A

Major Project

On

E-PILOTS : A SYSTEM TO PREDICT HARD LANDING DURING THE APPROACH PHASE OF COMMERCIAL FLIGHTS

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

B. VEDIKA (207R1A0569)

A. HARSHITHA NAIDU (207R1A0561)

B. ASHOK (207R1A0565)

Under the Guidance of

Dr. J. NARASIMHARAO

(Associate Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi) Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V), Medchal Road, Hyderabad-501401.

2020-2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled **"E-PILOTS : A SYSTEM TO PREDICT THE HARD LANDING DURING THE APPROACH PHASE OF COMMERCIAL FLIGHTS"** being submitted by **B. VEDIKA(207R1A0569), A. HARSHITHA NAIDU(207R1A0561) & B. ASHOK (207R1A0565)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2023-24.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Dr. J. Narasimharao
(Associate Professor)
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. K. Srujan Raju
HOD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on _____

ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **Dr. J. Narasimharao,** Associate Professor for his exemplary guidance, monitoring andconstant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC) **G. Vinesh Shanker, Dr. J. Narasimharao, Ms. Shilpa, & Dr. K. Maheswari** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

B. VEDIKA (207R1A0569)

A. HARSHITHA NAIDU (207R1A0561)

B. ASHOK (207R1A0565)

ABSTRACT

More than half of all commercial aircraft operation accidents could have been prevented by executing a go around. Making timely decision to execute a go-around manoeuvre can potentially reduce overall aviation industry accident rate. In this paper, we describe a cockpit-deployable machine learning system to support flight crew go around decision making based on the prediction of a hard landing event. The work presents a hybrid approach for hard landing prediction that uses features modelling temporal dependencies of aircraft variables as inputs to a neural network. Based on a large dataset of 58177 commercial flights, the results show that our approach has 85% of average sensitivity with 74% of average specificity at the go-around point. It follows that our approach is a cockpit-deployable recommendation system that outperforms existing approaches .

In general, although the overall occurrence of serious flight safety accidents in the aviation industry is very rare, the possibility of adverse events (e.g., flight exceedances) that may affect flight safety and further lead to severe accidents cannot be ignored. For example, it is not uncommon for the vertical acceleration to be too large at the touchdown moment, which is called the hard landing incident [4,5], and this incident may cause severe damage to the landing gears.

.

LIST OF FIGURES/TABLES

FIGURE NO	FIGURE NAME	PAGE NO
Figure 4.1	Project Architecture for E-Pilots : A system to predict hard landing during the approach phase of commercial flights	9
Figure 4.2	Use Case Diagram for E-Pilots : A system to predict hard landing during the approach phase of commercial flights.	11
Figure 4.3	Class Diagram for E-Pilots : A system to predict hard landing during the approach phase of commercial flights.	12
Figure 4.4	Sequence diagram for E-Pilots : A system to predict hard landing during the approach phase of commercial flights.	13
Figure 4.5	Activity diagram for E-Pilots : A system to predict hard landing during the approach phase of commercial flights.	14

LIST OF SCREENSHOTS

SCREENSHOT NO.	SCREENSHOT NAME	PAGE NO.
Screenshot 6.1	Upload Flight Landing Dataset	24
Screenshot 6.2	Select Folder	24
Screenshot 6.3	Dataset Loaded	25
Screenshot 6.4	Preprocess Dataset	25
Screenshot 6.5	Run SVM Algorithm	26
Screenshot 6.6	Run Logistic Regression Algorithm	26
Screenshot 6.7	Run AP2TD Algorithm	27
Screenshot 6.8	Run AP2DH Algorithm	27
Screenshot 6.9	Run DH2TD Algorithm	28
Screenshot 6.10	Comparision Graph	28
Screenshot 6.11	Sensitivity and Specificity for various algorithms	29

TABLE OFCONTENTS

ABSTRACT i
LIST OF FIGURES ii
LIST OF SCREENSHOTS iii
1. INTRODUCTION 1
1.1 PROJECT SCOPE 1
1.2 PROJECT PURPOSE 1
1.3 PROJECT FEATURES 1
2. LITERATURE SURVEY 2
3. SYSTEM ANALYSIS 4
3.1 PROBLEM DEFINITION 4
3.2 EXISTING SYSTEM 4
3.2.1 LIMITATIONS OF THE EXISTING SYSTEM 5
3.3 PROPOSED SYSTEM 5
3.3.1 ADVANTAGES OF PROPOSED SYSTEM 5
3.4 FEASIBILITY STUDY 6
3.4.1 ECONOMIC FEASIBILITY 6
3.4.2 TECHNICAL FEASIBILITY 6
3.4.3 SOCIAL FEASIBILITY 6
3.5 HARDWARE & SOFTWARE REQUIREMENTS 7
3.5.1 HARDWARE REQUIREMENTS 7
3.5.2 SOFTWARE REQUIREMENTS 8
4. ARCHITECTURE 9
4.1 PROJECT ARCHITECTURE 9
4.2 DESCRIPTION 10
4.3 USE CASE DIAGRAM 11
4.4 CLASS DIAGRAM 12
4.5 SEQUENCE DIAGRAM 13
4.6 ACTIVITY DIAGRAM 14
5. IMPLEMENTATION 15
5.1 SAMPLE CODE 13
6. SCREENSHOTS 24
7. TESTING 30
7.1 INTRODUCTION TO TESTING 30

TABLE OFCONTENTS

7.2 TYPES OF TESTING 30
7.2.1 UNIT TESTING 30
7.2.2 INTEGRATION TESTING 31
7.2.3 FUNCTIONAL TESTING 31
7.2.4 SYSTEM TEST 31
7.2.5 WHITE BOX TESTING 32
7.2.6 BLACK BOX TESTING 32
7.3 TEST STRATEGY AND APPROACH 32
7.4 TEST CASES 33
7.4.1 CLASSIFICATION 33
8. CONCLUSION & FUTURE SCOPE 34
8.1 PROJECT CONCLUSION 34
8.2 FUTURE SCOPE 34
9. BIBLIOGRAPHY & REFERENCES 35
9.1 REFERENCES 35
9.2 GITHUB LINK 36

1. INTRODUCTION

1. INTRODUCTION

1.1 PROJECT SCOPE

E-pilot system encompasses the development and implementation of a machine learning system aimed at enhancing aviation safety. The focus is on predicting hard landing events during the critical approach phase of commercial flights. The system is designed to be cockpit-deployable, providing real-time support to flight crews in decision-making. The approach involves a hybrid model utilizing features that capture temporal dependencies of various aircraft variables.

1.2 PROJECT PURPOSE

The Major objective is to develop and implement a machine learning system that serves as a predictive tool for flight crews. By utilizing a hybrid model incorporating features that capture temporal dependencies of various aircraft variables, the system aims to provide accurate predictions. The ultimate goal is to empower flight crews with timely information, enabling them to make informed decisions, such as executing a go-around maneuver when the risk of a hard landing is predicted.

1.3 PROJECT FEATURES

The main features of this project revolve around the development and implementation of a sophisticated machine learning system within the aircraft cockpit. At its core, the system employs a hybrid approach, integrating features that model the temporal dependencies of various critical aircraft variables. The use of neural networks enhances predictive capabilities, allowing the system to forecast the likelihood of a hard landing event. Ultimately, the project's main features lie in its innovative use of machine learning technology to proactively predict and mitigate the risk of hard landings, contributing significantly to aviation safety during the crucial approach phase of commercial flights.

2. LITERATURE SURVEY

1. LITERATURE SURVEY

Although much study has been done on the prediction of fly landing mishaps and other unsafely conditions the prediction of hard landing accidents has received less attention. Furthermore, the majority of recent work focuses on the prediction of HL for unmanned aerial vehicles (UAV), which have fundamentally different dynamical properties and flying rules than commercial aircraft. A Hard Landing (HL) is a condition in which the aeroplane makes an excessive contact on the ground while landing. Because this impact is directly tied to vertical (or normal) acceleration, HL may be defined as flights in which the vertical acceleration exceeds the aircraft type's restricted value during the landing phase.

Classifiers are divided into two types: machine learning and deep learning. Machine learning approaches use a classifier to analyse UAV flight data captured with the Quick Access Recorder (QAR) and sampled at a discrete set of heights that constitute the feature space. The values of variables characterising aircraft dynamics measured between 9 and 2 metres before TD are used in most techniques Others A hybrid model with a net architecture that has been optimised. We present a hybrid technique that combines features characterising the temporal interdependence of aircraft data as input to an optimised neural network. To prevent bias induced by a lack of convergence of sophisticated models (such as LSTM), we employ a conventional network and characterise potential temporal dependencies associated with unstable methods as the variability of various types of aircraft characteristics at a set of altitudes. The concatenation of such variability for variables classified into four major groups (physical, actuator, pilot operations, and all of them) is used as an input feature by various designs to identify the best subset.

Extensive comparison to SoA in a big commercial flight database. Our models have been evaluated and compared to SoA techniques on a huge database of Flight Management Systems (FMS) recorded data of an airline that is no longer in service, which comprises three distinct aircraft models (A319, A320, and A321). The results reveal that when all variable types are examined, the best classification network obtains an average recall of HL events of 85% with a specificity of 75%, outperforming

existing LSTM approaches found in the literature.

Assessment of classifiers and repressor. We studied the efficacy of regression and classification models in terms of flight height and numerous aircraft characteristics, including the effects of automation and pilot movements, with the ultimate objective of establishing a cockpit deployable recommendation system. The results of our huge dataset from commercial flights demonstrate that, while the regression networks perform similarly to SoA approaches (with MSE of 103 in estimations at TD), the accuracy for identifying HL is relatively poor (46% sensitivity). This suggests that regression models might not be the best choice for detecting HL events in a field deployable support system. Error sources and the ability to offer a workaround. Unlike earlier techniques, we investigate the ability of networks to identify HL at the decision elevation as well as the impact of the operational context. We also investigated the sources of mistakes, such as the optimum variable type, the ideal altitude range for predictions, aircraft type biases, and the competence of regression coefficients for HL prediction.

3. SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. The system analyst plays an important role of an interrogator and dwells deep into the working of the present system. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, “what must be done to solve the problem?” The system is viewed as a whole and the inputs to the system are identified. Once analysis is completed the analyst has a firm understanding of what is to be done.

3.1 PROBLEM DEFINITION

Given the challenge of potential hard landings during the approach phase of commercial flights, the project seeks to develop a predictive system that, based on a dataset of critical aircraft variables, identifies the need for a go-around maneuver. This addresses the current absence of real-time tools within the cockpit to proactively predict hard landings, aiming to enhance flight crew decision-making and ultimately improve aviation safety.

3.2 EXISTING SYSTEM

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. Apply a classifier to UAV flight data recorded using the Quick Access Recorder (QAR) sampled at a discrete set of heights that define the feature space. Deep learning approaches are mainly based on Long Short-Term Memory Recurrent Neural Network (LSTM) architectures.

3.2.1 LIMITATIONS OF EXISTINGSYSTEM

Following are the disadvantages of existing system:

- In existing system there is no implementation for source of errors and capability for go-around recommendation .
- There is no implementation using hybrid approach for hardlanding prediction that uses features modeling temporal dependencies of aircraft variables as inputs to a neural network.

3.3 PROPOSED SYSTEM

This paper presents an analysis of approaches for early prediction of hard-landing events in commercial flights using Hybrid model with optimized net architecture, Exhaustive comparison to SoA in a large database of commercial flights, Analysis of the performance of classifiers and regressors and Sources of errors and capability for go-around recommendation.

3.3.1 ADVANTAGES OF THE PROPOSEDSYSTEM

- 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.

3.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and a business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. Three key considerations involved in the feasibility analysis:

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

3.4.1 ECONOMIC FEASIBILITY

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on a project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also all the resources are already available, it give an indication that the system is economically possible for development.

3.4.2 TECHNICAL FEASIBILITY

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

3.4.3 SOCIAL FEASIBILITY

This includes the following questions:

- Is there sufficient support for the users?
- Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible

3.5 HARDWARE & SOFTWARE REQUIREMENTS

3.5.1 HARDWARE REQUIREMENTS:

Hardware interfaces specify the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

- Processor : Pentium IV
- Hard disk : 20 GB
- RAM : 4 GB(min)
- Input devices : Keyboard, mouse.
- Monitor : SVGA

3.5.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements,

- Operating system : Windows 7 Ultimate
- Languages : Python
- Front-End : Python
- Bach-End : Django-ORM
- Designing : HTML, CSS, JavaScript

4. ARCHITECTURE

4. ARCHITECTURE

4.1 PROJECT ARCHITECTURE

This project architecture shows the procedure followed for classification, starting from input to final prediction.

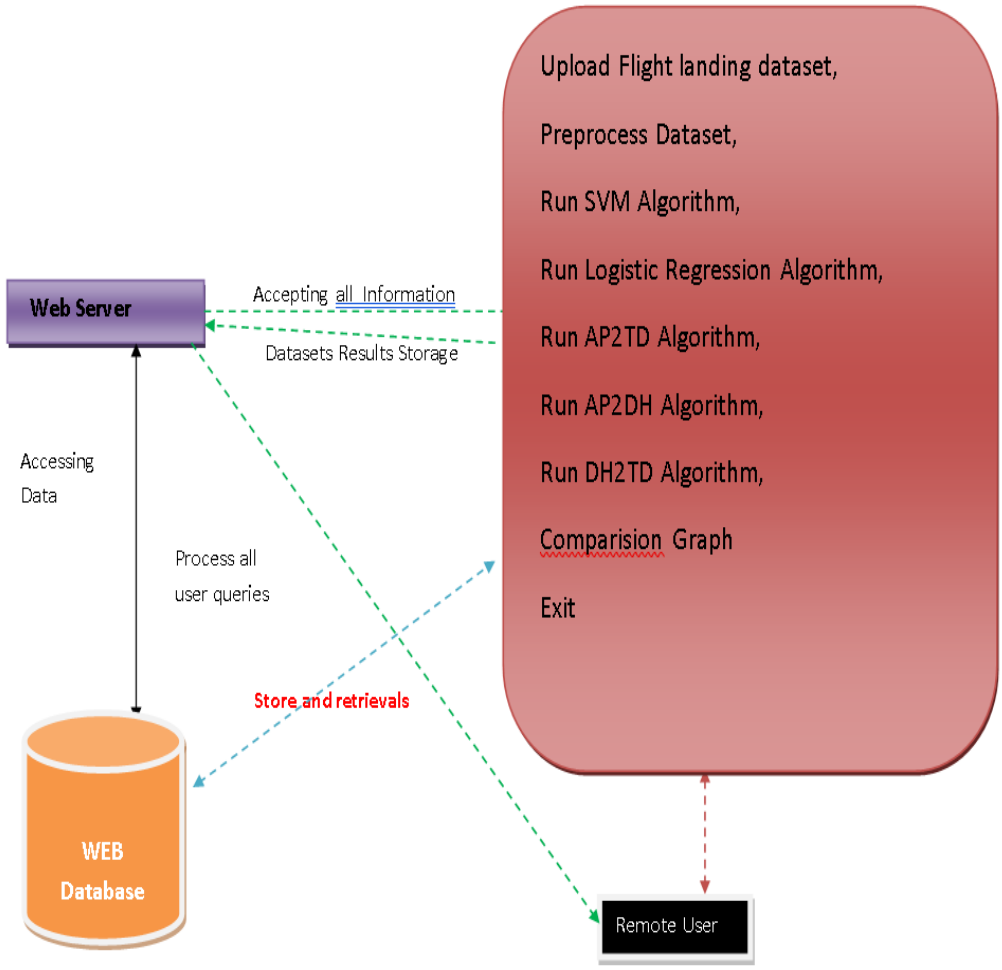


Figure 4.1: Project Architecture for E-Pilots: A system to predict hard landing during the approach phase of commercial flights

4.2 DESCRIPTION

Upload Flight Landing Dataset: Using this module we will upload dataset folder with 3 files and then application read all 3 files and then find and plot graph with number of HARD and NOT Hard Landing graph

Preprocess Dataset: Using this module we will normalize and shuffle dataset and then split dataset into train and test where application used 80% dataset for training and 20% for testing.

Run SVM Algorithm: Using this module we will train SVM with all features using 80% dataset and then perform prediction on 20% test data and then calculate SVM sensitivity and specificity score and then plot graph. Graph closer to 1 will reflect good performance of the algorithm

Run Logistic Regression Algorithm: Using this module we will train SVM with all features using 80% dataset and then perform prediction on 20% test data and then calculate SVM sensitivity and specificity score and then plot graph. Graph closer to 1 will reflect good performance of the algorithm

Run AP2TD Algorithm: This module train LSTM on PHYSICAL features and then perform prediction on test data and calculate sensitivity and specificity

Run AP2DH Algorithm: This module train LSTM on ACTUATOR features and then perform prediction on test data and calculate sensitivity and specificity

Run DH2TD Algorithm: This module train LSTM on PILOT features and then perform prediction on test data and calculate sensitivity and specificity. This module merge all modules to get HYBRID LSTM sensitivity and specificity values

Comparison Graph: Using this module we will plot sensitivity and specificity graph.

4.3 USE CASE DIAGRAM

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of usersthe system has. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

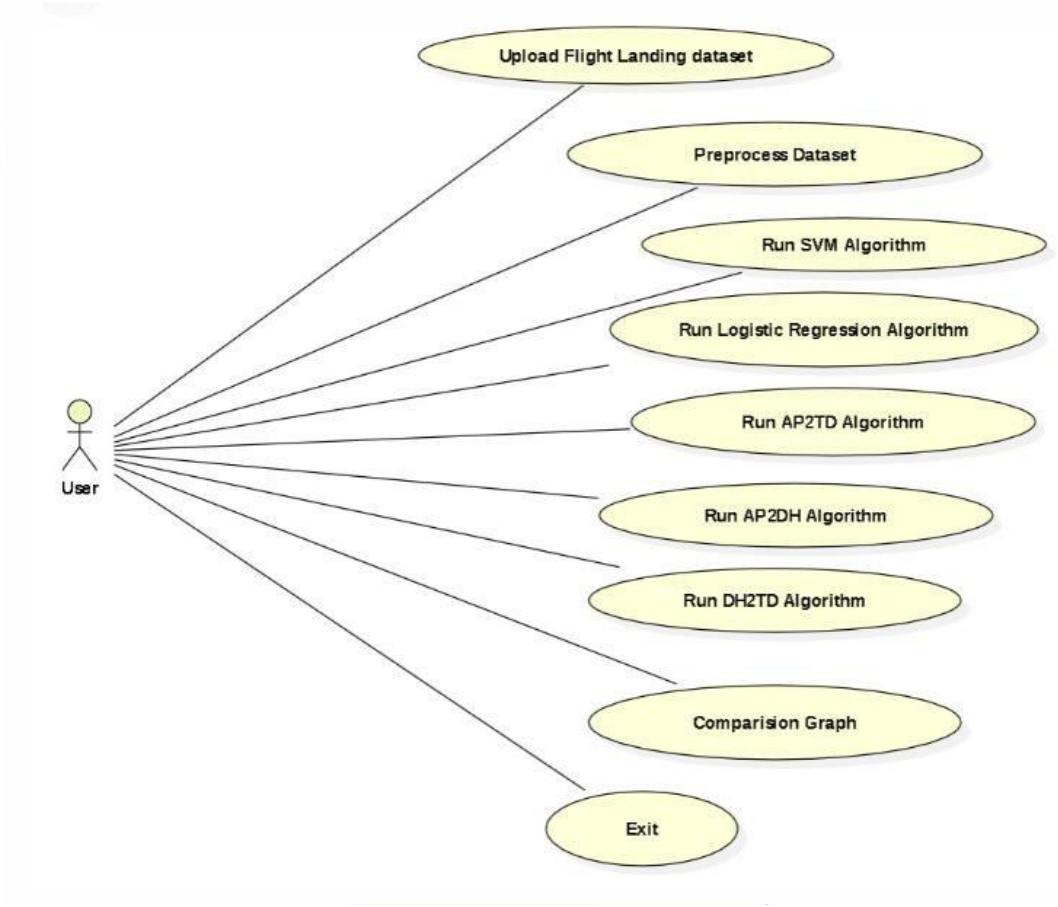


Figure 4.2: Use Case Diagram for E-Pilots: A system to predict hard landing during the approach phase of commercial flights

4.4 CLASS DIAGRAM

Class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations(or methods), and the relationships among objects.

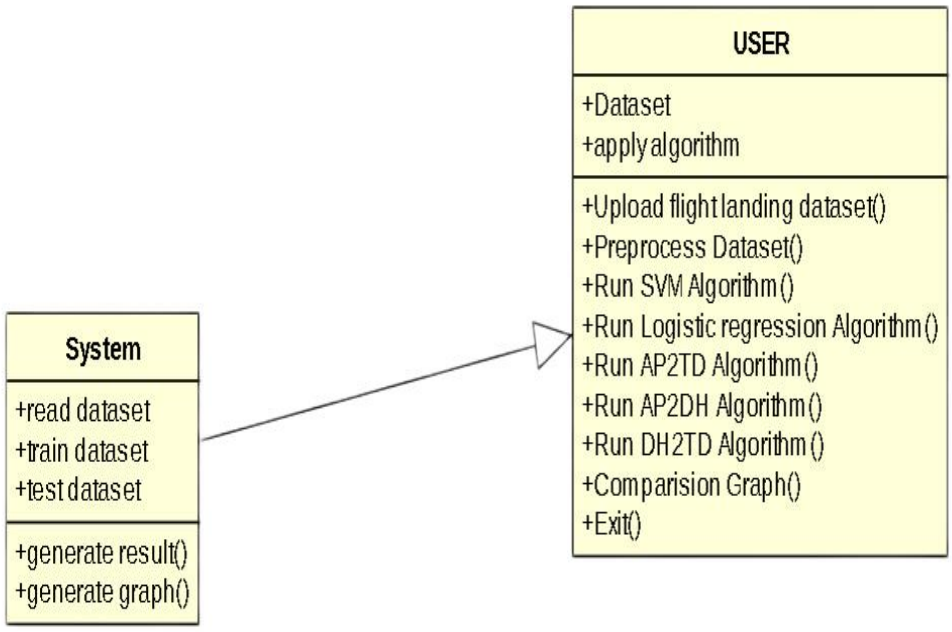


Figure 4.3: Class Diagram for E-Pilots: A system to predict hard landing during the approach phase of commercial flights

4.5 SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development.

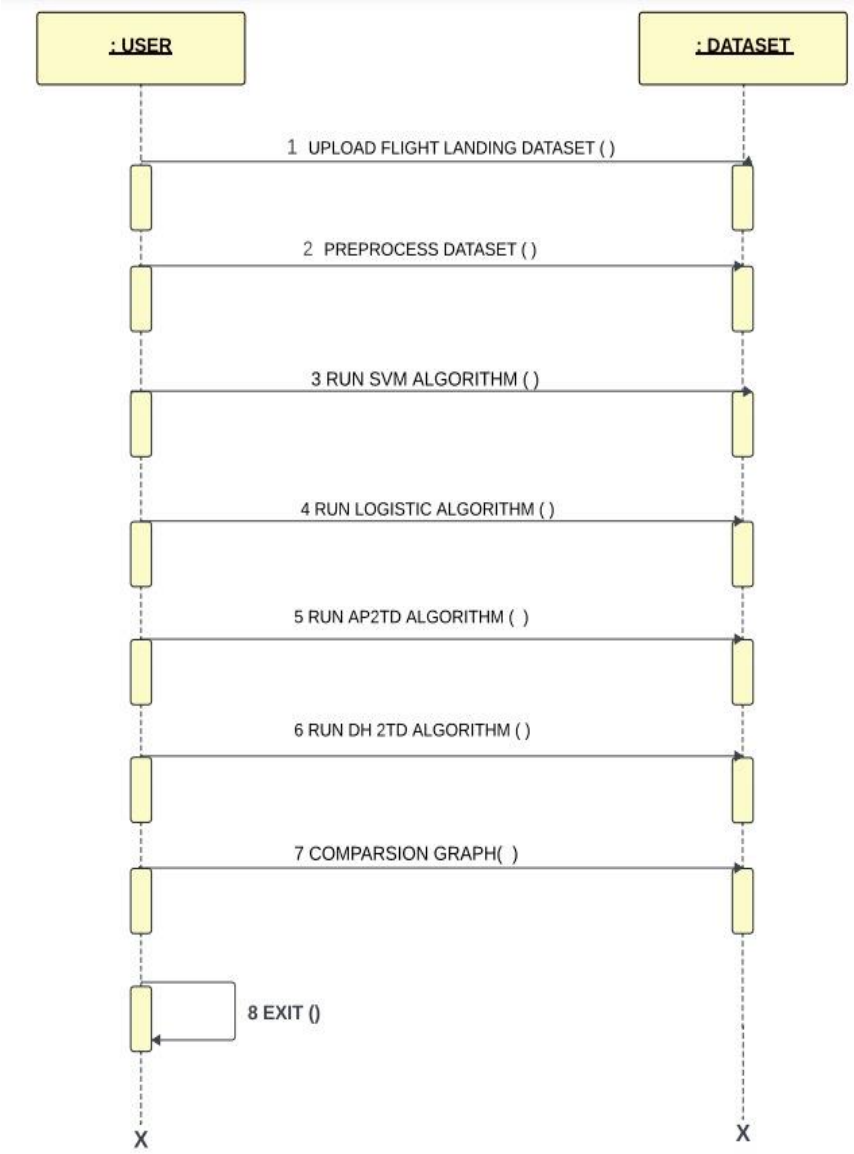


Figure 4.4: Sequence Diagram for E-Pilots: A system to predict hard landing during the approach phase of commercial flights

4.6 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. They can also include elements showing the flow of data between activities through one or more data stores.

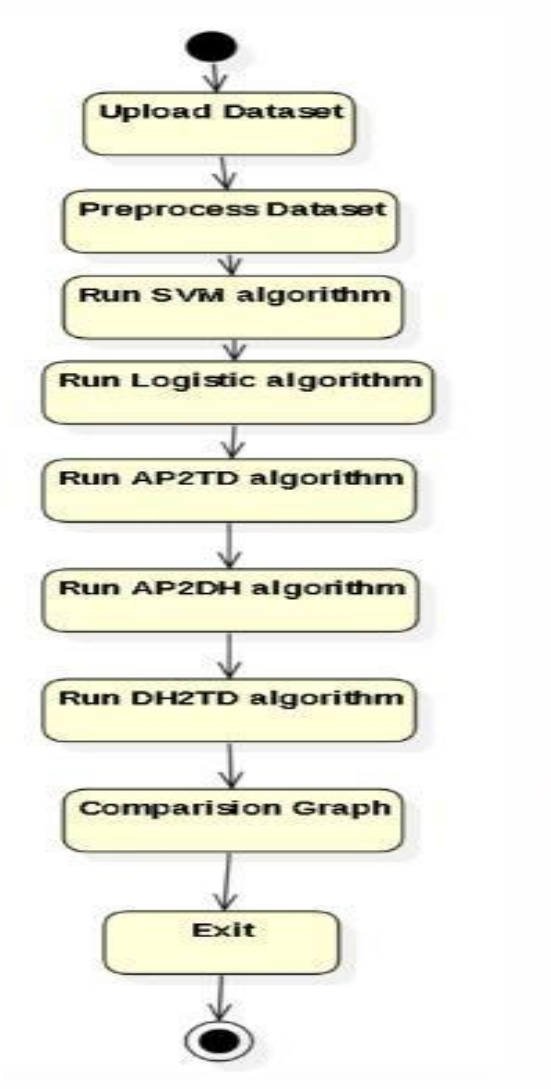


Figure 4.5: Activity Diagram for E-Pilots: A system to predict hard landing during the approach phase of commercial flights

5. IMPLEMENTATION

5.1 SAMPLE CODE

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
import matplotlib.pyplot as plt
from tkinter.filedialog import askopenfilename
from sklearn.metrics import recall_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn import svm
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import LSTM
from keras.utils.np_utils import to_categorical
from keras.models import model_from_json
import os
from sklearn.metrics import confusion_matrix

main = Tk()
main.title("E-Pilots: A System to Predict Hard Landing During the Approach Phase of
Commercial Flights")
main.geometry("1300x1200")


global filename
global dataset
global Y, all_data
global pilot_X_train, pilot_X_test, pilot_y_train, pilot_y_test
global actuator_X_train, actuator_X_test, actuator_y_train, actuator_y_test
global physical_X_train, physical_X_test, physical_y_train, physical_y_test
global all_X_train, all_X_test, all_y_train, all_y_test
global sensitivity, specificity
global pilot, actuator, physical
```

```
def uploadDataset():
    global pilot, actuator, physical, Y, all_data
    text.delete('1.0', END)
    filename = filedialog.askdirectory(initialdir = ".")
    pilot = pd.read_csv("Dataset/Pilot.csv")
    actuator = pd.read_csv("Dataset/Actuators.csv")
    physical = pd.read_csv("Dataset/Physical.csv")
    Y = physical['label'].values
    pilot.drop(['label'], axis = 1,inplace=True) #read pilot, actuator and physical dataset
    actuator.drop(['label'], axis = 1,inplace=True)
    physical.drop(['label'], axis = 1,inplace=True)
    all_data = [physical, actuator, pilot]
    all_data = pd.concat(all_data, axis=1)
    text.insert(END,"Pilot Dataset \n\n")
    text.insert(END,str(pilot.head())+"\n\n")

    text.insert(END,"Actuator Dataset \n\n")
    text.insert(END,str(actuator.head())+"\n\n")

    text.insert(END,"Physical Dataset \n\n")
    text.insert(END,str(physical.head())+"\n\n")
    text.update_idletasks()
    labels, count = np.unique(Y, return_counts = True)

    height = count
    bars = ('Not Hard Landing','Hard Landing')
    y_pos = np.arange(len(bars))
    plt.bar(y_pos, height)
    plt.xticks(y_pos, bars)
    plt.xlabel("Landing Type")
    plt.ylabel("Counts")
    plt.title("Different Landing Graphs in Dataset")
    plt.show()

def preprocessDataset():
    text.delete('1.0', END)
    global pilot_X_train, pilot_X_test, pilot_y_train, pilot_y_test
    global actuator_X_train, actuator_X_test, actuator_y_train, actuator_y_test
    global physical_X_train, physical_X_test, physical_y_train, physical_y_test
    global all_X_train, all_X_test, all_y_train, all_y_test
    global pilot, actuator, physical, Y, all_data
    #converting dataset into numpy array
    pilot = pilot.values
    actuator = actuator.values
```

```
    physical = physical.values
    all_data =  all_data.values
    #shuffling the dataset
    indices = np.arange(all_data.shape[0])
    np.random.shuffle(indices)
    all_data = all_data[indices]
    Y = Y[indices]
    pilot = pilot[indices]
    actuator = actuator[indices]
    physical = physical[indices]
    #normalizing dataset values
    scaler1 = StandardScaler()
    all_data = scaler1.fit_transform(all_data)
    scaler2 = StandardScaler()
    pilot = scaler2.fit_transform(pilot)
    scaler3 = StandardScaler()
    actuator = scaler3.fit_transform(actuator)
    scaler4 = StandardScaler()
    physical = scaler4.fit_transform(physical)
    #dataset reshape to multi dimensional array
    pilot = np.reshape(pilot, (pilot.shape[0], pilot.shape[1], 1))
    actuator = np.reshape(actuator, (actuator.shape[0], actuator.shape[1], 1))
    physical = np.reshape(physical, (physical.shape[0], physical.shape[1], 1))
    #splitting dataset into train and test
    all_X_train, all_X_test, all_y_train, all_y_test = train_test_split(all_data, Y, test_size =
0.2)
    Y = to_categorical(Y)
    pilot_X_train, pilot_X_test, pilot_y_train, pilot_y_test = train_test_split(pilot, Y,
test_size = 0.2)
    actuator_X_train, actuator_X_test, actuator_y_train, actuator_y_test =
train_test_split(actuator, Y, test_size = 0.2)
    physical_X_train, physical_X_test, physical_y_train, physical_y_test =
train_test_split(physical, Y, test_size = 0.2)

    text.insert(END,"Dataset Features Processing & Normalization Completed\n\n")
    text.insert(END,"Total records found in dataset            : "+str(all_data.shape[0])+"\n")
    text.insert(END,"All features found in dataset             : "+str(all_data.shape[1])+"\n")
    text.insert(END,"Total Pilot features found in dataset     : "+str(pilot.shape[1])+"\n")
    text.insert(END,"Total Actuator features found in dataset :
"+str(actuator.shape[1])+"\n")
    text.insert(END,"Total Physical features found in dataset :
"+str(physical.shape[1])+"\n\n")
    text.insert(END,"Dataset Train and Test Split\n\n")
    text.insert(END,"80% dataset records used to train ALL algorithms :
"+str(all_X_train.shape[0])+"\n")
```

```
        text.insert(END,"20% dataset records used to train ALL algorithms :
    "+str(all_X_test.shape[0])+"\n")


def calculateMetrics(algorithm, y_test, predict):
        cm = confusion_matrix(y_test, predict)
        total = sum(sum(cm))
        se = cm[0,0]/(cm[0,0]+cm[0,1])
        sp = cm[1,1]/(cm[1,0]+cm[1,1])
        se = accuracy_score(y_test, predict)
        sp = recall_score(y_test, predict)
        if sp == 0:
            sp = accuracy_score(y_test, predict)
        text.insert(END,algorithm+' Sensitivity : '+str(se)+"\n")
        text.insert(END,algorithm+' Specificity : '+str(sp)+"\n\n")
        text.update_idletasks()
        sensitivity.append(se)
        specificity.append(sp)
        if algorithm == 'DH2TD Pilot Features':
            se = sensitivity[2] + sensitivity[3] + sensitivity[4]
            sp = specificity[2] + specificity[3] + specificity[4]
            se = se / 3
            sp = sp / 3
            text.insert(END,'Hybrid LSTM Sensitivity : '+str(se)+"\n")
            text.insert(END,'Hybrid LSTM Specificity : '+str(sp)+"\n\n")
            text.update_idletasks()

        values = []
        values.append([se - 0.10, se])
        values.append([sp - 0.10, sp])

        data = pd.DataFrame(values, columns=['Sensitivity', 'Specificity'])
        data.plot(kind = 'box')
        plt.xticks(rotation=90)
        plt.title(algorithm+" Sensitivity & Specificity Graph")
        plt.show()

    def runSVM():
        text.delete('1.0', END)
        global sensitivity, specificity
        global all_X_train, all_X_test, all_y_train, all_y_test
        sensitivity = []
        specificity = []
        svm_cls = svm.SVC(kernel='poly', gamma='auto', C=0.1)
```

```
    svm_cls.fit(all_X_train, all_y_train)
    predict = svm_cls.predict(all_X_test)
    calculateMetrics("SVM", all_y_test, predict)


def runLR():
    lr_cls = LogisticRegression(max_iter=1,tol=300)
    lr_cls.fit(all_X_train, all_y_train)
    predict = lr_cls.predict(all_X_test)
    calculateMetrics("Logistic Regression", all_y_test, predict)

def runAP2TD():
    global physical_X_train, physical_X_test, physical_y_train, physical_y_test
    if os.path.exists('model/physical_model.json'):
        with open('model/physical_model.json', "r") as json_file:
            loaded_model_json = json_file.read()
            lstm_physical = model_from_json(loaded_model_json)
        json_file.close()
        lstm_physical.load_weights("model/physical_weights.h5")
        lstm_physical._make_predict_function()
    else:
        lstm_physical = Sequential()#defining deep learning sequential object
        #adding LSTM layer with 100 filters to filter given input X train data to select
relevant features
        lstm_physical.add(LSTM(100,input_shape=(physical_X_train.shape[1],
physical_X_train.shape[2])))
        #adding dropout layer to remove irrelevant features
        lstm_physical.add(Dropout(0.5))
        #adding another layer
        lstm_physical.add(Dense(100, activation='relu'))
        #defining output layer for prediction
        lstm_physical.add(Dense(physical_y_train.shape[1], activation='softmax'))
        #compile LSTM model
        lstm_physical.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
        #start training model on train data and perform validation on test data
        hist = lstm_physical.fit(physical_X_train, physical_y_train, epochs=20,
batch_size=16, validation_data=(physical_X_test, physical_y_test))
        #save model weight for future used
        lstm_physical.save_weights('model/physical_weights.h5')
        model_json = lstm_physical.to_json()
        with open("model/physical_model.json", "w") as json_file:
            json_file.write(model_json)
```

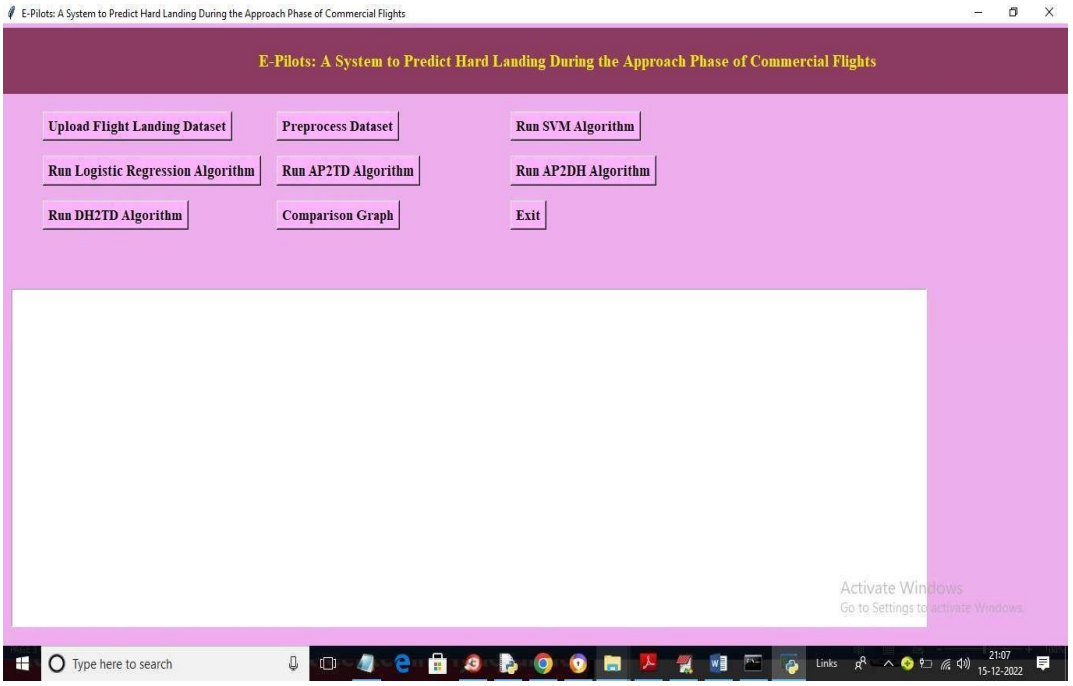
```
    json_file.close()
  print(lstm_physical.summary())
  #perform prediction on test data
  predict = lstm_physical.predict(physical_X_test)
  predict = np.argmax(predict, axis=1)
  testY = np.argmax(physical_y_test, axis=1)
  calculateMetrics("AP2TD Physical Features", testY, predict)

def runAP2DH():
  global actuator_X_train, actuator_X_test, actuator_y_train, actuator_y_test
  if os.path.exists('model/actuator_model.json'):
    with open('model/actuator_model.json', "r") as json_file:
      loaded_model_json = json_file.read()
      lstm_actuator = model_from_json(loaded_model_json)
    json_file.close()
    lstm_actuator.load_weights("model/actuator_weights.h5")
    lstm_actuator._make_predict_function()
  else:
    lstm_actuator = Sequential()#defining deep learning sequential object
    #adding LSTM layer with 100 filters to filter given input X train data to select
relevant features
    lstm_actuator.add(LSTM(100,input_shape=(actuator_X_train.shape[1],
actuator_X_train.shape[2])))
    #adding dropout layer to remove irrelevant features
    lstm_actuator.add(Dropout(0.5))
    #adding another layer
    lstm_actuator.add(Dense(100, activation='relu'))
    #defining output layer for prediction
    lstm_actuator.add(Dense(actuator_y_train.shape[1], activation='softmax'))
    #compile LSTM model
    lstm_actuator.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
    #start training model on train data and perform validation on test data
    hist = lstm_actuator.fit(actuator_X_train, actuator_y_train, epochs=20,
batch_size=16, validation_data=(actuator_X_test, actuator_y_test))
    #save model weight for future used
    lstm_actuator.save_weights('model/actuator_weights.h5')
    model_json = lstm_actuator.to_json()
    with open("model/actuator_model.json", "w") as json_file:
      json_file.write(model_json)
    json_file.close()
  print(lstm_actuator.summary())
  #perform prediction on test data
  predict = lstm_actuator.predict(actuator_X_test)
```

```
    predict = np.argmax(predict, axis=1)
    testY = np.argmax(actuator_y_test, axis=1)
    calculateMetrics("AP2DH Actuator Features", testY, predict)


def runDH2TD():
    global pilot_X_train, pilot_X_test, pilot_y_train, pilot_y_test
    if os.path.exists('model/pilot_model.json'):
        with open('model/pilot_model.json', "r") as json_file:
            loaded_model_json = json_file.read()
            lstm_pilot = model_from_json(loaded_model_json)
        json_file.close()
        lstm_pilot.load_weights("model/pilot_weights.h5")
        lstm_pilot._make_predict_function()
    else:
        lstm_pilot = Sequential()#defining deep learning sequential object
        #adding LSTM layer with 100 filters to filter given input X train data to select
relevant features
        lstm_pilot.add(LSTM(100,input_shape=(pilot_X_train.shape[1],
pilot_X_train.shape[2])))
        #adding dropout layer to remove irrelevant features
        lstm_pilot.add(Dropout(0.5))
        #adding another layer
        lstm_pilot.add(Dense(100, activation='relu'))
        #defining output layer for prediction
        lstm_pilot.add(Dense(pilot_y_train.shape[1], activation='softmax'))
        #compile LSTM model
        lstm_pilot.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
        #start training model on train data and perform validation on test data
        hist = lstm_pilot.fit(pilot_X_train, pilot_y_train, epochs=20, batch_size=16,
validation_data=(pilot_X_test, pilot_y_test))
        #save model weight for future used
        lstm_pilot.save_weights('model/pilot_weights.h5')
        model_json = lstm_pilot.to_json()
        with open("model/pilot_model.json", "w") as json_file:
            json_file.write(model_json)
        json_file.close()
    print(lstm_pilot.summary())
    #perform prediction on test data
    predict = lstm_pilot.predict(pilot_X_test)
    predict = np.argmax(predict, axis=1)
    testY = np.argmax(pilot_y_test, axis=1)
```

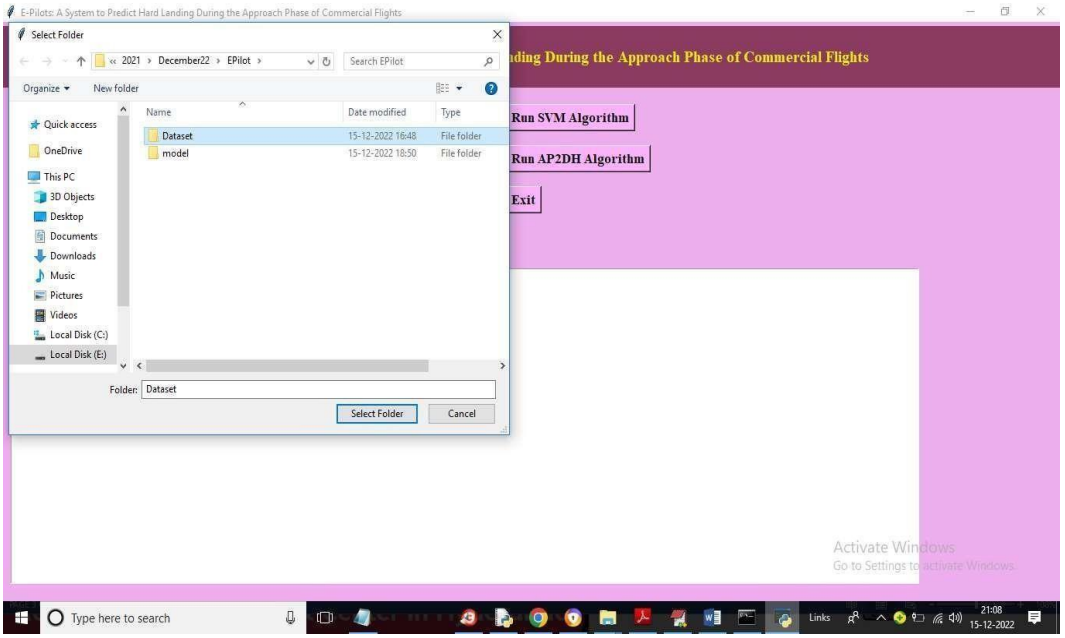
```
    calculateMetrics("DH2TD Pilot Features", testY, predict)

def graph():
   df =
pd.DataFrame([['SVM','Sensitivity',sensitivity[0]],['SVM','Specificity',specificity[0]],
                 ['Logistic Regression','Sensitivity',sensitivity[1]],['Logistic
Regression','Specificity',specificity[1]],
                 ['AP2TD','Sensitivity',sensitivity[2]],['AP2TD','Specificity',specificity[2]],
                 ['AP2DH','Sensitivity',sensitivity[3]],['AP2DH','Specificity',specificity[3]],
                 ['DH2TD','Sensitivity',sensitivity[4]],['DH2TD','Specificity',specificity[4]],
                ],columns=['Parameters','Algorithms','Value'])
   df.pivot("Parameters", "Algorithms", "Value").plot(kind='bar')
   plt.show()


def close():
   main.destroy()


font = ('times', 15, 'bold')
title = Label(main, text='E-Pilots: A System to Predict Hard Landing During the Approach
Phase of Commercial Flights')
title.config(bg='HotPink4', fg='yellow2')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)

font1 = ('times', 13, 'bold')
ff = ('times', 12, 'bold')

uploadButton = Button(main, text="Upload Flight Landing Dataset",
command=uploadDataset, bg='#ffb3fe')
uploadButton.place(x=50,y=100)
uploadButton.config(font=font1)

preprocessButton = Button(main, text="Preprocess Dataset",
command=preprocessDataset, bg='#ffb3fe')
preprocessButton.place(x=350,y=100)
preprocessButton.config(font=font1)

svmButton = Button(main,text="Run SVM Algorithm", command=runSVM, bg='#ffb3fe')
svmButton.place(x=650,y=100)
svmButton.config(font=font1)
```

```
lrButton = Button(main,text="Run Logistic Regression Algorithm", command=runLR,
bg='#ffb3fe')
lrButton.place(x=50,y=150)
lrButton.config(font=font1)

tdButton = Button(main,text="Run AP2TD Algorithm", command=runAP2TD,
bg='#ffb3fe')
tdButton.place(x=350,y=150)
tdButton.config(font=font1)

apButton = Button(main,text="Run AP2DH Algorithm", command=runAP2DH,
bg='#ffb3fe')
apButton.place(x=650,y=150)
apButton.config(font=font1)

dhButton = Button(main,text="Run DH2TD Algorithm", command=runDH2TD,
bg='#ffb3fe')
dhButton.place(x=50,y=200)
dhButton.config(font=font1)

graphButton = Button(main,text="Comparison Graph", command=graph, bg='#ffb3fe')
graphButton.place(x=350,y=200)
graphButton.config(font=font1)

closeButton = Button(main,text="Exit", command=close, bg='#ffb3fe')
closeButton.place(x=650,y=200)
closeButton.config(font=font1)

font1 = ('times', 13, 'bold')
text=Text(main,height=20,width=130)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=10,y=300)
text.config(font=font1)

main.config(bg='plum2')
main.mainloop()
```

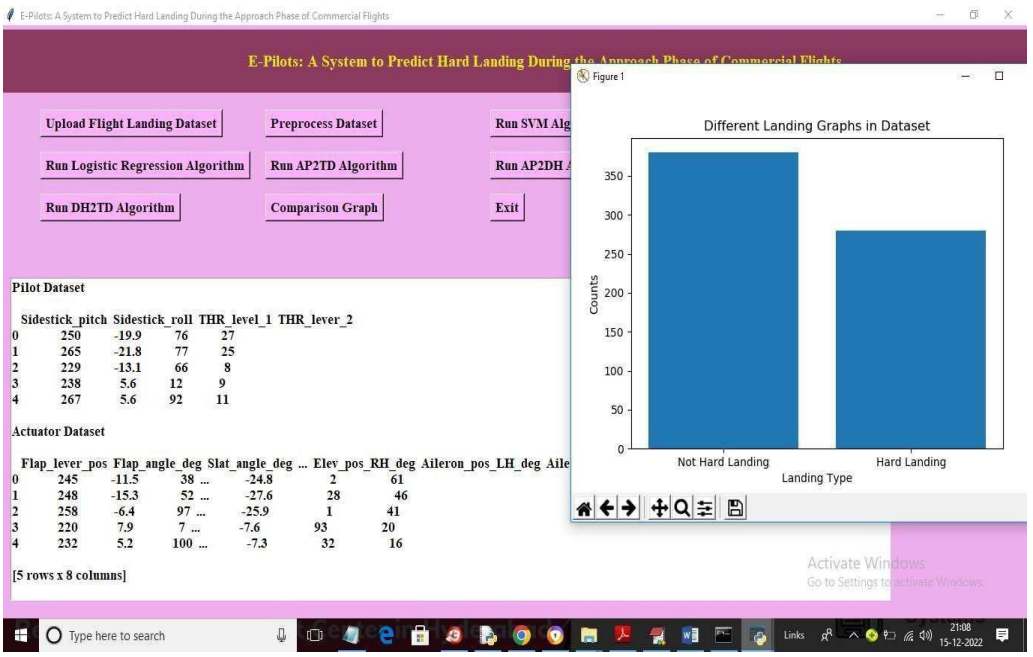
6.SCREENSHOTS



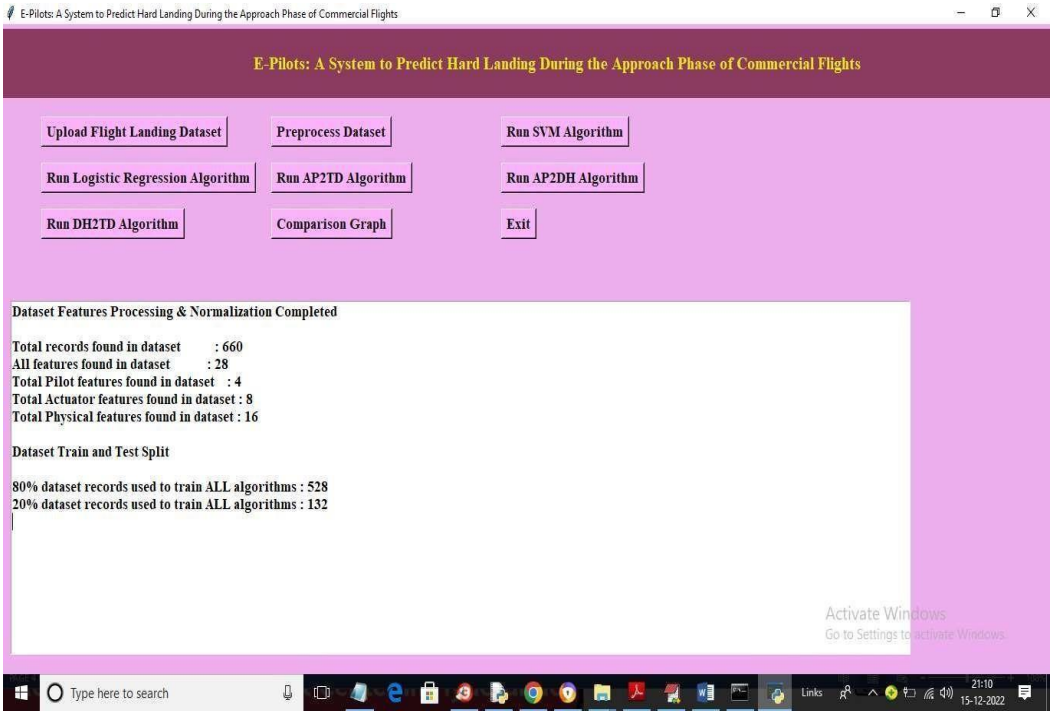
Screenshot 6.1: Upload Flight Landing Dataset



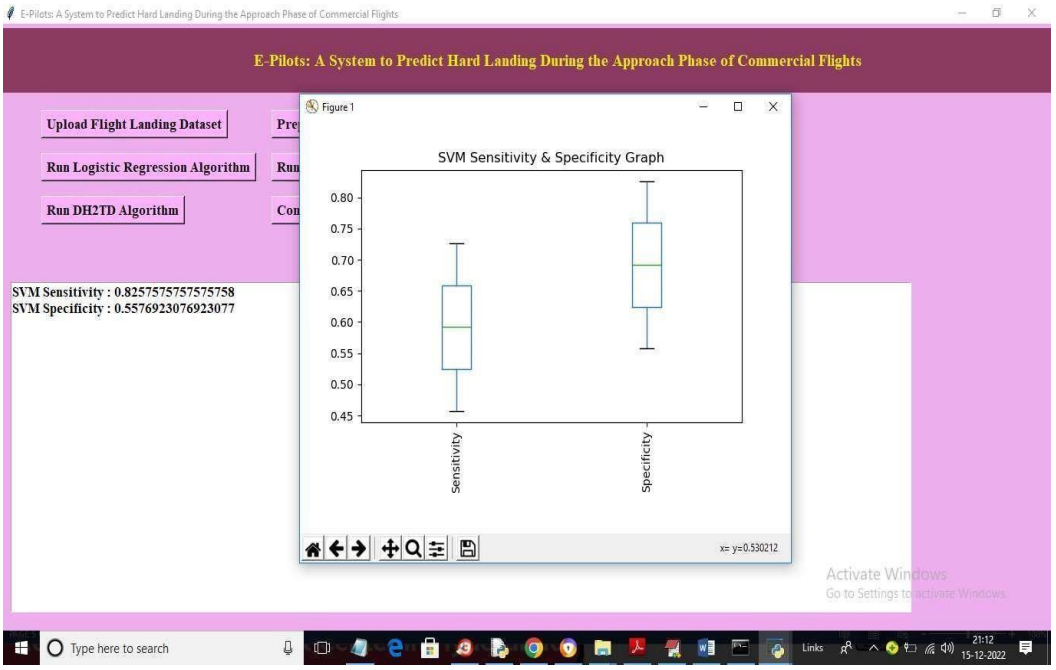
Screenshot 6.2: Select Folder



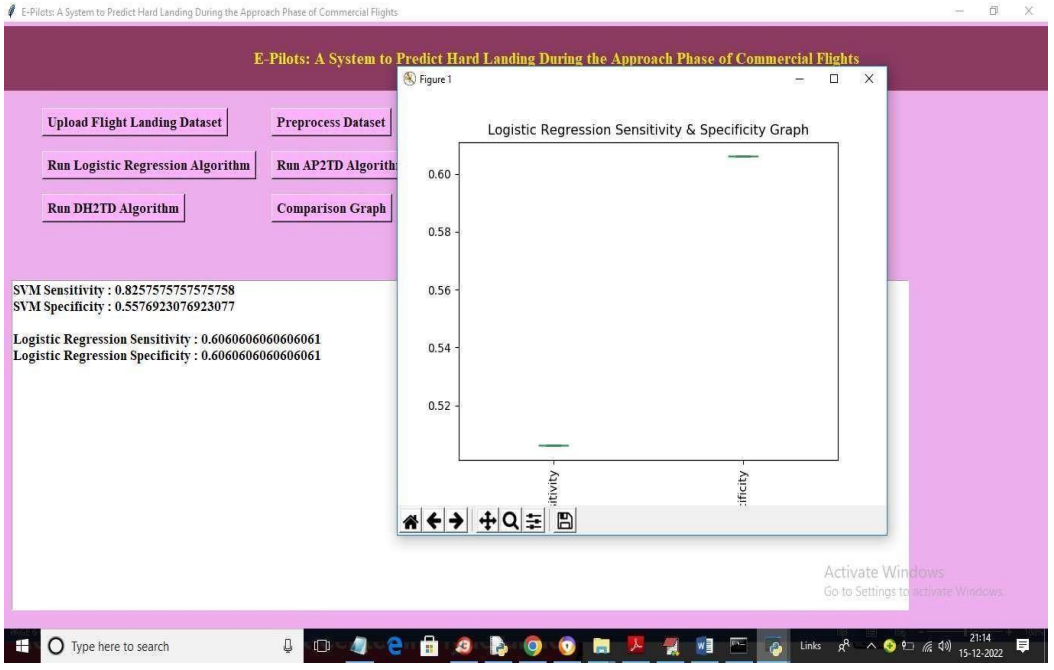
Screenshot 6.3: Dataset Loaded



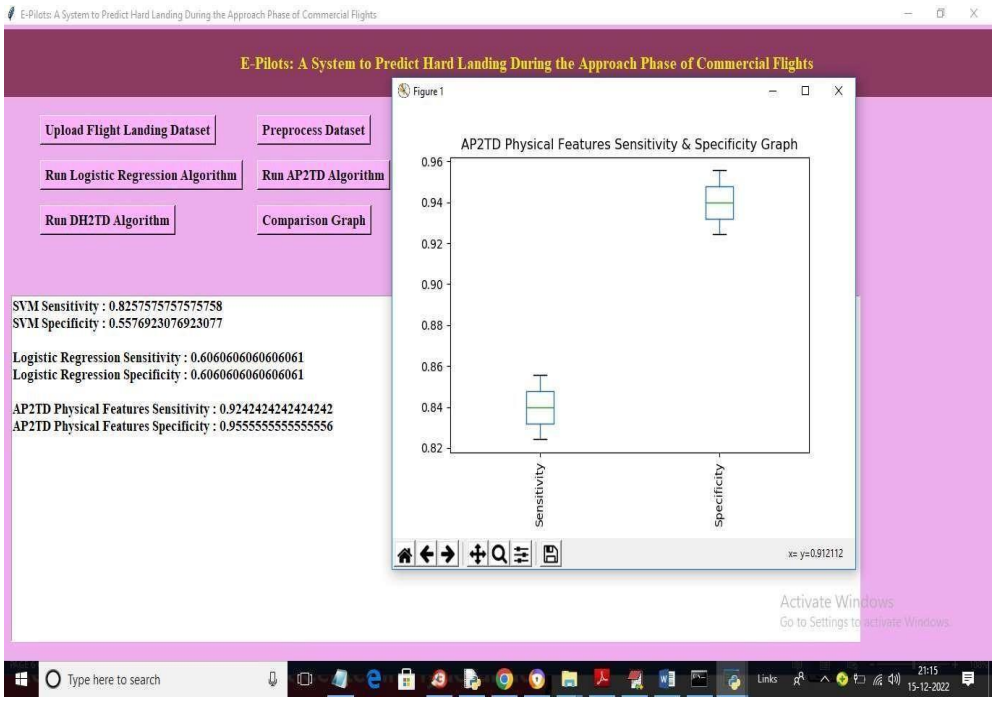
Screenshot 6.4: Preprocess Dataset



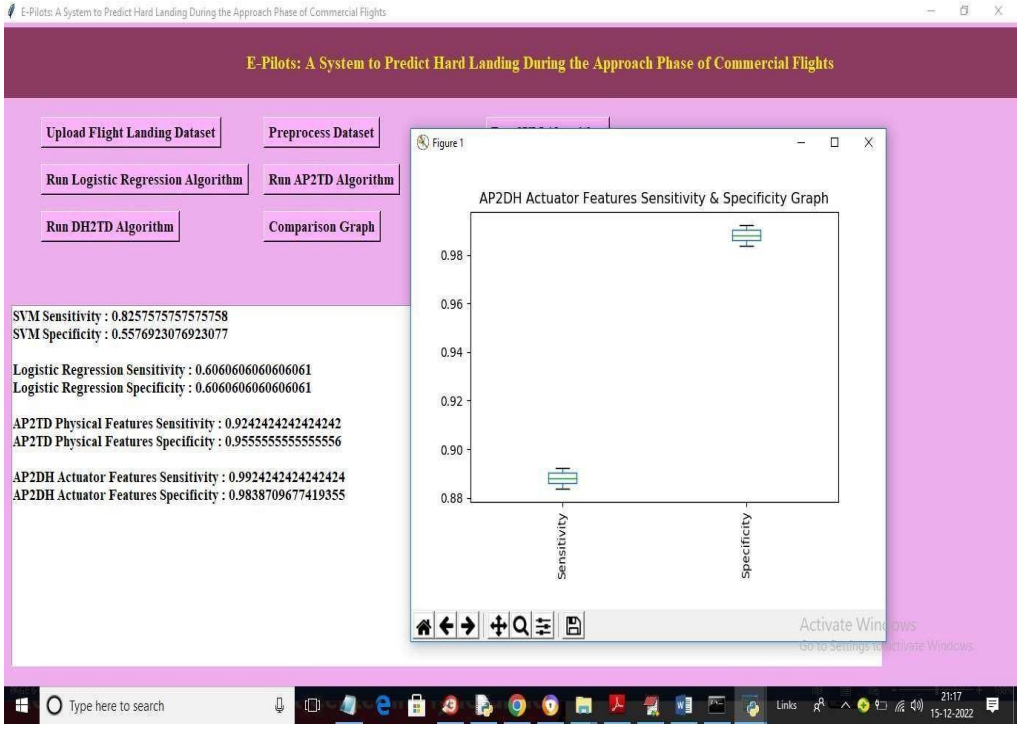
Screenshot 6.5: Run SVM Algorithm



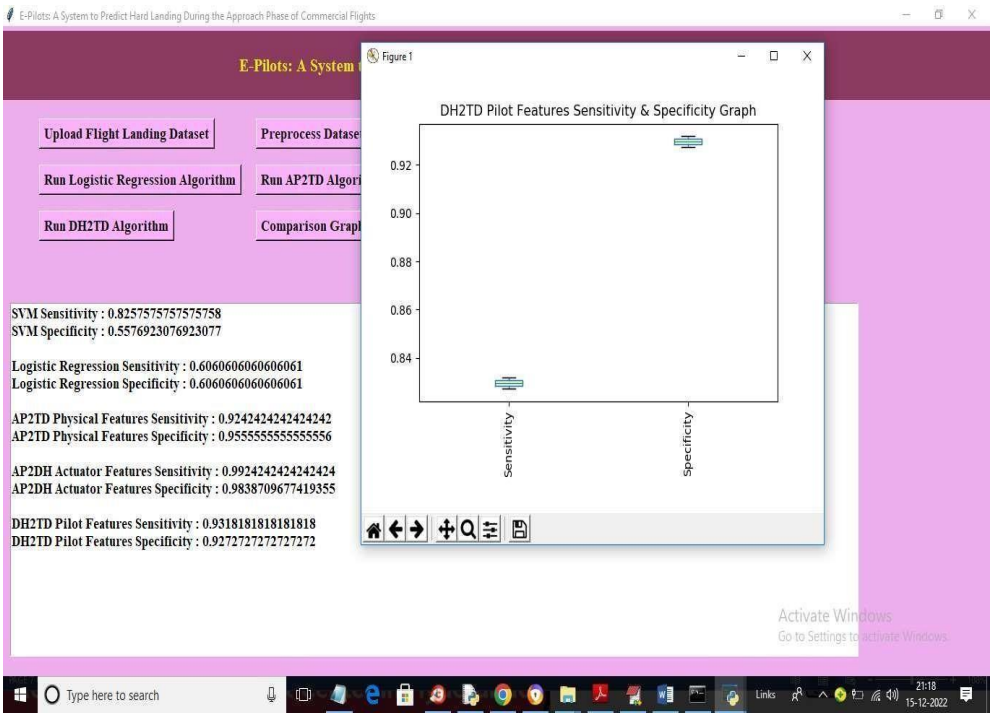
Screenshot 6.6 : Run Logistic Regression Algorithm



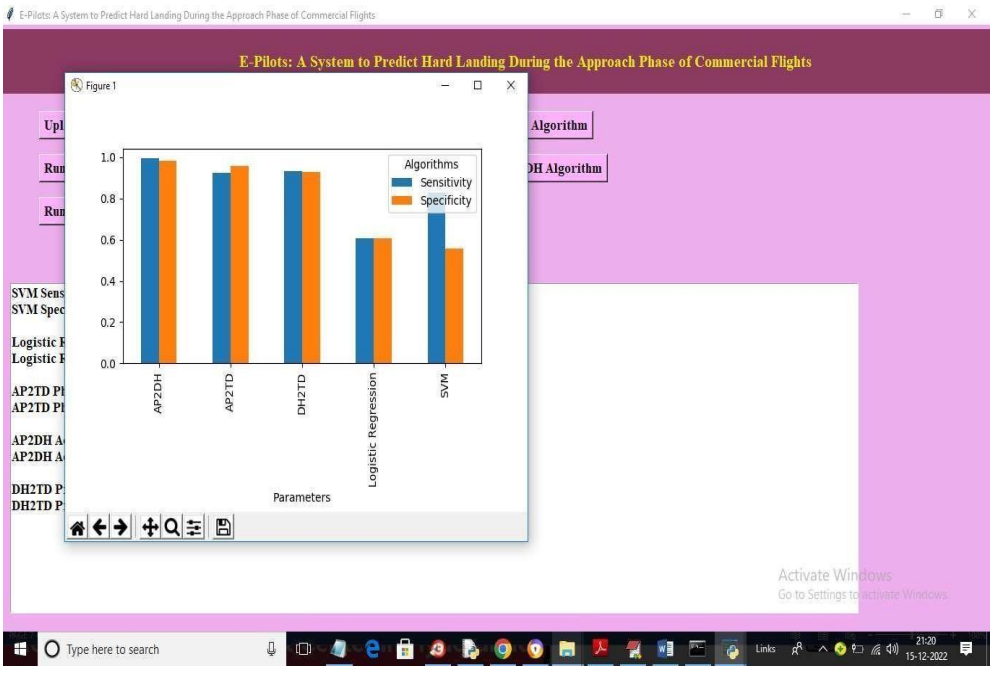
Screenshot 6.7 : Run AP2TD Algorithm



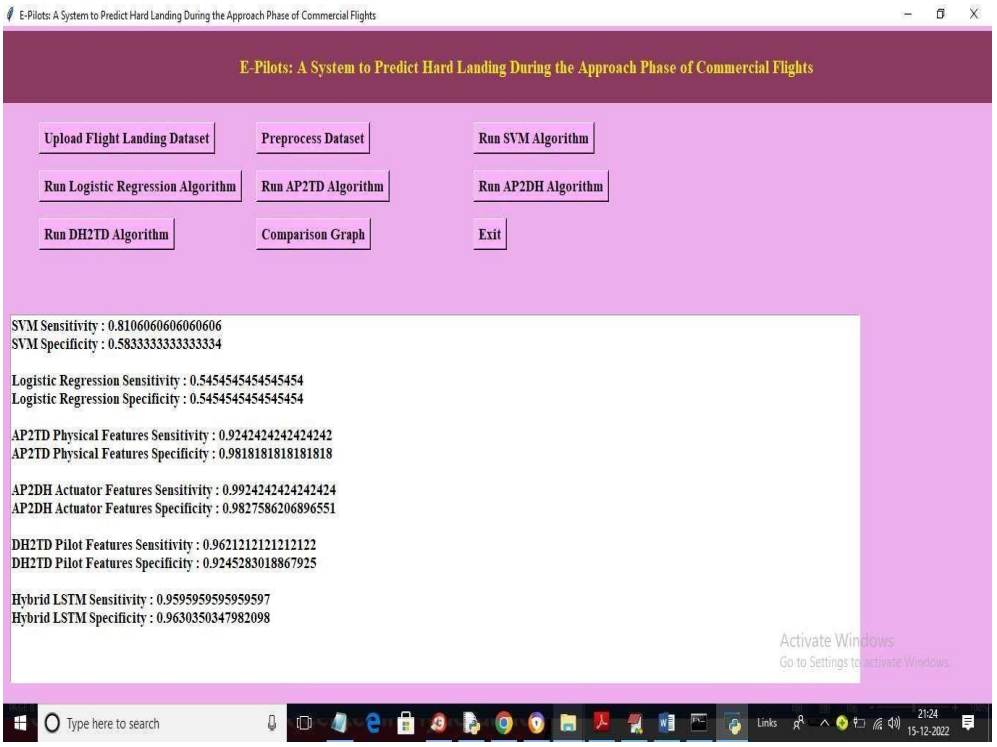
Screenshot 6.8 : Run AP2DH Algorithm



Screenshot 6.9 : Run DH2TD Algorithm



Screenshot 6.10 : Comparision Graph



Screenshot 6.11: Sensitivity and Specificity for Various algorithms

7. TESTING

7. TESTING

7.1 INTRODUCTION TO 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, subassemblies, 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.

7.2 TYPES OF TESTING

7.1.1 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. Unit testing is usually conducted as a 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. The purpose of the unit testing is to isolate a section of code, to verify the correctness of the code, to test every function and code and to help with code reuse. Unit testing of a software product is carried out during the development of an application.

7.1.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Integration tests demonstrate thatalthough the components were individually satisfactory, 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.

7.1.3 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 : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of applicationoutputs must be exercised.

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.

7.1.4 SYSTEM TEST

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.

7.1.5 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.

7.1.6 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.

7.2 TEST STRATEGY AND APPROACH

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

Test objectives

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

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed

All links should take the user to the correct page.

7.3 TEST CASES

7.3.1 CLASSIFICATION

Test case ID	Test case name	Purpose	Input	Output
1.	Uplaod dataset	If upload successfull.	Pass	If already uploaded then it fails.
2.	Click on Preprocess Dataset Button	To normalize, shuffle and split dataset into train and test	Pass	If button is not working then it fails.
3.	Click on run SVM algorithm button	To train SVM.	Pass	If button is not working then it fails.
4.	Click on run Logistic Regression Algorithm	To train Logistic Regression	Pass	If button is not working then it fails. .
5.	Click on Run AP2TD Algorithm button	To train LSTM on physical features	Pass	If button is not working then it fails.
6.	Click on Run AP2DH Algorithm Button	To train LSTM on actuator features	Pass	If button is not working then it fails.
7.	Click on Run DH2TD Algorithm button	To train LSTM on pilot features	Pass	If button is not working then it fails.
8.	Comparison Graphs	For our CNN models the Sensitivity and specificity Values are displayed	Pass	If Values not displayed Fail.

8. CONCLUSION
&
FUTURE SCOPE

8. CONCLUSION & FUTURE SCOPE

8.1 PROJECT CONCLUSION

The project "E-Pilots: A System to Predict Hard Landing During the Approach Phase of Commercial Flights" aims to enhance flight safety by predicting the likelihood of hard landings. The system utilizes machine learning algorithms, such as decision trees and SGD classifiers, to analyze historical flight data and generate predictions. By providing early warnings and proactive measures, the system can assist pilots and air traffic controllers in mitigating the risk of hard landings. The implementation of this project requires data collection, model training, system development, integration, and testing. The future scope includes refining the prediction models, integrating real-time data sources, facilitating collaborative decision-making, and expanding the system's capabilities. Overall, this project contributes to improving flight safety and reducing the occurrence of hard landings during commercial flights

8.2 FUTURE SCOPE

The future scopes aim to enhance the system's capabilities, improve flight safety, and provide more comprehensive support to pilots, air traffic controllers, and other stakeholders involved in the landing phase of commercial flights. In the future, such a system could be expanded to also include Air Traffic Management in which the information is shared with the Air Traffic Controller in order to anticipate the likely. .

9. BIBLIOGRAPHY
&
REFERENCES

9. BIBLIOGRAPHY & REFERENCES

9.1 REFERENCES

[1] Statistical Summary of Commercial Jet Airplane Accidents-Worldwide Operations|1959-2017, Boeing Commercial Airplanes, Aviation Saf., Seattle, WA, USA, 2018.

[2] "Developing standardized FDM-based indicators," Eur. Aviation Saf. Plan 2012-2015, Cologne, Germany, 2016.

[3] "Advisory circular ac no:91-79a mitigating the risks of a runway overrun upon landing," Federal Aviation Admin., Washington, Dc, USA, 2016.

[4] M. Coker and L.S. Pilot, " Why and when to perform a go-around maneu-ver," Boeing Edge, vol. 2014, pp. 5-11, 2014.

[5] T. Blajev and W. Curtis, "Go-around decision making and execution project: Final report to flight safety foundation, "Flight Saf. Found ., Alexandria, VA, USA, Mar. 2017.

[6] "European action plan for the prevention of runway excursions," Eurocon-trol, Brussels, Belgium, 2013.

[7] "Artificial intelligence roadmap-A human-centric approach to ai in aviation," Eur. Union Aviation Saf. Agency, Cologne, Germany, 2019.

[8] D. Zhou, X. Zhuang, H. Zuo, H. Wang, and H. Yan, " Deep learning-based approach for civil aircraft hazard identification and prediction,".

[9] Wenbing Chang, Shenghan Zhou, and Silin Qian. A better aeroplane.

[10] Xiang Zhang and YL Luo. Infrared thermal imaging and integrated svm are used to identify faults on aircraft electrical boards. 12:012-012, Meas

Control Technol., 2012.

[11] Hu C , Zhou S H , Xie Y , et al. The study on hard landing prediction model with optimized parameter SVM method[C]// Control Conference. IEEE, 2016.

[12] Luo X X , Wang C Y . Prediction method of landslide geological hazard based on AdaBoost model[J]. Journal of Physics: Conference Series, 2021, 1966(1):012019 (7pp).

[13] Gil D , Hernandez-Sabate A , Enconniere J , et al. E-Pilots: A System to Predict Hard Landing During the Approach Phase of Commercial Flights[J]. IEEE Access, 2021, PP(99):1-1.

[14] Senol, M.B. Evaluation and prioritization of technical and operational airworthiness factors for flight safety. Aircr. Eng. Aerosp. Technol. 2020, 92, 1049–1061. [CrossRef].

[15] Kong, Y.; Zhang, X.; Mahadevan, S. Bayesian Deep Learning for Aircraft Hard Landing Safety Assessment. IEEE Trans. Intell. Transp. Syst. 2022, 23, 17062–17076. [CrossRef]

9.2 GITHUB LINK

https://github.com/burrevedika/EPilot