📝 Notebook Summary: Step 2 – Feature Engineering

This notebook generates time-aware statistical features and encodes categorical variables to prepare the dataset for machine learning modeling.

📌 Objectives

- Sort data by machine and timestamp
- Generate rolling window statistics (mean, std, min, max)
- One-hot encode `operating_mode`
- Output a clean, feature-rich dataset for model training

🔍 Section-by-Section Overview

1. Import & Load

- Loaded pandas, numpy, matplotlib, seaborn
- Read cleaned dataset (`agri_sensor_data_cleaned.csv`) created in Notebook 01

2. Sort by Time

- Ensured chronological order of events for each machine using `groupby` on `machine_id`

3. Rolling Window Features

- For each sensor: vibration, current, temp, torque, and RPM:
    - Computed 30-second rolling `mean` , `std` , `min` , and `max`
- These features help detect trends and anomalies prior to failure

4. Fill NaNs

- Filled NaNs from rolling windows with zero to avoid breaking model inputs

5. One-Hot Encoding

- Converted categorical `operating_mode` into binary columns (e.g., `operating_mode_idle`)
- Retained all sensor + contextual signals

6. Drop and Save

- Dropped timestamp (not needed for modeling)
- Saved the new dataset to `data/processed/agri_features.csv`

📈 Recommendations

- Log sensor behavior over time for 1–2 sample machines (trend plots)
- Investigate correlation between rolling features and failure events
- Validate that no leakage occurs from label columns (`failure_label` , `RUL`) into feature space

🔗 Link to Next Notebook

This dataset feeds directly into `03_model_training.ipynb` , where models are trained to:

- Classify failure events
- Predict remaining useful life (RUL) via regression

✅ **Status**: Features generated and dataset exported for ML training.

In [1]:
```
# 📦 Import Required Libraries
# These libraries support data handling, visualization, and statistical feature generation.

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pathlib import Path
```

In [2]:
```
# 📂 Define Paths and Load Cleaned Dataset
# We use the cleaned output from Notebook 01 to begin feature engineering.

INPUT_PATH = Path("../data/processed/agri_sensor_data_cleaned.csv")
OUTPUT_PATH = Path("../data/processed/agri_features.csv")

df = pd.read_csv(INPUT_PATH, parse_dates=['timestamp'])
print("✅ Cleaned dataset loaded. Shape:", df.shape)
```

```
✅ Cleaned dataset loaded. Shape: (32400, 11)
```

In [3]:
```
# 🧭 Sort Data by Machine and Timestamp
# Ensures temporal ordering before generating rolling features.

df = df.sort_values(by=['machine_id', 'timestamp'])
```

In [4]:
```
# 📊 Generate Rolling Window Features
# For each machine, compute rolling statistics (mean, std, min, max) over a 30-second window.

window_size = 30
sensor_cols = ['vibration_level', 'motor_current', 'motor_temp', 'torque', 'rpm']

for col in sensor_cols:
    df[f'{col}_mean'] = df.groupby('machine_id')[col].transform(lambda x: x.rolling(window_size, min_periods=1).mean())
    df[f'{col}_std'] = df.groupby('machine_id')[col].transform(lambda x: x.rolling(window_size, min_periods=1).std())
    df[f'{col}_min'] = df.groupby('machine_id')[col].transform(lambda x: x.rolling(window_size, min_periods=1).min())
    df[f'{col}_max'] = df.groupby('machine_id')[col].transform(lambda x: x.rolling(window_size, min_periods=1).max())
```

In [5]:
```
# 🧹 Fill Missing Values from Rolling Stats
# Rolling std at start of window may result in NaNs — we fill them with 0.

df.fillna(0, inplace=True)
```

In [6]:
```
# 🔠 One-Hot Encode Operating Mode
# Converts the 'operating_mode' categorical column into binary columns.

df = pd.get_dummies(df, columns=['operating_mode'])
```

In [7]:
```
# 🧼 Drop Non-Numeric or Unused Columns
# We drop columns not used directly for ML modeling at this stage.

df_model = df.drop(columns=['timestamp'])  # machine_id retained for traceability
```

In [8]:
```
# 💾 Save Feature-Engineered Dataset
# This file will be used in Notebook 03 for model training and evaluation.

df_model.to_csv(OUTPUT_PATH, index=False)
print(f"📁 Feature-engineered dataset saved to: {OUTPUT_PATH}")
print("📊 Final shape:", df_model.shape)
```

```
📁 Feature-engineered dataset saved to: ..\data\processed\agri_features.csv
📊 Final shape: (32400, 33)
```