Project Report

On

“Predictive Equipment Maintenance Application Using IoT Data”

Submitted in partial fulfillment of the requirements of the degree of

Master of Technology (M.Tech)

Presented by

Amit Singh Kushwaha (24204141107)

Rohit Kumar (24204041120)

Department of Mathematics, Bioinformatics & Computer Applications

MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY BHOPAL

2024-2026

ABSTRACT

This project focuses on the development and deployment of a **Predictive Maintenance System** leveraging **IoT sensor data** and **machine learning algorithms**. Industrial equipment is equipped with IoT sensors that continuously monitor parameters like temperature, vibration, and pressure. The data collected from these sensors is preprocessed to handle noise, outliers, and missing values, ensuring consistency and accuracy for analysis.

Using historical sensor data, machine learning models such as **Random Forest** and **Gradient Boosting** are trained to predict equipment failures before they occur. The trained model is deployed on a **Virtual Machine (VM)**, enabling real-time monitoring of equipment and anomaly detection. Alerts are generated for the maintenance team when anomalies or potential failures are detected, allowing proactive maintenance and minimizing unexpected downtime.

The system's deployment includes a robust data pipeline for real-time data ingestion, preprocessing, and predictions. A user-friendly interface provides visualizations and insights into equipment health, empowering decision-makers to act swiftly. The project demonstrates the potential of IoT and machine learning to reduce maintenance costs, improve operational efficiency, and enhance equipment lifespan.

This work lays the foundation for advanced predictive maintenance systems with the potential for integration with **cloud platforms**, **mobile applications**, and **streaming analytics**, paving the way for a more reliable and efficient industrial environment.

Introduction

Background of Predictive Maintenance and IoT

Predictive maintenance is a proactive approach to maintenance that aims to predict when equipment failure might occur and perform maintenance activities just before failure. IoT (Internet of Things) sensors play a key role in this, as they collect real-time data (such as temperature, pressure, vibration) from industrial machines, which is then analyzed to predict future breakdowns.

In this project, we combine IoT sensor data with machine learning models to predict equipment failures. The system helps to reduce downtime and maintenance costs by alerting maintenance teams before a failure occurs

Project Motivation and Goals

The project was initiated to create a system capable of monitoring industrial equipment and predicting maintenance requirements. The motivation is to reduce unexpected downtime in manufacturing environments, minimize maintenance costs, and improve operational efficiency.

Tools and Technologies Used

- ## Streamlit

  Streamlit is an open-source Python library that allows you to create and share beautiful, custom web applications for machine learning and data science projects with minimal coding. Its straightforward interface enables quick prototyping of interactive apps using only Python scripts, making it accessible for data scientists and developers alike. Streamlit supports a variety of widgets and integrates seamlessly with popular data manipulation and visualization libraries, offering an easy way to showcase results and build user-friendly dashboards.

- ## Pandas

  Pandas is a powerful data manipulation and analysis library for Python, widely used in data science for handling structured data. It provides data structures like DataFrames, which allow for efficient data cleaning, transformation, and analysis. Pandas excels at handling large datasets with numerous operations and complex groupings, making it indispensable for data preprocessing tasks. Its user-friendly syntax and rich functionality make it a go-to tool for analysts and researchers.

- ## Numpy

  NumPy is the fundamental package for numerical computing in Python, providing support for arrays, matrices, and a wide range of mathematical functions. It serves as the foundation for many other scientific computing libraries, enabling efficient numerical operations and matrix manipulation. NumPy's high-performance capabilities stem from its use of optimized C and Fortran libraries under the hood, making it essential for tasks that require fast numerical computations.

- ## Matplotlib

  Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Known for its flexibility, it enables users to produce publication-quality plots and complex multi-plot figures. Matplotlib works well with NumPy and Pandas, making it a versatile tool for data visualization. Its extensive customization options and support for various plot types make it a preferred choice for both simple and advanced graphical representations of data.

- ## Seeborn

  Seaborn is a statistical data visualization library built on top of Matplotlib, designed to make it easier to create attractive and informative visualizations. It provides a high-level interface for drawing complex statistical plots, with built-in themes and color palettes that improve the aesthetics of the plots. Seaborn excels in visualizing complex data relationships and distributions, making it particularly useful for exploratory data analysis and communicating insights effectively.

- ## Scikit-learn

  Scikit-learn is a versatile machine learning library for Python, offering simple and efficient tools for data mining and analysis. It provides a wide range of supervised and unsupervised learning algorithms, along with utilities for model selection, preprocessing, and evaluation. Scikit-learn's consistent API and comprehensive documentation make it easy to implement machine learning workflows, from data preprocessing to model training and validation, making it a staple in the data scientist's toolkit.

- ## Google colab

  Google Colab is a cloud-based Jupyter notebook environment that allows users to write and execute Python code in their browsers. It provides free access to powerful computing resources, including GPUs and TPUs, which are particularly useful for training machine learning models. Colab supports collaborative work and integrates seamlessly with Google Drive, making it an ideal platform for sharing projects and conducting experiments without worrying about local hardware constraints.

- ## VsCode

  Visual Studio Code (VSCode) is a popular source-code editor developed by Microsoft, known for its versatility and extensibility. It supports a wide range of programming languages and features built-in Git integration, debugging tools, and a vast library of extensions that enhance its functionality. VSCode's lightweight nature and robust feature set make it suitable for various development tasks, from scripting and web development to complex application programming, making it a favorite among developers.

- ## Docker

  Docker is a platform that enables developers to create, deploy, and run applications inside lightweight, portable containers. Containers encapsulate an application and its dependencies, ensuring consistent environments across development, testing, and production stages. Docker streamlines the development workflow, reduces compatibility issues, and enhances scalability. Its efficiency in isolating applications makes it invaluable for DevOps practices and microservices architectures.

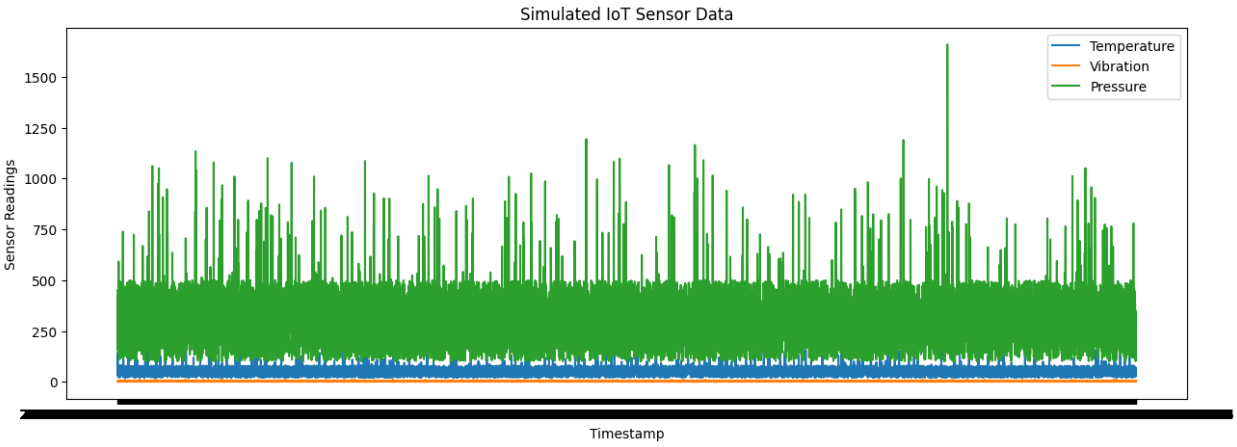
- ## Azure

  Azure is Microsoft's cloud computing platform, offering a comprehensive suite of services for building, deploying, and managing applications through Microsoft-managed data centers. Azure supports a wide range of solutions, including virtual machines, databases, machine learning, and IoT. Its integration with other Microsoft products, robust security features, and extensive global network make it a powerful choice for enterprises looking to leverage cloud computing for scalability, reliability, and innovation.

Flow Process of the System

The system follows these primary steps:

1. **Collect IoT Sensor Data:**
   Data is continuously collected from various sensors installed on industrial equipment. The data includes parameters such as temperature, pressure, vibration, and operating hours.



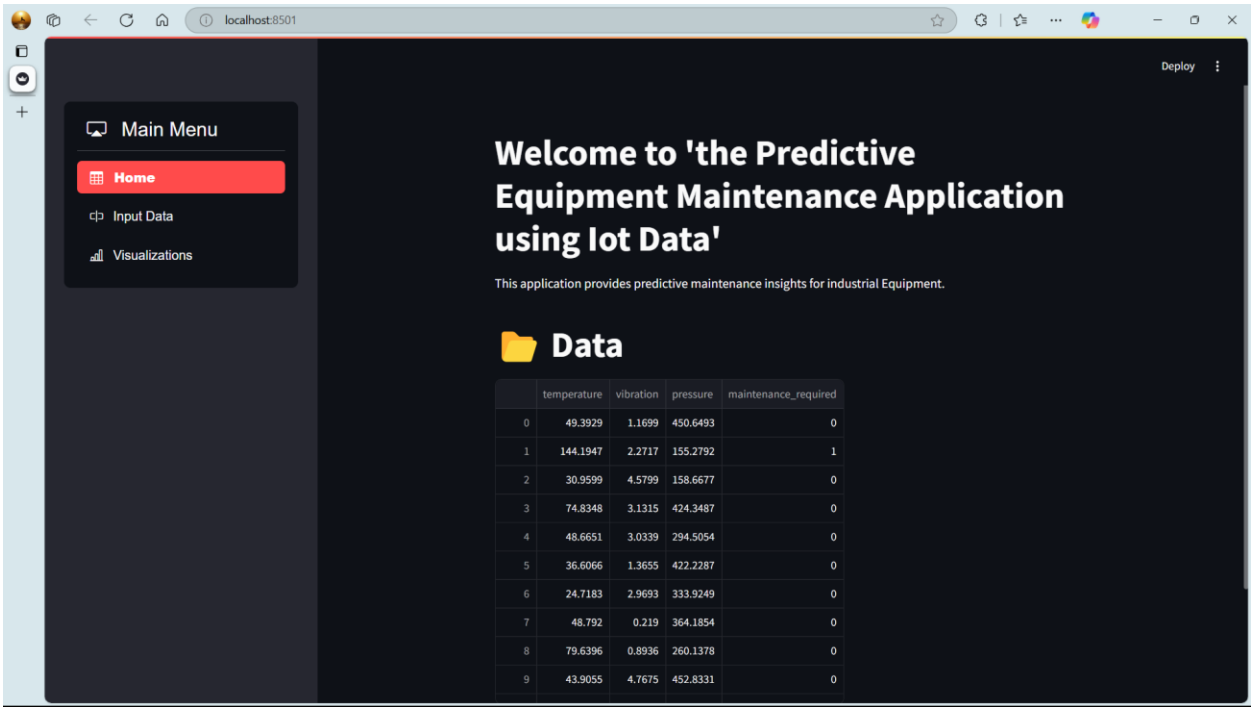
2. **Data Preprocessing:**
   The collected raw data is cleaned by handling missing values, noise, and outliers. It is then normalized and transformed to be fed into machine learning models.

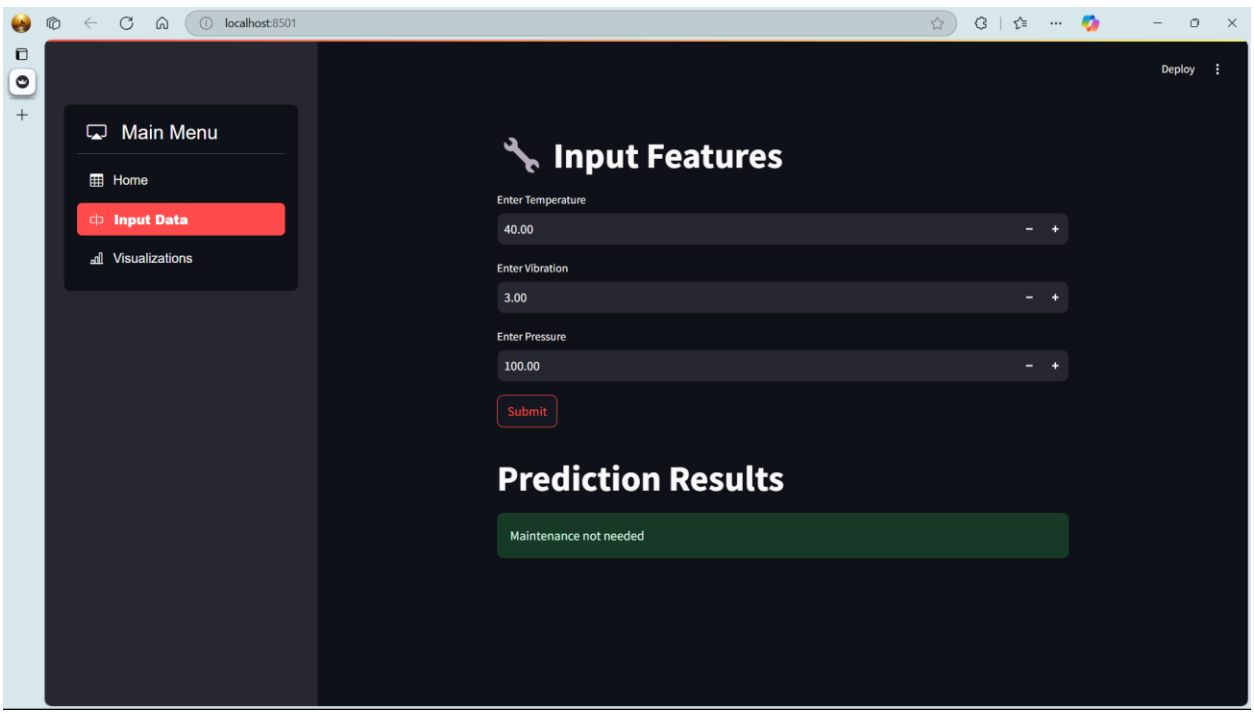
```
df = df.drop(columns=['Timestamp'])
print(df)
```

```
       Temperature (°C)  Vibration (mm/s)  Pressure (Pa)  Maintenance Required
0              0.548793          0.509840       0.619918                     1
1              0.715185          0.587573       0.802121                     1
2              0.602748          0.294453       0.965546                     1
3              0.544862          0.500041       0.519955                     0
4              0.423622          0.031649       0.323663                     0
...                 ...               ...            ...                   ...
43795          0.719535          0.251835       0.403438                     0
43796          0.883550          0.633510       0.669265                     1
43797          0.516298          0.338688       0.206265                     0
43798          0.861137          0.141975       0.368326                     0
43799          0.917329          0.794854       0.626119                     1

[43800 rows x 4 columns]
```

3. **Train Machine Learning Model**:
   A machine learning model (e.g., Random Forest or Gradient Boosting) is trained using the historical data to predict equipment failure.
4. **Make Web APP:**
   Using **Streamlit** make a user interface to use model to predict





5. **Making Image of Webapp :** Make a image of webapp using Docker and push it to the DockerHub
6. **Deploy the Model:** Make a VM on azure and then install docker on VM. After that pull the image from docker hub and run it.

Results and Observations

- **Model Performance:**
  The machine learning model is evaluated using various metrics such as precision, recall, and F1-score to determine its accuracy in predicting maintenance needs.

- **Prediction Accuracy:**
  The model accurately detects anomalies based on historical data, reducing the likelihood of unexpected failures.

- **Real-time Alerts:**
  Maintenance teams are alerted well in advance, enabling them to perform maintenance before any breakdown.

```
accuracy = accuracy_score(y_test, y_pred)
print(f"Classification Model Accuracy: {accuracy}")
print(classification_report(y_test, y_pred))
```

```
Classification Model Accuracy: 0.9995
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1860
           1       1.00      0.99      1.00       140

    accuracy                           1.00      2000
   macro avg       1.00      1.00      1.00      2000
weighted avg       1.00      1.00      1.00      2000
```

Conclusion and Future Scope

Conclusion:

This predictive maintenance system successfully uses IoT data and machine learning to predict equipment failures in real time. The system helps reduce downtime and maintenance costs by predicting potential issues before they cause significant disruptions.

Future Scope:

- **Integration of Advanced ML Models**: Implement advanced models like LSTM or Deep Learning for better handling of sequential data.
- **Real-time Streaming Data**: Use cloud-based platforms to handle real-time data streaming from IoT sensors.
- **Mobile Alerts**: Integrate mobile applications for quicker response times and on-the-go notifications for maintenance teams.

Project Code

```
import streamlit as st
from streamlit_option_menu import option_menu
import pandas as pd
import numpy as np
import joblib
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

data=pd.read_csv('created_iot_data.csv')
df=data
model=joblib.load('predictive_maintenance_model.pkl')

# Data Prepration
data['maintenance_required'] = np.where(
    (data['temperature'] > 80) | (data['vibration'] > 5) | (data['pressure'] > 500), 1, 0)

data = data.drop(columns=['timestamp'])
X = data.drop(columns=['maintenance_required'])
y = data['maintenance_required']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# # Model training
# model = RandomForestClassifier(n_estimators=100, random_state=42)
# model.fit(X_train, y_train)

# #Testing model
# y_pred = model.predict(X_test)
# y_prob = model.predict_proba(X_test)[:, 1]

# Evaluating
# accuracy = accuracy_score(y_test, y_pred)
# print(f"Classification Model Accuracy: {accuracy}")
# print(classification_report(y_test, y_pred))

def predict_maintenance(features):
    pred = model.predict([features])
    return pred
```

```
# Streamlit code
st.set_page_config(

    initial_sidebar_state="collapsed"

)
with st.sidebar:
    selected = option_menu(
        menu_title="Main Menu",
        options=["Home", "Input Data", "Visualizations"],
        icons=[ "table", "input-cursor", "bar-chart-line"],
        menu_icon="cast",
        default_index=0,
    )

if selected == "Home":
    st.title("Welcome to 'the Predictive Equipment Maintenance Application using Iot Data' ")
    st.markdown("""
    This application provides predictive maintenance insights for industrial Equipment.""")
    st.title("📂 Data")
    st.write(data.head(15))

elif selected == "Input Data":
    st.title("🔧 Input Features")

    Temperature=st.number_input('Enter Temperature', value=40.0)
    Vibratio=st.number_input('Enter Vibration', value=3.0)
    Pressure=st.number_input('Enter Pressure',value=100.0)

    if st.button('Submit'):
        st.title(" Prediction Results")
        input_features=[Temperature,Vibratio,Pressure]
        prediction = predict_maintenance(input_features)
        if prediction[0]==1:
            st.error(f'Maintenance Required!')
        else :
            st.success(f'Maintenance not needed')

elif selected == "Visualizations":
    st.title("📊 Data Visualizations")

    st.subheader("Histogram of Sensors Readings")
```

```
st.write('Histogram for Temprature')
fig=plt.figure(figsize=(16,8))
sns.histplot(df['temperature'], bins=30, kde=True).set_title('Temperature')
st.pyplot(fig)

st.write('Histogram for Vibration')
fig2=plt.figure(figsize=(16,8))
sns.histplot(df['vibration'], bins=30, kde=True).set_title('Vibration')
st.pyplot(fig2)

st.write('Histogram for Pressure')
fig3=plt.figure(figsize=(16,8))
sns.histplot(df['pressure'], bins=30, kde=True).set_title('Pressure')
st.pyplot(fig3)

st.title("📊 Sensors Data")
fig4=plt.figure(figsize=(16, 10))
sns.lineplot(df, x='timestamp', y='temperature', label='Temperature')
sns.lineplot(df, x='timestamp', y='vibration', label='Vibration')
sns.lineplot(df, x='timestamp', y='pressure', label='Pressure')
plt.xlabel('Timestamp')
plt.ylabel('Sensor Readings')
plt.legend()
plt.title('Created IoT Sensor Data')
st.pyplot(fig4)

importances = model.feature_importances_
indices = np.argsort(importances)[::-1]

st.title("📊 Feature Importance")
fig5=plt.figure(figsize=(10, 5))
plt.title("Feature Importances")
plt.bar(range(X.shape[1]), importances[indices], align='center')
plt.xticks(range(X.shape[1]), X.columns[indices], rotation=90)
plt.tight_layout()
st.pyplot(fig5)
```

References

1. **"Predictive Maintenance Using IoT and Machine Learning"**

2. Source: https://www.sciencedirect.com

3. **"IoT in Predictive Maintenance: A Case Study"**
   Source: https://www.researchgate.net
4. **"Machine Learning for Predictive Maintenance"**
   Source: https://www.medium.com