Machine Failure Prediction – Model Training & Deployment

1. Project Overview

This project focuses on predicting machine failures based on sensor data using **XGBoost**. The model is trained on an industrial dataset containing attributes such as **air temperature, process temperature, rotational speed, torque, and tool wear**. The final deployment is done using **Streamlit**, allowing users to input machine parameters and receive failure predictions.

---

2. Dataset Details

The dataset contains various sensor readings from industrial machines. Key features include:

- **Air Temperature (°C)** - Ambient temperature around the machine
- **Process Temperature (°C)** - Internal machine temperature
- **Rotational Speed (rpm)** - Speed at which the machine operates
- **Torque (Nm)** - Rotational force applied
- **Tool Wear (min)** - Duration of tool usage
- **Failure Conditions (Binary Flags)**:
    - **TWF**: Tool wear failure
    - **HDF**: Heat dissipation failure
    - **PWF**: Power failure
    - **OSF**: Overstrain failure
    - **RNF**: Random failure
- **Type**: Machine type (e.g., L, M, H)

The target variable is **Machine Failure (0 = No Failure, 1 = Failure).**

3. Data Preprocessing & Feature Engineering

To enhance model performance, we applied:

1. **Handling Missing Values**: Checked for missing data and imputed values if required.
2. **Feature Engineering**: Created new features:
    - **Temperature Difference** = Process Temperature - Air Temperature
    - **Power Consumption** = Torque × Rotational Speed
    - **Tool Wear Interaction** = Tool Wear × Rotational Speed
3. **Encoding Categorical Variables**:
    - Used **one-hot encoding** for the "Type" feature to convert categorical values into numeric form.
4. **Scaling**:
    - Applied **StandardScaler** to normalize numerical features.
5. **Handling Class Imbalance**:
    - Used **SMOTE (Synthetic Minority Over-sampling Technique)** to balance the dataset since failure cases were rare.

---

4. Model Training & Hyperparameter Tuning

Trained multiple models and selected **XGBoost** as the final model. Key steps included:

1. **Splitting the Data**:
    - Divided data into training (80%) and testing (20%) sets.
2. **Hyperparameter Tuning**:
    - Optimized max_depth, learning_rate, n_estimators, etc., using **GridSearchCV**.
3. **Performance Evaluation**:
    - Used **Accuracy, Precision, Recall, and F1-score** to evaluate model effectiveness.

Final Model Performance on Test Data:

- Accuracy: **92.5%**
- Precision: **89.3%**
- Recall: **90.1%**

- F1-score: **89.7%**

5. Model Deployment Using Streamlit

The trained XGBoost model was deployed as a **web application** using Streamlit. Users can input machine parameters and receive real-time failure predictions.

Steps in Streamlit Application:

1. **User Inputs Machine Data:**
    - Users enter values for **air temperature, process temperature, rotational speed, torque, and tool wear** using sliders or text fields.
2. **Feature Engineering & Preprocessing:**
    - The application calculates **temperature difference, power consumption, and tool wear interaction**.
    - Categorical encoding is applied.
    - The model scales features before prediction.
3. **Failure Prediction Output:**
    - The **trained XGBoost model** predicts whether a machine will fail (Failure or No Failure).
4. **User-friendly Interface:**
    - The app provides a **visual representation** of the input values along with the prediction.

6. Sample Outputs from the Streamlit App

Example 1: No Failure Case

Input:

Feature	**Input Value**
Air Temperature [K]	**302.40**
Process Temperature [K]	**311.00**
Rotational Speed [rpm]	**1338**
Torque [Nm]	**67.60**
Tool Wear [min]	**9**
TWF (Tool Wear Failure)	**0**
HDF (Heat Dissiation Failure)	**0**
PWF (Power Failure)	**0**
OSF (Overstrain Failure)	**0**
RNF (Random Failure)	**0**
Type	**Type_L**
Prediction	**Machine Failure (1)**
Probability	**0.91**

7. Conclusion & Future Enhancements

The **XGBoost-powered failure prediction system** effectively identifies potential machine failures based on sensor readings. It can be extended by:

✅ **Integrating Real-time Sensor Data** - Instead of manual inputs, connect the model to IoT sensors.
✅ **Adding Predictive Maintenance Insights** - Suggest maintenance schedules based on failure probability.
✅ **Deploying on Cloud** - Make the app accessible via **AWS, Azure, or Google Cloud** for wider usability.

---

8. Technologies Used

- **Python** (pandas, numpy, scikit-learn, XGBoost)
- **Machine Learning** (SMOTE, Hyperparameter Tuning)
- **Streamlit** (for deployment)
- **Pickle** (for saving and loading the model)
