Week 2 – Data Preprocessing and Feature Engineering

Introduction:

Raw observations by themselves rarely provide enough insight or predictive power since industrial processes generate constant streams of sensor data. Converting this raw data into a rich, organized format appropriate for machine learning is the main goal of week two of SmartFactory.AI. To improve the model's capacity to predict failures and maximize performance, this comprises preprocessing, temporal feature engineering, and advanced analytics. The official start of turning data into intelligence occurs this week, which builds on the exploratory data analysis completed in Week 1.

Objective:

The following are the main goals for Week 2:

- Clean up and standardize unprocessed sensor data.
- Create reliable time-aware features like cumulative statistics, rolling windows, and delays.
- To record temporal behavior, extract date-time components.
- Determine characteristics that point to stability, trends, or change.
- Validate the statistical and practical significance of engineered traits by visualizing them.
- Keep the preprocessed data for use in further modeling stages.

Methodology:

A Python script with the following structure implements the preprocessing pipeline and feature engineering:

- **Preparation and Data Loading:**
    - The script creates a pandas DataFrame from **synthetic_sensor_data.json**.
    - To preserve the order of machine operation, **timestamp** data are processed as **datetime** objects and sorted.
    - To maintain temporal continuity, forward fill **(method='ffill')** is used to handle missing data.

- **Extraction of Datetime Features:** The following datetime components are extracted in order to record periodic behavior:
    - For identifying long-term patterns and seasonal impacts, use **year, month, and day.**
    - **hour:** Documents trends of the day, such as a rise in failures during night shifts.
    - **weekday:** Helpful for contrasting operational abnormalities on weekdays and weekends.
- **Rolling Statistics (Window = 5):** The following rolling statistics are calculated for the primary sensor metrics (temperature, vibration, rpm, pressure, humidity, and voltage):
    - **rolling_mean:** highlights long-term dynamics by smoothing down high-frequency noise.
    - **rolling_std:** Indicates the volatility of the signal; wear or instability may be shown by spikes.

    Because this is done per machine, contextual dynamics are kept local to each unit.

- **Lag Features:**
    - The script calculates the values from the latest 1 and 2 readings, respectively, for each measure in order to give the model instant historical context.
    - For sequential models (such as RNNs or temporal ensembles), they are essential because they enable them to identify momentum or behavioral reversals.
- **Rate of Change (RoC):**
    - This is calculated as the percentage change between successive sensor readings:

    $$ROC = \frac{x_t - x_{t-1}}{x_{t-1}}$$

    - As a result, the model can capture system dynamics accelerations and decelerations, which are frequently more revealing than raw numbers.
- **Cumulative Features:**
    - The script determines the cummax of temperature and vibration each machine in order to comprehend peak stress levels. This keeps track of the greatest stress level that has been seen thus far and is pertinent to wear prediction.

Advance Analytics and Visualization:

Following feature engineering, a number of investigations are carried out to investigate the new features' predictive value:

- **Failure Counts by Machine:**
    - The top ten machines with the greatest failure counts are displayed in a bar plot.
    - This makes it easier to designate units for more regular inspection and prioritize machinery for repair.
- **Failure Rate by Hour:**
    - The average failure rate for each hour of the day is displayed in a line chart.
    - This identifies risk hotspots across time, such as failures that peak in the early morning.
- **The correlation Heatmap:**
    - A heatmap displays the relationships between failure and rolling means.
    - The importance of features like **temperature_rolling_mean** and **vibration_rolling_mean** as inputs to predictive models is demonstrated by their somewhat good association with failure likelihood.
- **Vibration Std Dev vs. Failure Boxplot:**
    - For failed samples, **vibration_rolling_std** clearly increases.
    - This illustrates how vibration instability, as opposed to absolute quantities, is a powerful failure indication.
- **Monthly Failure Trend:**
    - Seasonal drifts and anomalies are displayed in a line plot of monthly failures.
    - Seasonality-aware features or models can be incorporated based on these trends.
- **RPM Rate of Change Distribution:**
    - The spread and tails of the **rpm_roc** histogram are visible.
    - This is crucial for determining if high RoC levels, which signify stress circumstances, are associated with breakdowns.
- **Engineered Features Pairplot:**
    - Observable clusters may be seen in a pairplot (sample size = 2000) of **temperature_rolling_mean**, **vibration_rolling_mean**, and **rpm_rolling_mean** colored by failure status.
    - Because these designed characteristics exhibit separability, their application in classification models is justified.

Summary:

Raw, noisy data is transformed into a rich, organized dataset with a multi-scale picture of operational behavior via the Week 2 feature engineering pipeline. Among the major accomplishments are:

- Recording behavior on an hourly, daily, and cumulative basis is known as temporal granularity.
- Standard deviations and rate-of-change measurements are used in volatility modeling.
- **Contextual memory:** Through cumulative peaks and lag characteristics.
- **Failure alignment: These** traits and failure events have a significant correlation, according to visual analytics.

High importance for downstream modeling tasks such as failure prediction, anomaly detection, and optimization is suggested by these findings, which also verify the direction of feature building.

Conclusion:

A strict pipeline for preprocessing and feature engineering in SmartFactory.AI is established in Week 2. The outputs, which include volatility measurements, lag indicators, and temporal data, will greatly increase the model's sensitivity to early warning indications of machine failure. The project is now prepared to proceed to Weeks 3–4 for model development, assessment, and integration after the processed data was stored as processed_sensor_data_week2.json.

URL to Week 2 – Data Preprocessing and Feature Engineering.ipynb