Hamedan University of Technology

Special Topics in the Industrial Applications of Machine Learning

Dr. Ghasem Alipoor

Electrical Engineering Department

Hamedan University of Technology

E-mail: g.alipoor@gmail.com, alipoor@hut.ac.ir

Fall 2025

Chapter 3: Data Preprocessing

Industrial datasets are often **noisy**, **incomplete**, or **inconsistent**. Proper preprocessing is essential for building reliable ML models.

In this chapter, we will cover:

- **Data Cleaning:** handling missing and invalid values, outliers, duplicates
- **Data Transformation:** scaling, normalization, encoding categorical data
- **Feature Extraction:** extracting meaningful features from signals and images
- **Data Splitting & Validation:** train/test splits, cross-validation methods

Chapter 3: Data Preprocessing

- Section 1: Motivation
  Why preprocessing is essential for reliable industrial ML.
- Section 2: Data Cleaning
  Handling duplicates, missing values, outliers, and noise.
- Section 3: Data Transformation
  Adjusting scales, distributions, and categories to prepare data for ML models.
- Section 4: Feature Extraction
  Converting raw data into informative representations.
- Section 5: Data Splitting & Validation
  Ensuring fair evaluation and generalization of ML models.

Motivation

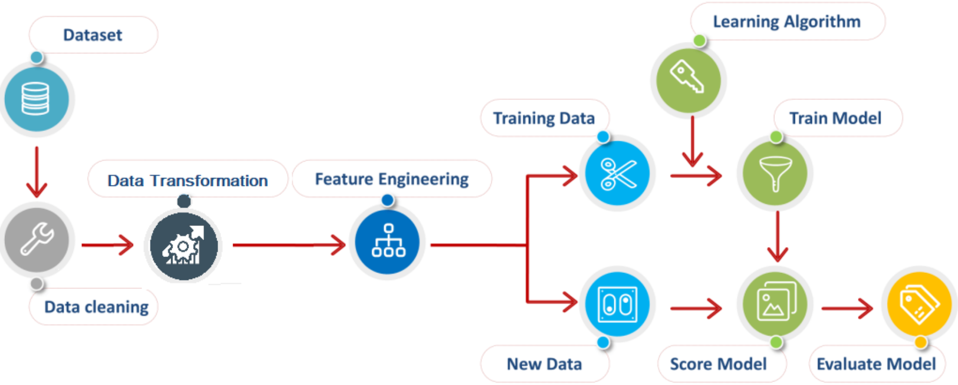
Why Preprocessing Matters in Industrial ML:

- ✓ **Industrial datasets are often noisy or incomplete** → cleaning ensures reliable data.
- ✓ **Signals may be on different scales or types** → transformation makes them suitable for ML models.
- ✓ **Raw signals may not be informative** → feature extraction captures useful features.
- ✓ **Improper data splits can bias evaluation** → proper splitting & validation ensures generalization.

Additional Points:

- ➢ Proper preprocessing can prevent models from learning **spurious patterns or overfitting**.
- ➢ In industrial ML, **70–80% of effort** is often spent on data preparation.

ML Pipeline


Dataset
Data cleaning
Data Transformation
Feature Engineering
Training Data
New Data
Learning Algorithm
Train Model
Score Model
Evaluate Model

Nature of Industrial Data

Raw industrial data often includes:

- **Missing values** – gaps due to faulty sensors or communication loss
- **Outliers**– sudden unrealistic spikes in measurements
- **Noise**– background fluctuations or unhelpful signals
- **Inconsistencies** – data from different devices, units, or protocols
- **Duplicates** – repeated or redundant records

Industrial Examples:

- **Power grid logs with gaps** – missing voltage or current samples
- **Abnormal vibration peaks** – faulty sensor spikes in machinery data
- **Noisy temperature readings** – fluctuating IoT sensor outputs
- **Inconsistent units in process data** – e.g., pressure logged in bar vs. psi
- **Duplicate event logs** – repeated fault alarms in SCADA records

Chapter 3: Data Preprocessing

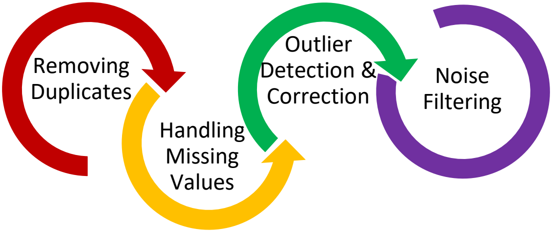
- Section 1: Motivation
  Why preprocessing is essential for reliable industrial ML.
- Section 2: Data Cleaning
  Handling duplicates, missing values, outliers, and noise.
- Section 3: Data Transformation
  Adjusting scales, distributions, and categories to prepare data for ML models.
- Section 4: Feature Extraction
  Converting raw data into informative representations.
- Section 5: Data Splitting & Validation
  Ensuring fair evaluation and generalization of ML models.

Data Cleaning

Without preprocessing, machine learning models may:

- **Learn spurious patterns** – pick up on noise instead of useful features
- **Overfit noisy samples** – perform well on training but fail on test data
- **Produce inaccurate predictions** – give wrong results on real data
- **Generalize poorly in practice** – unreliable when deployed in real settings

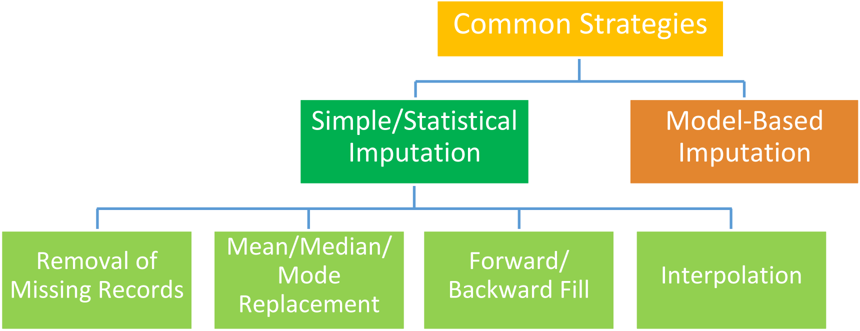
Typical Data Cleaning Tasks:



Handling Missing Values

Why It Matters:

- Missing values are common in industrial datasets due to sensor failures, network issues, or human errors.
- Many ML algorithms cannot handle missing values directly, so preprocessing is essential.



Outlier Detection and Correction

Why It Matters:

Outliers are extreme points that can distort ML learning.
Often caused by sensor faults, abnormal machine behavior, or errors.

Common Detection Methods:

Statistical approaches: beyond 3σ or outside IQR.
Domain rules: thresholds from engineering knowledge (e.g., max pressure, voltage).
Model-based methods: isolation forest, DBSCAN, etc.

Correction Strategies:

Removal: discard only if clearly erroneous (impossible values or confirmed sensor faults).
Capping/Winsorization: limit extreme values while keeping valid extremes.
Imputation: replace extreme but plausible values with median, mean, or predicted values.

Noise Filtering

Why It Matters:

- Sensor data often contains random noise.
- Noise can mask true patterns and degrade model accuracy.

Common Techniques:

Noise Smoothing

- Moving average
- Savitzky–Golay filter

Frequency Filtering

- Low-pass filter
- Band-pass filter

Wavelet Denoising

Removes high-frequency noise while preserving trends

Industrial Note:

- Filtering must balance noise removal with preservation of useful signal details.

Chapter 3: Data Preprocessing

- Section 1: Motivation
  Why preprocessing is essential for reliable industrial ML.
- Section 2: Data Cleaning
  Handling duplicates, missing values, outliers, and noise.
- Section 3: Data Transformation
  Adjusting scales, distributions, and categories to prepare data for ML models.
- Section 4: Feature Extraction
  Converting raw data into informative representations.
- Section 5: Data Splitting & Validation
  Ensuring fair evaluation and generalization of ML models.

Data Transformation

Why It Matters:

- Industrial variables differ in units, scales, or distributions.
- Transformation ensures comparability and faster convergence.

Scaling:

Normalization

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Standardization

$$x' = \frac{x - \mu}{\sigma}$$

Distribution transform:

Power transform

$$x' = \frac{x^{\lambda} - 1}{\lambda}$$

Log transform

$$x' = \log(1 + x)$$

Categorical Encoding:

Labe encoding

$$"Motor" \rightarrow 0$$
$$"Pump" \rightarrow 1$$
$$"Gear" \rightarrow 2$$

One-hot encoding

$$x = \text{Pump} \rightarrow [0, 1, 0]$$

- Ensure consistent types and formats before ML.
- Ordinal Encoding preserves order if categories have ranking.

Chapter 3: Data Preprocessing

- Section 1: Motivation
  Why preprocessing is essential for reliable industrial ML.
- Section 2: Data Cleaning
  Handling duplicates, missing values, outliers, and noise.
- Section 3: Data Transformation
  Adjusting scales, distributions, and categories to prepare data for ML models.
- Section 4: Feature Extraction
  Converting raw data into informative representations.
- Section 5: Data Splitting & Validation
  Ensuring fair evaluation and generalization of ML models.

Feature Extraction

Why It Matters:

- Converts raw signals into informative features for ML.
- Reduces data dimensionality while retaining key patterns.
- Improves model performance and interpretability.

Common Feature Types:

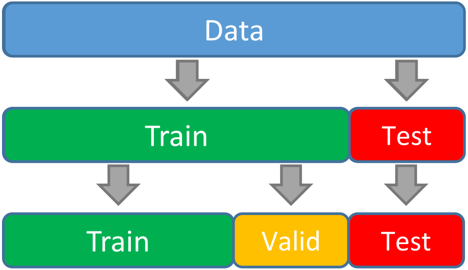
- **Time-domain:** mean, std, RMS, skewness, kurtosis
- **Frequency-domain:** FFT, dominant frequency, spectral energy
- **Other:** envelope, peak-to-peak, signal energy
- **Images:** histogram, edge detection, texture

Chapter 3: Data Preprocessing

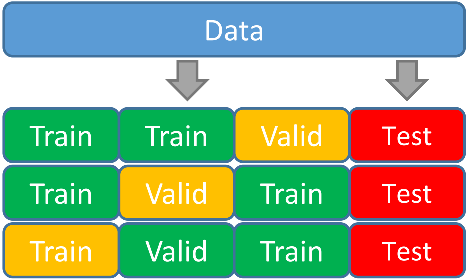
- Section 1: Motivation
  Why preprocessing is essential for reliable industrial ML.
- Section 2: Data Cleaning
  Handling duplicates, missing values, outliers, and noise.
- Section 3: Data Transformation
  Adjusting scales, distributions, and categories to prepare data for ML models.
- Section 4: Feature Extraction
  Converting raw data into informative representations.
- Section 5: Data Splitting & Validation
  Ensuring fair evaluation and generalization of ML models.

Data Splitting & Validation

Hold-out split



K-fold cross-validation



Industrial Note:

- **Validation set**: for hyperparameter tuning, keep test set untouched.
- **Stratification**: preserve class balance in imbalanced datasets.
- **Random shuffling** helps avoid bias, never shuffle time-series data.
- For predictive maintenance and forecasting: use **time-based splits** to prevent leakage.