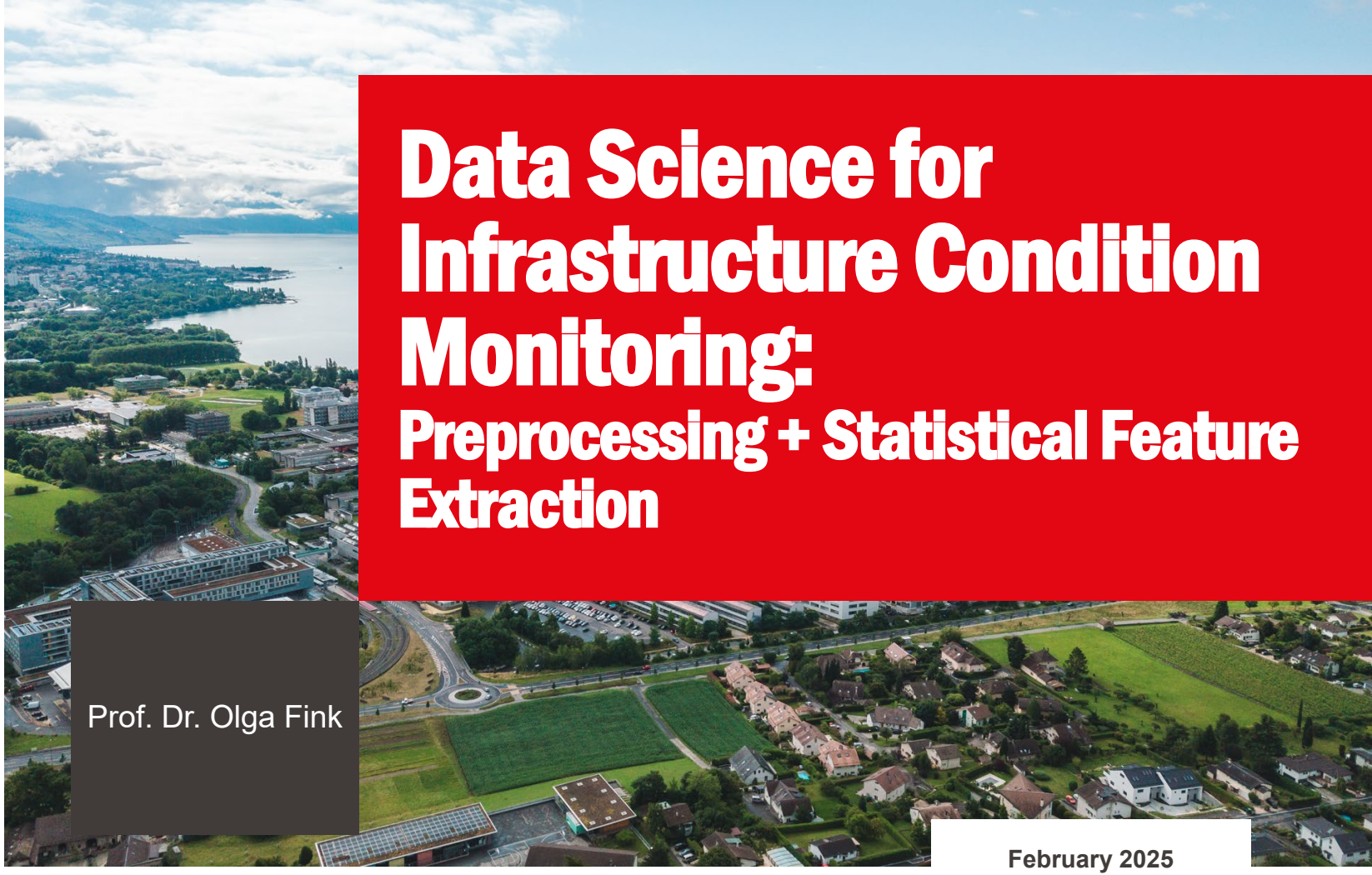


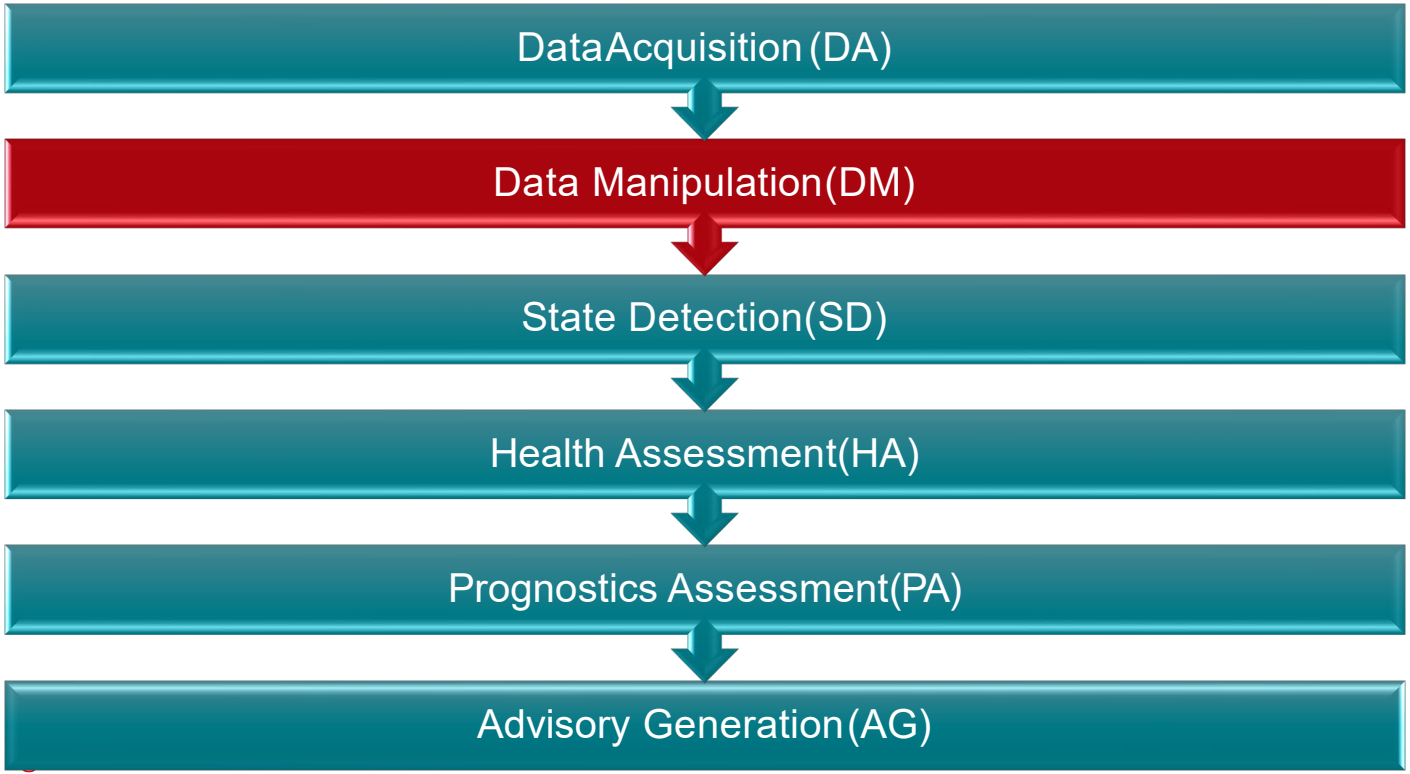
Data Science for Infrastructure Condition Monitoring:
Preprocessing + Statistical Feature Extraction

Prof. Dr. Olga Fink

École polytechnique fédérale de Lausanne

February 2025

PHM Process


DataAcquisition (DA)
Data Manipulation(DM)
State Detection(SD)
Health Assessment(HA)
Prognostics Assessment(PA)
Advisory Generation (AG)

Typical steps to follow in a machine learning (ML) project (1/3)

1. Problem Definition and Understanding
   - Clearly articulate the problem you aim to solve.
   - Identify the objectives and success criteria.
   - Ask the right questions: Identify the questions your data can answer to solve the business problem.
   - Understand the data: Learn the context, limitations, and opportunities within the available data.
2. Identify the Value:
   - Determine the potential value and impact of solving the problem (both from business and from the scientific perspective)
   - Assess how the solution will benefit stakeholders and align with business goals.
3. Collect Data:
   - Gather relevant data from various sources.
   - Make sure that your data is representative for the test data (application data) → be aware of the variability of the operating conditions (→ domain shift)
   - Ensure data quality and completeness.
   - Acquire labels for your data if possible (ensure the quality of labels)
4. Explore and Understand the Data:
   - Perform exploratory data analysis.
   - Visualize data to understand patterns and relationships.
   - Understand the data distribution (Explore the statistical properties, distributions, and trends in the data)
   - Identify data quality issues (Look for missing values, outliers, and inconsistencies that may affect analysis).
   - Generate hypotheses (Form hypotheses based on patterns observed during data exploration).

Typical steps to follow in a machine learning (ML) project (2/3)

5. Prepare Data:
   - Clean the data (handle missing values, outliers).
   - Feature engineering (create meaningful features, transform existing ones).
   - Split data into training, validation, and test sets.
   - Scaling and normalization (Prepare the data for modeling by applying appropriate scaling techniques)
   - Address class imbalances: If needed, use techniques like oversampling, undersampling, or synthetic data generation to balance the dataset.
6. Select and Train Models:
   - Select appropriate models (Choose models based on the problem type (regression, classification, clustering, etc.) and data characteristics)
   - Split the dataset (Use train-test splits (or cross-validation) to ensure your model's generalization to unseen data)
   - Baseline model (Start with a simple model as a baseline to compare more complex models).
   - Tune Hyperparameters (Optimize model hyperparameters for better performance).
   - Iterate and improve (Experiment with different models and tuning hyperparameters)
   - Avoid overfitting (Implement regularization techniques or apply cross-validation to avoid overfitting the training data).
7. Evaluate Models:
   - Choose appropriate metrics (Select evaluation metrics that are relevant to the problem (e.g., accuracy, precision, recall, F1-score, RMSE))
   - Validate on unseen data (Always validate your model on a test set or through cross-validation to assess its performance)
   - Compare models: Compare different models based on performance metrics and business value.

Typical steps to follow in a machine learning (ML) project (3/3)

8. Model Interpretation
   - What do the results mean?
   - Explainability (Ensure that you understand how your model works, and use XAI techniques such as SHAP values or feature importance to explain the results).
   - Check assumptions (Ensure that the model's assumptions hold and are consistent with the problem and data context)
9. Test Model:
   - Evaluate the final model on the test dataset to gauge its real-world performance.
10. Deploy Model:
    - Integrate the model into a production environment.
11. Monitor and Maintain:
    - Continuously monitor model performance.
    - Update the model as needed based on new data and changing conditions.
    - Monitor how your model is actually used
12. Documentation and Communication
    - Document thoroughly (Keep detailed notes on every step, including data sources, preprocessing steps, model choices, evaluation, and deployment procedures)
    - Communicate results (Present findings to stakeholders in a clear, actionable manner. Tailor communication based on the audience (e.g., technical team vs. business executives).)

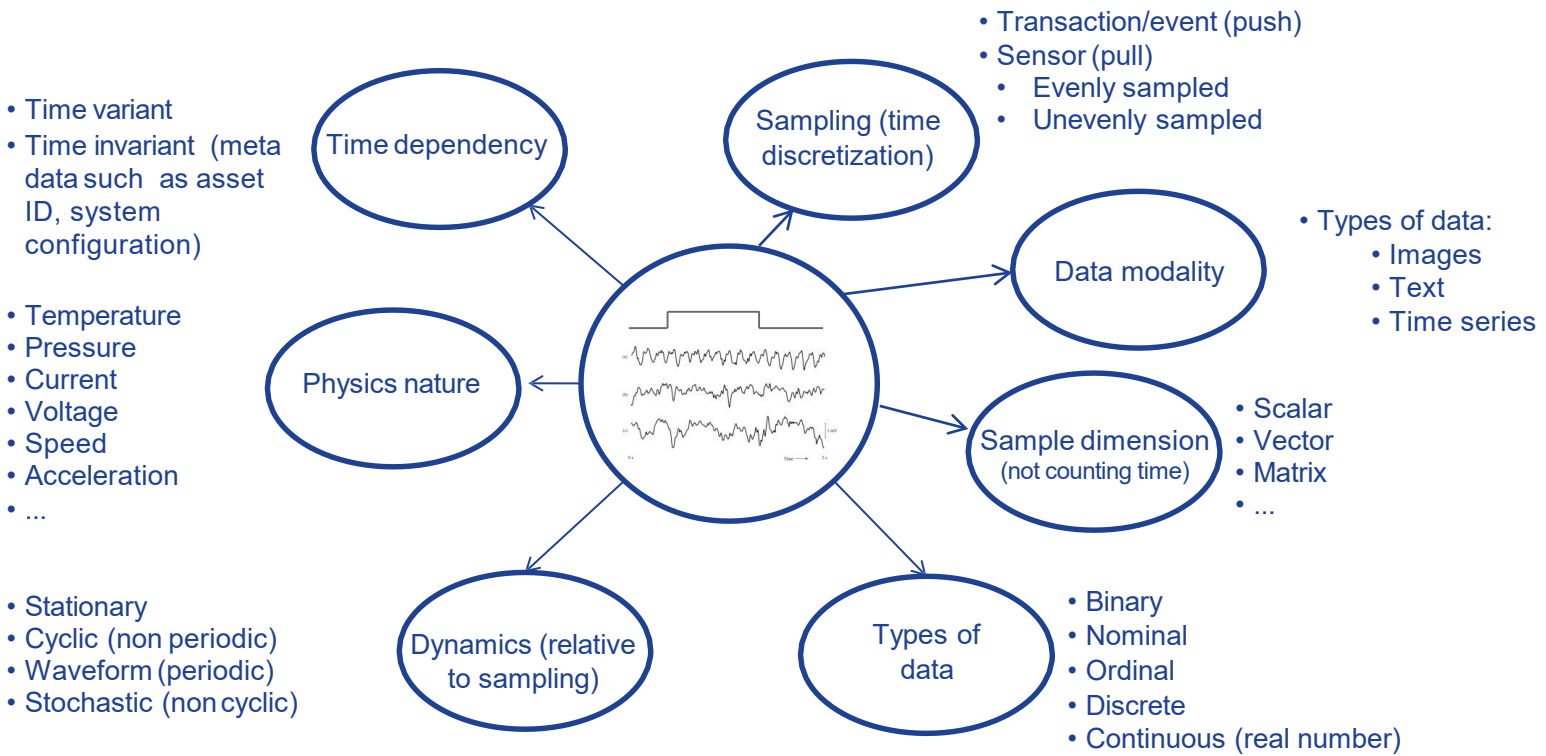
Why data pre-processing?

- Data in the real world is “dirty”
  - incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
  - noisy: containing errors or outliers
  - inconsistent: containing discrepancies in codes or names

Main tasks in data pre-processing

- Data profiling
  - examining, analyzing and reviewing data
  - collect statistics about its quality.
- Data cleaning
  - Fill in missing values, smooth noisy data,
  - identify or remove outliers, and resolve inconsistencies
- Data integration (if needed)
  - Integration of multiple databases, data cubes, or files
- Data transformation
  - Normalization and aggregation
  - Structuring unstructured data
- Data reduction
  - Obtains reduced representation in volume but produces the same or similar analytical results
- Data discretization (if required)
- Data enrichment
  - Feature engineering
- Data validation
  - Assessing the dataset for quality assurance

Data properties



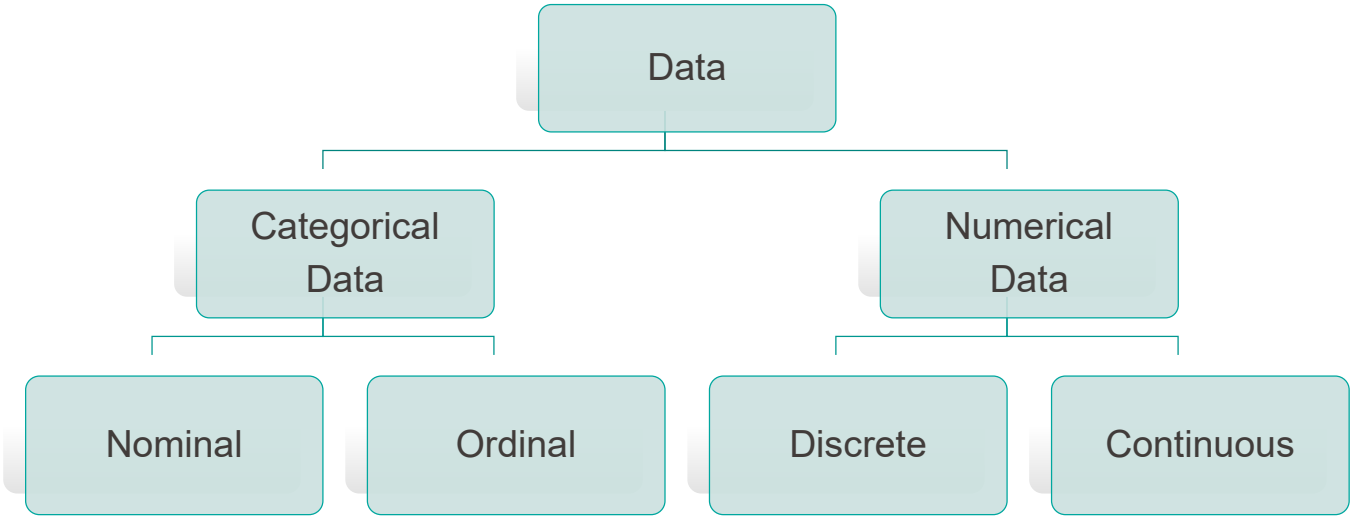
Source: Wang, 2012

Data Sampling

- Transaction/event (data are “pushed” by data originator)
  - Data records occur only at the specified event / transaction / time stamp
  - Data between the time stamps / events are undefined.
- Sensor (data are “pulled” from data originator)
  - Data samples are acquired only at the specified time stamp
  - Data between the time stamps are just not observed.
  - Sampling rate
    - Evenly sampled – controlled (e.g. 100 Hz)
    - Unevenly sampled - triggered

Source: Wang, 2012

Categorical vs. Numerical Data



Examples:

Operating mode, Event code, asset ID

Performance level; severity level; friction level (Low-Medium-High → ranked levels)

Number of occurring diagnostic events, number of occurring faults / interruptions

Temperature, pressure, acceleration (most sensors)

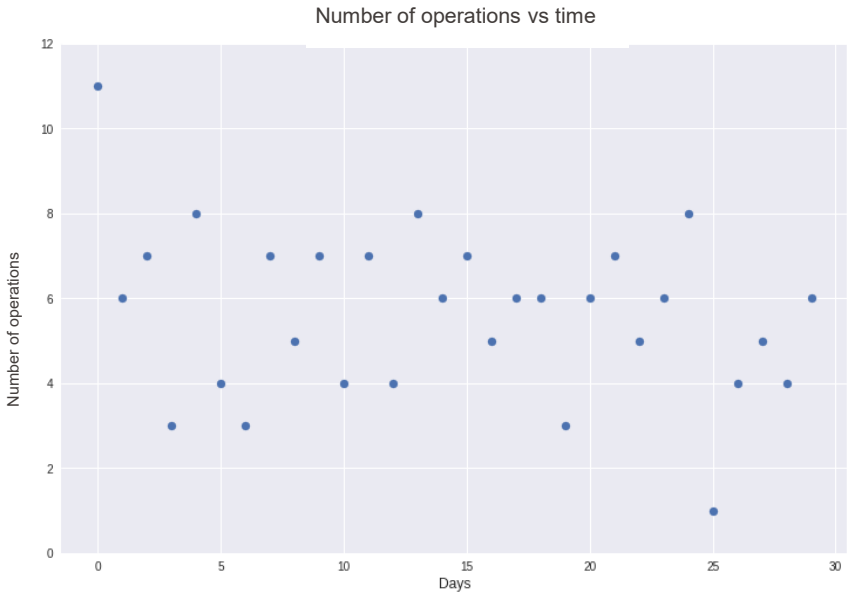
Binary data

- A type of categorical data in which there are only two categories
- Binary data can either be nominal or ordinal

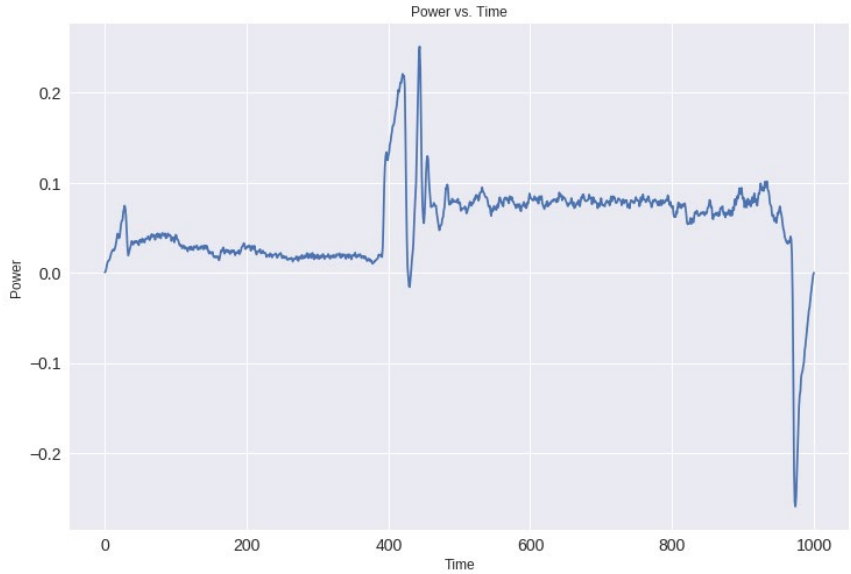
- Examples: event status, on/off sensor

Numerical Data: Discrete vs. Continuous

Discrete Data



Continuous Data

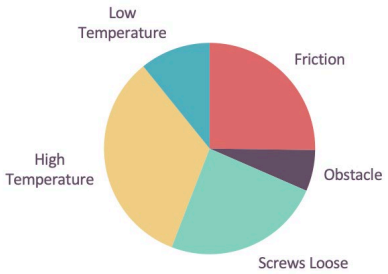


Categorical Data: Representations

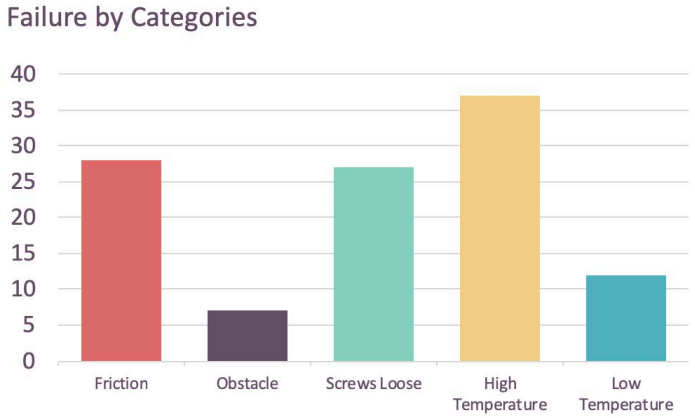
Frequency Tables

Cause of Error	*Number of Occurence*	*Percentage*
Friction	28	25.2%
Obstacle	7	6.3%
Screws Loose	27	24.3%
High Temperature	37	33.3%
Low Temperature	12	10.8%

Pie Charts



Bar Charts

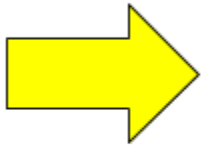


Handling of categorical data

- Replacing values
  - Freely assign numbers to the categories according to the use case / expert knowledge
- Encoding labels
  - convert each categorical value in a column to a number between 0 and n_categories-1
- One-hot encoding
  - convert each category value into a new column and assign a 1 or 0 (True/False) value to the column
- Binary encoding
  - first the categories are encoded as ordinal, then those integers are converted into binary code, then the digits from that binary string are split into separate columns
- …

Example of one-hot encoding

Color
Red
Red
Yellow
Green
Yellow



Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

Binary Encoding in Machine Learning

Why Use Binary Encoding for Categorical Variables?

- When working with categorical data in machine learning, raw categorical values (e.g., country names, product types) need to be transformed into numerical representations for models to process them. Binary encoding is one such method that is particularly useful when:
  - The number of unique categories is large.
  - One-hot encoding would result in high-dimensional data.
  - You want a compressed and less redundant representation.

How Binary Encoding Works

1. Convert each category label into a **unique integer** (ordinal encoding).
2. Convert each integer into its **binary representation**.
3. Store the binary digits in separate columns.

Binary Encoding

Temperature
Hot
Cold
Very Hot
Warm
Hot
Warm
Warm
Hot
Hot
Cold

Order
1
2
3
4
1
4
4
1
1
2

Binary
001
010
011
100
001
100
100
001
001
010

Temperature_0	Temperature_1	Temperature_2
0	0	1
0	1	0
0	1	1
1	0	0
0	0	1
1	0	0
1	0	0
0	0	1
0	0	1
0	1	0

Advantages / Disadvantages of Binary Encoding

+ **Reduces Dimensionality:** Compared to one-hot encoding, fewer columns are needed, making it useful for large categorical datasets.
+ **Avoids Dummy Variable Trap:** Unlike one-hot encoding, binary encoding does not introduce perfect multicollinearity.
+ **Handles High Cardinality Efficiently:** Works well when the number of unique categories is large, as it scales logarithmically.
+ **Preserves Some Ordinal Information:** Since binary encoding is based on integer representations, it retains some level of similarity between categories.

- **Interpretability:** Unlike one-hot encoding, binary-encoded features are less interpretable.
- **Assumes Ordinal Relationships:** Though it preserves some ordinal information, this assumption may not always be correct.

Handling of categorical data

- Some categorical indicators can be used to split the problem in sub-problems (e.g. indicator of the operating conditions for base and part load → developing two models for the two types of operating conditions)

Signal dynamics (relative to sampling)

Stationary (constant + white noise)

- Power, speed, temperature in steady state

Stochastic (non-cyclic)

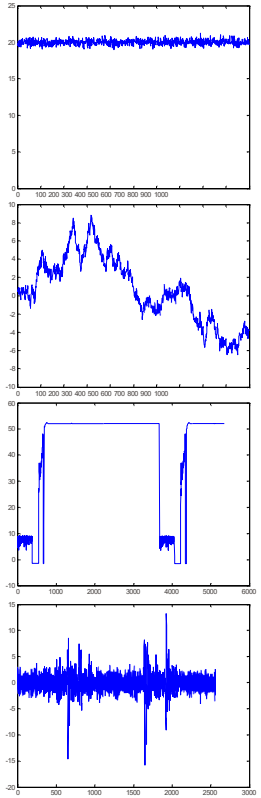
- Power, torque, speed

Cyclic (consider each period individually)

- Power, speed, one switching cycle of a railway switch, one passage of a truck / a railway wheel

Waveform (consider multiple periods together)

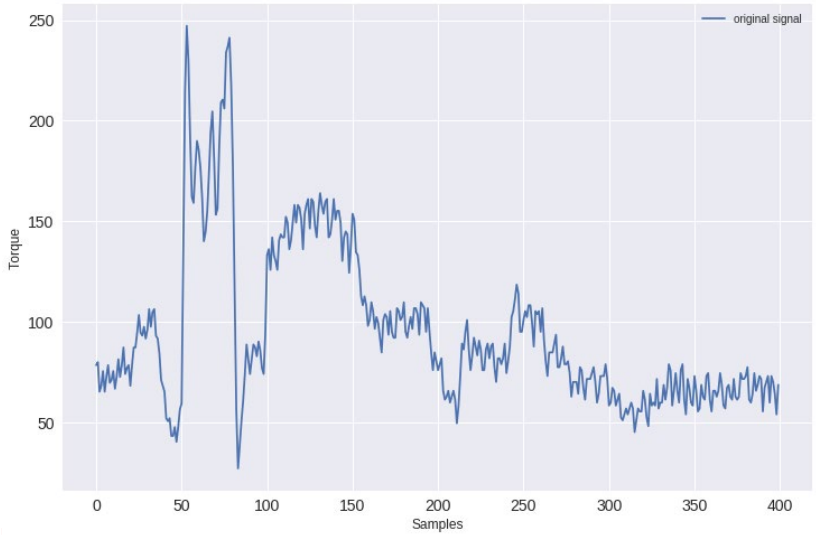
- Vibration sensors, acoustic sensors

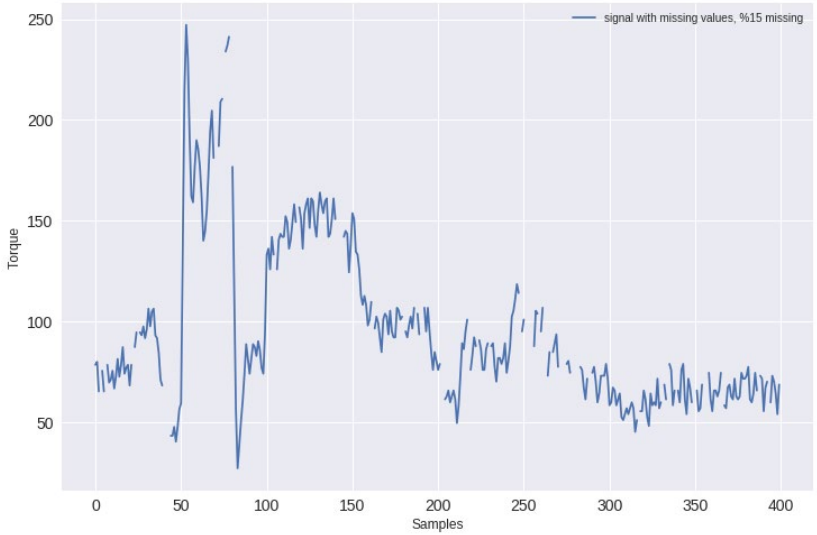


Source: Wang, 2012

Missing Data Problem

- Times series observed with 15% missing data





Missing, noisy, inconsistent data

- Missing data
  - Data imputation approaches (next slide)
- Noisy data
  - Binning
  - Filtering
  - Clustering
  - Remove manually
  - Apply denoising algorithms
- Inconsistent data
  - External references
  - Knowledge engineering tools

Data imputation

- Complete case analysis: Delete any record that has missing values from the data set.
- Nearest neighbors: to impute variable x average the value x of the k closest data points with no missing values.
- Average method: Average the value of x for the non-missing values.
- Hot deck: pick a “similar” record at random and use its value of x.
- Predictive: Fit a model to the data with variable x as the target and use it to predict the value (e.g. kernel regression)
- Single imputation: Draw a value at random from the conditional distribution of x given the other variables
- Multiple imputation: Repeatedly draw values at random from the conditional distribution of x given the other variables (e.g. as above), creating new data sets. Make the predictions with these now complete datasets and average the predictions.

Caution with the different imputation approaches

- Complete case analysis: can result in a bias
- Nearest neighbors: The definition of the “close points” and the value of k required
- Average method: Easy to implement but crude
- Hot deck: A definition of “similar” is required
- Predictive: Better suitable but understates the uncertainty in the imputation process.
- Single imputation: Better suitable, respects the uncertainty. However, just a single value is sampled.
- Multiple imputation: generally regarded as the best method (a sample is better than a single observation)

Why Do We Need Normalization or Standardization?

- Machine learning models, especially distance-based models (e.g., k-NN, SVM, PCA, clustering) and gradient-based models (e.g., neural networks, linear regression), can be **affected by differences in scale** among features. If features have vastly different ranges, models may:
  - Be **biased toward higher-magnitude features**.
  - Take **longer to converge** during training.
  - Exhibit **poor generalization** due to inconsistencies in input scales.

Data normalization / Standardization

- Feature scaling

(→ also referred to as min-max Nomalization)

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

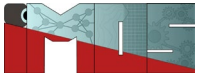
- Standard score

(particularly suitable for normally distributed data)

(→ also referred to as Z-Score Standardization)

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

Alternatives (particularly for data with outliers)



- Quantile Transformation
- Power Transformation (non-linear transformation)

When to Use Normalization?

- When features have different scales and do not follow a normal distribution.
- When working with bounded data (e.g., pixel intensities in images [0, 255]).
- When using distance-based models like k-NN, k-Means clustering.
- When the data has outliers, normalization compresses their impact.

When to Use Standardization?

- When data follows or approximately follows a normal distribution.
- Algorithms that assume the input features are normally distributed with zero mean and unit variance, such as Support Vector Machines, Logistic Regression, etc.
- When using PCA or models where feature variance impacts performance.
- Standardization can be a better choice if your data contains many outliers as it scales the data based on the standard deviation.

Data Discretization

- Create bins (e.g. equal depth, equal size) → e.g. different categories of part-load conditions of a gas turbine
- Additional smoothing possible → replace the values in a bin by their mean

What are Features and Why do we need them?

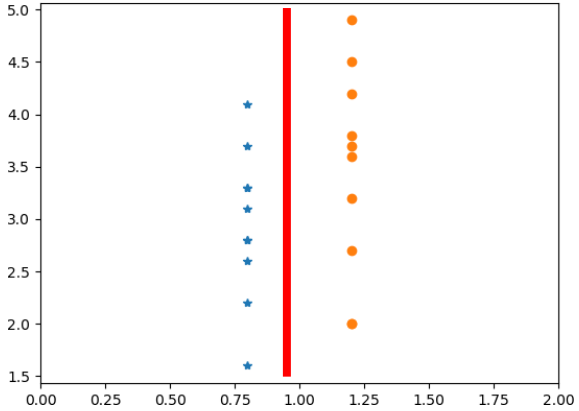
- Transform raw signals into more informative signatures (or fingerprints) of a system
- Reduce size / complexity of the dataset
- Provide a physical description / representation
- Reduce resources necessary for further processing
- Achieve intended objectives

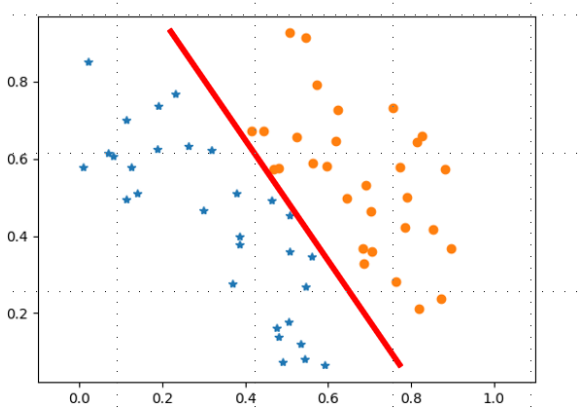
Why are features important

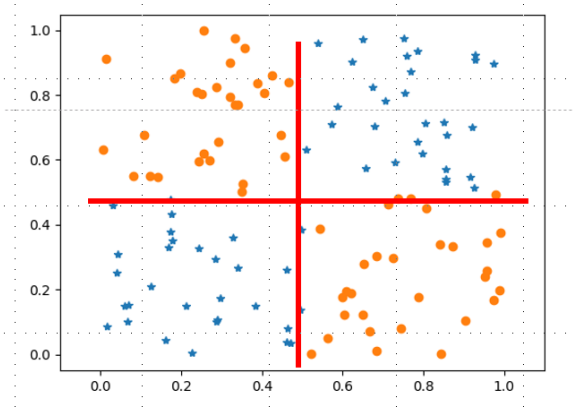
- Features are known to be the most crucial point in machine learning
- “At the end of the day, some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used.”

- Pedro Domingos, in “A Few Useful Things to Know about Machine Learning”

Univariate versus multivariate feature engineering

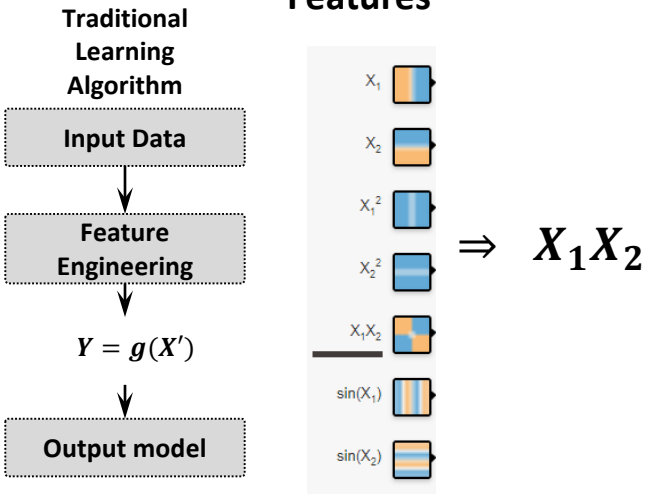


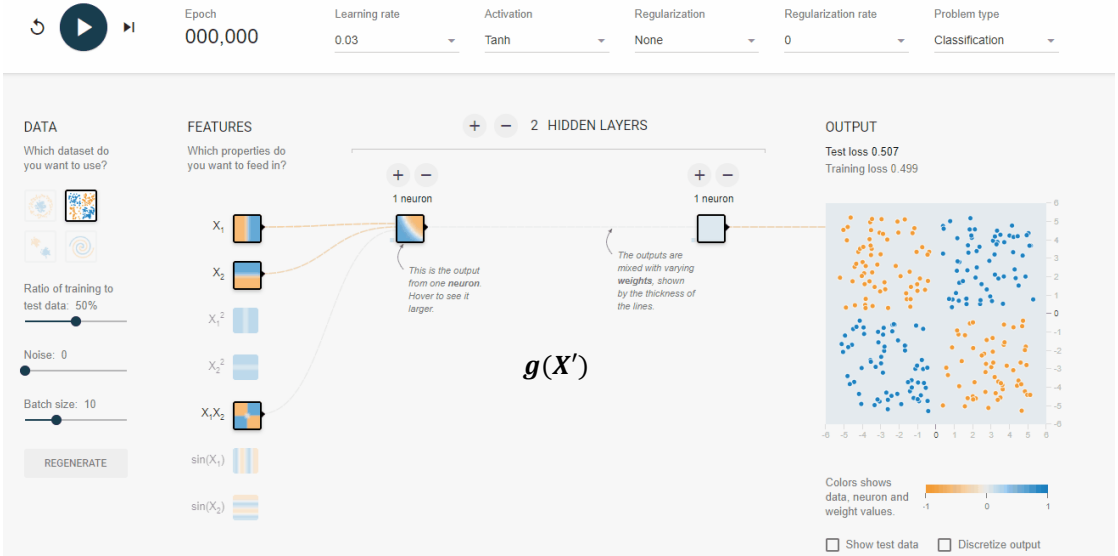




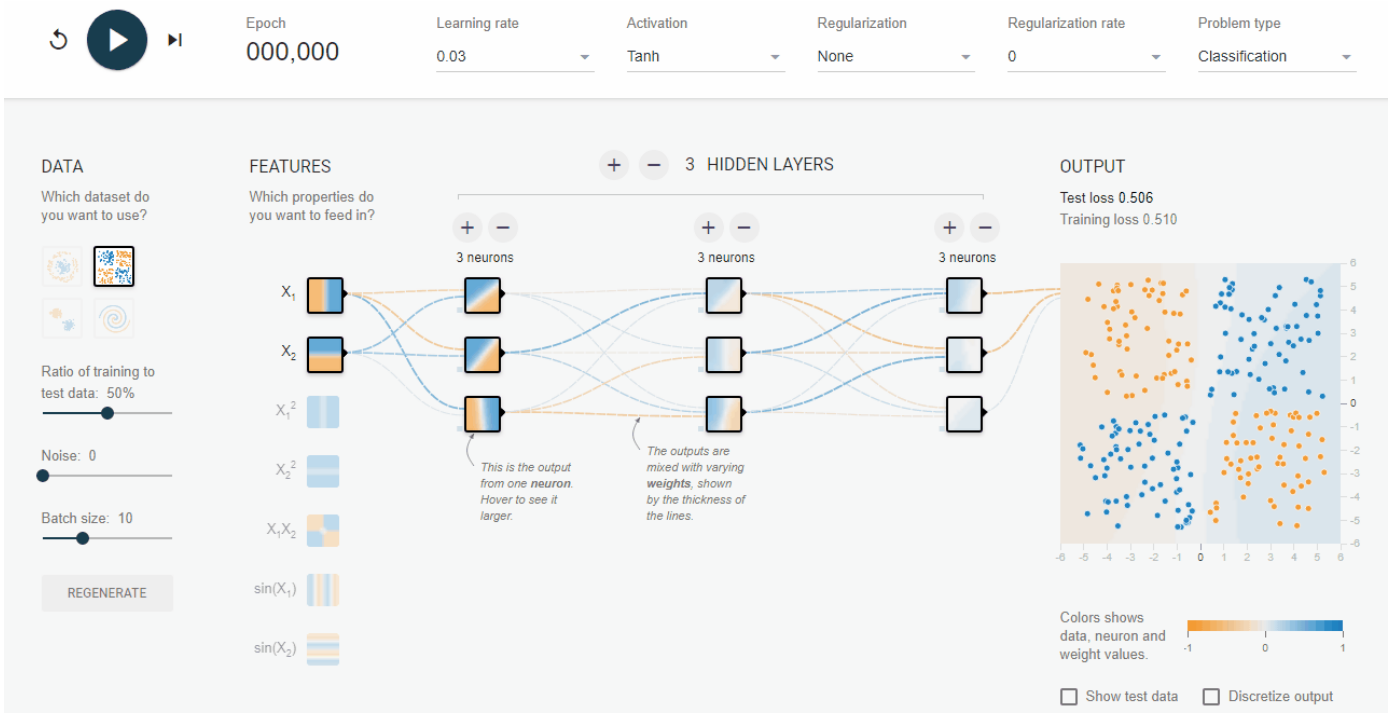
Example of feature engineering

Option 1: good feature engineering





Deep Learning: go deeper and learn the features automatically



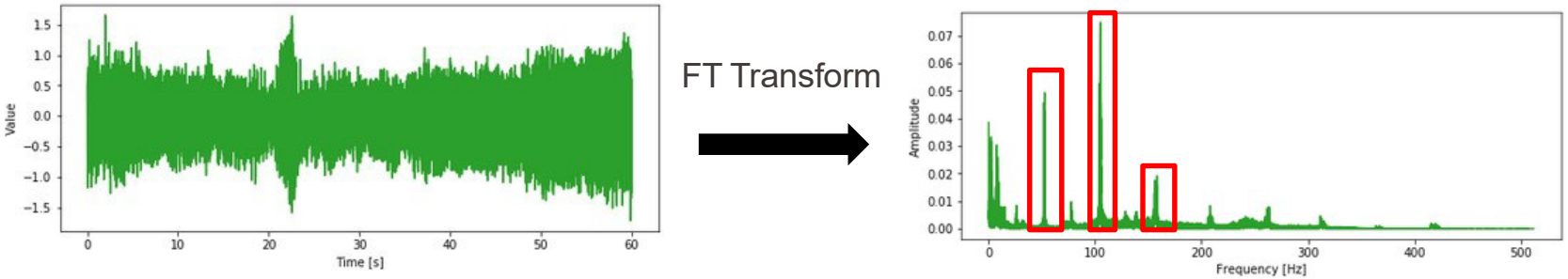
Source: https://playground.tensorflow.org

Example feature extraction in images

ML algorithm (e.g. classifier)

Prediction (label)

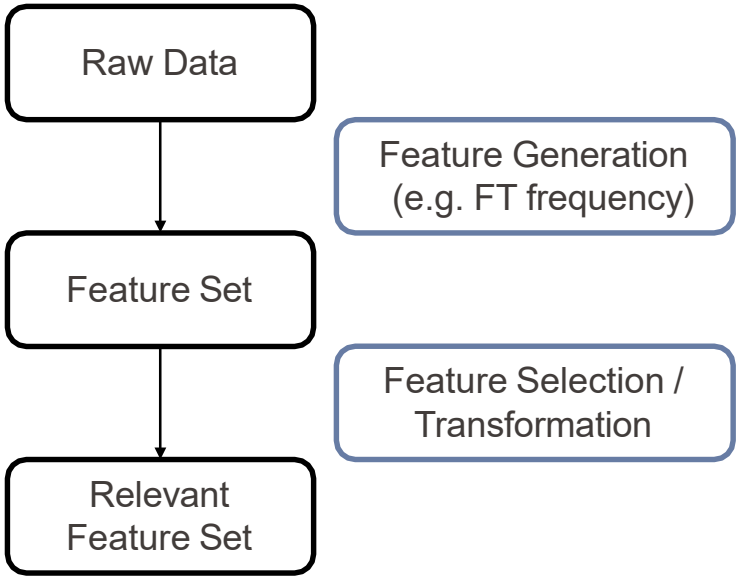
Example high frequency data



Features: Magnitude of 3 carefully chosen frequency bands

In the future, we can compare these features for any measurements

Feature Extraction Process:



- **Raw Data:** Can be of different type and nature and size
- **Generation:** exhaustive, ad-hoc,
  - prior knowledge (domain, physics), modify variable type (cat. to numeric)
- **Feature Set:**
  - Various representation and size (dimensionality change)
- **Dimensionality reduction:**
  - select best subset
  - transform in space of lower dimensions

What matters for the process

- Kind of data

→ available type of features

- Aim of the application and domain

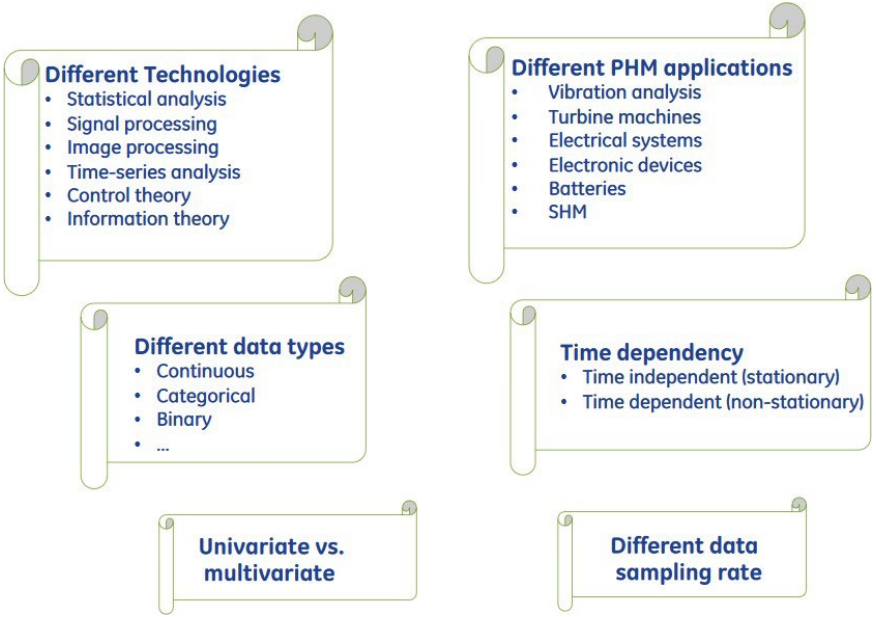
→ (e.g. monitoring, detecting, predicting)

→ (vibration, images, torque,…)

- Kind of methods that will be used

→ To extract feature,

→ To use the features for the application



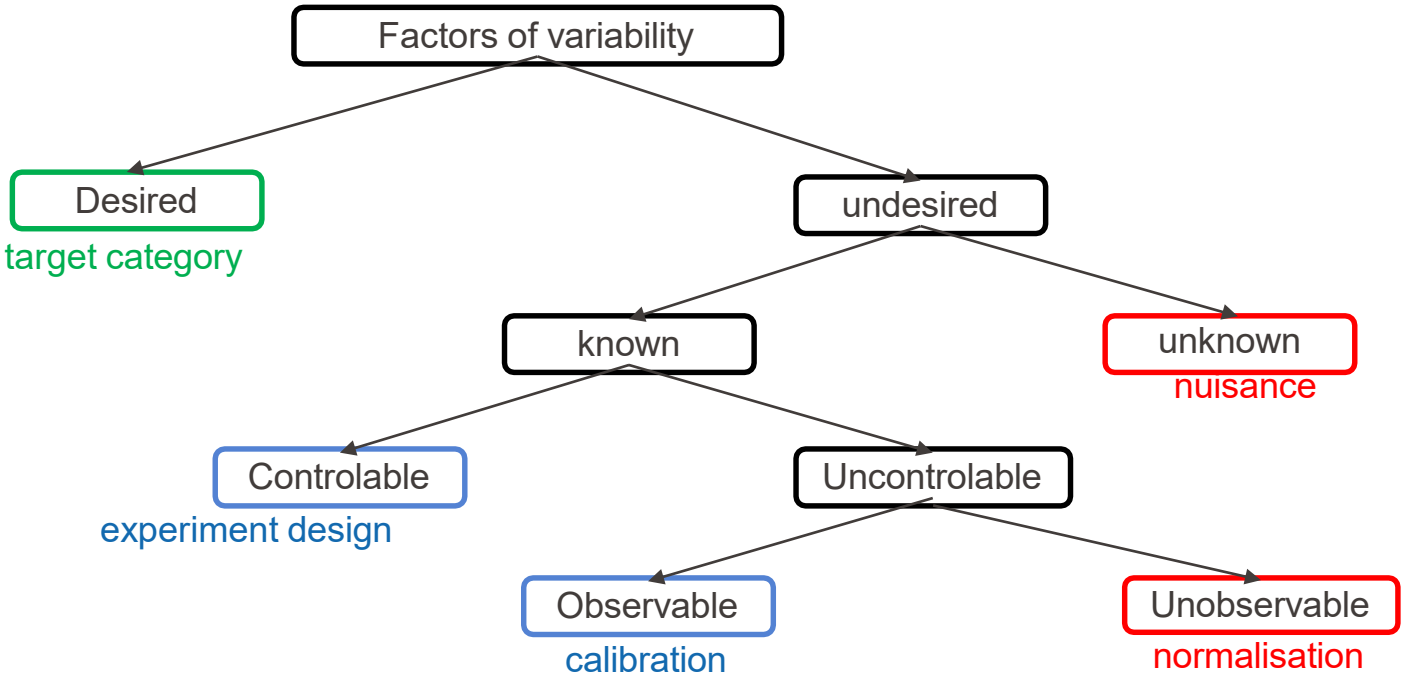
Aims and properties of features

- Explainability
- Parsimony
- Robustness (e.g. to missing data)
- Uncertainty handling
- Link to knowledge and physics…

In fact, many features come from domain know-how

- Domain: Mechanical, structural, thermal, electrical,…
- Kind of system: railway tracks, bridge…
- Components: motor, switching mechanism…

Feature Variability



What is Feature Extraction?

- Feature extraction is the process of transforming raw data into a set of meaningful features that can improve the performance of machine learning models. It reduces the dimensionality of data while preserving essential information, making it easier for algorithms to recognize patterns and make accurate predictions.

- **Reduces Dimensionality**: Helps remove irrelevant or redundant information, improving model efficiency.
- **Enhances Interpretability**: Extracted features are often more meaningful and easier to understand.
- **Improves Model Performance**: Helps machine learning models generalize better by removing noise.
- **Reduces Computational Cost**: Smaller feature sets require less storage and processing power.

Feature extraction

- For failure detection:
  - Features should be dependent on the failure (and failure type)
- Feature tuned to operating conditions
  - feature relevance depends on the OC
- Feature designed based on knowledge:
  - Known relationships
  - known behavior of components

Some feature extraction Approaches

- Descriptive statistical features:
  - For regularly sampled data: moments, correlation, RMS, …
  - For event based data: count, rate, duration, delay, …
- Descriptive models:
  - Distribution/histogram
  - Information based (mutual information)
  - Regression models, curve fitting
  - Classification, clustering (class label as a feature), sequence matching
- Mathematical Transformation:
  - derivative, cumulative sum, power, log, ….(e.g. log if noise depends on amplitude, log normal)
- Time-Series Specific Features
  - Rolling Mean/Variance: Captures local trends.
  - Autocorrelation: Measures how a time-series correlates with its past values.
  - Peak-to-Peak Distance: Identifies signal periodicity.
- Domain-Specific Features
  - Structural Health Monitoring: Modal frequencies, damping ratios.
  - IoT/Industrial Sensors: RMS (Root Mean Square), Zero Crossing Rate.

Statistical features examples (1/3)

Category	Feature	Formula	Interpretation	Use Case
Central Tendency	Mean (μ)	$\mu = (1/N) \sum x_i$	Average value, represents dataset's central tendency.	Detecting overall bridge vibration levels.
	Median	Middle value in sorted data	Robust to outliers, better for skewed data.	Measuring typical structural response.
	Mode	Most frequent value	Identifies the most common occurrence.	Identifying repeating load patterns.
Dispersion (Variability)	Variance (σ^2)	$\sigma^2 = (1/N) \sum (x_i - \mu)^2$	Measures spread; high variance means high variability.	Analyzing variations in structural response.
	Standard Deviation (σ)	$\sigma = \sqrt{\sigma^2}$	Measures how much values deviate from the mean.	Detecting abnormal bridge vibrations.
	Range	max(x) - min(x)	Difference between max and min values.	Checking extreme deflections.
	Interquartile Range (IQR)	Q3 - Q1	Spread of the middle 50% of data, robust to outliers.	Identifying variations in load distribution.

Statistical features examples (2/3)

Category	Feature	Formula	Interpretation	Use Case
Shape (Higher Moments)	Skewness (μ_3)	$\mu_3 = (1/N) \sum [(x_i - \mu)/\sigma]^3$	Measures asymmetry; >0 (right-skewed), <0 (left-skewed).	Detecting imbalance in vibrations.
	Kurtosis (μ_4)	$\mu_4 = (1/N) \sum [(x_i - \mu)/\sigma]^4$	Measures peakness; >3 (leptokurtic), <3 (platykurtic).	Identifying sudden impacts or shocks.
Signal Strength	Root Mean Square (RMS)	$\sqrt{(1/N) \sum x_i^2}$	Measures signal energy, useful for power analysis.	Identifying increasing load stress.
	Peak-to-Peak (P2P)	max(x) - min(x)	Measures the total amplitude range.	Monitoring bridge movement limits.
	Crest Factor	max(|x|) / RMS	Identifies impulsive signals. Higher values suggest faults.	Detecting sudden structural damage.

Statistical features examples (3/3)

Category	Feature	Formula	Interpretation	Use Case
Frequency-Domain	Dominant Frequency	f_max in FFT	Frequency with the highest power in the spectrum.	Identifying resonance frequencies.
	Spectral Entropy	-∑ P(f) log P(f)	Measures randomness in frequency content.	Detecting irregular load distributions.
	Bandwidth	f_high - f_low	Spread of dominant frequency components.	Checking vibration stability.
Time-Series Specific	Autocorrelation	R(τ) = ∑ x_t x_{t+τ}	Measures how similar a signal is to itself at different times.	Identifying fatigue in bridges.
	Entropy (Shannon)	-∑ P(x) log P(x)	Measures signal randomness; higher entropy means more disorder.	Detecting unexpected structural changes.

Statistical Features: Moments

$$\mu_n = \int_{-\infty}^{\infty} (x-c)^n \, f(x) \, \mathrm{d}x = \int_{-\infty}^{\infty} (x-c)^n \, \mathrm{d}F(x)$$
$$= \mathrm{E}\left[(X-c)^n\right]$$

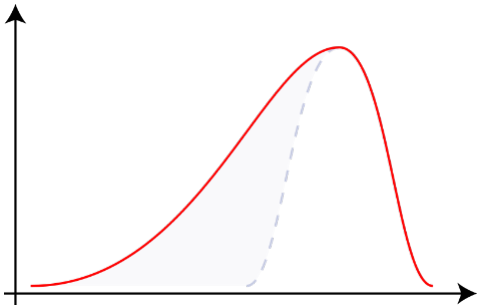
- c = 0 : raw moment
- c =average(X): central moment

- Normalized (central) moment:

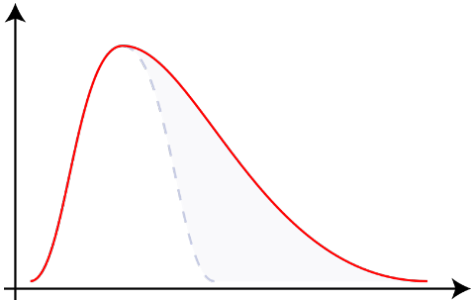
$$\frac{\mu_n}{\sigma^n} = \frac{\mathrm{E}\left[(X-\mu)^n\right]}{\sigma^n}$$

n	**Raw Moment**	**Central M**	**Normalised M**
1	Mean	0	0
2		Variance	1
3			Skewness
4			kurtosis
5			Hyperskewness
6			Hypertailedness
7			

Skewness and Kurtosis

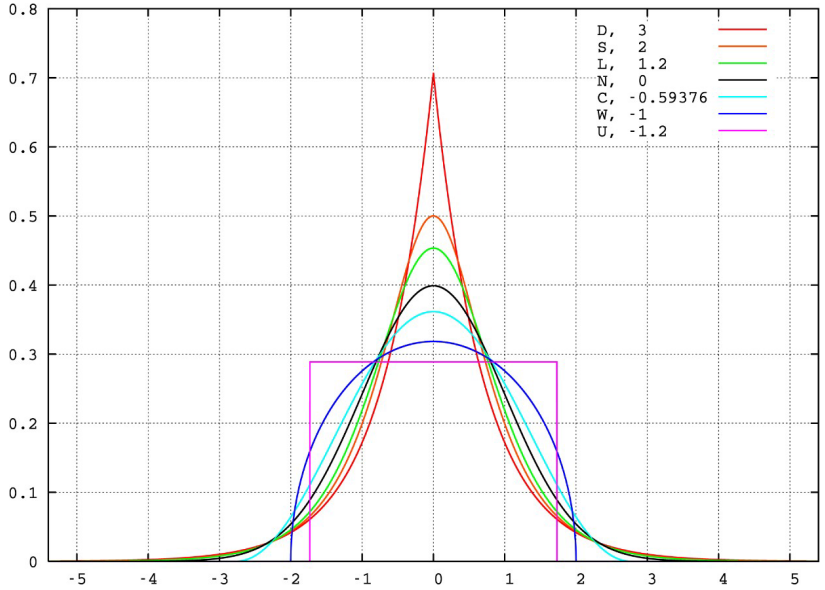


negative skew

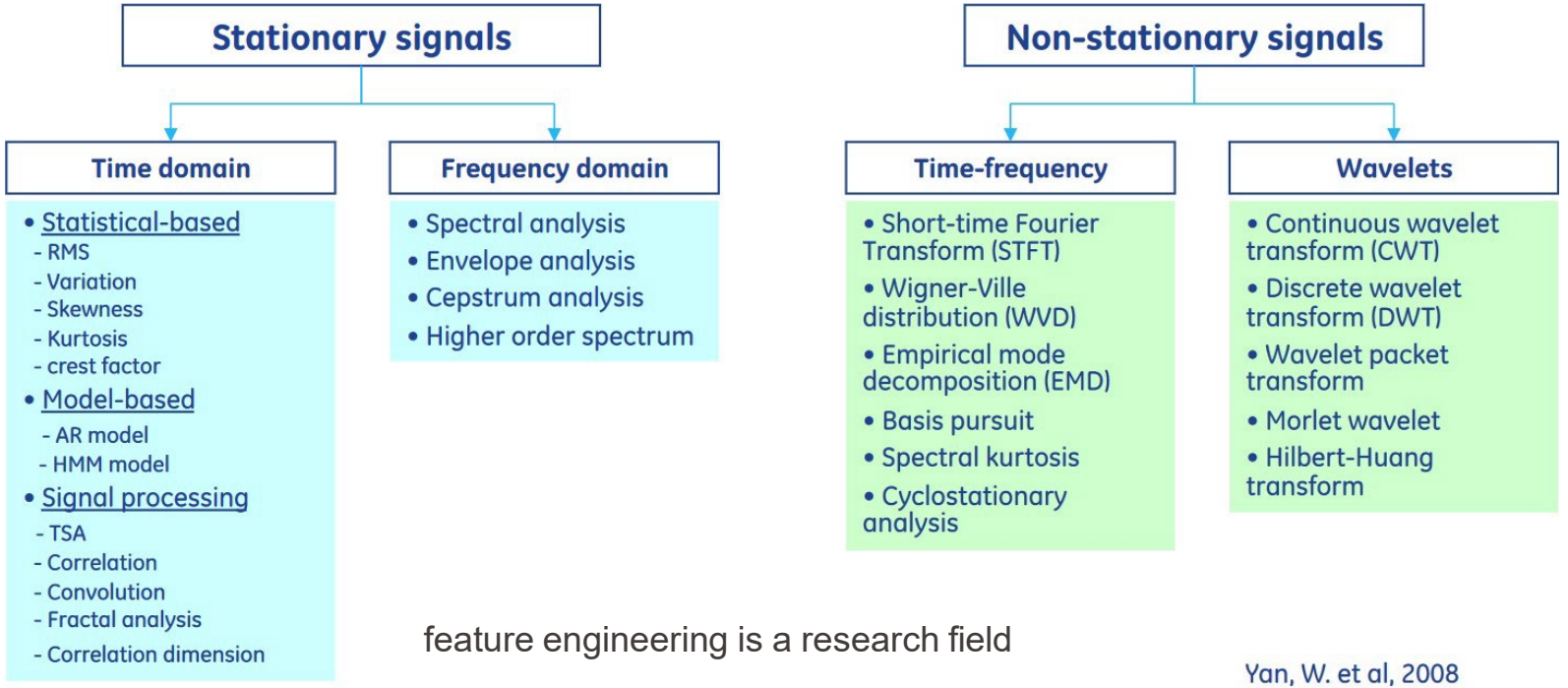


positive skew

Kurtosis



Example: Feature extraction for vibration analysis



feature engineering is a research field

Yan, W. et al, 2008

Some feature extraction approaches

domain know-how

- Physics based :
  - expected input-output relations, etc.
  - comparison to expected output (model)

- Special procedures for data processing:
  - operational regime segmentations, envelop analysis, etc…
  - Time synchronous averaging

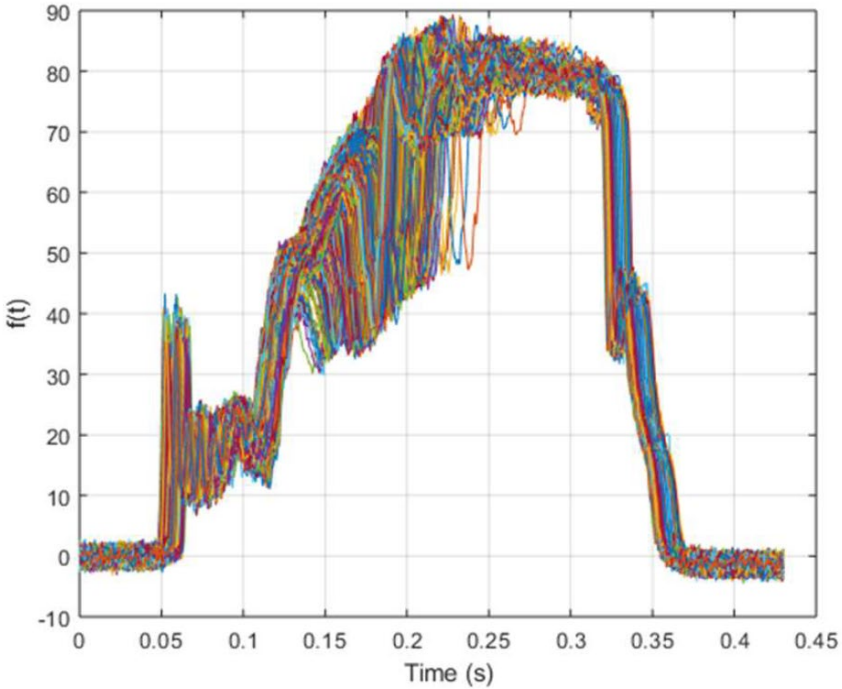
Take away

- Procedure:
  - feature extraction + dimension reduction
- What to extract:
  - data property vs. application domain vs. algorithm requirements
- Feature extraction vs. signal processing
- Feature goodness:
  - Relevance and redundancy
- Feature selection:
  - wrapper approach vs. filter approach vs. embedding
- Feature consistency and sensitivity issues

Feature Engineering vs. Feature Learning

Aspect	Feature Engineering	Feature Learning
Definition	Manually designing features based on domain knowledge.	Automatically extracting features using deep learning.
Approach	Requires human expertise and predefined rules.	Learns representations directly from raw data.
Example	Selecting statistical features in SHM (mean, variance).	Using CNNs to automatically learn patterns in images.

Possible problem here?



Curve registration

- The time may not be directly relevant to the internal dynamics of many real-life systems, observations may be shifted, also missing or imprecise synchronization possible

→ Transform curves by transforming their arguments (aligning the curves)
→ *curve registration*

Shift registration

- We can align curves by fixing the location of a feature, such as the starting time of the process for example
- This works provided the location of this feature is easy to determine in each curve
- We can also align curves by using the entire curve.
- This is always possible, but needs an explicit criterion for alignment

Least squares criterion for shift alignment

- First we estimate a mean function $\hat{\mu}(t)$ for t in T. If the individual functional observations x_i are smooth, we can estimate $\hat{\mu}$ by the sample average $\bar{x}$.
- Then we can minimize this criterion with respect to δ_i.

$$R = \sum_{i=1}^{T} \int_T [x_i(t+\delta_i) - \hat{\mu}(t)]^2 ds = \sum_{i=1}^{T} \int_T [x_i^*(t) - \hat{\mu}(t)]^2 ds$$

- We then iterate this process, by re-computing the mean $\mu(t)$ from the registered curves $x^*_i(t)$, and re-computing a new set of shifts δ_i.
- These iterations usually converge in a few cycles