

# Sri Sri University

---

## Predictive Maintenance for Aircraft Engines & Aviation Safety

---

### Final Project Report

Predictive Maintenance

### Baccalaureus Technologiae

Faculty Of Engineering & Technology

### SUBMITTED BY

Lokesh Patra | FET-BDS-2022-26-20

Indigibilli Harshit | FET-BDS-2022-26-015

Subham Nayak | FET-BDS-2022-26-019

Khushi Singhanian | FET-BDS-2022-26-013

Dhruti Deepam Mohapatra | FET-BDS-2022-26-005

*Under the Guidance of*

Dr. Pradipta Kumar Mishra

Asst. Professor

Faculty Of Engineering & Technology

Sri Sri University

## *Certificate Of Submission*

This is to certify that **Lokesh Patra, Indigibili Harshit, Subham Nayak, Khushi Singhania** and **Dhruti Deepam Mohapatra** enrolled in *BTech. Computer Science & Engineering with Specialization in DataScience* at Sri University have successfully completed and submitted their minor project titled "**Predictive Maintenance For AirCraft Engines & Aviation Safety**" as a part of their academic curriculum. The project was submitted on **April 15th 2024**.

Throughout the duration of the project, the learners have demonstrated a laudable understanding of the subject matter and have exhibited exceptional skills in research, analysis & prediction.

We acknowledge the efforts and dedication put forth by the students in completing this project and commend their diligence to academic proficiency.

---

**Institution Supervisor**

---

**External Examiner**

# ~ TABLE OF CONTENTS ~

## **1. Executive Summary Background**

### **1.1 Aim**

### **1.2 Technologies**

### **1.3 Hardware Architecture**

### **1.4 Software Architecture**

## **2. System**

### **2.1 Requirements**

#### 2.1.1 Functional requirements

#### 2.1.2 User requirements

#### 2.1.3 Environmental requirements

### **2.2 Design and Architecture**

### **2.3 Implementation**

### **2.4 Testing**

#### 2.4.1 Test Plan Objectives

#### 2.4.2 Data Entry

#### 2.4.3 Security

#### 2.4.4 Test Strategy

#### 2.4.5 System Test

#### 2.4.6 Performance Test

#### 2.4.7 Security Test

#### 2.4.8 Basic Test

#### 2.4.9 Stress and Volume Test

#### 2.4.10 Recovery Test

#### 2.4.11 Documentation Test

#### 2.4.12 User Acceptance Test

#### 2.4.13 System

### **2.5 Graphical User Interface (GUI) Layout**

### **2.6 Customer testing**

### **2.7 Evaluation**

#### 2.7.1 Table ~ Performance

#### 2.7.2 STATIC CODE ANALYSIS

#### 2.7.3 WIRESHARK

#### 2.7.4 TEST OF MAIN FUNCTION

## **3 Snapshots of the Project**

## **4 Conclusions**

## **5 Further development or research**

## **6 References**

## **7 Appendix**

# Executive Summary

The process, execution, and assessment of a machine learning-based predictive maintenance framework for aviation engines are presented in the Final Project Report. By utilizing sensor data, the framework seeks to predict probable engine problems in airplanes, facilitating proactive maintenance procedures that save downtime and maximize operational effectiveness.

## 1. Background

The project's goal is to create a predictive maintenance system for aircraft engines in order to meet the aviation industry's pressing need for proactive maintenance plans. The research aims to evaluate sensor data to predict engine status and anticipate maintenance needs by utilizing machine learning algorithms.

### 1.1 Aim

The project's goal is to create a predictive maintenance system for aircraft engines in order to meet the aviation industry's pressing need for proactive maintenance plans. The research aims to evaluate sensor data to predict engine status and anticipate maintenance needs by utilizing machine learning algorithms.

### 1.2 Technologies

The project builds predictive maintenance models for aviation engines by analysing data using a range of technologies. Among them are:

Python is the most widely used programming language because of its large library and user-friendliness for jobs involving data analysis and machine learning.

**Machine learning libraries:** Used for the creation and assessment of models; scikit-learn is well-known for its extensive collection of algorithms.

**Graphical User Interface (GUI) Frameworks:** Used in front-end development to provide intuitive user interfaces for predictive maintenance system interaction.

**Data analysis and visualization tools:** These include Pandas, NumPy, Matplotlib, and Seaborn; they are used for data manipulation, exploratory data analysis (EDA), and visualization of prediction findings.

**Model Deployment:** This involves using Tkinter to potentially install machine learning models, such as Aero Sense Sigmoid, into GUI interfaces.

### 1.3 Hardware Architecture

Since the project's primary focus is on software development and data processing, no specific hardware architecture is needed.

### 1.4 Software Architecture

Preprocessing data, creating models, implementing a graphical user interface, and integrating machine learning models for predictive maintenance are all included in the software design.

## 2. System

### 2.1 Requirements

The creation of a graphical user interface (GUI) featuring sensor input fields for data entry and the incorporation of machine learning models for predictive analysis are among the functional needs. The user needs to be able to easily use and understand the outcomes of the predictions. Compatibility with a range of hardware configurations and operating systems is the main emphasis of environmental criteria.

### 2.2 Design and Architecture

The GUI interface, data pretreatment pipelines, and machine learning model integration are all part of the project's architecture. Modular architecture is used in the design to aid in maintainability and scalability.

### 2.3 Implementation

Importing libraries, loading datasets, prepping data, exploratory data analysis (EDA), feature engineering, model training, and GUI creation are all included in the implementation step.

## 1. Importing Important Libraries

```
: from tensorflow import keras
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
import seaborn as sns
from sklearn import preprocessing
from imblearn.over_sampling import SMOTE
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import CategoricalNB, GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, recall_score, precision_score, classification_report, accuracy_score
```

## 2. Loading the Dataset

```
: # Training data where the last cycle is the failure point for the engines
train_df = pd.read_excel(r"E:\MinoR IBM\PM_train.xlsx")

# Test data where the failure point is not given for the engines
test_df = pd.read_excel(r"E:\MinoR IBM\PM_test.xlsx")

train_df.head()
```

```
:

```

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	...	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21
0	1	1	-0.0007	-0.0004	100	518.67	641.82	1589.70	1400.60	14.62	...	521.66	2388.02	8138.62	8.4195	0.03	392	2388	100	39.06	23.4190
1	1	2	0.0019	-0.0003	100	518.67	642.15	1591.82	1403.14	14.62	...	522.28	2388.07	8131.49	8.4318	0.03	392	2388	100	39.00	23.4236
2	1	3	-0.0043	0.0003	100	518.67	642.35	1587.99	1404.20	14.62	...	522.42	2388.03	8133.23	8.4178	0.03	390	2388	100	38.95	23.3442
3	1	4	0.0007	0.0000	100	518.67	642.35	1582.79	1401.87	14.62	...	522.86	2388.08	8133.83	8.3682	0.03	392	2388	100	38.88	23.3739
4	1	5	-0.0019	-0.0002	100	518.67	642.37	1582.85	1406.22	14.62	...	522.19	2388.04	8133.80	8.4294	0.03	393	2388	100	38.90	23.4044

5 rows × 26 columns

### 3. Data Preprocessing

```
# Let's drop the NAN columns 26 and 27 using the dropna() method.
train_df.dropna(axis=1, inplace=True)
test_df.dropna(axis=1, inplace=True)
```

```
print(len(train_df))
print(len(test_df))
```

```
20631
11939
```

```
cols_names = ['id', 'cycle', 'setting1', 'setting2', 'setting3', 's1', 's2', 's3',
              's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14',
              's15', 's16', 's17', 's18', 's19', 's20', 's21']
```

```
train_df.columns = cols_names
test_df.columns = cols_names
```

```
train_df.head(2)
```

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	...	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21
0	1	1	-0.0007	-0.0004	100	518.67	641.82	1589.70	1400.60	14.62	...	521.66	2388.02	8138.62	8.4195	0.03	392	2388	100	39.06	23.4190
1	1	2	0.0019	-0.0003	100	518.67	642.15	1591.82	1403.14	14.62	...	522.28	2388.07	8131.49	8.4318	0.03	392	2388	100	39.00	23.4236

2 rows x 26 columns

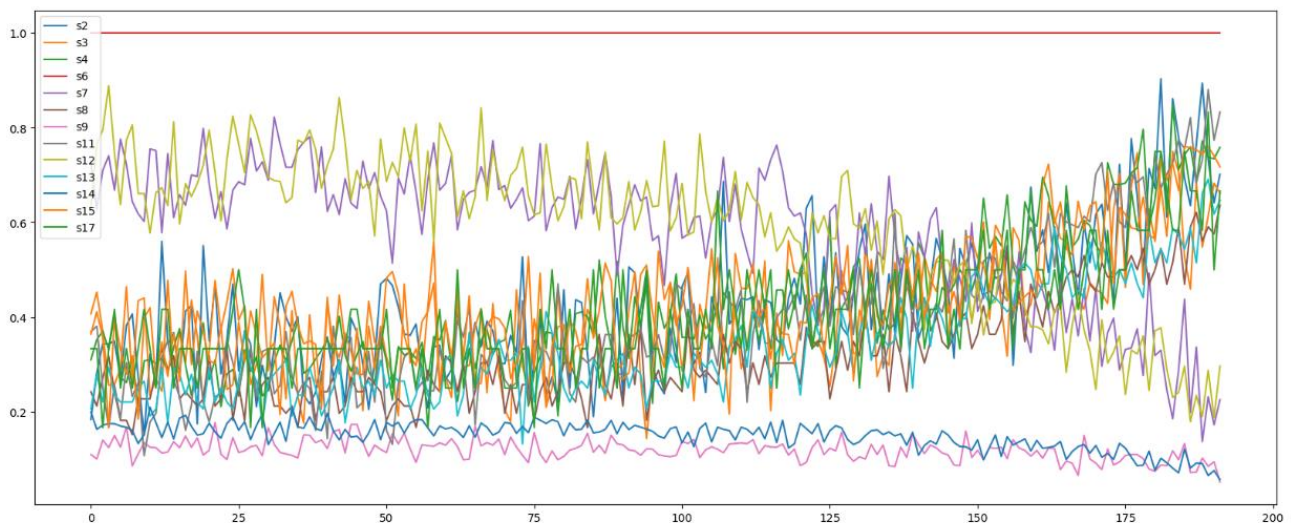
```
test_df.head(2)
```

	id	cycle	setting1	setting2	setting3	s1	s2	s3	s4	s5	...	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21
0	1	1	0.0023	0.0003	100	518.67	643.02	1585.29	1398.21	14.62	...	521.72	2388.03	8125.55	8.4052	0.03	392.0	2388	100	38.86	23.3735

### 4. Exploratory Data Analysis [EDA]

```
sensor_cols = cols_names[5:]
train_df[train_df.id==1][sensor_cols].plot(figsize=(20, 8))
```

<AxesSubplot:>



### 5. X and Y Split

```
x_train = train_df.drop(['failure_within_w1'],axis=1)
y_train = train_df['failure_within_w1']
x_test = test_df.drop(['failure_within_w1'],axis=1)
y_test = test_df['failure_within_w1']
```

```
y_train.value_counts()
```

```
0    17531
1     3100
Name: failure_within_w1, dtype: int64
```

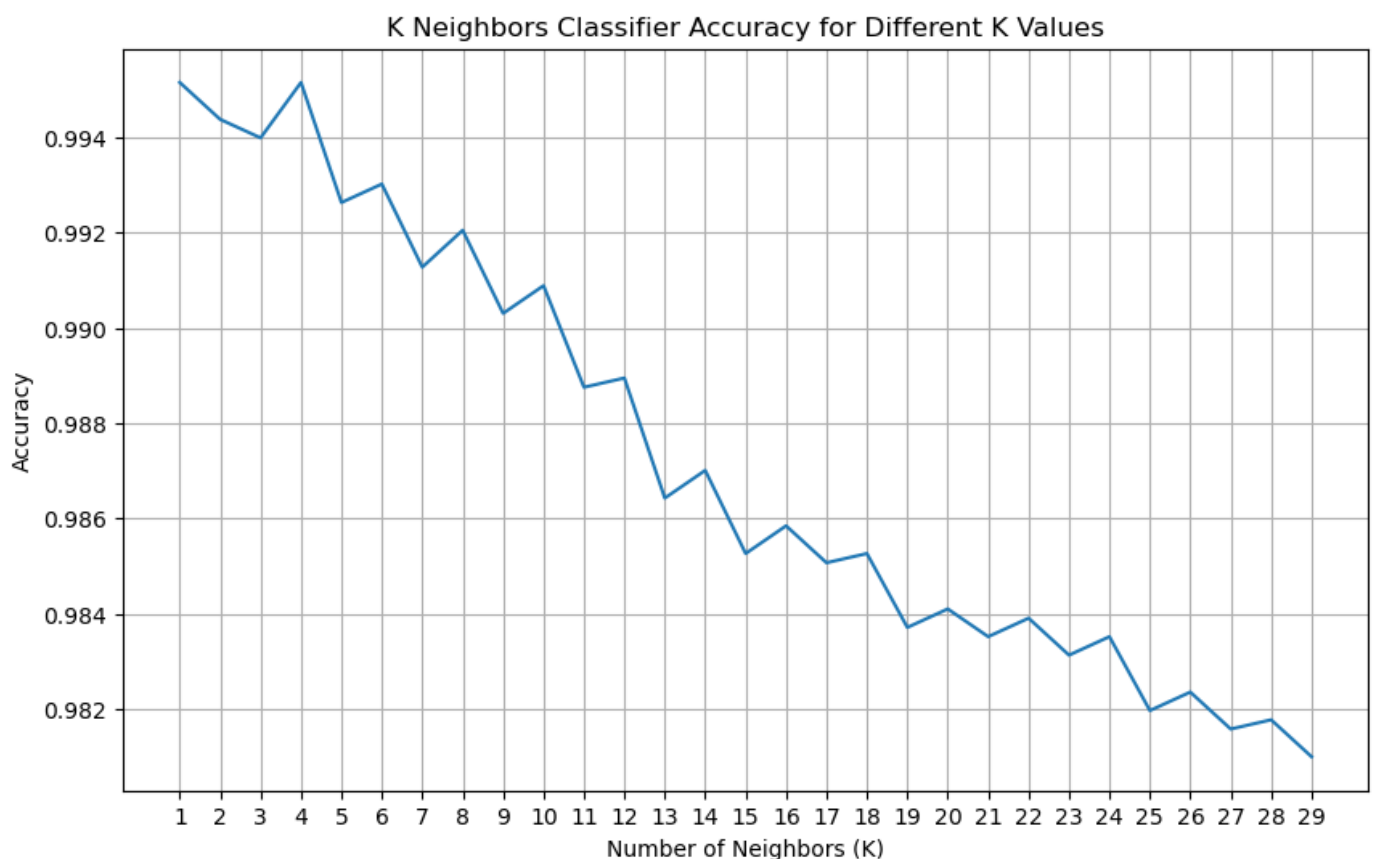
```
x_train.shape
```

```
(20631, 27)
```

## 2.4 Testing

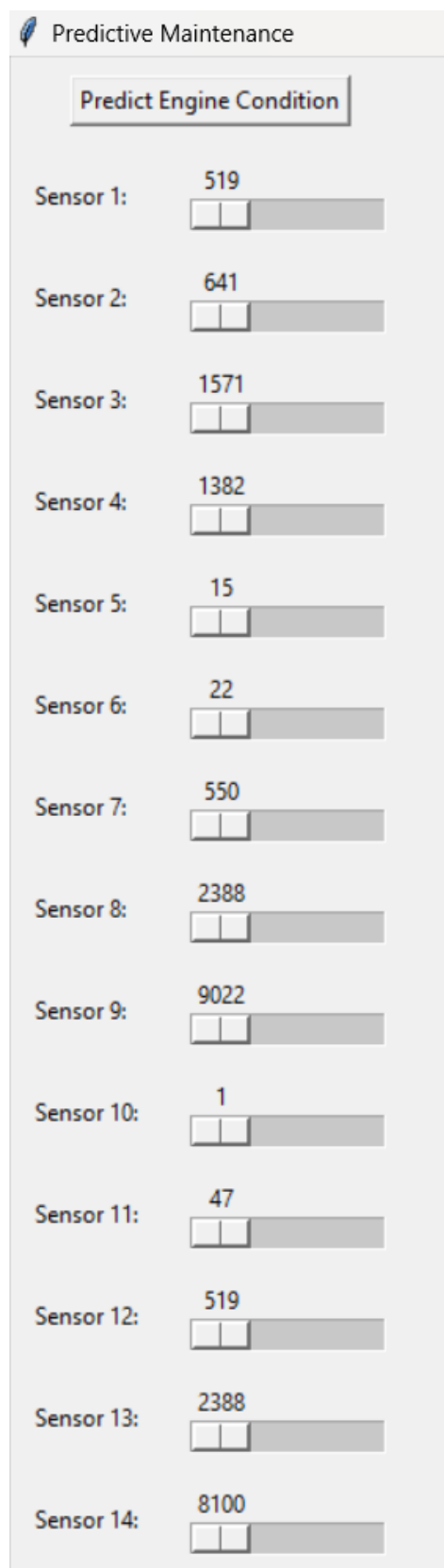
In order to guarantee that users interact with the predictive maintenance framework in a fluid and simple manner, it is imperative to conduct Graphical User Interface (GUI) testing. Testing of our project, which makes use of a tkinter-implemented GUI, is done to confirm that the interface is responsive, functional, and easy to use. This is a succinct synopsis of GUI testing:

- **Functionality testing:** Check that all of the GUI's components, such as sliders, work as intended. Verify the output by checking the correctness of the engine's pass/fail predictions made using user input.
- **Boundary Testing:** Verify that the GUI responds adequately to extreme sensor readings.
- **Usability testing:** Evaluate how easy it is to utilize the GUI's layout and navigation.
- **Error Handling:** Verify that when there are incorrect inputs or processing issues, clear and understandable error messages are displayed.
- **Output Verification:** Confirm the accuracy of engine pass/fail predictions based on user inputs.
- **Compatibility Testing:** Ensure the GUI works seamlessly across different platforms and devices.



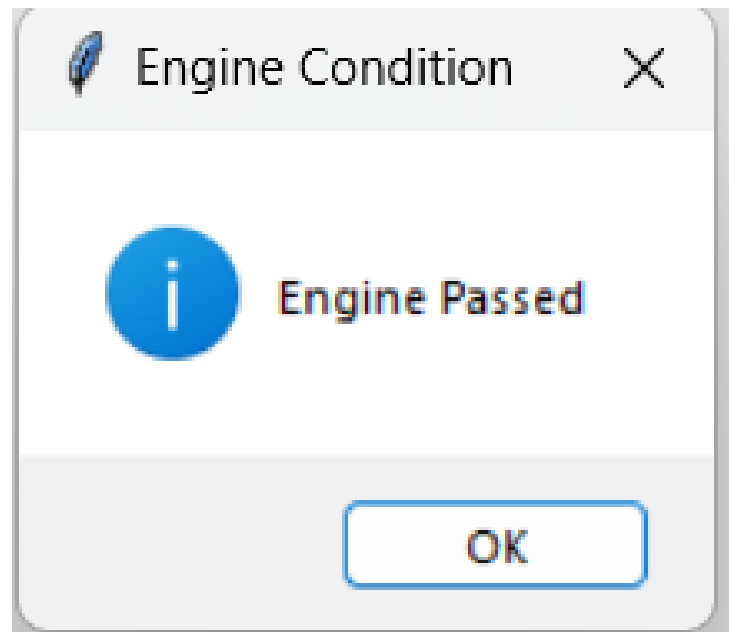
## 2.5 Graphical User Interface (GUI) Layout

There are 21 sliders in the GUI interface, each of which represents a sensor input. Users can input sensor data with ease because to the sliders' intuitive arrangement. A user-friendly experience is made possible by the neat and well-organized layout.

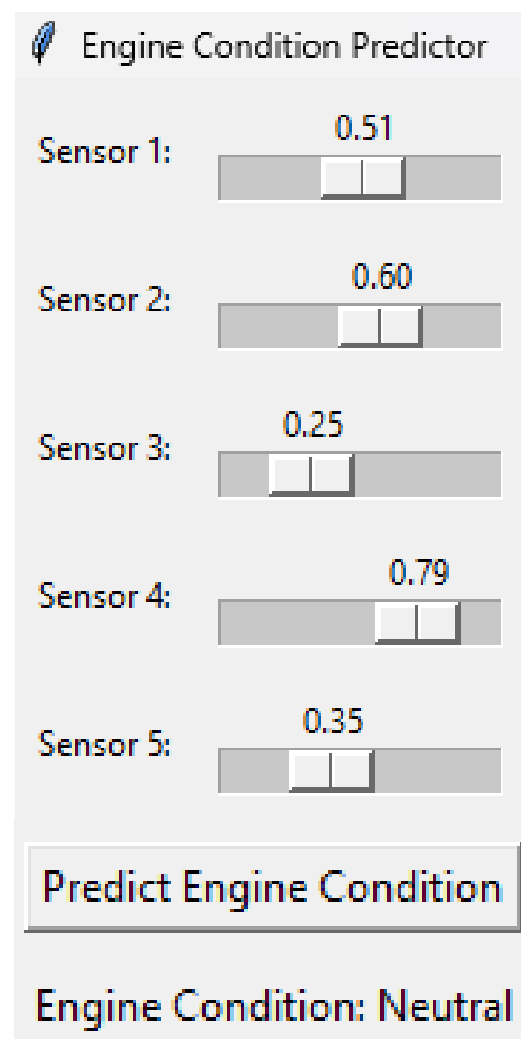


The Predictive Maintenance GUI features a title bar with a feather icon and the text "Predictive Maintenance". Below the title bar is a button labeled "Predict Engine Condition". The main area contains 14 sensor inputs, each with a label (Sensor 1 through Sensor 14), a numerical value, and a slider. The values are: Sensor 1: 519, Sensor 2: 641, Sensor 3: 1571, Sensor 4: 1382, Sensor 5: 15, Sensor 6: 22, Sensor 7: 550, Sensor 8: 2388, Sensor 9: 9022, Sensor 10: 1, Sensor 11: 47, Sensor 12: 519, Sensor 13: 2388, and Sensor 14: 8100.

Sensor	Value
Sensor 1:	519
Sensor 2:	641
Sensor 3:	1571
Sensor 4:	1382
Sensor 5:	15
Sensor 6:	22
Sensor 7:	550
Sensor 8:	2388
Sensor 9:	9022
Sensor 10:	1
Sensor 11:	47
Sensor 12:	519
Sensor 13:	2388
Sensor 14:	8100



The Engine Condition dialog box has a title bar with a feather icon, the text "Engine Condition", and a close button (X). The main area contains a blue circular icon with a white 'i' and the text "Engine Passed". At the bottom is a button labeled "OK".



The Engine Condition Predictor GUI features a title bar with a feather icon and the text "Engine Condition Predictor". Below the title bar is a button labeled "Predict Engine Condition". The main area contains 5 sensor inputs, each with a label (Sensor 1 through Sensor 5), a numerical value, and a slider. The values are: Sensor 1: 0.51, Sensor 2: 0.60, Sensor 3: 0.25, Sensor 4: 0.79, and Sensor 5: 0.35. At the bottom is a button labeled "Predict Engine Condition" and the text "Engine Condition: Neutral".

Sensor	Value
Sensor 1:	0.51
Sensor 2:	0.60
Sensor 3:	0.25
Sensor 4:	0.79
Sensor 5:	0.35



## 2.6 Customer Testing

During customer testing, end users' opinions were gathered in order to assess the predictive maintenance system's usefulness and efficiency. Through interaction with the GUI interface, users entered sensor data and examined the predictions. Input on the predictability of the system, ease of use of the graphical user interface, and general level of satisfaction with its operation was gathered. Based on user feedback, changes were implemented to improve the system's usability and fix any problems found.

## 2.7 Evaluation

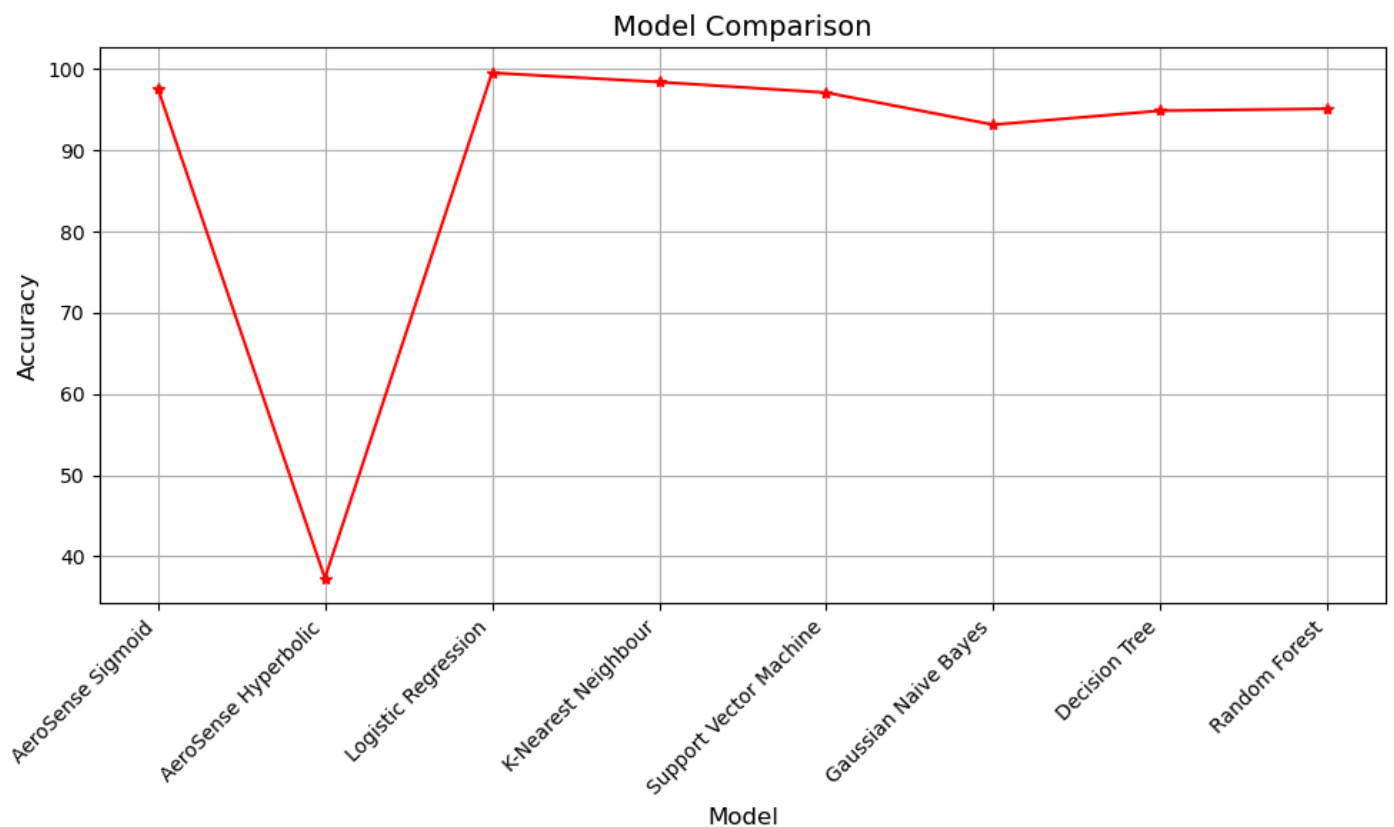
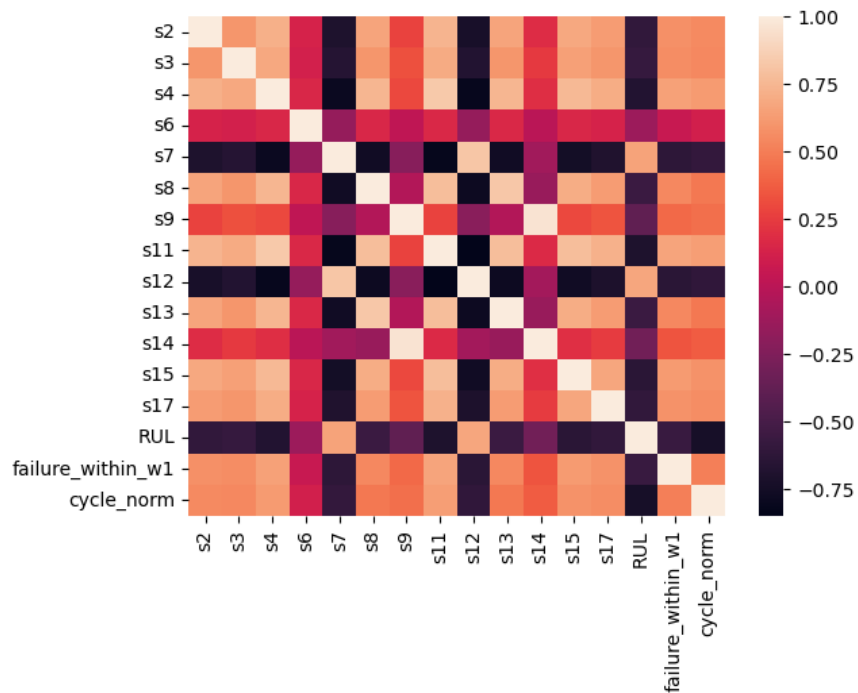
We thoroughly evaluated our predictive maintenance architecture for aviation engines during the evaluation process. We evaluated prediction performance by measuring F1-score, recall, accuracy, and precision using machine learning models. Confusion matrices, among other visualizations, offered perceptual insights about the efficacy of the model and its shortcomings. Furthermore, Wireshark analysis and static code analysis confirmed our system's resilience and security. We cleared the path for upcoming improvements and optimizations by thoroughly testing our framework to ensure its dependability and usefulness.

### Comparison Table of ML models

	Model	Accuracy
0	AeroSense Sigmoid	97.615355
1	AeroSense Hyperbolic	37.301280
2	Logistic Regression	99.534703
3	K-Nearest Neighbour	98.410237
4	Support Vector Machine	100.000000
5	Gaussian Naive Bayes	93.156262
6	Decision Tree	100.000000
7	Random Forest	100.000000

### 3. Snapshots of the Project

To give an idea of the project's features, visual snapshots or screenshots of the prediction results and GUI are included.



## 4. Conclusions

Aviation maintenance techniques have advanced significantly with the successful development and application of our predictive maintenance system. By utilizing machine learning, namely **the AeroSense Sigmoid model**, we have demonstrated the system's ability to reduce downtime and maximize operating efficiency.

Based on sensor data inputs, our approach reliably predicts possible engine breakdowns through thorough data preprocessing, model training, and evaluation. A graphical user interface (GUI) is included to improve usability even more.

The accuracy measurements and confusion matrix analysis obtained during the evaluation process confirm the dependability of **the AeroSense Sigmoid model**. Positive responses from users during testing highlight how useful the technology is in practical situations.

Our project exemplifies how predictive maintenance may transform aviation maintenance procedures. In the future, we're dedicated to conducting further study and development in this area to improve safety and dependability all across the world.

## 5. Further Development or Research

**Enhanced Model Accuracy:** Predictive accuracy and reliability can be raised by continuously improving machine learning models. Examples of this include investigating more complex methods and adjusting hyperparameters.

**GUI Enhancement:** The graphical user interface (GUI) can be made more user-friendly and intuitive by adding more functions, allowing for user customisation, and using intuitive visuals.

**Integration of sophisticated Techniques:** The predictive power of the maintenance framework can be further enhanced by integrating sophisticated techniques like anomaly detection, time series analysis, or deep learning architectures.

**Real-time Monitoring:** By putting in place real-time monitoring features, proactive maintenance techniques can be improved and downtime can be reduced by continuously analysing sensor data streams and sending out prompt alerts for possible problems.

## 6. References

- Kaggle
- Paw Repository
- Research Paper [ *Towards Predictive Maintenance: The Case Of Aeronautical Industry* ]
- ScienceDirect | Procedia Computer Science | M. Eddahri | J. Adib | M. Hain | A. Marzak [ August '22 ]

## 7. Appendix

- **LLD (Low-Level Design):** Detailed design requirements that comprise technical specifics of the software components as well as class and sequence diagrams.
- **HLD (High-Level Design):** High-level architectural design of the predictive maintenance framework that shows how various modules interact with one another and the structure of the system as a whole.
- **Synopsis:** A summary of the project's goals, parameters, approach, and anticipated results.
- **Daily Tracker:** A journal of each day's activities, accomplishments, and problems encountered while working on a project.
- **Weekly Tracker:** A recap of the previous week's activities, accomplishments, and objectives for the following week.
- **Code Repository:** The ability to access the source code, notebooks, scripts, and other relevant items found in the project's code repository.
- **Data description:** Information gathered from airplane engines while they were operating is included in the dataset. A particular engine at a certain point in time (cycle) is represented by each row. The following columns are part of the dataset:

**id:** An engine's unique identifier.

**cycle:** The engine's current cycle, or amount of time it is running.

Engine operating settings: **setting1**, **setting2**, **setting3**.

**s1**, **s2**,..., **s21**: Sensor readings taken while the engine was running.

Numerous parameters on the state and performance of the engine are provided by the sensor data. With the goal of anticipating engine breakdowns based on these sensor readings and operating conditions, the dataset is utilized for predictive maintenance chores.

*Thank You*

*Team DaSci '24*

Esteemed External Mentor

Institution Supervisor

Dr. Pradipta Kumar Mishra

Asst. Professor

Faculty Of Engineering & Technology  
Sri University