with unstable approaches as the variability of different types of aircraft variables at a selected set of altitudes.

The concatenation of such variability for variables categorized into 4 main types (physical, actuator, pilot operations and all of them) are the input features of different architectures in order to determine the optimal subset.

2) Exhaustive comparison to Soa in a large database of commercial flights.

A main contribution compared to existing works is that our models have been tested and compared to Soa methods on a large database of Flight Management System (FMS) recorded data of an airline no longer in operation that includes 3 different aircraft models (A319, A320, A321).

Results show that the optimal classification network when all variable types are considered achieves an average recall of HL events of 85% with a specificity of 75% in average, which outperforms current LSTM methods found in the literature.

3) Analysis of the performance of classifiers and regressors.

With the final goal of developing a cockpit deployable recommendation system we have conducted a study of the performance of classification and regression models in terms of the flight height and different aircraft variables including the impact of automation and pilot man oeuvres.

Results on our large dataset of commercial flights, show that although our regression networks perform similarly to Soa methods the accuracy for detecting HL is very poor. This indicates that regression models might not be the most appropriate for the detection of HL events in a cockpit deployable support system.

4) Sources of errors and capability for go-around recommendation.

Unlike previous approaches, we analyze the capability of networks for the detection of HL before the decision height, as well as, the influence of the operational context. We have also performed an analysis of the sources of errors, including selection of the best variable type, optimal altitude range used for predictions, biases due to aircraft type and capability of regressors for HL prediction.

3.2.1 ADVANTAGES OF PROPOSED SYSTEM

- The machine learning approach can also be improved in several aspects. Although results appear superior to existing methods, our models would benefit from a more complex analysis of temporal dependencies using a convolutional neural network to extract deep dependencies.
- In the proposed system, for a cockpit-deployable machine learning system to support flight crew go-around decision, some results regarding the hardware and software requirements, especially for the speed of networks should be investigated.
- The system provides real-time predictions of potential hard landings, allowing flight crews to take corrective actions before landing, thereby preventing incidents.
- By predicting hard landings in advance, the system enhances flight safety, reducing the risk of aircraft damage and protecting passengers and crew from the potential consequences of a hard landing.
- Robust Decision-Making: Decision trees offer clear rules, while Naïve Bayes provides probabilistic insights, leading to better predictive power.
- Improved Classification Accuracy: Using both algorithms can help cross-validate results for better reliability.
- Scalability: These models can be easily scaled as the project grows.

3.3 PROJECT ALGORITHM

Naïve Bayes Algorithm: It is a probabilistic machine learning model based on **Bayes' Theorem**, which describes the likelihood of an event occurring given prior knowledge of related conditions. It is particularly useful for classification tasks, as it calculates the probability of different classes based on input features and assigns the most probable class label.

- The algorithm is termed "naïve" because it assumes that all features used for classification are **independent** of each other, which may not always be the case in real-world scenarios. Despite this simplifying assumption, Naïve Bayes often performs well

in various applications, including text classification, spam detection, medical diagnosis, and even predictive modeling in aviation.

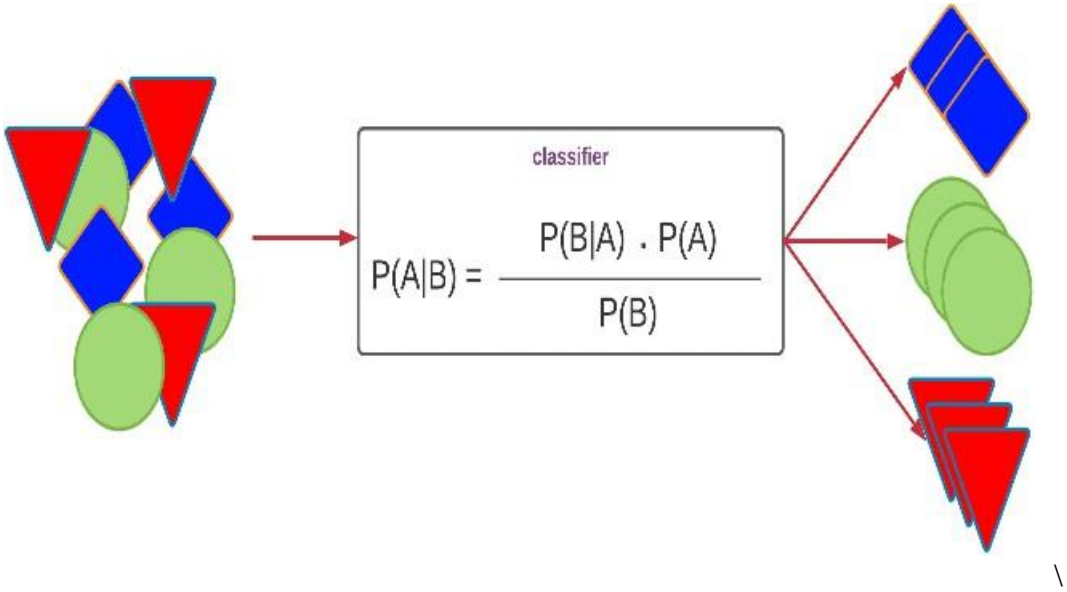

Fig:3.3.1: Naïve Bayes Classifier

- In operation, the algorithm calculates the probability of an instance belonging to a particular category by multiplying the prior probability of that category with the likelihood of the observed features occurring within it. The classification decision is then made by selecting the category with the highest computed probability.
- Naïve Bayes is computationally efficient and works well with both small and large datasets. It requires minimal training time, making it a good choice for real-time applications where quick decisions are necessary. However, one limitation of the algorithm is that its assumption of feature independence may not always hold true, potentially affecting accuracy in complex datasets where features are interdependent. Despite this, Naïve Bayes remains a powerful tool for many classification problems, especially when probabilistic reasoning and interpretability are essential.
- The core of the solution is the machine learning model, which is likely to use algorithms such as **decision trees, Naïve bayes classifier** along with **CNN** for data collection
- **Decision tree:** Decision tree modeling is one of several supervised learning techniques that are particularly well-suited for virtual training applications because they employ a repository of solved problems to draw inferences.

DECISION TREE ALGORITHM: Decision tree modeling is one of several supervised learning techniques that are particularly well-suited for virtual training applications because they employ a repository of solved problems to draw inferences.

As the name says all about it, it is a tree which helps us by assisting us in decision making. Used for both classification and regression, it is a very basic and important predictive learning algorithm.

- It is different from others because it works intuitively i.e., taking decisions one-by-one.
- Non-parametric: Fast and efficient.
- It consists of nodes which have parent-child relationship**.**

Decision Tree considers the most important variable using some fancy criterion and splits the dataset based on it. It is done to reach a stage where we have homogenous subsets that are giving predictions with utmost surety.

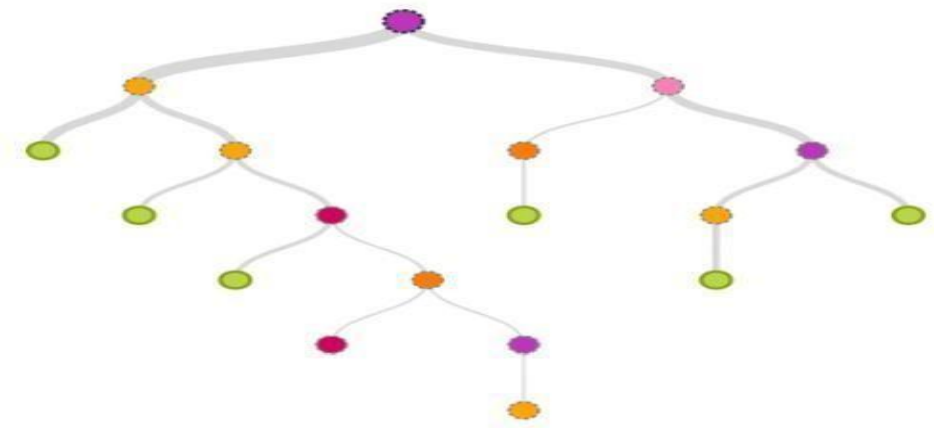

Fig.3.3.2 Decision Tree

3.4. ML SYSTEM ARCHITECTURAL DIAGRAM

EXPLANATION:

- The process begins with data collection, the data is collected from different sources as well as the real time data is collected.
- **Data cleaning:** Where the data is refined by removing unwanted data or null data and prepared for analysis.

- **Algorithms:** Algorithms like decision tree and naïve bayes classifier are then applied to the cleaned data, resulting in a model. This model undergoes evaluation before being used for prediction.
- In essence, the image provides a visual overview of the steps involved in transforming raw data into actionable insights through machine learning classification problems, especially when probabilistic reasoning and interpretability are essential.

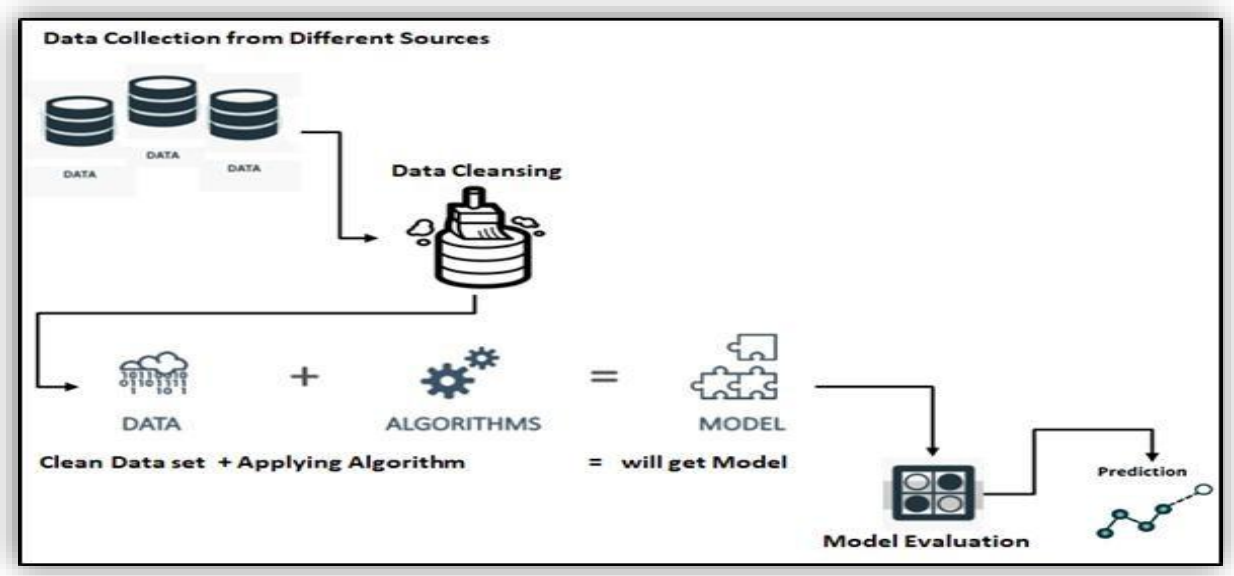

Fig:3.4:ML Architectural diagram.

Steps to Implement Algorithms:

1. **Data Preprocessing**:

● Handle missing data (impute or remove missing values)

● Encode categorical variables (e.g., one-hot encoding, label encoding)

● Split the data into training and testing sets

2. **Feature Scaling** (optional but recommended):

● Scale the features to a common range (e.g., standardization, normalization)

● This step can improve the convergence of the algorithm

3. **Importing the algorithms:**

● In Python, you can import the algorithms from the scikit-learn library:

from sklearn.tree import DecisionTreeClassifier

from sklearn. ensemble import NaivebayesClassifier

4. **Creating the Models on Algorithms**:

● You can specify the parameters for the model, such as the solver, regularization method, and regularization strength.

5. **Train the Model**:

● Fit the above-mentioned models to the training data.

model. Fit (train, y_train), where X_train is the training feature data, and y_train is the target variable for the training set.

6. **Make Predictions**:

● Use the trained models to make predictions on the test data.

7. **Interpret the Model**:

● Analyse the model performance, understand its workings and also verify accuracy.

8. **Deploy the Model**:

● Once you have a satisfactory model, you can deploy it to make predictions on new, unseen data.

3.5 SYSTEM SPECIFICATIONS:

3.5.1 HARDWARE REQUIREMENTS

- Processor- Intel (R) Core (TM) i3-4200U
- CPU - 1.6GHz
- RAM:4 GB
- Hard Disk: 500 GB.

3.5.2 SOFTWARE REQUIREMENTS

- Operating System - Windows 10
- Server - XAMPP Server
- Front End - HTML, CSS, JS
- Back End - Python
- Data base - MYSQL
- IDE - Py charm

3.6 FEASIBILITY STUDY

An important outcome of preliminary investigation is the determination that the system request is feasible. This is possible only if it is feasible within limited resource and time. The different feasibilities that have to be analyzed are

- **Operational Feasibility**
- **Economic Feasibility**
- **Technical Feasibility**

3.6.1 Operational Feasibility

Operational Feasibility deals with the study of prospects of the system to be developed. This system operationally eliminates all the tensions of the admin and helps him in effectively tracking the project progress. This kind of automation will surely reduce the time and energy, which previously consumed in manual work. Based on the study, the system is proved to be operationally feasible.

3.6.2 Economic Feasibility

Economic Feasibility or Cost-benefit is an assessment of the economic justification for a computer-based project. As hardware was installed from the beginning & for lots of purposes thus the cost on project of hardware is low. Since the system is a network based, any number of employees connected to the LAN within that organization can use this tool from at any time. The Virtual Private Network is to be developed using the existing resources of the organization. So, the project is economically feasible.

3.6.3 Technical Feasibility

According to Roger S. Pressman, Technical Feasibility is the assessment of the technical resources of the organization. The organization needs IBM compatible machines with a graphical web browser connected to the Internet and Intranet. The system is developed for platform independent environment. Python, JavaScript, HTML, SQL server and WebLogic Server are used to develop the system. The technical feasibility has been carried out. The system is technically feasible for development and can be developed with the existing facility.

CHAPTER-4

SYSTEM DESIGN

It is a process of planning a new business system or replacing an existing system by defining its components or modules to satisfy the specific requirements. Before planning, you need to understand the old system thoroughly and determine how computers can best be used in order to operate efficiently.

4.1 SYSTEM ARCHITECTURE

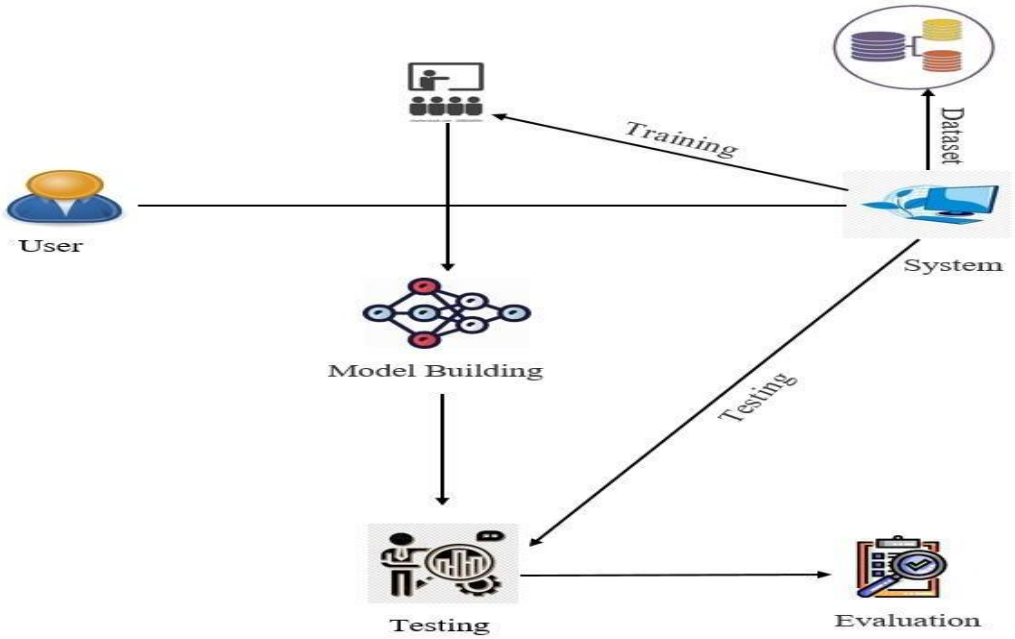

Fig 4.1 System Architecture

The diagram appears to illustrate a system for predicting type of crime and detecting crime hotspots. It involves various steps:

- Data Collection
- Training
- Model Building
- Testing
- Evaluation
- Prediction

4.2 MODULES

In this Proposed System. Modules are:

Admin

In this module, the admin has to login by using valid user name and password. After login successful he can do some operations such as Login, Browse Flight Landing Data Sets and Train & Test, View Flight Landing Trained and Tested Accuracy in Bar Chart, View Flight Landing Trained and Tested Accuracy Results, View Prediction of Flight Landing Type, View Flight Landing Type Ratio, Download Predicted Data Sets, View Flight Landing Ratio Results, View All Remote Users.

User

In this module, there are n numbers of users are present. User should register before doing any operations. Once user registers, their details will be stored to the database. After registration successful, he has to login by using authorized user name and password. Once Login is successful user will do some operations like Register and Login, Predict Flight Landing Type, View Your Profile.

Data Collection Module

This module gathers real-time and historical flight data from various sources, including Flight Data Recorders (FDRs), Quick Access Recorders (QARs), and Air Traffic Control (ATC) systems. It collects crucial flight parameters such as vertical speed, descent rate, wind shear, airspeed, altitude, runway conditions, and aircraft weight. The collected data is preprocessed to remove noise and inconsistencies before being used for analysis.

Feature Extraction and Preprocessing Module

In this module, raw flight data is transformed into structured input features required for machine learning models. Important parameters such as approach angle, vertical acceleration, thrust levels, and environmental factors are extracted. Preprocessing techniques such as normalization, missing value handling, and feature scaling are applied to ensure the data is ready for analysis.

Machine Learning Model Development Module

This module is responsible for training predictive models to classify landing types as "hard" or "safe." Various machine learning algorithms such as Naïve Bayes, Random Forest, Support Vector Machines (SVM), and Deep Learning models (e.g., LSTMs and CNNs) are implemented and compared for performance.

4.3 DATA FLOW DIAGRAM

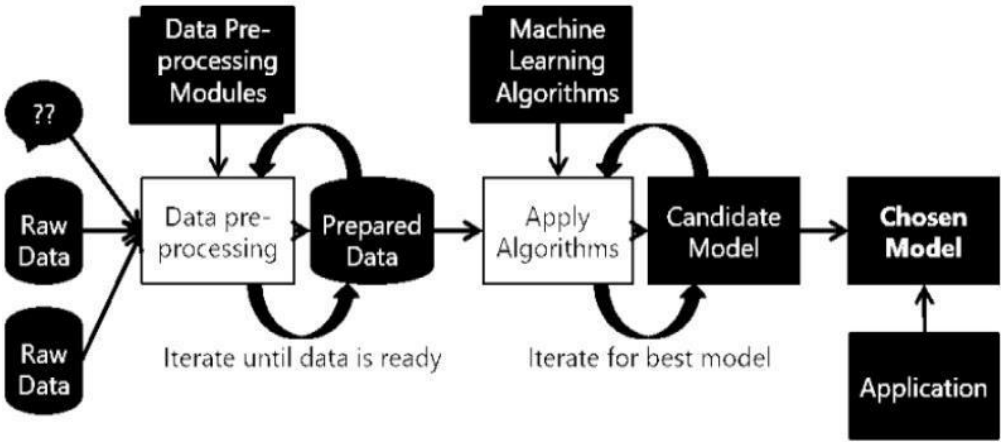

Fig4.3: Data flow Diagram

4.4 UML DIAGRAMS

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 modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling 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 mostlygraphical 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.

As the strategic value of software increases for many companies, the industry looks for techniques to automate the production of software and to improve quality andreduce cost and time-to-market. These techniques include component technology, visual programming, patterns and frameworks. Businesses also seek techniques to manage the complexity of systems as they increase in scope and scale. In particular, they recognize the need to solve recurring architectural problems, such as physical distribution, concurrency, replication, security, load

balancing and fault tolerance.

Additionally, the development for the World Wide Web, while making some things simpler, has exacerbated these architectural problems. The Unified Modeling Language (UML) was designed to respond to these needs. Simply, Systems design refers to the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements which can be done easily through UML diagrams.

Object-Oriented (OO) Concepts

UML can be described as the successor of object-oriented (OO) analysis and design.

An object contains both data and methods that control the data. The data represents the state of the object. A class describes an object and they also form a hierarchy to model the real-world system. The hierarchy is represented as inheritance and the classes can also be associated in different ways as per the requirement.

Objects are the real-world entities that exist around us and the basic concepts such as abstraction, encapsulation, inheritance, and polymorphism all can be represented using UML.

UML is powerful enough to represent all the concepts that exist in object-oriented analysis and design. UML diagrams are representation of object-oriented concepts only. Thus, before learning UML, it becomes important to understand Object Oriented concept in detail.

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

- **Objects** − Objects represent an entity and the basic building block.
- **Class** − Class is the blue print of an object.
- **Abstraction** − Abstraction represents the behavior 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.

Object Oriented Analysis and Design

Object-Oriented can be defined as an investigation and to be more specific, it is the investigation of objects. Design means collaboration of identified objects.

Thus, it is important to understand the Object-Oriented analysis and design concepts. The most important purpose of Object-Oriented analysis is to identify objects of a system to be designed. This analysis is also done for an existing system.

Now an efficient analysis is only possible when we are able to start thinking in a way where objects can be identified. After identifying the objects, their relationships are identified and finally the design is produced.

The purpose of Object-Oriented analysis and design can describe as −

- Identifying the objects of a system.
- Identifying their relationships.
- Making a design, which can be converted to executables using OO languages.

There are three basic steps where the OO concepts are applied and implemented. The steps can be defined as

Object-Oriented Analysis → Object-Oriented Design → Object-Oriented implementation using Object-Oriented languages

The above three points can be described in detail as −

- During Object-Oriented analysis, the most important purpose is to identify objects and describe them in a proper way. If these objects are identified efficiently, then the next job of design is easy. The objects should be identified with responsibilities. Responsibilities are the functions performed by the object. Each and every object has some type of responsibilities to be performed. When these responsibilities are collaborated, the purpose of the system is fulfilled.
- The second phase is Object-Oriented design. During this phase, emphasis is placed on the requirements and their fulfilment. In this stage, the objects are collaborated according to their intended association. After the association is complete, the design is also complete.

- The third phase is Object-Oriented implementation. In this phase, the design is implemented using Object-Oriented languages such as Java, C++, etc.

ROLE OF UML IN OBJECT-ORIENTED DESIGN

UML is a modeling language used to model software and non-software systems. Although UML is used for non-software systems, the emphasis is on modeling Object-Oriented software applications. Most of the UML diagrams discussed so far are used to model different aspects such as static, dynamic, etc. Now whatever be the aspect, the artifacts are nothing but objects.

Hence, the relation between Object-Oriented design and UML is very important to understand. The Object-Oriented design is transformed into UML diagrams according to the requirement.

In this project, basic UML diagrams have been explained

1) Use Case Diagram
2) Class Diagram
3) Sequence Diagram
4) Collaboration Diagram
5) Activity Diagram
6) Deployment Diagram

4.4.1 CLASS DIAGRAM

UML class diagrams model static class relationships that represent the fundamental architecture of the system. Note that these diagrams describe the relationships between classes, not those between specific objects instantiated from those classes. Thus, the diagram applies to all the objects in the system.

A class diagram consists of the following features:

- **Classes:** These titled boxes represent the classes in the system and contain information about the name of the class, fields, methods and access specifies. Abstract roles of the Class in the system can also be indicated
- **Interfaces:** These titled boxes represent interfaces in the system and contain information about the name of the interface and its methods. Relationship Linesthat

model the relationships between classes and interfaces in the system.

- **Dependency:** A dotted line with an open arrowhead that shows one entity depends on the behavior of another entity. Typical usages are to represent that one class instantiates another or that it uses the other as an input parameter.

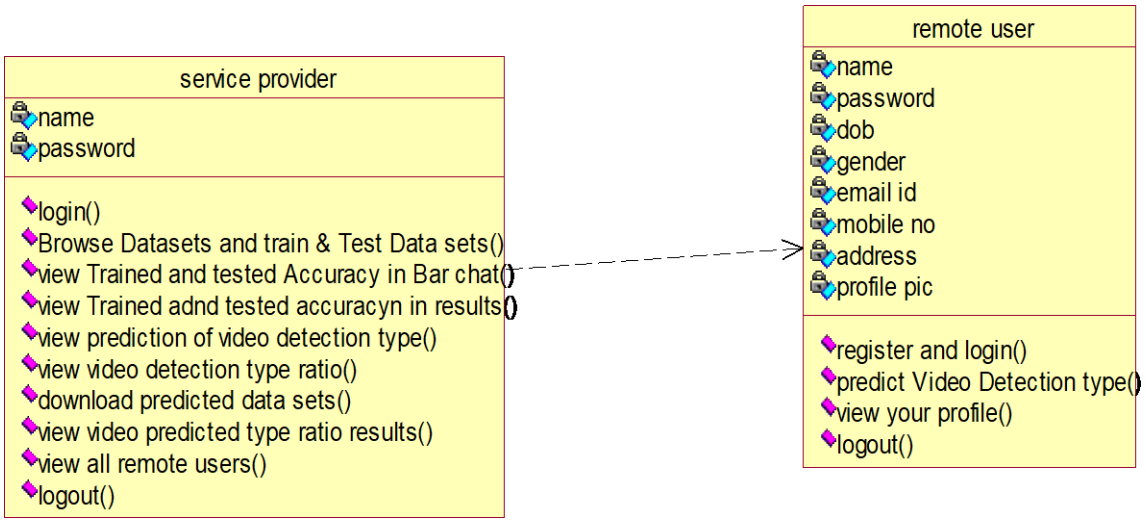

Fig 4.4.1. Class Diagram

4.4.2 USECASE DIAGRAM

A use case diagram in the Unified Modelling 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.

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements.The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It consists of a group of elements (for example, classes and interfaces) that can be used together in a way that will have an effect larger than the sum of the separate elements combined.

The use case should contain all system activities that have significance to the users. A use case can be thought of as a collection of possible scenarios related to a particular goal, indeed, the use case and goal are sometimes considered to be synonymous. 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.

Parts of use case Diagram

System boundary boxes (optional)

A rectangle is drawn around the use cases, called the system boundary box, to indicate the scope of system. Anything within the box represents functionality that is in scope and anything outside the box is not

Relationships.

Include

In one form of interaction, a given use case may include another. "Include is a Directed Relationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case". The first use case often depends on the outcome of the included use case.

This is useful for extracting truly common behaviors from multiple use cases into a single description. The notation is a dashed arrow from the including to the included use case, with the label "«include»". This usage resembles a macro expansion where the included use case behavior is placed inline in the base use case behavior.

There are no parameters or return values. To specify the location in a flow of events in which the base use case includes the behavior of another, you simply write include followed by the name of use case you want to include, as in the following flow for track order.

Extend

In another form of interaction, a given use case (the extension) may extend another. This relationship indicates that the behavior of the extension use case may be inserted in the extended use case under some conditions.

The notation is a dashed arrow from the extension to the extended use case, with the label "«extend»".

The notes or constraints may be associated with this relationship to illustrate the conditions under which this behavior will be executed.

Modelers use the «extend» relationship to indicate use cases that are "optional" to the base use case.

Depending on the modeler's approach "optional" may mean "potentially not executed with the base use case" or it may mean "not required to achieve the base use case goal"

4.4.2.1 USE CASE DIAGRAM FOR REMOTE USER

1. Remote User: A pilot or aviation professional accessing the e-Pilot system remotely.

Use Cases:

1. Predict Hard Landing: The remote user inputs flight data, and the ML model predicts the likelihood of a hard landing.

- Primary Actor: Remote User
- Secondary Actor: Machine Learning Model

- **Username:** Unique identifier for the user.
    1. Username/Email: User's username or email address.
    2. Password: User's password
- **Email:** User's email address.
- **Password**: User's password.
- **Confirm Password:** Verification of the user's password

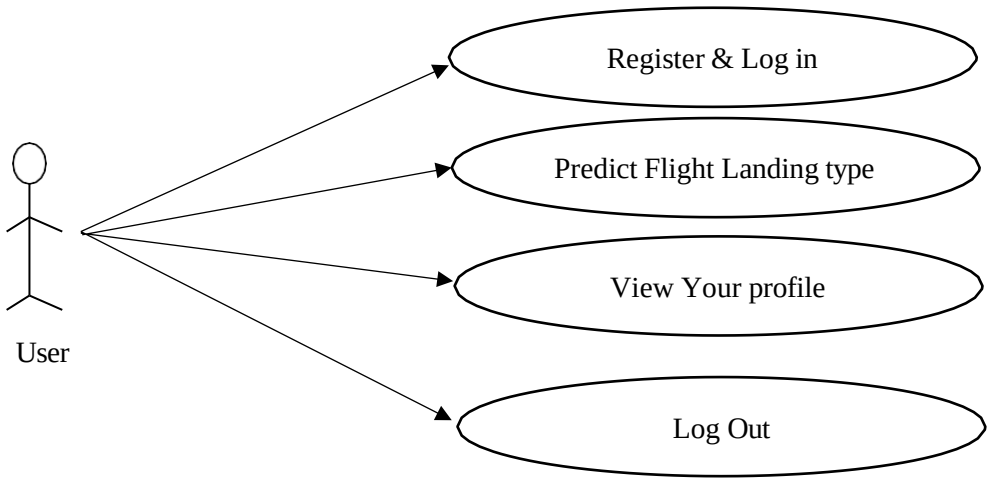

Fig 4.4.2.1 Use Case Diagram for Remote User.

4.4.2.2USE CASE DIAGRAM FOR SERVICE PROVIDER

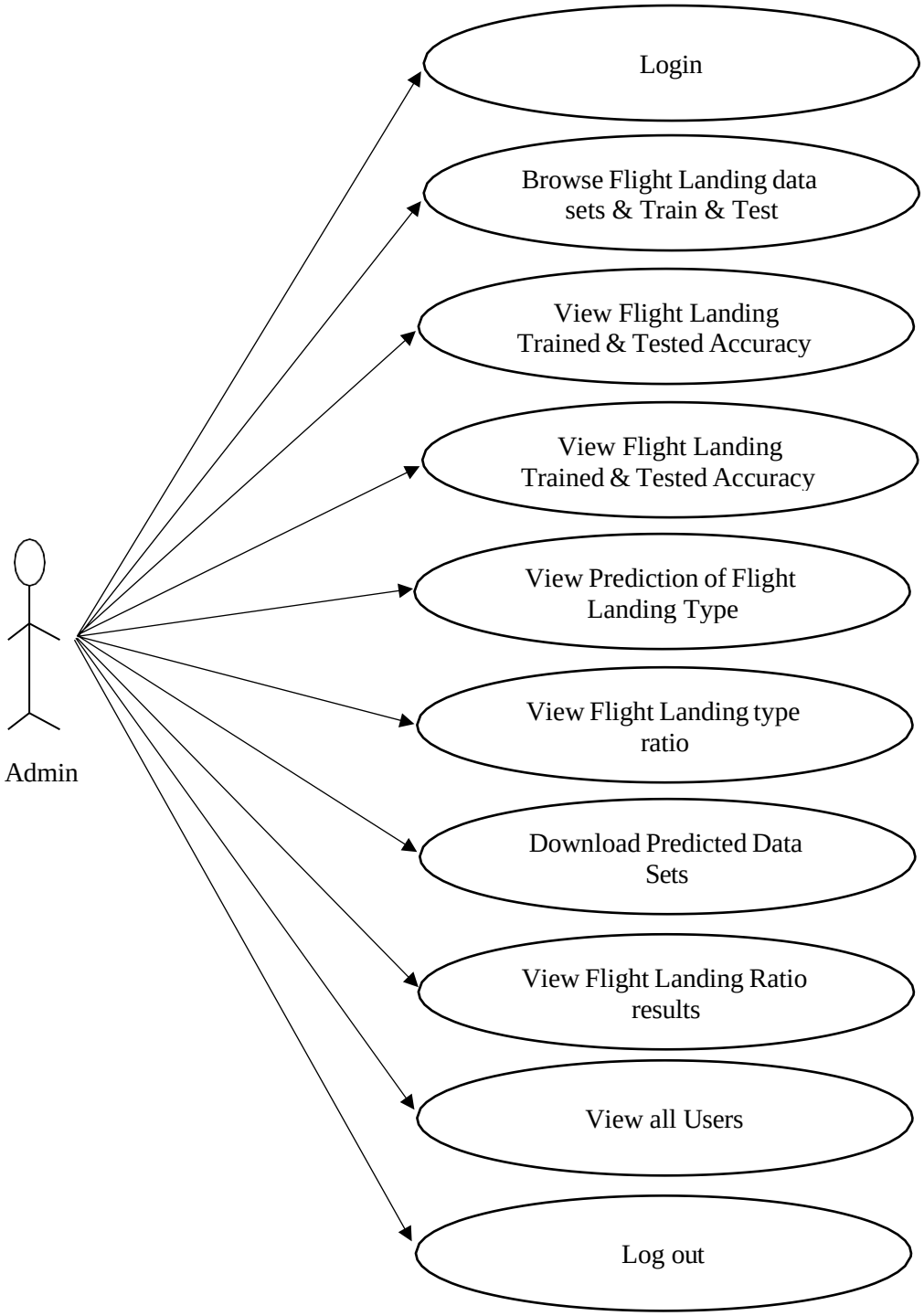

Fig 4.4.2.2 Use case Diagram for service provider

4.4.3. 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 a construct of a Message Sequence Chart. A Sequence diagram depicts the sequence of actions that occur in a system. The invocation of methods in each object, and the order in which the invocation occurs is captured in a Sequence diagram. This makes the Sequence diagram a very useful tool to easily represent the dynamic behavior of a system.

Elements of sequence diagram

The sequence diagram is an element that is used primarily to showcase the interaction that occurs between multiple objects. This interaction will be shown over certain period of time. Because of this, the first symbol that is used is one that symbolizes the object.

Lifeline

A lifeline will generally be generated, and it is a dashed line that sits vertically,and the top will be in the form of a rectangle. This rectangle is used to indicate both theinstance and the class. If the lifeline must be used to denote an object, it will be underlined.

Messages

To showcase an interaction, messages will be used. These messages will come in the form of horizontal arrows, and the messages should be written on top of the arrows. If the arrow has a full head, and it’s solid, it will be called a synchronous call. If the solid arrow has a stick head, it will be an asynchronous call. Stick heads with dash arrows are used to represent return messages.

Objects

Objects will also be given the ability to call methods upon themselves, and theycan add net activation boxes. Because of this, they can communicate with others to show multiple levels of processing. Whenever an object is eradicated or erased from memory, the "X" will be drawn at the lifeline's top, and the dash line will not be drawnbeneath it. This will often occur as a result of a message. If a message is sent from the outside of the diagram, it can be used to define a message that comes from a circle thatis filled in. Within a UML based model, a Super step is a collection of steps which resultfrom outside stimuli.

Steps to Create Sequence Diagram

- Set the stage for the interaction by identifying which object play a role in interaction.
- Set their time or each object.

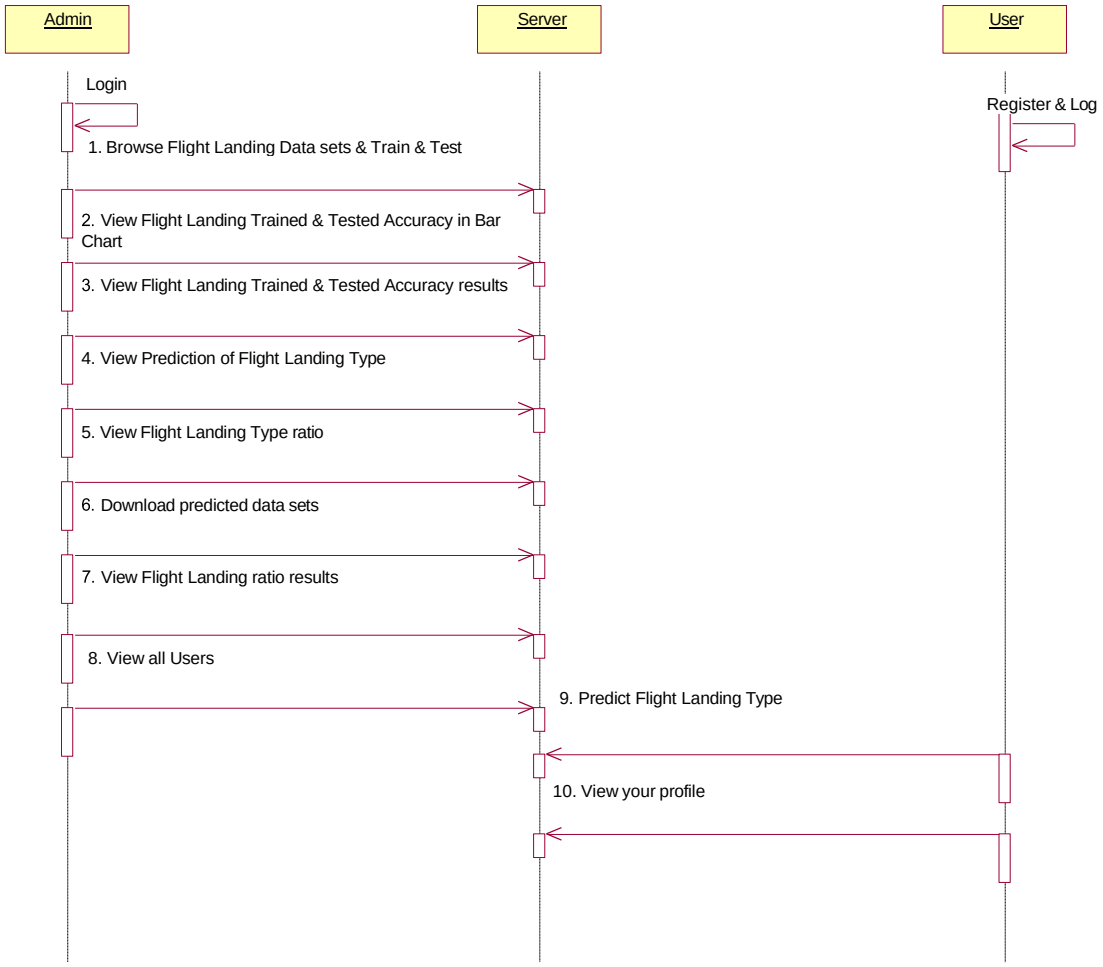

Fig 4.4.3. Sequence Diagram

4.4.4. ACTIVITY DIAGRAM

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So, the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deal with all type of flow control by using different elements like fork, join etc.

How to draw Activity Diagram?

Activity diagrams are mainly used as a flow chart consists of activities performed by the system. But activity diagram is not exactly a flow chart as they havesome additional capabilities. These additional capabilities include branching, parallel flow, swim lane etc. Before drawing an activity diagram, we must have a clearunderstanding about the elements used in activity diagram. The main element of an activity diagram is the activity itself. An activity is a function performed by the system.After identifying the activities, we need to understand how they are associated with constraints and conditions. So before drawing an activity diagram we should identify the following elements.

- Activities
- Association
- Conditions
- Constraints

The following are the basic notational elements that can be used to make up a diagram:

Initial state

An initial state represents a default vertex that is the source for a single transitionto the default state of a composite state. There can be at most one initial vertex in a region. The outgoing transition from the initial vertex may have a behavior, but not a trigger or guard. It is represented by Filled circle, pointing to the initial state.

Final state

A special kind of state signifying that the enclosing region is completed. If the enclosing region is directly contained in a state machine and all other regions in the state machine also are completed, then it means that the entire state machine iscompleted. It is represented by Hollow circle containing a smaller filled circle, indicating the final state.

Rounded rectangle

It denotes a state. Top of the rectangle contains a name of the state. Can containa horizontal line in the middle, below which the activities that are done in that state areindicated.

Arrow

It denotes transition. The name of the event (if any) causing this transition labels the arrow body.

4.4.4.1 ACTIVITY DIAGRAM FOR REMOTE USER

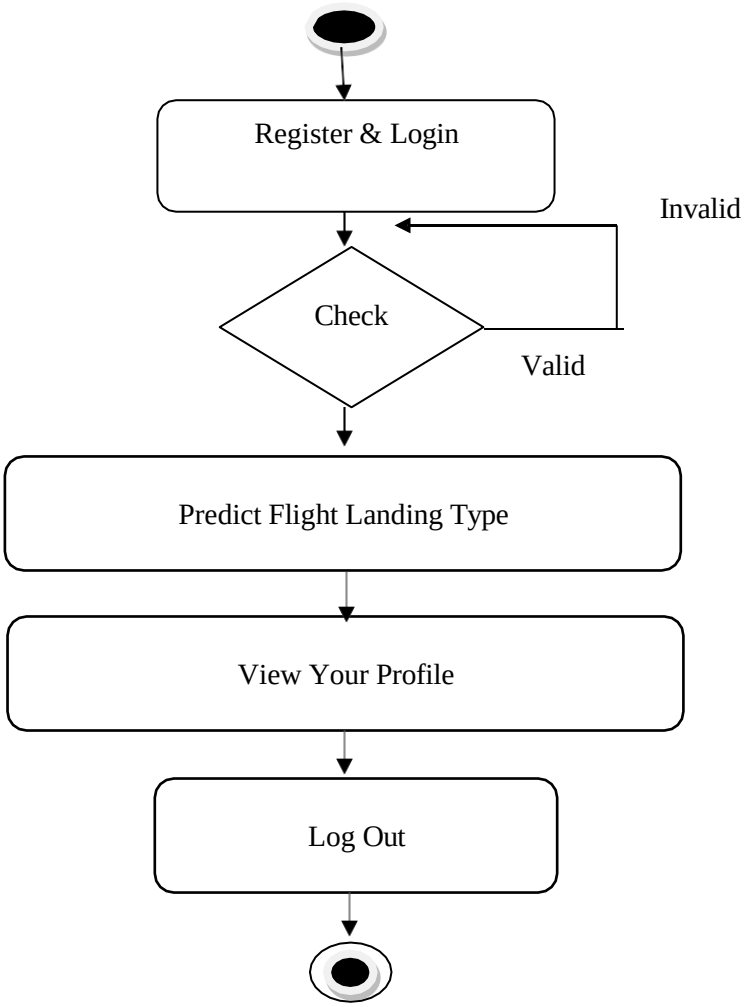

Fig 4.4.4.1 Activity diagram For Remote User

4.4.4.2 ACTIVITY DIAGRAM FOR SERVICE PROVIDER

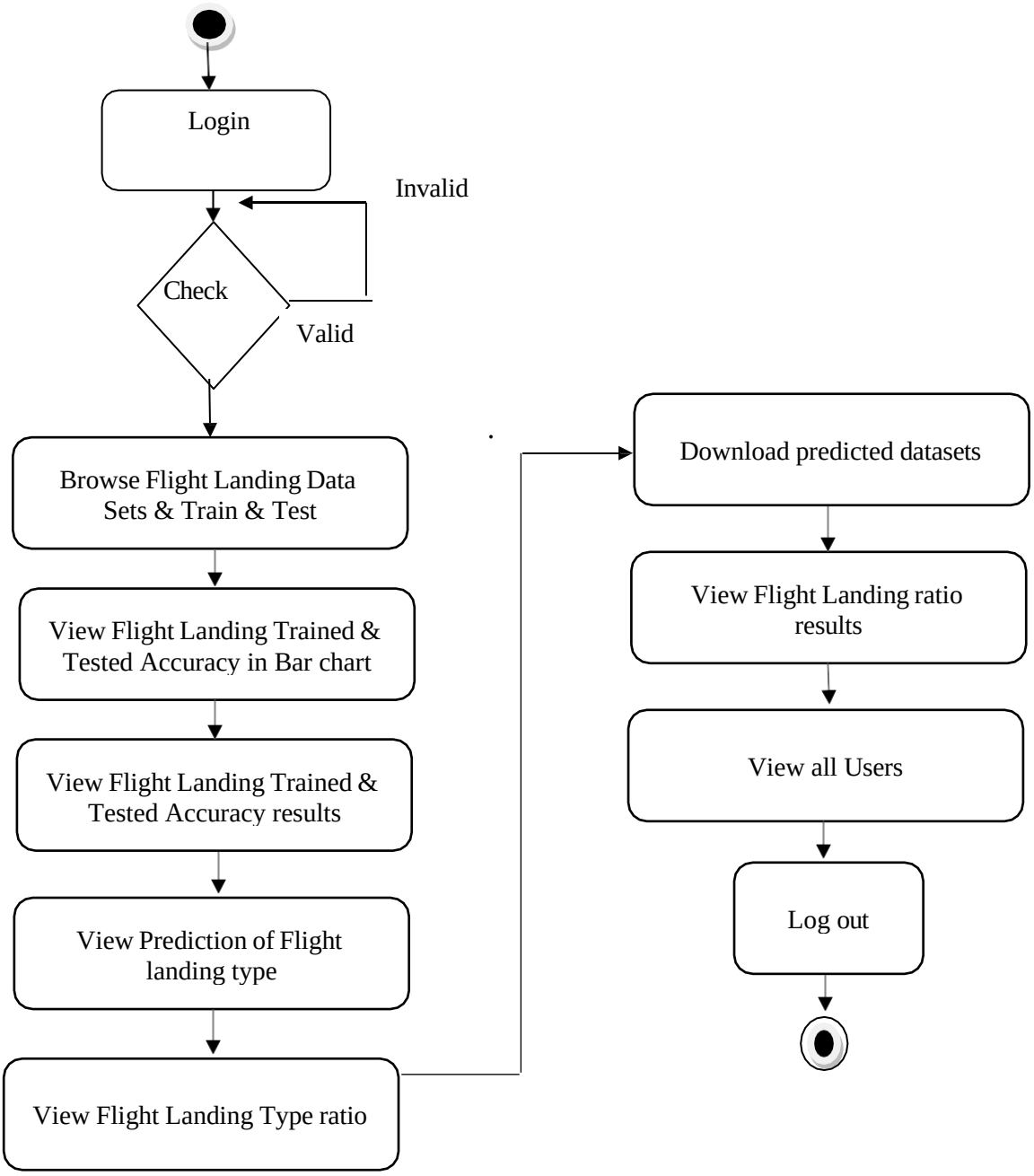

Fig 4.4.4.2 Activity diagram For Service provider

4.4.5. DEPLOYMENT DIAGRAM

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

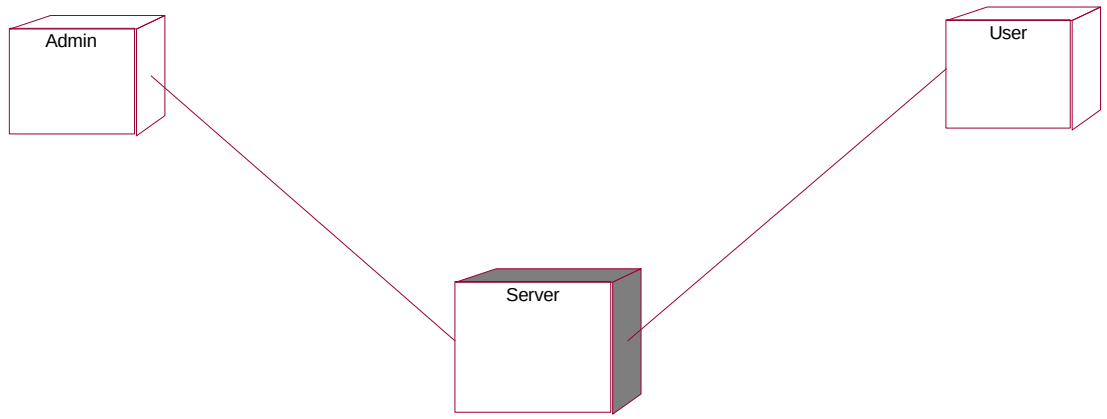

Fig 4.4.5. Deployment Diagram

4.4.6. COLLABORATION DIAGRAM

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object.

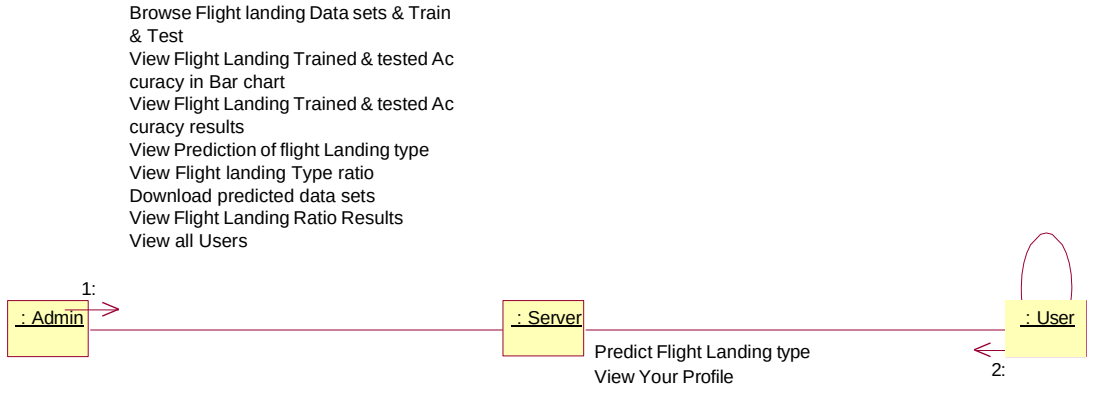

Fig 4.4.6. Collaboration Diagram

CHAPTER-5
SYSTEM IMPLEMENTATION

5.1 SOFTWARE DESCRIPTION

Documentation gives aircraft technicians contextual information for performing maintenance specific to the aircraft. Unfortunately, it's not uncommon for owners or operators to forget to provide some of these necessary documents during aircraft maintenance – potentially resulting in higher costs and a longer downtime.

As an MRO, we know what aircraft documentation is necessary to ensure maintenance is accomplished efficiently and in accordance with aviation authority regulations. In this article, we provide a quick overview of what documents to include with your aircraft.

5.2 INTRODUCTION TO 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 a greater number of Python Programmers due to its application in Machine Learning, Artificial Intelligence etc.

What Is a 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/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 where as 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++, Algol-68, 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 fairly 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 byte-code 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 a number of 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

5.2.1 Python Packages

NUMPY

In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called **ndarray**, it provides a lot of supporting functions that make working with **ndarray** very easy. Arrays are very frequently used in data science, where speed and resources are very important.

PANDAS

Pandas allows us to analyze big data and make conclusions based on statistical theories. Pandas can clean messy data sets, and make them readable and relevant. Relevant data is very important in data science.

MATPLOTLIB

Matplotlib is a low-level graph plotting library in python that serves as a visualization utility. Matplotlib was created by John D. Hunter. Matplotlib is open source and we can use it freely. Matplotlib is mostly written in python, a few segments are written in C, Objective-C and JavaScript for Platform compatibility.

SCIKIT AND SKLEARN

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon **NumPy, SciPy** and **Matplotlib**.

Pillow A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.

SQL Alchemy

A database library

Twisted

The most important tool for any network application developer. It has a very beautiful ape and is used by a lot of famous python developers.

Numbly

How can we leave this very important library? It provides some advance math functionalities to python.

Skippy

When we talk about numbly then we have to talk about spicy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.

Pygmy

Which developer does not like to play games and develop them? This library will help you achieve your goal of 2d game development.

Piglet

A 3d animation and game creation engine. This is the engine in which the famous python port of mine craft was made

Pit.

A GUI toolkit for python. It is my second choice after python for developing GUIs for my python scripts.

Scaly

A packet sniffer and analyzer for python made in python.

Pywin32.

A python library which provides some useful methods and classes for interacting with windows.

Notch. Natural Language Toolkit –

most people won't be using this one, but it's generic enough. It is a very useful library if you want to manipulate strings.

5.2.2 Python Modules

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be imported into other modules or into the main module.

There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.

This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-

program and called when needed. That means that a function is a piece of code written to carry out a specified task. To carry out that specific task, the function might or might not need multiple inputs. When the task is carved out, the function can or cannot return one or more values. There are three types of functions in python:

Help (), min (), print ().

Generally speaking, a **namespace** (sometimes also called a context) is a naming system for making names unique to avoid ambiguity. Everybody knows a name spacing system from daily life, i.e. the naming of people in first name and family name (surname).

An example is a network: each network device (workstation, server, printer,) needs a unique name and address. Yet another example is the directory structure of file systems. The same file name can be used in different directories, the files can be uniquely access via the pathnames. Many programming languages use namespaces or contexts for identifiers. An identifier defined in a namespace is associated with that name space. This way, the same identifier can be independently defined in multiple namespaces. (Like the same file names in different directories) Programming languages, which support namespaces, may have different rules that determine to which namespace an identifier belongs.

Namespaces in Python are implemented as Python dictionaries; this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

- **global names** of a module
- **local names** in a function or method invocation
- **built-in names**: this namespace contains built-in functions (e.g. abs (), camp (), ...) and built-in exception names

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

5.2.3 Python XML Parser

XML is a portable, open-source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language.

What is XML? The Extensible Mark-up Language XML is a mark-up language much like HTML or SGML.

This is recommended by the World Wide Web Consortium and available as an open standard.

XML is extremely useful for keeping track of small to medium amounts of data without requiring a SQL-based backbone.

XML Parser Architectures and APIs the Python standard library provides a minimal but useful set of interfaces to work with XML.

The two most basic and broadly used APIs to XML data are the SAX and DOM interfaces.

Simple API for XML SAX: Here, you register callbacks for events of interest and then let the parser proceed through the document.

This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory.

Document Object Model DOM API: This is a World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical tree − based form to represent all the features of an XML document.

SAX obviously cannot process information as fast as DOM can when working with large files. On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files. SAX is read-only, while DOM allows changes to the XML file. Since these two different APIs literally complement each other, there is no reason why you cannot use them both for large projects.

Python allows parsing these XML documents using two modules namely, the xml. tree. Element Tree module and Minidome (Minimal DOM Implementation). Parsing means to read information from a file and split it into pieces by identifying parts of that particular XML file. Let’s move on further to see how we can use these modules to parse XML data.

There are two ways to parse the file using 'Element Tree' module. The first is by using the **parse () function** and the second is **fromstring () function**. The parse () function parses XML document which is supplied as a file whereas, fromstring parses XML when supplied as a string i.e. within triple quotes.

5.3 WEB FRAMEWORK

5.3.1 Python Web frameworks

A web framework is a code library that makes a developer's life easier when building reliable, scalable and maintainable web applications.

Why are web frameworks useful?
Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web. Frameworks make it easier to reuse code for common HTTP operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.
Common web framework functionality

Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:

URL routing
HTML, XML, JSON, and other output format tinplating
Database manipulation
Security against Cross-site request forgery (CSRF) and other attacks
Session storage and retrieval
Not all web frameworks include code for all of the above functionality. Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer. Some frameworks take the "batteries-included" approach where everything possibly comes bundled with the framework while others have a minimal core package that is amenable to extensions provided by other packages.

Comparing web frameworks
There is also a repository called compare-python-web-frameworks where the same web application is being coded with varying Python web frameworks, tinplating engines and object.

Web framework resources

When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing.

Frameworks is a really well-done short video that explains how to choose between web frameworks. The author has some particular opinions about what should be in a framework. For the most part I agree although I've found sessions and database ORMs to be a helpful part of a framework when done well.

What is a web framework? Is an in-depth explanation of what web frameworks being and their relation to web servers?

Jingo vs. Flash vs. Pyramid: Choosing a Python web framework contains background information and code comparisons for similar web applications built in these three big Python frameworks.

This fascinating blog post takes a look at the code complexity of several Python web frameworks by providing visualizations based on their code bases.

Python's web frameworks benchmarks is a test of the responsiveness of a framework with encoding an object to JSON and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but the output is fun to read about nonetheless.

What web frameworks do you use and why are they awesome? Is a language agnostic Reedit discussion on web frameworks? It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main Python frameworks.

This user-voted question & answer site asked "What are the best general purpose Python web frameworks usable in production?" The votes aren't as important as the list of the many frameworks that are available to Python developers.

Web frameworks learning checklist

Choose a major Python web framework (Jingo or Flask are recommended) and stick with it. When you're just starting it's best to learn one framework first instead of bouncing around trying to understand every framework.

Work through a detailed tutorial found within the resource's links on the framework's page.

Study open-source examples built with your framework of choice so you can take parts of those projects and reuse the code in your application.

Build the first simple iteration of your web application then go to the deployment section to make it accessible on the web.

Flask Framework

A Web Application Framework or a simply a Web Framework represents a collection of libraries and modules that enable web application developers to write applications without worrying about low- level details such as protocol, thread management, and so on.

Flask is a web framework; it's a Python module that lets you develop web applications easily. It has a small and easy-to-extend core: it's a micro framework that doesn't include an ORM (Object Relational Manager) or such features. It does have many cool features like URL routing, template engine. It is a WSGI web app framework.

Fig 5.3.1. Flash Framework Diagram

Django Framework

Django is one of the most popular python frameworks available now. It is a python framework with DRY (don't repeat yourself) concept. It has a model-view-template architectural pattern and emphasizes on reusability of components. Its flexibility makes it a desired choice for start-ups that are working on a restrained budget.

Highlights

- Follows MVT (model-view-template) pattern
- Vast library collections for full-stack web development
- Supports URL routing
- Data is delivered as object relational mapping (ORM)

- Supports databases like MySQL, SQLite, Oracle and PostgreSQL.
- Adaptable to third party drivers
- Built-in authentication system

Due to availability of multiple libraries and its ability to migrate from one database to another, Django is one of the most popular web development frameworks in use at present.

Features of Django

✧ Rapid Development

✧ Secure

✧ Scalable

✧ Fully loaded

✧ Versatile

✧ Open Source

✧ Vast and Supported Community

Rapid Development

Django was designed with the intention to make a framework which takes less time to build web application. The project implementation phase is a very time taken but Django creates it rapidly.

Secure

Django takes security seriously and helps developers to avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery etc. Its user authentication system provides a secure way to manage user accounts and passwords.

Scalable

Django is scalable in nature and has ability to quickly and flexibly switch from small to large scale application project.

Fully loaded

Django includes various helping task modules and libraries which can be used to handle common Web development tasks. Django takes care of user authentication, content administration, site maps, RSS feeds etc.

Versatile

Django is versatile in nature which allows it to build applications for different-different domains. Now a days, Companies are using Django to build various types of applications like: content management systems, social networks sites or scientific computing platforms etc.

Open Source

Django is an open-source web application framework. It is publicly available without cost. It can be downloaded with source code from the public repository. Open source reduces the total cost of the application development.

Vast and Supported Community

Django is a one of the most popular web frameworks. It has widely supportive community and channels to share and connect.

CHAPTER-6

SYSTEM TESTING

6.1 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 tests. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTS

6.2.1 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 a 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.

6.2.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event 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.

6.2.3 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.

6.2.4 System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

6.2.5 White Box Testing

White Box Testing is a testing in which 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.

6.2.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as 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 testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

6.3 UNIT TESTING:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

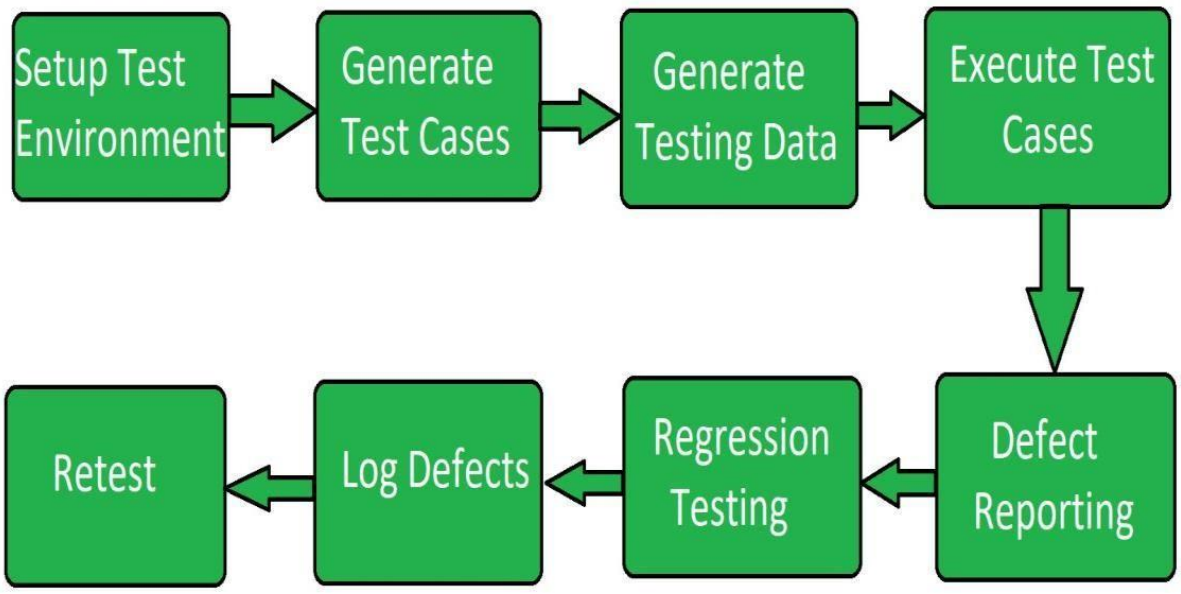

Fig.6.3: System Testing Process

6.3.1 Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

6.3.2 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.

6.3.3 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.

6.4 INTEGRATION TESTING:

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.

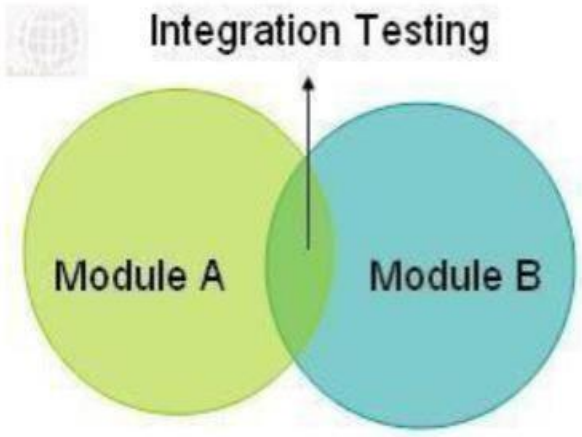

Fig:6.4: Integration Testing

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

6.5 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.

Acceptance testing for Data Synchronization:

- The Acknowledgements will be received by the Sender Node after the Packets are received
- by the Destination Node
- The Route add operation is done only when there is a Route request in need
- The Status of Nodes information is done automatically in the Cache Updating process

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

6.6 TEST CASES:

6.6.1 Test Case-1:

Service provider Home Page

Expected Output: Service provider Home Page is successful and Admin main menu is displayed

Actual Output: When Service provider username and Password is entered into login page. On Successful credentials, Service provider home page is displayed and main menu is displayed in Service provider home page

6.6.2 Test Case-2:

User Home Page

Expected Output: User Home Page is successful and User main menu is displayed

Actual Output: When User's username and Password is entered into login page. On Successful credentials, user home page is displayed and main menu is displayed in user home

CHAPTER-7

EXPERIMENTAL RESULTS

7.1 EXECUTION PROCEDURE

The Execution procedure is as follows:

1. In this research work with data with attributes are observable and then all of them are floating data. And there's a decision class/class variable. This data was collected from Kaggle machine learning repository.
2. In this research 70% data use for train model and 30% data use for testing purpose.
3. Naïve Bayes is used as Classifier.
4. In the classification report we were able to find out the desired result
5. In this analysis the result depends on some part of this research. However, which algorithm gives the best true positive, false positive, true negative, and false negative are the best algorithms in this analysis.

7.2 OUTPUT SCREENSHOTS

7.2.1 Execution Steps

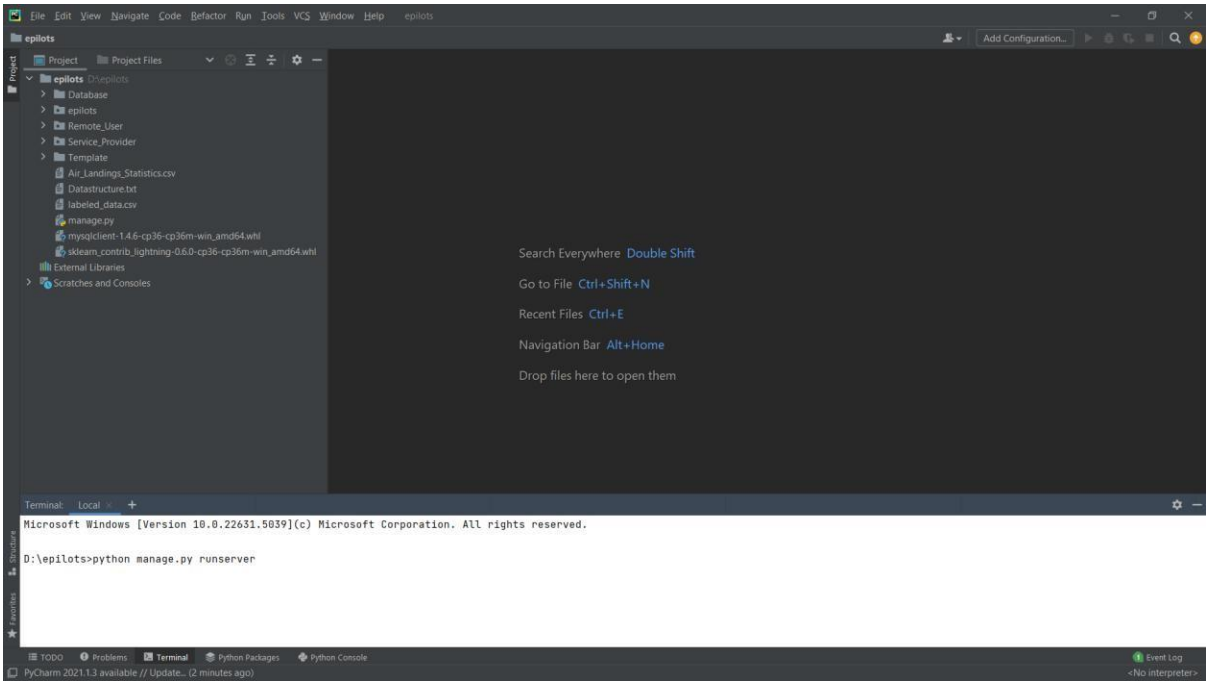

Fig 7.2.1 Running Program on console

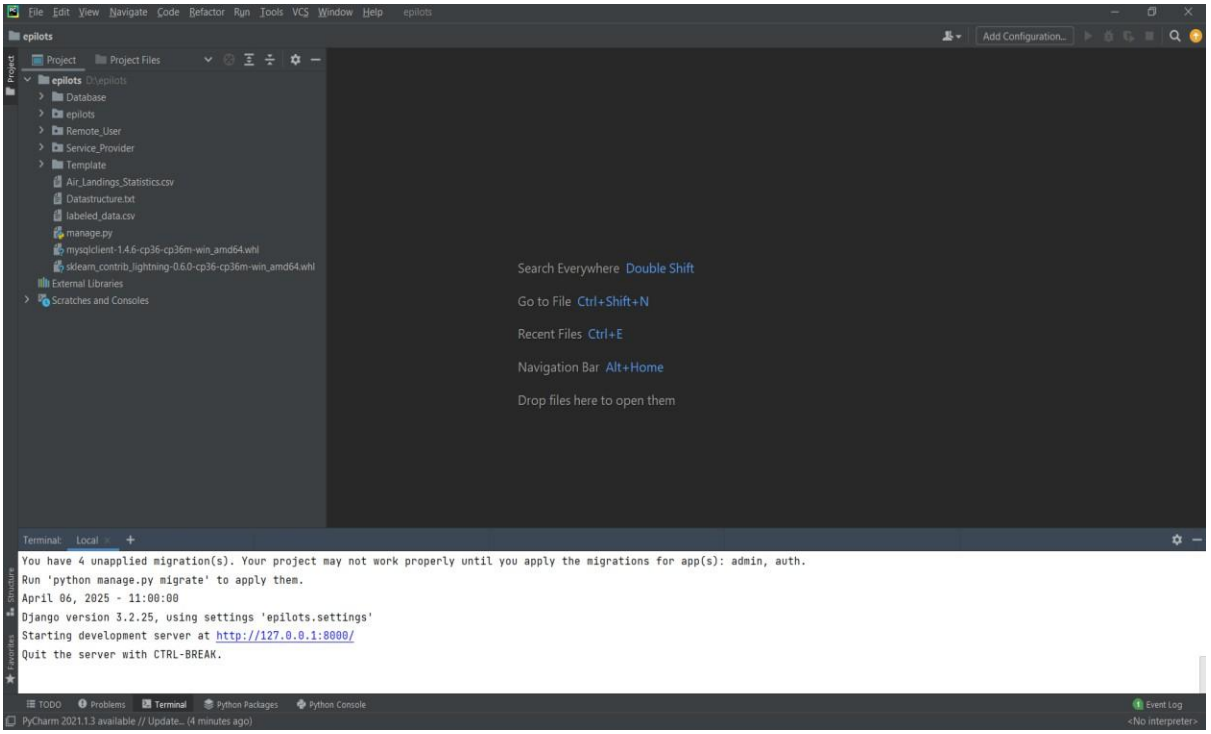

Fig7.2.2. Generating Link

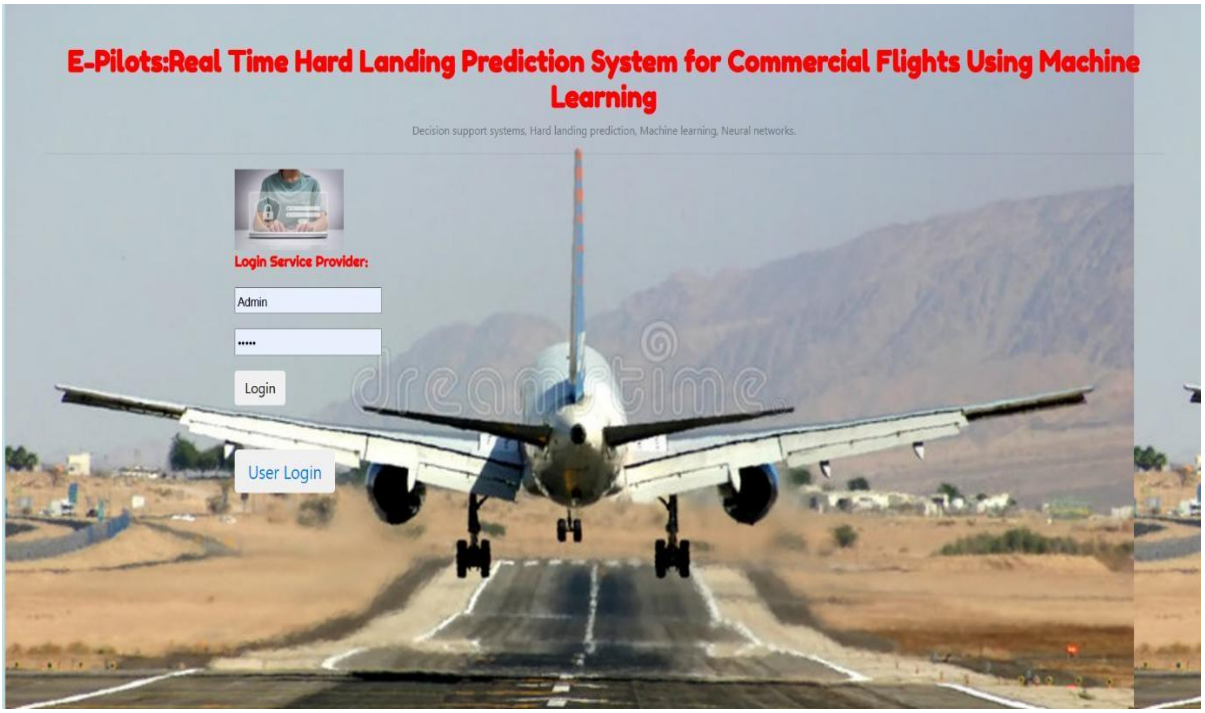

Fig 7.2.3. Admin Login

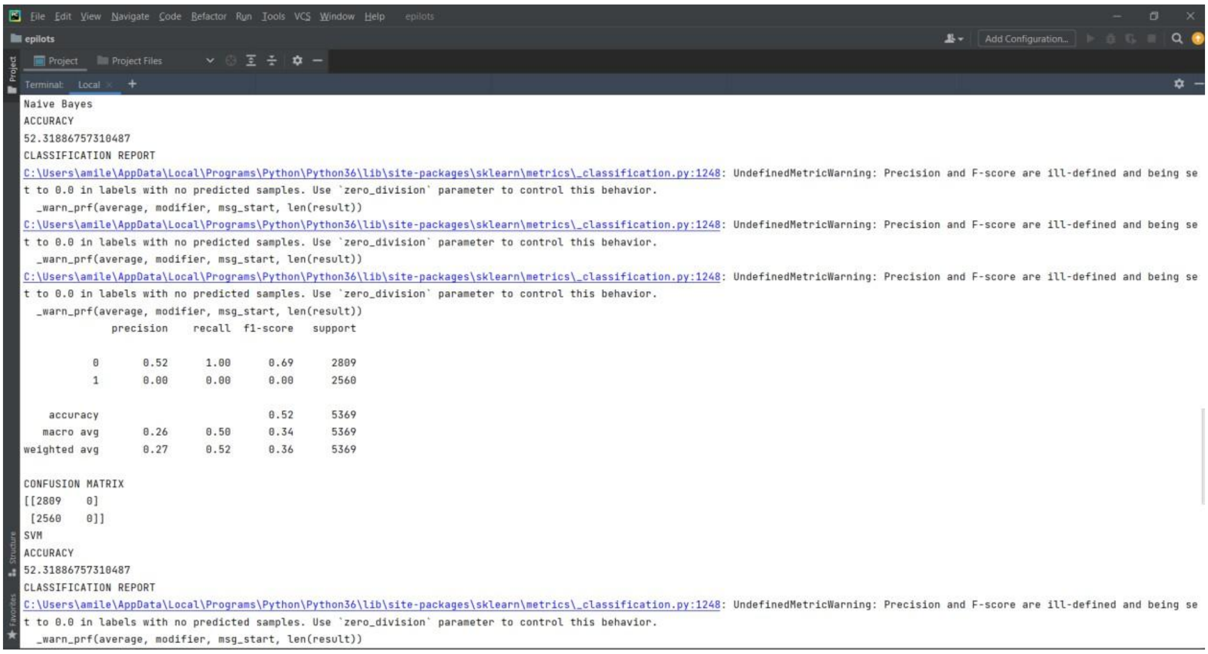

Fig 7.2.4 Training Dataset

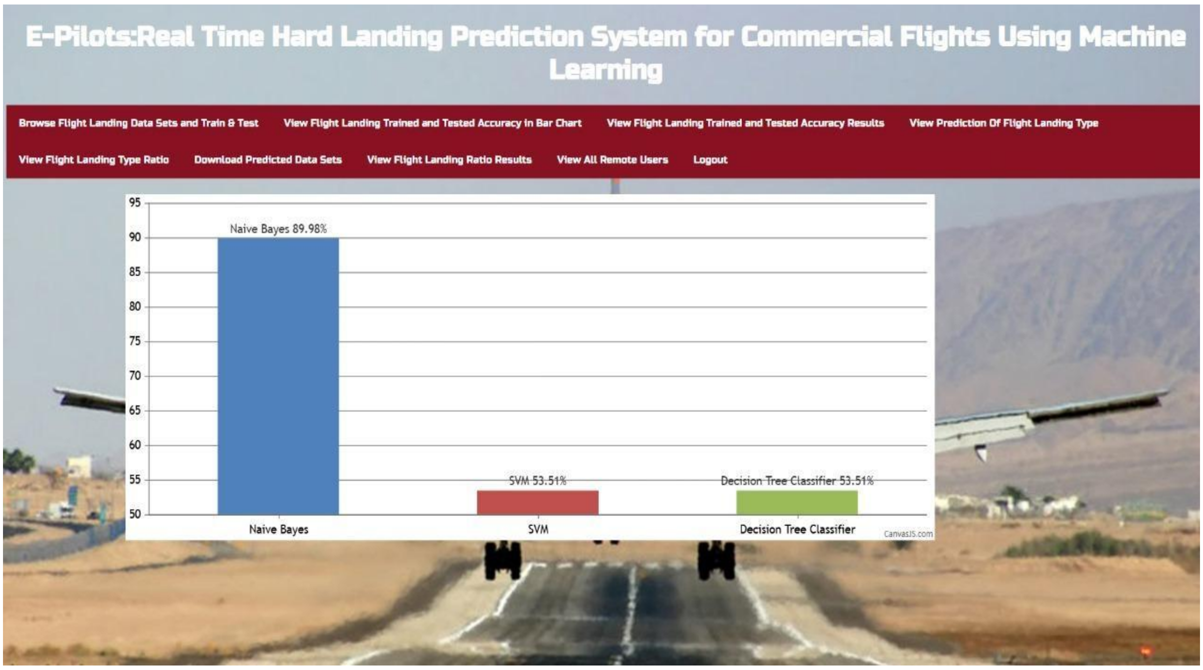

Fig 7.2.5 Trained and Tested Accuracy

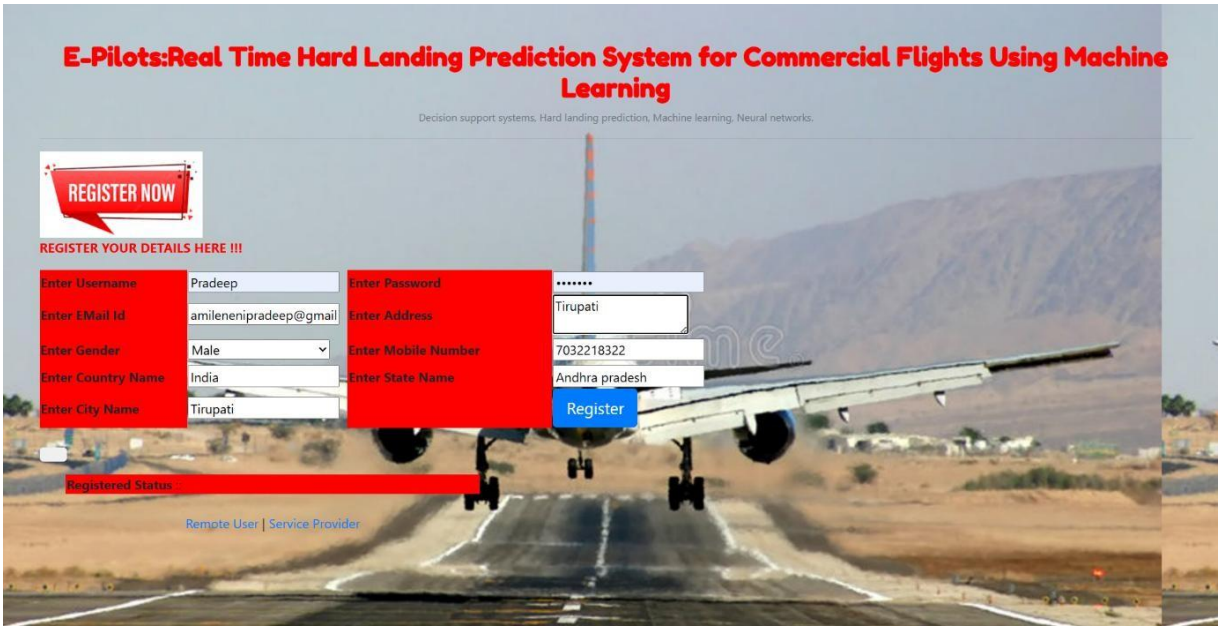

Fig.7.2.6 User Registration

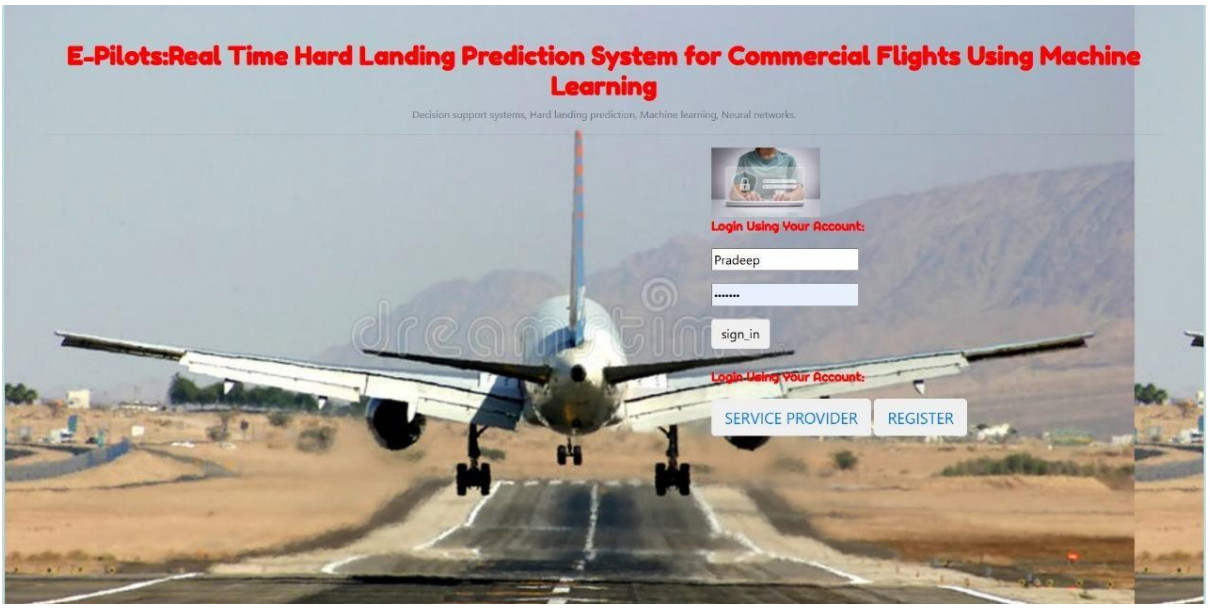

Fig.7.2 7. User Login

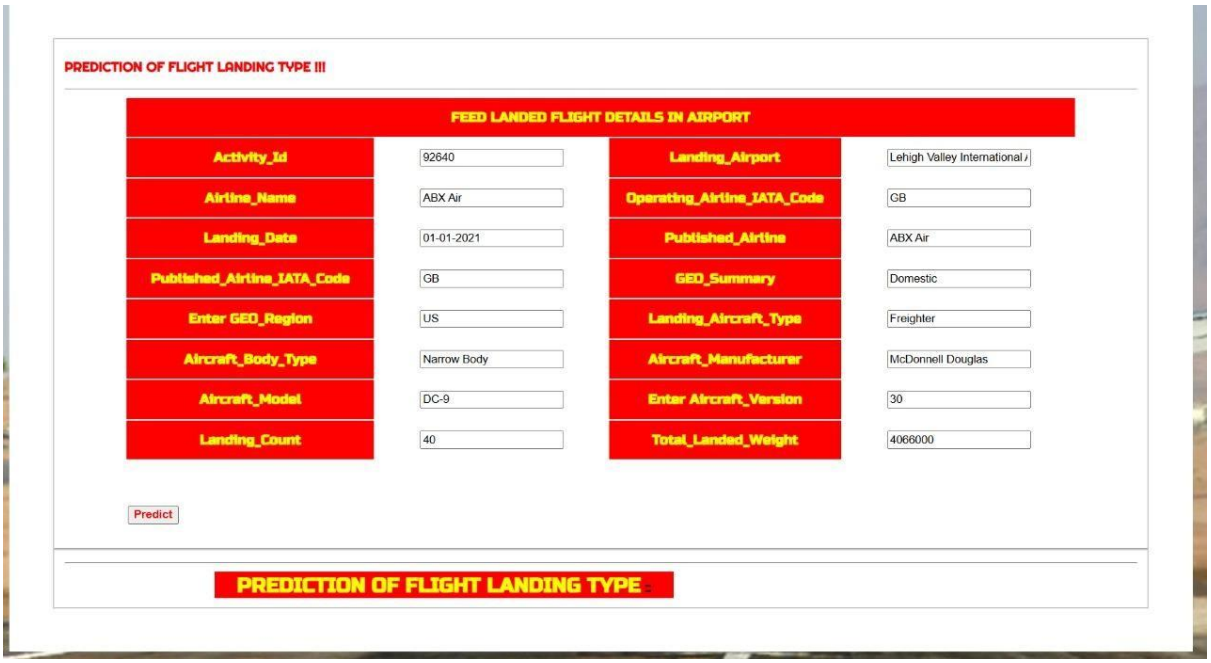

Fig.7.2 8. Enter Values for Prediction

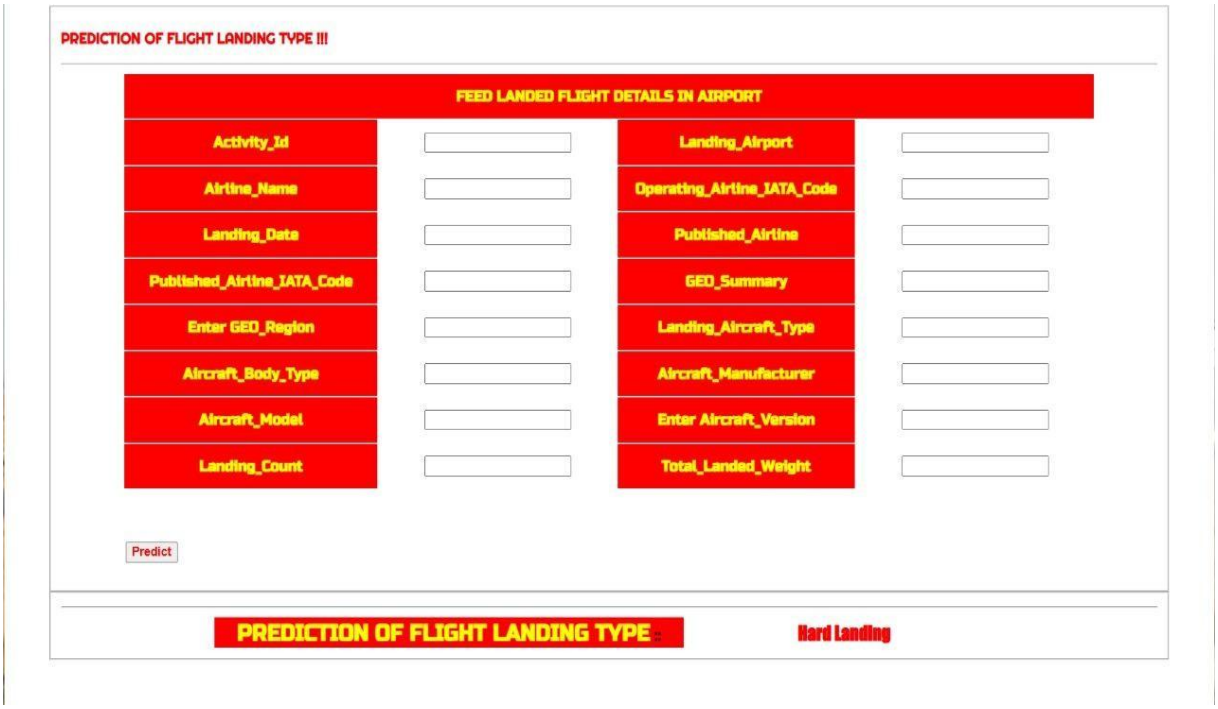

Fig.7.2 9. Prediction Result

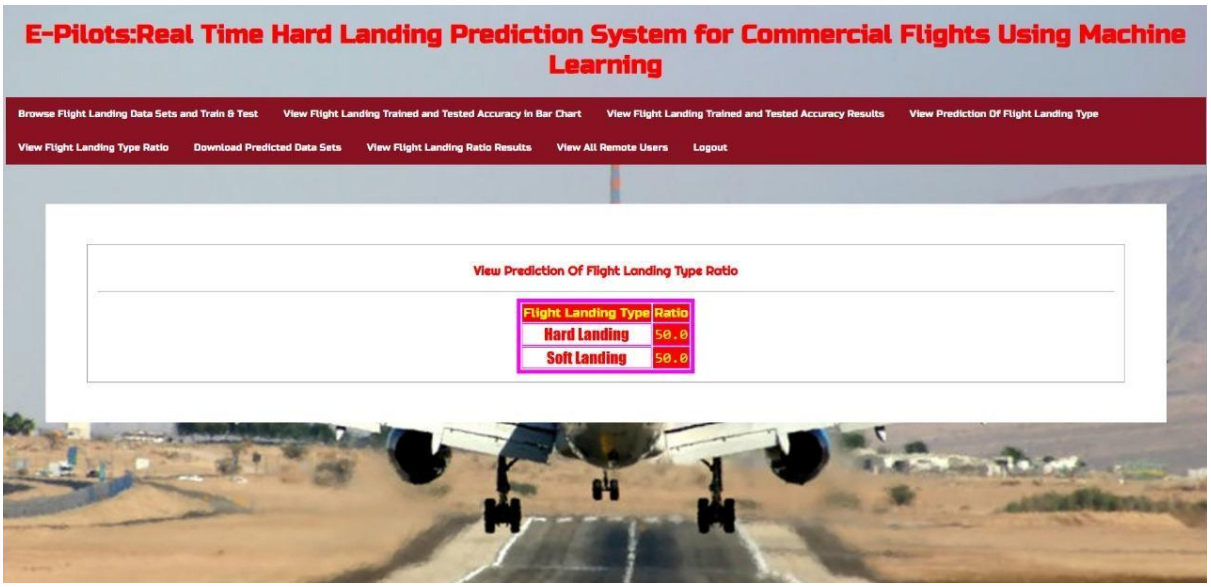

Fig.7.210. Flight Landing Prediction Type Ratio

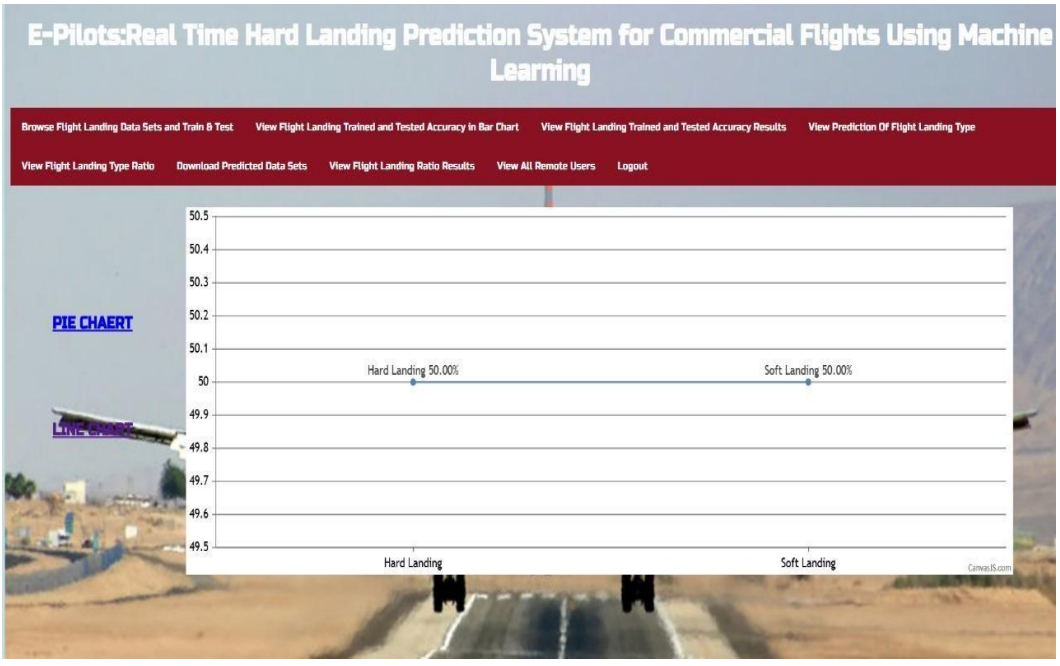

Fig 7.2.11 Graph of Soft Landing and Hard Landing

CHAPTER-8
CONCLUSION AND FUTURE SCOPE

8.1. CONCLUSION

This work presents a hybrid approach for hard landing prediction that uses features modeling temporal dependencies of aircraft variables as inputs to a neural network. Based on a large dataset of commercial flights, the results show that our approach has good sensitivity with specificity at the go-around point.

The system's integration with existing flight management and monitoring infrastructure ensures seamless operation without adding complexity to the pilot's workflow. Its adaptability to various aircraft types and continuous learning from updated data make it a scalable and robust solution for the aviation industry. In addition to improving safety, the system reduces costs associated with aircraft maintenance and repairs resulting from hard landings. It also fosters passenger confidence by prioritizing safety and adopting advanced technological measures.

8.2. FUTURE SCOPE

The advancements in aviation technology, artificial intelligence, and real-time data analytics continue to evolve. One of the key areas of future development is the integration of deep learning models, such as LSTMs and Transformers, to enhance predictive accuracy by capturing complex temporal dependencies in flight data. These models can help refine predictions by analyzing patterns from a vast dataset of previous landings, allowing airlines to proactively address potential risks.