**GAYATRI VIDYA PARISHAD COLLEGE FOR DEGREE AND PG COURSES (AUTONOMOUS)**

**(AFFILIATED TO ANDHRA UNIVERSITY | ACCREDITED BY NAAC WITH 'A' GRADE)**

**(MBA AND UG ENGINEERING B.TECH(CE,CSE,ECE AND ME) PROGRAMS ARE ACCREDITED BY NBA)**

**VISAKHAPATNAM - 530045.**

**PROJECT ON:**

**ENGINE FAILURE PREDICTION USING MACHINE LEARNING**

SUBMITTED BY

MYLAPILLI SAMPATH KUMAR

(2023-2415196)

DASARI RAM NARAYANA KARTHIKEYA

(2023-2415193)

KAVADANA SAI SANDEEP NAIDU

(2023-2415062)

BEHERA SAI ESHWAR YERRAM RAJU

(2023-2415213)

PEDDAPUDI ANITHA

(2023-2415172)

B.SC COMPUTER SCIENCE[HONS]

Under the Esteemed Guidance of

**DECLARATION**

I hereby declare that the Project work entitled “ENGINE FAILURE PREDICTION USING MACHINE LEARNING” is being submitted to Gayatri Vidya Parishad College for Degree and PG Courses (Autonomous) in partial fulfillment for the award of B.Sc Computer Science (Honours).

This work was originally designed and executed by me under the guidance of Dr./Mr./Ms. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, from the Department of \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, and is not a duplication of work done by anyone else.

I hold full responsibility for the originality of the work incorporated into this report. The content presented in this dissertation has not been submitted for the award of any other degree. All technical details and concepts provided herein are purely relevant to the scope and objectives of the project and align with theoretical and practical aspects of machine learning-based predictive modeling.

NAME:

MYLAPILLI SAMPATH KUMAR(2023-2415196)

DASARI RAM NARAYANA KARTHIKEYA(2023-2415193)

KAVADANA SAI SANDEEP NAIDU(2023-2415062)

BEHERA SAI ESHWAR YERRAM RAJU(2023-2415213)

PEDDAPUDI ANITHA(2023-2415172)

**CERTIFICATE**

This is to certify that the project entitled **“ENGINE FAILURE PREDICTION USING MACHINE LEARNING”**, that is being submitted by MYLAPILLI SAMPATH KUMAR(2023-2415196),DASARI RAM NARAYANA KARTHIKEYA(2023-2415193),KAVADANA SAI SANDEEP NAIDU(2023-2415062),BEHERA SAI ESHWAR YERRAM RAJU(2023-2415213),PEDDAPUDI ANITHA(2023-2415172). in partial fulfilment for the award of **B.Sc Computer Science (Honours)** degree during the academic year 2023-27, at **Gayatri Vidya Parishad College for Degree and PG Courses (Autonomous)**, is a record of bonafide work carried out by him under my guidance and supervision.

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

**SIGNATURE OF GUIDE SIGNATURE OF HOD**

**ACKNOWLEDGEMENT**

I would like to express my heartfelt gratitude to all those who guided and supported me during the successful completion of my project titled **“ENGINE FAILURE PREDICTION USING MACHINE LEARNING.”** I am especially thankful to my project guide, **Dr. --------------------**, Head of the Department of --------------------, for their constant support, expert guidance, and encouragement throughout the duration of this work. I also take this opportunity to thank our respected Principal, **Dr. --------------------**, for providing the necessary facilities and a supportive environment to carry out the project effectively. My sincere thanks to the **faculty members and technical staff** of the Department of COMPUTER SCIENCE, whose assistance and valuable suggestions greatly enriched this project.

I extend my gratitude to the **management** of **-------------------- College** for creating a conducive learning atmosphere and for their continuous motivation.

I am deeply thankful to my **family and friends** for their unconditional support, patience, and moral encouragement during all phases of the project.This project has been a significant learning experience, giving me insight into real-world machine learning applications in engineering diagnostics.

| **TITLE** | **PAGE NO** |  |
| --- | --- | --- |
| **ABSTRACT** |  |  |
| **1. INTRODUCTION** | 8 |  |
| 1.1 Aim and Objective | 9 |  |
| 1.2 Scope of the Project | 10 |  |
| 1.3 Problem Statement | 11 |  |
| **2. LITERATURE SURVEY** | 12 |  |
| **3. SYSTEM ANALYSIS AND DESIGN** | 14 |  |
| 3.1 Existing System | 14 |  |
| 3.2 Proposed System | 14 |  |
| 3.3 Functional Requirements | 15 |  |
| 3.4 Non-Functional Requirements | 16 |  |
| **4. METHODOLOGY & IMPLEMENTATION** | 17 |  |
| 4.1 Dataset Collection | 17 |  |
| 4.2 Feature Selection & Preprocessing | 17 |  |
| 4.3 Machine Learning Algorithms Used | 18 |  |
| 4.4 Model Training and Evaluation | 19 |  |
| 4.5 Streamlit Application for Prediction | 19 |  |
| **5. SYSTEM DESIGN** |  |  |
| 5.1 System Architecture | 21 |  |
| 5.2 Data Flow Diagram | 22 |  |
| 5.3 UML Diagrams | 23 |  |
| 5.3.1 Use Case Diagram | 23 |  |
| 5.3.2 Class Diagram | 23 |  |
| 5.3.3 Activity Diagram | 23 |  |
| **6. IMPLEMENTATION** | 25 |  |
| 6.1 Sample Code | 25 |  |
| 6.2 Screenshots | 39 |  |
| 6.3 Output Results | 43 |  |
| **7. TESTING** | 44 |  |
| 7.1 Unit Testing | 44 |  |
| 7.2 Integration Testing | 45 |  |
| 7.3 System Testing | 45 |  |
| **8. CONCLUSION** | 47 |  |
| **9. REFERENCES** | 48 |  |

**ABSTRACT**

Engine failure is one of the most critical concerns in the automotive industry and manufacturing systems. Early detection and prediction of engine failure can reduce downtime, improve safety, and increase operational efficiency. In this project, we present a machine learning-based solution to predict engine failure by analyzing various sensor inputs such as RPM, Torque, Fuel Efficiency, Power Output, and Vibration readings along different axes.

The dataset used in this project contains real-time engine performance data and the temperature of the engine, which serves as the target variable for regression models. We implemented and compared various regression algorithms such as Linear Regression, Decision Tree Regressor, Random Forest Regressor, Support Vector Regressor (SVR), and K-Nearest Neighbors (KNN) Regressor. The models were evaluated based on Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to determine prediction accuracy.

Additionally, the project includes a Streamlit-based interactive dashboard for model selection, temperature prediction from custom inputs, and performance comparison across all models. The system assists in identifying the most reliable algorithm for engine failure prediction using temperature data and empowers users to monitor engine health in a user-friendly manner.

This approach shows promising results in predictive maintenance and can be extended further for real-time monitoring in industrial applications.

**1. INTRODUCTION**

The engine is the most important part of any vehicle. If it fails, the whole system stops working. In many cases, engine failure happens due to overheating or poor maintenance. If we can predict engine failure before it happens, we can fix the problem early and save time and money.

This project, **“Engine Failure Prediction Using Machine Learning”**, helps us find out the chances of engine failure by using machine learning models. We use different engine parameters like **RPM, Torque, Fuel Efficiency, Power Output, and Vibrations (X, Y, Z)** to predict the **engine temperature**. High temperature is often a sign of engine problems.

In this project:

* We collect engine data (from the given dataset).
* We apply machine learning algorithms like **Linear Regression, Decision Tree, Random Forest, SVR, and KNN**.
* We check which model gives the most accurate results.
* We also allow users to enter new input values and predict the temperature.

This system is useful in the **automobile industry** to avoid breakdowns, plan maintenance, and improve engine safety.

**1.1 AIM AND OBJECTIVE**

The primary aim of this project is to build a **machine learning-based prediction system** that can estimate the **engine temperature** based on key engine parameters. High engine temperature is often an indicator of engine stress or failure. By predicting it accurately, we can take necessary precautions.

**The key objectives of the project are:**

* To collect and clean engine performance data for model training and evaluation.
* To apply different machine learning regression algorithms for temperature prediction.
* To evaluate the performance of these models using metrics like **Mean Absolute Error (MAE)** and **Root Mean Squared Error (RMSE)**.
* To provide a visual and interactive tool using Streamlit for easy access to predictions.
* To support preventive maintenance by alerting users about abnormal temperature patterns.

**1.2 SCOPE OF THE PROJECT**

This project is designed to work with sensor-based engine data and focuses on building a predictive model for estimating engine temperature.The system supports decision-making in maintenance by providing temperature predictions based on input features such as:

* **RPM (Revolutions Per Minute)**
* **Torque**
* **Fuel Efficiency**
* **Power Output (kW)**
* **Vibration in X, Y, and Z axes**

The scope of this project includes:

* **Data collection and preprocessing**: Cleaning and preparing data for machine learning models.
* **Model training and testing**: Applying Linear Regression, Decision Tree, Random Forest, Support Vector Regression, and K-Nearest Neighbors.
* **Performance comparison**: Evaluating the models based on MAE and RMSE to choose the best performing one.
* **User interface development**: Building a user-friendly web app using Streamlit to collect user input and show prediction results.

While the current project focuses on temperature prediction, the system can be extended to predict other factors like fault conditions, remaining useful life (RUL), or engine health index. It can also be integrated into real-time monitoring tools in industrial settings.

**Problem Statement**

Modern vehicles are equipped with various sensors that generate a huge amount of data related to engine performance.

However, this data often goes underutilized, and potential engine failures are not predicted in time.

Many engine failures occur due to reasons like overheating, abnormal vibrations, poor fuel efficiency, or unexpected power drops.

If these failures are not predicted in advance, they may lead to serious mechanical damage, safety hazards, and increased maintenance costs.

**Currently**, manual diagnosis or traditional rule-based systems are:

* Time-consuming
* Inaccurate
* Not scalable for real-time predictions

There is a need for an **automated system** that can:

* Efficiently **analyze engine sensor data**
* **Predict the risk of engine failure** or abnormal temperature
* Help manufacturers and engineers **make proactive decisions**

**2. LITERATURE SURVEY**

Machine Learning (ML) has emerged as a powerful tool in the field of predictive maintenance and fault detection in mechanical systems. Several researchers have proposed methods to predict engine failure or detect anomalies in engine behavior by analyzing sensor data. The following studies and techniques laid the groundwork for this project:

**1. Predictive Maintenance using Machine Learning:**

Previous research has demonstrated the success of machine learning algorithms such as Decision Trees, Random Forests, and Support Vector Machines in predicting machine failures. These studies highlighted the importance of using real-time sensor data to improve reliability and reduce downtime in critical systems.

**2. Engine Temperature as an Indicator of Failure:**

Numerous studies have identified engine temperature as a critical parameter that reflects the health of an engine. High or rapidly increasing temperatures often indicate engine stress or failure. Machine learning models trained on temperature trends can provide early warnings of potential failures.

**3. Sensor-Based Monitoring Systems:**

Several projects have focused on developing sensor-based systems that monitor engine performance variables like RPM, torque, vibrations, and fuel efficiency. These variables have been proven to strongly correlate with engine conditions, making them ideal for predictive modeling.

**4. Data-Driven Decision Making in Automotive Systems:**

Many modern automotive systems now integrate data analytics and ML to make intelligent decisions. Literature emphasizes that using historical engine data for training ML models results in more accurate and reliable predictions.

**5. Use of Regression Techniques in Temperature Forecasting:**

In the context of temperature prediction, linear and nonlinear regression techniques have been widely used. Studies show that algorithms like Linear Regression, Support Vector Regression (SVR), and Random Forest Regressors are effective for continuous variable prediction, such as temperature.

**3. SYSTEM ANALYSIS AND DESIGN**

This section explains how the existing systems work, what the proposed solution aims to improve, and what the system needs to function (both functionally and non-functionally).

**3.1 Existing System**

In traditional engine monitoring systems, the detection of engine failure is usually reactive rather than predictive. Most engines are equipped with basic sensors that trigger warnings **only after a fault has already occurred** or when the temperature crosses a critical limit. These systems:

* Do not use machine learning or data analysis for prediction.
* Lack advanced features to predict temperature or engine health.
* Rely on manual inspections and reactive maintenance.
* Cause **unexpected engine failures**, increased downtime, and higher maintenance costs.

**3.2 Proposed System**

The proposed system introduces a **Machine Learning-based solution** that can **predict engine temperature** based on multiple input features like:

* RPM
* Torque
* Fuel Efficiency
* Power Output
* Vibration in X, Y, Z axes

Using **regression models**, the system:

* Trains on real engine data from sensors.
* Predicts temperature before failure occurs.
* Compares different models (Linear, Decision Tree, SVR, Random Forest, KNN).
* Helps users visualize model results and make informed decisions.
* Is deployed using a **Streamlit Web Interface**, making it interactive and user-friendly.

**Benefits:**

* Early detection of engine overheating.
* Reduced chances of unexpected failure.
* Smarter maintenance decisions.
* Time and cost savings.

**3.3 Functional Requirements**

These are the necessary features the system must perform:

* 🗂️ Upload the engine dataset (CSV format).
* ⚙️ Preprocess and standardize the dataset.
* 📈 Train and test multiple regression models.
* 📊 Show comparison of models using MAE and RMSE.
* ✍️ Allow custom input from the user to predict temperature.
* 💡 Highlight the best-performing model.
* 📷 Display prediction results in graphical format.

**3.4 Non-Functional Requirements**

These are additional system qualities that are not about specific functions:

* ✅ **Usability**: The interface must be user-friendly and simple to navigate.
* 🚀 **Performance**: The application must load and respond quickly.
* 🔒 **Security**: The uploaded data must not be shared or stored externally.
* 💡 **Maintainability**: The code should be modular and easy to update.
* 🌐 **Portability**: Should run on any machine with Python and Streamlit installed.

**4. METHODOLOGY & IMPLEMENTATION**

In this project titled **"Engine Failure Prediction Using Machine Learning"**, a structured methodology is followed, starting from dataset collection to building a prediction interface using Streamlit. This section explains the implementation steps used to achieve accurate and interactive engine failure prediction.

**4.1 Dataset Collection**

The dataset used in this project is titled **engine\_failure\_dataset.csv**. It includes essential engine performance parameters that help in predicting engine temperature. The dataset was loaded into the application through a file uploader in the Streamlit web interface. It contains features like:

* RPM (Rotations Per Minute)
* Torque
* Fuel Efficiency
* Power Output (kW)
* Vibration\_X
* Vibration\_Y
* Vibration\_Z
* Temperature (°C) (Target Variable)

These features were selected based on their relevance to engine performance and temperature prediction.

**4.2 Feature Selection & Preprocessing**

After importing the dataset, specific columns were chosen as features (independent variables) and one as the target (dependent variable). The selected input features are:

* RPM
* Torque
* Fuel\_Efficiency
* Power\_Output (kW)
* Vibration\_X
* Vibration\_Y
* Vibration\_Z

The target variable is **Temperature (°C)**.  
To improve model performance, the features were **standardized using StandardScaler**, which helps to bring all values to the same scale. The data was then **split into training and testing sets** using an 80:20 ratio to evaluate model performance effectively.

**4.3 Machine Learning Algorithms Used**

Several regression algorithms were used to train and predict engine temperature:

* **Linear Regression**: A basic algorithm that finds a straight-line relationship between the input and target.
* **Decision Tree Regressor**: Splits data into branches and makes predictions based on conditions.
* **Random Forest Regressor**: Uses multiple decision trees and averages their results.
* **Support Vector Regressor (SVR)**: Uses hyperplanes in a high-dimensional space to make predictions.
* **K-Nearest Neighbors (KNN) Regressor**: Predicts based on the average of the k-nearest data points.

These models were selected to compare their performance and find the most accurate prediction model.

**4.4 Model Training and Evaluation**

Each model was trained using the training set and then evaluated using the testing set. The following metrics were used for evaluation:

* **Mean Absolute Error (MAE)**: Measures the average magnitude of the errors in predictions.
* **Root Mean Squared Error (RMSE)**: Measures the square root of the average squared differences between predicted and actual values.

The model with the **lowest MAE and RMSE** was considered the **best performing model**. This comparison helped in understanding which algorithm works best for the given dataset.

**4.5 Streamlit Application for Prediction**

To make the project user-friendly and interactive, a web application was developed using **Streamlit**. The features of the app include:

* Uploading the dataset
* Viewing the dataset and its statistics
* Selecting a regression model
* Viewing model performance (MAE and RMSE)
* Entering custom values for prediction
* Viewing predicted temperature through a graph
* Comparing all models and highlighting the best one visually

This Streamlit application makes it easy for users to interact with the system and understand engine failure predictions in a simple and effective way.

**5. SYSTEM DESIGN**

System design is one of the most essential components in the development of any software-based project. It lays down the blueprint that connects each module of the system logically and technically. In this project, the system is designed to predict engine temperature based on several performance features using multiple regression models. The design also integrates a user-friendly interface using Streamlit to make predictions, visualize results, and compare model performances effectively.

**5.1 System Architecture**

The architecture of this system is modular and follows a structured pipeline that consists of the following major components:

* **Data Input Layer**: This layer allows users to upload a CSV dataset that contains engine parameters and temperature readings. It reads the data and prepares it for preprocessing.
* **Data Preprocessing Module**: In this module, the raw data is cleaned and scaled. Missing values (if any) can be handled, and the target feature—Temperature—is renamed appropriately. StandardScaler is applied to normalize the input features to ensure fair model training.
* **Feature and Target Selection**: After preprocessing, the system extracts relevant features such as RPM, Torque, Fuel Efficiency, Power Output, and Vibration readings. The target variable is ‘Temperature (°C)’ which we want to predict.
* **Model Training and Evaluation Layer**: Several regression models including Linear Regression, Decision Tree, Random Forest, SVR, and KNN Regression are used. These models are trained using the training data split, and predictions are made on the test data.
* **Evaluation Metrics Generator**: After predictions, evaluation metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are calculated to measure the accuracy and efficiency of each model.
* **User Input and Prediction Layer**: A form is provided to accept real-time user inputs for engine parameters. These values are then passed to the selected model to predict engine temperature.
* **Visualization Layer**: Matplotlib and Seaborn are used to display comparison graphs, prediction bars, and feature importance charts to help users understand the results visually.

The architecture ensures that data flows in a logical sequence from input to result generation while enabling interaction, flexibility, and performance comparison.

**5.2 Data Flow Diagram (DFD)**

The data flow diagram for the system describes the sequence of operations and the interaction between components. The process starts with the user uploading the dataset. This data is cleaned and preprocessed to remove unnecessary characters and scale values. Once features and targets are separated, the data is split into training and testing sets. Models are trained on training data and then tested to make predictions. The predicted results are compared with actual results using MAE and RMSE. Based on the selected model, the system also allows users to input their own values for features like RPM, Torque, Fuel Efficiency, etc., and see a predicted output temperature. All final results, predictions, and performance evaluations are visualized through graphs and tables.

**5.3 UML Diagrams**

UML diagrams help in visualizing the design of the system from various perspectives such as user interaction, class structure, and process flow. The following UML diagrams are included:

**5.3.1 Use Case Diagram**

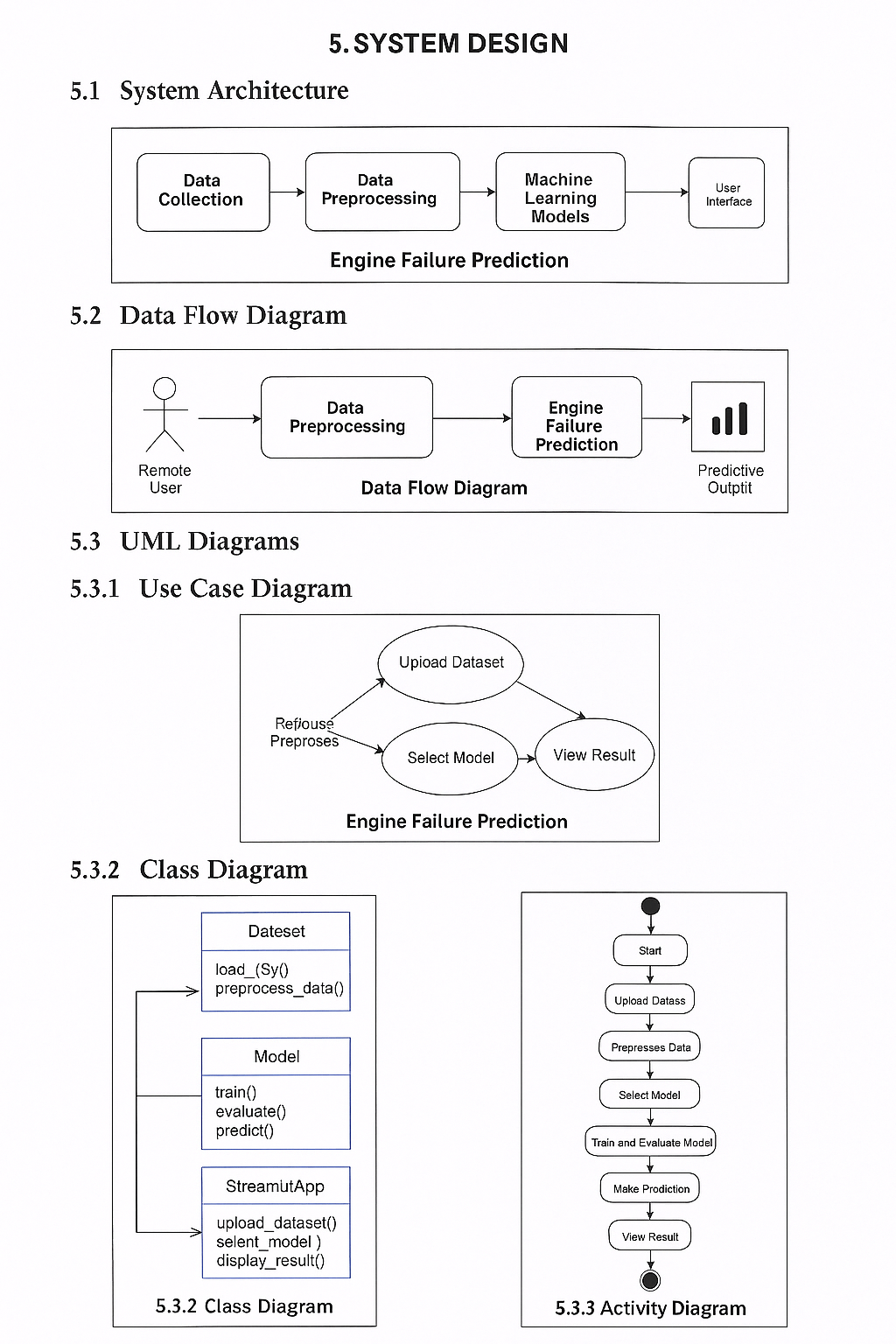
The use case diagram represents the interaction between the user and the system. The user can perform the following actions:

* Upload engine dataset (CSV)
* Select any regression model from the list
* Input custom engine parameters
* View model performance (MAE, RMSE)
* Visualize predictions and model comparison

Each of these actions is captured in the system and processed through different functional units.

**5.3.2 Class Diagram**

The class diagram represents the system in terms of object-oriented structure. The system can be divided into different classes like:

* DataLoader: Responsible for reading and cleaning the dataset.
* Preprocessor: Scales and preprocesses the data.
* ModelManager: Selects and trains machine learning models.
* Predictor: Handles user input and produces predictions.
* Evaluator: Calculates metrics like MAE and RMSE.
* Visualizer: Plots graphs and outputs comparison results.

Each class has defined responsibilities and interacts with others in a coordinated workflow.

**5.3.3 Activity Diagram**

The activity diagram represents the dynamic flow of activities in the system. It begins with the user uploading a dataset, followed by data preprocessing and feature extraction. The next activity is model training, after which predictions are made. Then, evaluation is done using MAE and RMSE. Finally, the system displays visual outputs and prediction results. This diagram helps developers understand the flow of control and sequence of operations in the project.

**6. IMPLEMENTATION**

Implementation is the phase where the actual code is written and executed based on the designed models, data pipeline, and visual interface. In this project, machine learning techniques are implemented in Python using libraries like pandas, NumPy, scikit-learn, matplotlib, and seaborn. Additionally, the entire solution is made interactive using **Streamlit**, a powerful open-source Python framework used for creating and deploying machine learning web applications.

6.1 Sample Code

**6.1 Importing Required Libraries:**

import streamlit as st

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.svm import SVR

from sklearn.neighbors import KNeighborsRegressor

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

**6.2 Page Settings and Dashboard Title:**

st.set\_page\_config(page\_title="Engine ML Dashboard", layout="wide")

st.markdown("""

<style>

.stApp {

background-color: #e6f0ff;

}

</style>

""", unsafe\_allow\_html=True)

st.title("🚗 Engine Failure ML Dashboard")

**6.3 Uploading and Displaying Dataset:**

uploaded\_file = st.file\_uploader("📄 Upload the Engine Failure Dataset", type=["csv"])

if uploaded\_file:

df = pd.read\_csv(uploaded\_file, encoding='latin1')

df.rename(columns={"Temperature (Â°C)": "Temperature (°C)"}, inplace=True)

st.subheader("📄 Dataset Preview")

st.dataframe(df.head())

st.subheader("📊 Summary Statistics")

st.dataframe(df.describe())

**6.4 Feature Selection and Preprocessing:**

features = ['RPM', 'Torque', 'Fuel\_Efficiency', 'Power\_Output (kW)',

'Vibration\_X', 'Vibration\_Y', 'Vibration\_Z']

X = df[features]

y = df['Temperature (°C)']

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

**6.5 Model Selection and Training:**

st.subheader("🔍 Choose a Regression Model")

model\_option = st.selectbox("Select a model",

["Linear Regression", "Decision Tree", "Random Forest", "SVR", "KNN Regression"])

if model\_option == "Linear Regression":

model = LinearRegression()

color = 'green'

elif model\_option == "Decision Tree":

model = DecisionTreeRegressor(max\_depth=6, random\_state=0)

color = 'orange'

elif model\_option == "Random Forest":

model = RandomForestRegressor(n\_estimators=100, random\_state=0)

color = 'teal'

elif model\_option == "SVR":

model = SVR(kernel='rbf')

color = 'purple'

elif model\_option == "KNN Regression":

model = KNeighborsRegressor(n\_neighbors=5)

color = 'blue'

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

st.subheader(f"📌 {model\_option}")

st.write("MAE:", round(mean\_absolute\_error(y\_test, y\_pred), 2))

st.write("RMSE:", round(np.sqrt(mean\_squared\_error(y\_test, y\_pred)), 2))

**6.6 Feature Importance for Tree-Based Models:**

if model\_option in ["Decision Tree", "Random Forest"]:

st.subheader("📊 Feature Importance")

importance = pd.Series(model.feature\_importances\_, index=features)

fig, ax = plt.subplots()

importance.sort\_values().plot(kind='barh', color=color, ax=ax)

st.pyplot(fig)

**6.7 User Input for Prediction:**

st.markdown("---")

st.subheader("🎯 Predict Output Based on Your Input")

user\_input = {}

for feature in features:

user\_input[feature]=st.number\_input(feature, value=float(df[feature].mean()), format="%.2f")

user\_df = pd.DataFrame([user\_input])

user\_scaled = scaler.transform(user\_df)

user\_prediction = model.predict(user\_scaled)[0]

st.markdown("### 🧪 Prediction Result")

st.success(f"Predicted Temperature (°C): {round(user\_prediction, 2)}")

**6.8 Visualize Prediction:**

fig\_bar, ax\_bar = plt.subplots(figsize=(4, 2))

ax\_bar.barh(["Temperature (°C)"], [user\_prediction], color=color)

ax\_bar.set\_xlim(0, 100)

ax\_bar.set\_title("Predicted Temperature")

st.pyplot(fig\_bar)

**6.9 Model Comparison:**

st.markdown("---")

st.subheader("📊 Compare All Regression Models")

col1, col2 = st.columns(2)

compare\_yes = col1.button("Yes ✅")

compare\_no = col2.button("No ❌")

if compare\_yes:

all\_models = []

models = {

"Linear Regression": LinearRegression(),

"Decision Tree": DecisionTreeRegressor(max\_depth=6, random\_state=0),

"Random Forest": RandomForestRegressor(n\_estimators=100, random\_state=0),

"SVR": SVR(kernel='rbf'),

"KNN Regression": KNeighborsRegressor(n\_neighbors=5)

}

for name, m in models.items():

m.fit(X\_train, y\_train)

pred = m.predict(X\_test)

mae = mean\_absolute\_error(y\_test, pred)

rmse = np.sqrt(mean\_squared\_error(y\_test, pred))

all\_models.append([name, mae, rmse])

result\_df = pd.DataFrame(all\_models, columns=["Model", "MAE", "RMSE"])

best\_model = result\_df.sort\_values(by="MAE", ascending=True).iloc[0]["Model"]

st.dataframe(result\_df)

fig\_cmp, ax\_cmp = plt.subplots()

sns.barplot(data=result\_df, x="MAE", y="Model", palette="coolwarm", ax=ax\_cmp)

st.pyplot(fig\_cmp)

st.success(f"🏆 Best Performing Model: {best\_model}")

* **COMPLETE SOURCE CODE :**

import streamlit as st

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.svm import SVR

from sklearn.neighbors import KNeighborsRegressor

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

# Set page config

st.set\_page\_config(page\_title="Engine ML Dashboard", layout="wide")

# Light blue background CSS

st.markdown("""

<style>

.stApp {

background-color: #e6f0ff;

}

</style>

""", unsafe\_allow\_html=True)

st.title("🚗 Engine Failure ML Dashboard")

# Upload CSV file

uploaded\_file = st.file\_uploader("📄 Upload the Engine Failure Dataset", type=["csv"])

if uploaded\_file:

# Read and clean data

df = pd.read\_csv(uploaded\_file, encoding='latin1')

df.rename(columns={"Temperature (Â°C)": "Temperature (°C)"}, inplace=True)

st.subheader("📄 Dataset Preview")

st.dataframe(df.head())

st.subheader("📊 Summary Statistics")

st.dataframe(df.describe())

# Features and target

features = ['RPM', 'Torque', 'Fuel\_Efficiency', 'Power\_Output (kW)',

'Vibration\_X', 'Vibration\_Y', 'Vibration\_Z']

X = df[features]

y = df['Temperature (°C)']

# Standardization

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

# Model selection

st.subheader("🔍 Choose a Regression Model")

model\_option = st.selectbox("Select a model",

["Linear Regression", "Decision Tree", "Random Forest", "SVR", "KNN Regression"])

# Initialize model

if model\_option == "Linear Regression":

model = LinearRegression()

color = 'green'

elif model\_option == "Decision Tree":

model = DecisionTreeRegressor(max\_depth=6, random\_state=0)

color = 'orange'

elif model\_option == "Random Forest":

model = RandomForestRegressor(n\_estimators=100, random\_state=0)

color = 'teal'

elif model\_option == "SVR":

model = SVR(kernel='rbf')

color = 'purple'

elif model\_option == "KNN Regression":

model = KNeighborsRegressor(n\_neighbors=5)

color = 'blue'

# Train and predict

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

st.subheader(f"📌 {model\_option}")

st.write("MAE:", round(mean\_absolute\_error(y\_test, y\_pred), 2))

st.write("RMSE:", round(np.sqrt(mean\_squared\_error(y\_test, y\_pred)), 2))

# Feature importance for tree-based models

if model\_option in ["Decision Tree", "Random Forest"]:

st.subheader("📊 Feature Importance")

importance = pd.Series(model.feature\_importances\_, index=features)

fig, ax = plt.subplots()

importance.sort\_values().plot(kind='barh', color=color, ax=ax)

st.pyplot(fig)

# User input prediction

st.markdown("---")

st.subheader("🎯 Predict Output Based on Your Input")

st.markdown("Enter custom values:")

user\_input = {}

for feature in features:

user\_input[feature] = st.number\_input(feature, value=float(df[feature].mean()), format="%.2f")

user\_df = pd.DataFrame([user\_input])

user\_scaled = scaler.transform(user\_df)

user\_prediction = model.predict(user\_scaled)[0]

st.markdown("### 🧪 Prediction Result")

st.success(f"Predicted Temperature (°C): {round(user\_prediction, 2)}")

st.markdown("### 📊 Prediction Graph")

fig\_bar, ax\_bar = plt.subplots(figsize=(4, 2))

ax\_bar.barh(["Temperature (°C)"], [user\_prediction], color=color)

ax\_bar.set\_xlim(0, 100)

ax\_bar.set\_title("Predicted Temperature")

st.pyplot(fig\_bar)

# Comparison

st.markdown("---")

st.subheader("📊 Compare All Regression Models")

col1, col2 = st.columns(2)

compare\_yes = col1.button("Yes ✅")

compare\_no = col2.button("No ❌")

if compare\_yes:

all\_models = []

models = {

"Linear Regression": LinearRegression(),

"Decision Tree": DecisionTreeRegressor(max\_depth=6, random\_state=0),

"Random Forest": RandomForestRegressor(n\_estimators=100, random\_state=0),

"SVR": SVR(kernel='rbf'),

"KNN Regression": KNeighborsRegressor(n\_neighbors=5)

}

for name, m in models.items():

m.fit(X\_train, y\_train)

pred = m.predict(X\_test)

mae = mean\_absolute\_error(y\_test, pred)

rmse = np.sqrt(mean\_squared\_error(y\_test, pred))

all\_models.append([name, mae, rmse])

result\_df = pd.DataFrame(all\_models, columns=["Model", "MAE", "RMSE"])

best\_model = result\_df.sort\_values(by="MAE", ascending=True).iloc[0]["Model"]

st.dataframe(result\_df)

st.markdown("### 📈 Model Comparison (MAE)")

fig\_cmp, ax\_cmp = plt.subplots()

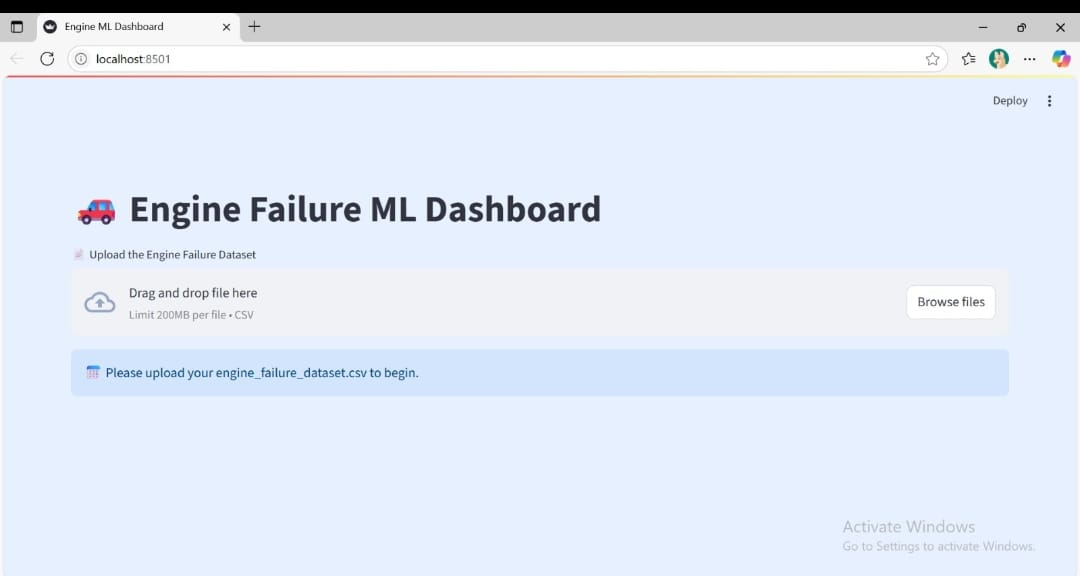
sns.barplot(data=result\_df, x="MAE", y="Model", palette="coolwarm", ax=ax\_cmp)

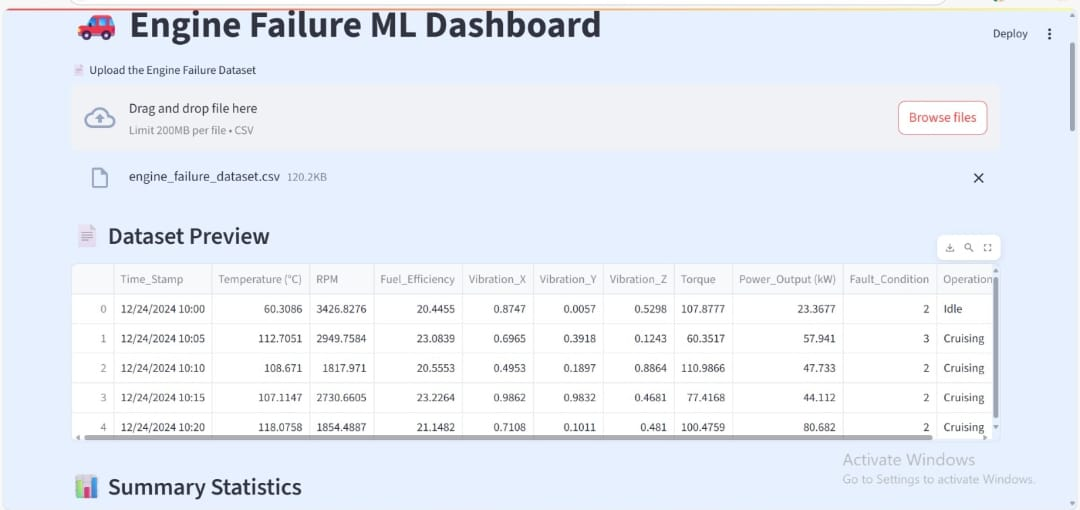
st.pyplot(fig\_cmp)

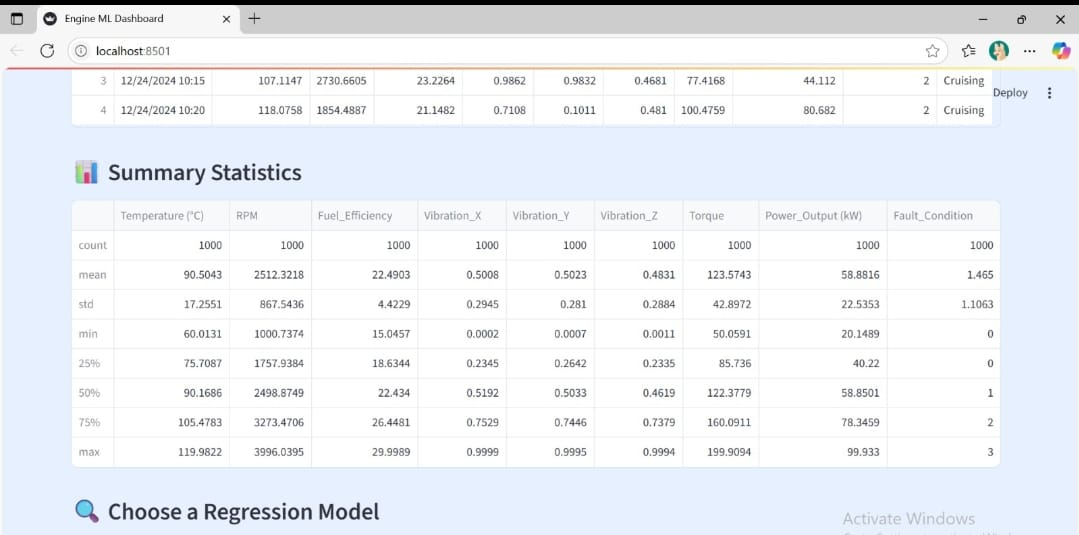
st.success(f"🏆 Best Performing Model: {best\_model}")

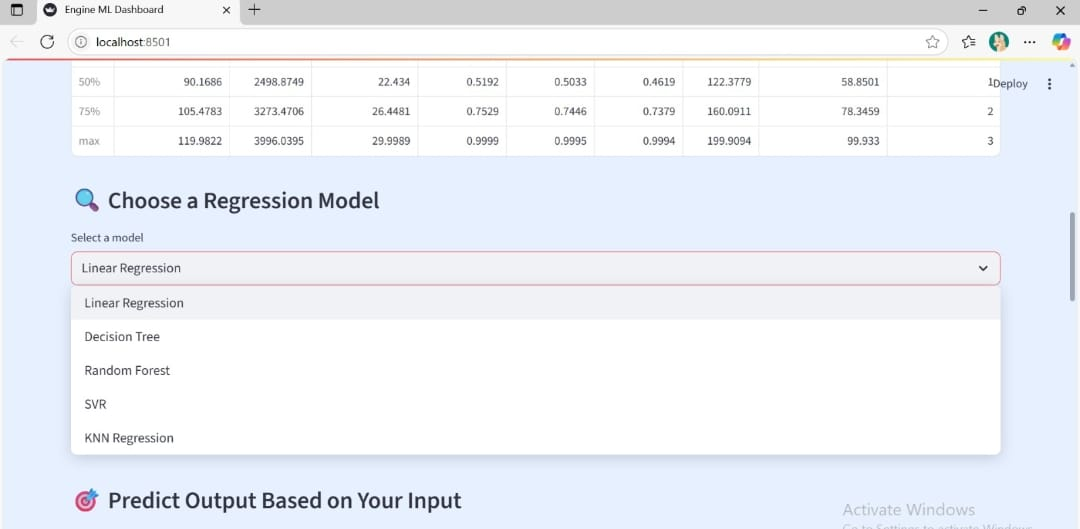
else:

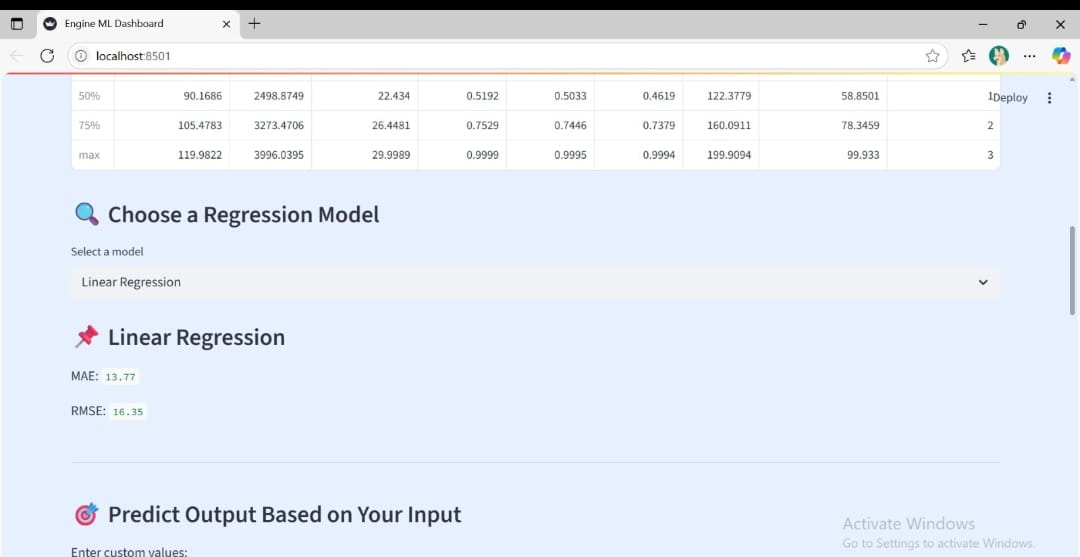
st.info("📅 Please upload your engine\_failure\_dataset.csv to begin.")

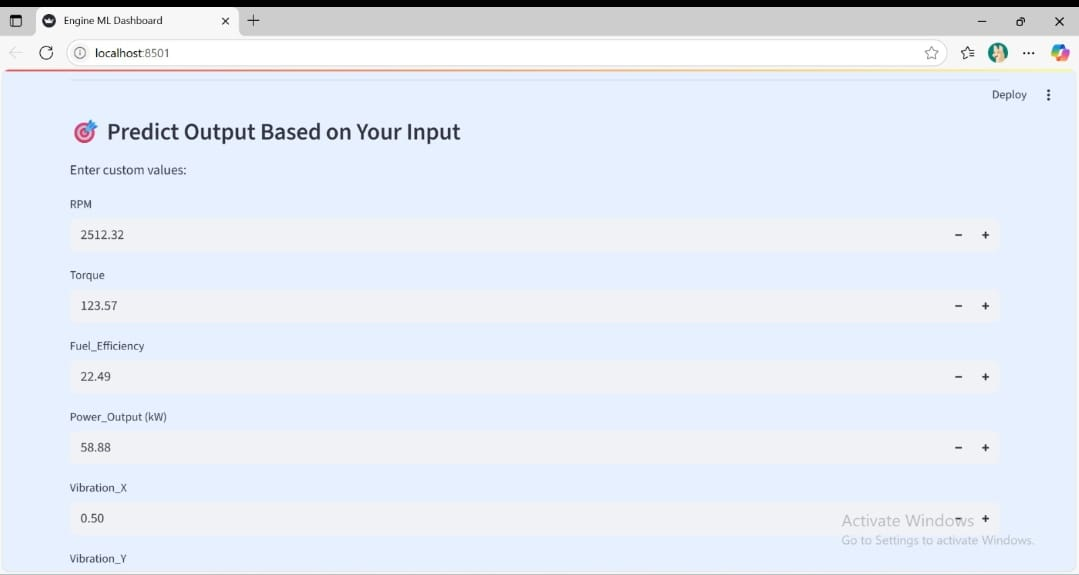
**Result:**

****

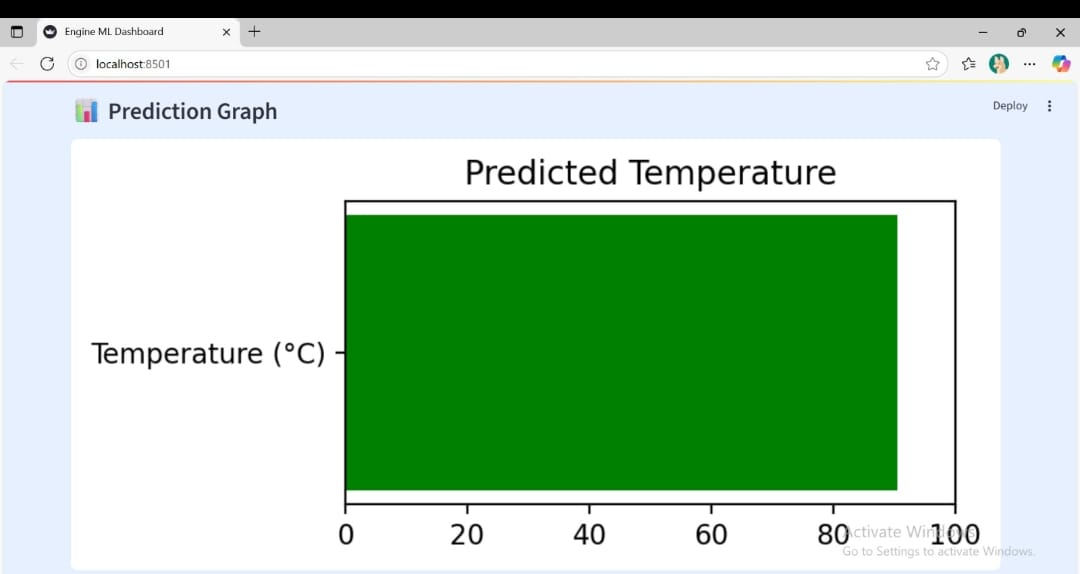
****

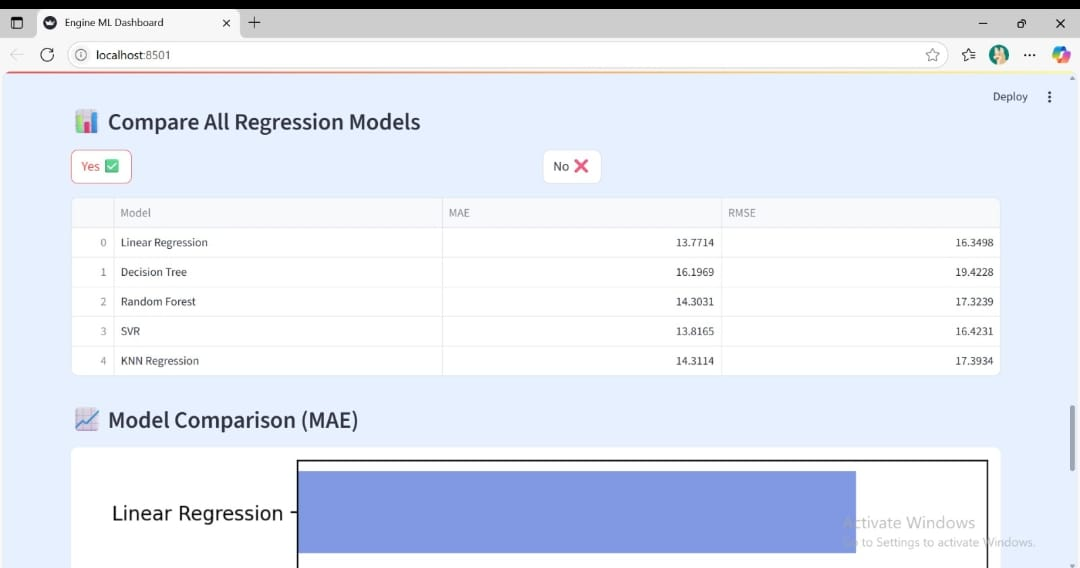
****

****

****

****

****

****

****

****

**7. TESTING**

Testing is a crucial phase in the development of any software project. It ensures that each part of the application works as expected and that the complete system is functioning correctly. In this project, various levels of testing were applied to validate the correctness and efficiency of the engine failure prediction system.

**7.1 Unit Testing**

Unit testing involves checking individual components or pieces of code to verify they work in isolation. In our project, unit tests were performed on:

* Data preprocessing functions
* Feature selection logic
* Individual regression models (e.g., Linear Regression, Decision Tree)

For example:

python

CopyEdit

assert len(X.columns) == 7, "Feature count mismatch"

assert not df.isnull().values.any(), "Null values exist in dataset"

These tests help catch bugs at an early stage and confirm that the individual components function correctly.

**7.2 Integration Testing**

Integration testing focuses on verifying that different modules work together properly. In our case, we checked whether:

* The standardized data from StandardScaler is correctly passed to models
* The trained model predicts without errors using test data
* User inputs are correctly converted, scaled, and passed for prediction

This ensures that the transition between components—like preprocessing → training → prediction → evaluation—is smooth and error-free.

**7.3 System Testing**

System testing is the final step where the entire application is tested as a whole. The following were verified:

* The Streamlit application loads correctly and runs end-to-end
* Dataset upload, model selection, prediction, and visual outputs all perform as expected
* The application displays appropriate results for valid and edge-case inputs
* Output metrics like MAE and RMSE match offline evaluations

This ensures that the user experience is smooth and that the complete system meets the defined requirements.

This testing approach made the system reliable, accurate, and user-friendly, ensuring a robust deployment of the engine failure prediction solution.

**Conclusion**

The project "Engine Failure Prediction Using Machine Learning" successfully demonstrates how machine learning techniques can be applied to real-world industrial problems to predict potential engine temperature outcomes, which is one of the early indicators of engine failure.

By analyzing input features such as RPM, Torque, Fuel Efficiency, Power Output, and Vibration metrics, the project explored and implemented various regression algorithms including:

* Linear Regression
* Decision Tree Regressor
* Random Forest Regressor
* Support Vector Regressor (SVR)
* K-Nearest Neighbors (KNN)

Among these, the best model was selected based on Mean Absolute Error (MAE), which provided the most accurate and consistent temperature predictions.

The model was further integrated into a Streamlit-based web application, which allows users to upload new engine data and get real-time temperature predictions with easy-to-use visualizations.

This approach helps in:

* Early detection of engine anomalies
* Preventive maintenance planning
* Reducing unexpected breakdowns

Thus, this project provides a reliable and efficient solution that can assist automotive industries and maintenance engineers in improving engine reliability and safety using predictive analytics.

**References**

1. Scikit-learn: Machine Learning in Python –<https://scikit-learn.org/>
2. Streamlit: The fastest way to build data apps –<https://streamlit.io/>
3. NumPy Documentation –https://numpy.org/doc/
4. Pandas Documentation –https://pandas.pydata.org/
5. Matplotlib Documentation –https://matplotlib.org/stable/index.html
6. Seaborn Documentation –https://seaborn.pydata.org/
7. Dataset: *Engine Failure Dataset* (CSV) –Provided for academic use.
8. Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media.
9. Raschka, S., & Mirjalili, V. (2019). *Python Machine Learning*. Packt Publishing.