Machine Failure Prediction

Dataset Overview

This dataset captures sensor readings from machines alongside labels indicating whether a machine failure occurred. The aim is to predict machine failures in advance by analyzing the relationships between various sensor readings and the failure indicator. This type of analysis can help prevent unexpected downtime, reduce maintenance costs, and improve operational efficiency.

Columns Description

Each column represents a feature or a label in your dataset:

1. **footfall**
   - **Definition: Number of people or objects passing near the machine.**
   - **Usefulness: May indirectly indicate machine usage or external interference that could affect performance.**
2. **temp Mode**
   - **Definition: Operational temperature mode or setting of the machine.**
   - **Usefulness: Affects the operating conditions and could influence the likelihood of failure.**
3. **AQ (Air Quality)**
   - **Definition: Air quality index near the machine.**
   - **Usefulness: Poor air quality could lead to faster wear and tear or sensor degradation.**
4. **USS (Ultrasonic Sensor)**
   - **Definition: Measures proximity, possibly detecting objects or abnormal distances in machine operation.**
   - **Usefulness: Variations may signal operational disruptions.**
5. **CS (Current Sensor)**
   - **Definition: Electrical current usage of the machine.**
   - **Usefulness: Abnormal current levels could indicate electrical faults or mechanical issues.**
6. **VOC (Volatile Organic Compounds)**

    - Definition: Levels of volatile organic compounds near the machine.
    - Usefulness: High VOC levels may indicate leaks or environmental factors affecting the machine.
7. RP (Rotational Position or RPM)
    - Definition: Rotational speed of machine parts (measured in revolutions per minute).
    - Usefulness: Deviations from normal RPM may point to mechanical issues.
8. IP (Input Pressure)
    - Definition: Pressure entering the machine's system.
    - Usefulness: Abnormal input pressure could affect machine performance or signal blockages.
9. Temperature
    - Definition: Actual operating temperature of the machine.
    - Usefulness: High or low operating temperatures can lead to failure or reduced efficiency.
10. fail

- Definition: A binary indicator of machine failure, where `1` represents failure and `0` represents no failure.
- Usefulness: The target variable for predictive modeling.

Objective

Your project focuses on building a predictive model that identifies patterns in the sensor readings to forecast machine failures. This involves:

1. Exploratory Data Analysis (EDA): Understanding trends, correlations, and distributions among the features.
2. Feature Engineering: Extracting meaningful features from raw data to improve model accuracy.
3. Model Training: Building a machine learning model to classify machines as at risk of failure or not.
4. Evaluation: Assessing the model's performance using metrics like accuracy, precision, recall, and F1-score.
5. Deployment/Insights: Applying the model to real-world scenarios to optimize machine maintenance schedules.

Step-by-Step Code Breakdown

1. Import Libraries

python
Copy code
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
import gradio as gr
```

- **pandas**: Used for data manipulation and analysis.
- **sklearn**: Provides ML tools for training and evaluating models.
- **Gradio**: Used to create an interactive web-based interface.

---

2. Load and Clean the Dataset

python
Copy code
```
data = pd.read_csv('/path/to/your/dataset.csv')
data_cleaned = data.drop_duplicates()
```

- **Load Dataset**: Reads the dataset from a CSV file.
- **Remove Duplicates**: Eliminates redundant rows to ensure data integrity.

---

3. Split Features and Target

python
Copy code
```
X = data_cleaned.drop('fail', axis=1)
y = data_cleaned['fail']
```

- **Features (`X`)**: Columns used to predict machine failure (`fail`). These include:
    - `footfall`, `tempMode`, `AQ`, `USS`, etc.
- **Target (`y`)**: The column `fail` (1 = machine failed, 0 = no failure).

---

4. Split Data into Training and Testing Sets

python

Copy code
```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42, stratify=y)
```

- **Train-Test Split**:
  - 80% data for training the model.
  - 20% data for testing its performance.
- **Stratify**: Ensures the training and testing sets have the same class distribution as the original data.

---

5. Train the Random Forest Model

python
Copy code
```
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
```

- **Why Random Forest?**
  - It's a robust and efficient model for classification problems.
  - Combines multiple decision trees to reduce overfitting and improve accuracy.
- **Training (`fit`)**: The model learns patterns from the training data.

---

6. Evaluate the Model

python
Copy code
```
y_pred = rf_model.predict(X_test)
print("Classification Report:\n", classification_report(y_test,
y_pred))
```

- **Predict (`predict`)**: Uses the test data to predict whether machines fail.
- **Evaluation Metrics**:
  - **Precision, Recall, F1-Score**: Measures the accuracy of predictions.
  - **Accuracy**: The overall percentage of correct predictions.

---

7. Define Prediction Function for Gradio

python
Copy code
```
def predict_failure(footfall, tempMode, AQ, USS, CS, VOC, RP, IP,
Temperature):
    input_data = pd.DataFrame({
        'footfall': [footfall],
        'tempMode': [tempMode],
        'AQ': [AQ],
        'USS': [USS],
        'CS': [CS],
        'VOC': [VOC],
        'RP': [RP],
        'IP': [IP],
        'Temperature': [Temperature]
    })
    prediction = rf_model.predict(input_data)[0]
    return "Failure" if prediction == 1 else "No Failure"
```

- Accepts user inputs for sensor data.
- Converts inputs into a DataFrame to match the training data format.
- Makes predictions using the trained model.

---

8. Build the Gradio Interface

python

Copy code

```
with gr.Blocks() as demo:
    gr.Markdown("# Machine Failure Prediction")

    with gr.Row():
        footfall = gr.Number(label="Footfall")
        tempMode = gr.Number(label="Temperature Mode")
        AQ = gr.Number(label="Air Quality (AQ)")
        USS = gr.Number(label="Ultrasonic Sensor (USS)")
        CS = gr.Number(label="Current Sensor (CS)")
        VOC = gr.Number(label="VOC")
        RP = gr.Number(label="Rotational Position (RP)")
        IP = gr.Number(label="Input Pressure (IP)")
        Temperature = gr.Number(label="Temperature")

    predict_button = gr.Button("Predict")
    output = gr.Textbox(label="Prediction")

    predict_button.click(
        predict_failure,
        inputs=[footfall, tempMode, AQ, USS, CS, VOC, RP, IP,
Temperature],
        outputs=output
    )
demo.launch()
```

- **Interface**:
    - `Markdown`: Adds a title.
    - `Number`: Allows users to input numerical values for sensor data.
    - `Button`: A clickable element to trigger the prediction.
    - `Textbox`: Displays the prediction result.
- **Integration**:
    - The `predict_failure` function is called when the "Predict" button is clicked.
- **Launch**:
    - Opens the Gradio app on a local or public link.

Why Use Machine Learning Models?

- **Automate Predictions**: Models can predict outcomes (e.g., machine failures) based on input data without human intervention.
- **Scalability**: Handle large datasets efficiently, identifying patterns humans may miss.
- **Accuracy**: Improve decision-making with consistent and precise predictions.
- **Cost Efficiency**: Proactively detect issues (e.g., machine failures) to save maintenance costs.