YOUR SCIENCE

CREATING OPPORTUNITY

# Applied Analytics
# AI & ML for BI & Automation

## 3. Data Cleaning and Preparation with Industrial Applications


Your Science

Mathematical Consulting

Prof. Norbert Poncin


2025


**Your Science Center**: 31, Boulevard Prince Henri, L-1724 Luxembourg

**Phone**: +352 621 674 917, **Email**: info@yourscience.eu, **Website**: yourscience.eu

**Business Permit**: 10154927, **LBR**: A44409, **VAT**: LU35024328

YOUR SCIENCE

CREATING OPPORTUNITY

# Applied Analytics
# AI & ML for BI & Automation

## 3. Data Cleaning and Preparation with Industrial Applications


Your Science

Mathematical Consulting

Prof. Norbert Poncin


2025


**Your Science Center**: 31, Boulevard Prince Henri, L-1724 Luxembourg

**Phone**: +352 621 674 917, **Email**: info@yourscience.eu, **Website**: yourscience.eu

**Business Permit**: 10154927, **LBR**: A44409, **VAT**: LU35024328

# Contents

four key layers, with self-explanatory names: *Source Layer, Transformation Layer, Storage Layer, and Presentation Layer*:

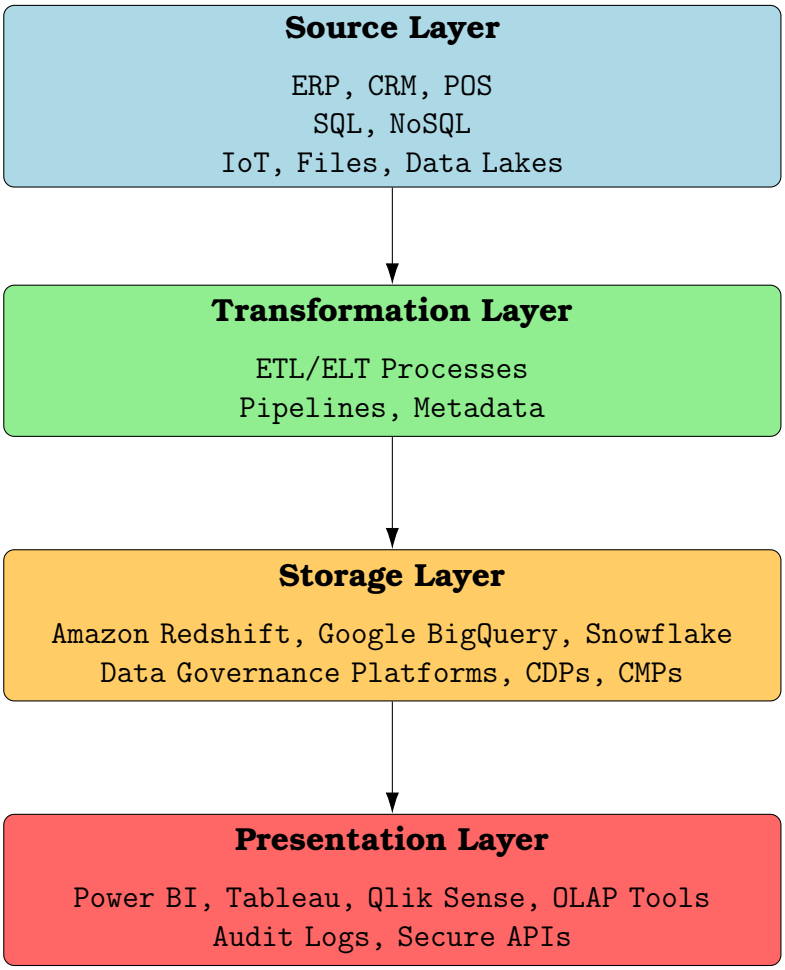



Figure 2: Data Flow Architecture in a Warehouse System

We encourage readers to explore some of the keywords within these layers using their preferred AI tools. The keyword **ETL** in the second layer stands for **Extract, Transform, Load**. The process is fundamental: it *extracts* data from multiple sources, *cleans and standardizes* it for usability and consistency, and *loads* the transformed data into a target system, such as a data warehouse.

This chapter will explore Python-based techniques for transforming data for analysis, focusing on key processes such as **deduplication, imputa-**

**tion, outlier handling, normalization, encoding, scaling, aggregation, splitting, and bias mitigation**.

Python is a leading choice in data workflows (for instance, ChatGPT uses Python for all Data Science tasks) due to its unmatched flexibility and comprehensive capabilities, excelling in:

- **Integration:** Python seamlessly connects diverse tools, platforms, and databases, acting as a *bridge across the data ecosystem*.
- **Extensive Libraries:** With libraries like **NumPy**, **Pandas**, **Scikit-learn**, **TensorFlow**, and **PyTorch**, *Python supports all stages of data processing, analysis, and machine learning*.
- **Automation:** *Automates repetitive tasks*, streamlines workflows, and manages complex pipelines.
- **Customization:** Enables *tailored solutions for unique challenges*, making it adaptable to specific use cases and industries.
- **Data Visualization and Reporting:** Tools like **Matplotlib**, **Seaborn**, and **Plotly** make Python ideal for creating *insightful visualizations and dashboards*.
- **Readability and General Purpose:** Python's *simple syntax and beginner-friendly design* make it accessible for all skill levels and suitable for a *wide range of applications*, from scripting to full-scale software development.
- **Community and Support:** Python's extensive and active community ensures *abundant resources, tools, and ongoing innovation*.
- **Web Development:** Frameworks like **Django** and **Flask** enable the development of *robust web applications and APIs*.

# 2 Deduplication Methods Including Fuzzy and Phonetic Matching

Deduplication refers to suppressing duplicates in the data. Fundamental techniques include **Exact Matching** (identifying identical records) and **Rule-Based Matching** (detecting potential duplicates based on criteria such as similar email addresses). Advanced techniques include **Fuzzy Matching** ('fuzzy' translates to 'flou' in French and 'unscharf' in German, referring to approximate matchings like 'Jon Smith' with 'John Smyth') and **Phonetic Matching** (e.g., matching "Katherine" with "Catherine").

The following code generates a customer feedback dataset.

```
import pandas as pd

# Mock data representing customer feedback from a CRM system
data = {
    "Customer Name": [
        "Alice Johnson", "Alyce Jonson", "Bob Smith", "Robert Smithe", "Charlie Brown",
        "Dave Wilson", "David Wilson", "Eve Davis", "Eva Davies", "Frank Miller",
        "Grace Lee", "Gracie Lee", "Henry White", "Henri White", "Ivy Green"
    ],
    "Email": [
        "alice@example.com", "alyce@example.com", "bob.smith@domain.com", "robert@domain.com", "charlie.brown@domain.com",
        "dave.w@another.com", "david.w@another.com", "eve@feedback.com", "eva@feedback.com", "frank@domain.com",
        "grace@sample.com", "gracie@sample.com", "henry@domain.com", "henry@domain.com", "ivy.green@paperfeedback.com"
    ],
    "Phone Number": [
        "123 456 890", "123 456 890", "456 789 234", "456 789 234", "789 123 567",
        "123 456 890", "123 456 890", "987 654 210", "987 654 210", "456 789 234",
        "654 321 987", "654 321 987", "789 123 567", "789 123 567", "321 654 870"
```

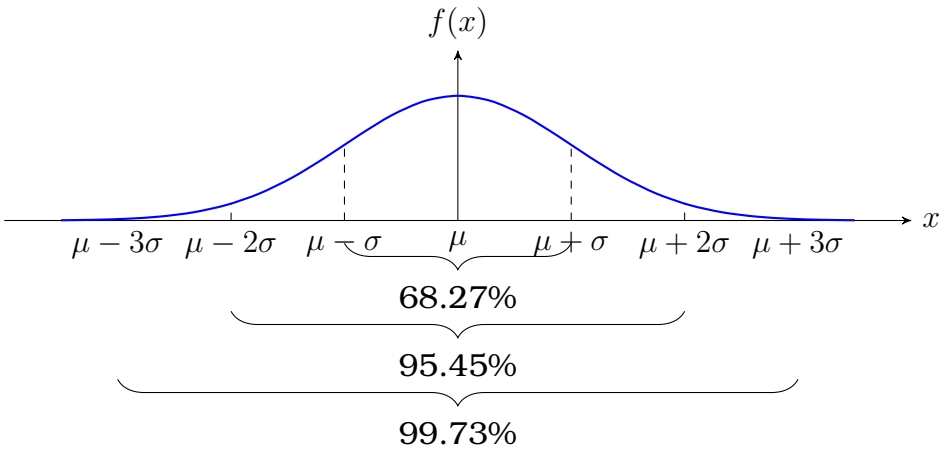



Figure 3: Probability Density of the Normal Distribution $N(\mu, \sigma)$

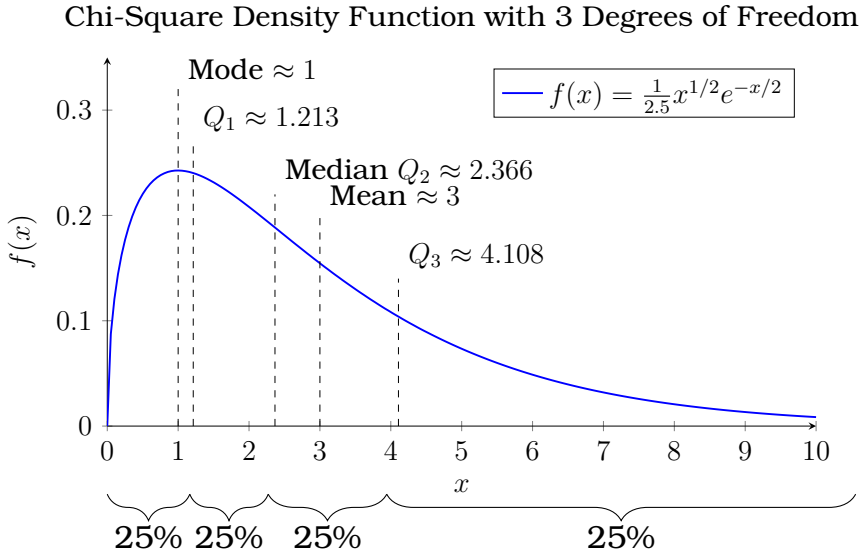



Figure 4: Mode, Mean, Median, Quartiles, and Quartile Intervals

For handling missing data, the choice of imputation method depends on the characteristics of the dataset:

- Mean Imputation ($\mu$): Used for symmetric data distributions without significant outliers, as the mean is sensitive to extreme values.
- Median Imputation ($Q_2$): Suitable for skewed data or datasets with outliers, as the median is robust to extreme values. The median, also

## Standard Deviation and Median Absolute Deviation (MAD)

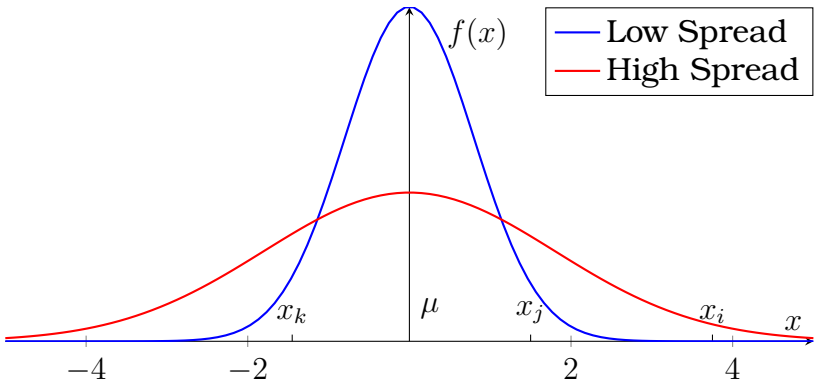



Figure 6: Gaussian Distributions with Different Levels of Spread

A natural approach to quantifying the **spread** of data or the **dispersion** within a distribution, is to compute the mean deviation $x_i - \mu$ of data points $x_i$ from their mean $\mu$ – in order to avoid negative deviations, either as the mean of the absolute deviations $|x_i - \mu|$ or, more commonly, the mean of the squared deviations $(x_i - \mu)^2$. This provides a single value that reflects how much data points tend to differ from the mean, thereby capturing the overall spread of the data.

1. **Standard Deviation**: This is defined as the square root of the mean of the squared deviations from the mean:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

   where $N$ is the number of data points. Here, the squaring of deviations (to eliminate negative values) and the square root (to revert to the original units) are essential steps in the formula. Standard deviation works well with normally distributed data.

2. **Median Absolute Deviation** (MAD): MAD, unlike the standard devia-

tion, computes the median of the absolute deviations from the median:

$$\text{MAD} = \text{median}(|x_i - \text{median}(x)|) \ ,$$

where notations are self-explanatory. By using the median (in place of the mean) and the absolute value (instead of squaring), MAD becomes a robust measure, less sensitive to extreme values and better suited to non-normal distributions or datasets with outliers.

# 5 Outlier Detection Using Visual and Statistical Approaches

When computing the average salary of the 10 working inhabitants of a small village in Luxembourg, whose monthly salaries range between € 1,500 and € 8,750, one outlier – a billionaire finance expert earning € 100,000 per month – would heavily skew the mean salary. This highlights the importance of addressing outliers to ensure more accurate statistics.

We will confine ourselves to simple outlier detection techniques:

- *Visual Inspection*: histograms, box plots, scatter plots (ideal for low-dimensional data).
- *Statistical Methods*: Z-score (for normally distributed data), IQR (robust to non-normality), and Z-score based on MAD (robust to skewed or heavy-tailed distributions).

## Visual Techniques for Outlier Detection: A Concrete Example

The following code generates satisfaction scores, manually adds outliers, and provides clear definitions for both **mild and strong outliers**. It then visualizes these outliers using a histogram, a boxplot, and a scatter plot, making it easy to identify each type visually.

A **boxplot** provides a concise summary of the data distribution by displaying the three **quartiles** – values that divide the data into four equal parts, or quarters: the **median** ($Q_2$), the 25th ($Q_1$) and 75th ($Q_3$) **percentiles**, and any outliers. The 'box' represents the **interquartile range** (IQR), computed as IQR $= Q_3 - Q_1$, with '**whiskers**' extending to the smallest and largest data points within 1.5 times the IQR from $Q_1$ and $Q_3$, respectively. Any data points beyond these whiskers are marked as outliers.

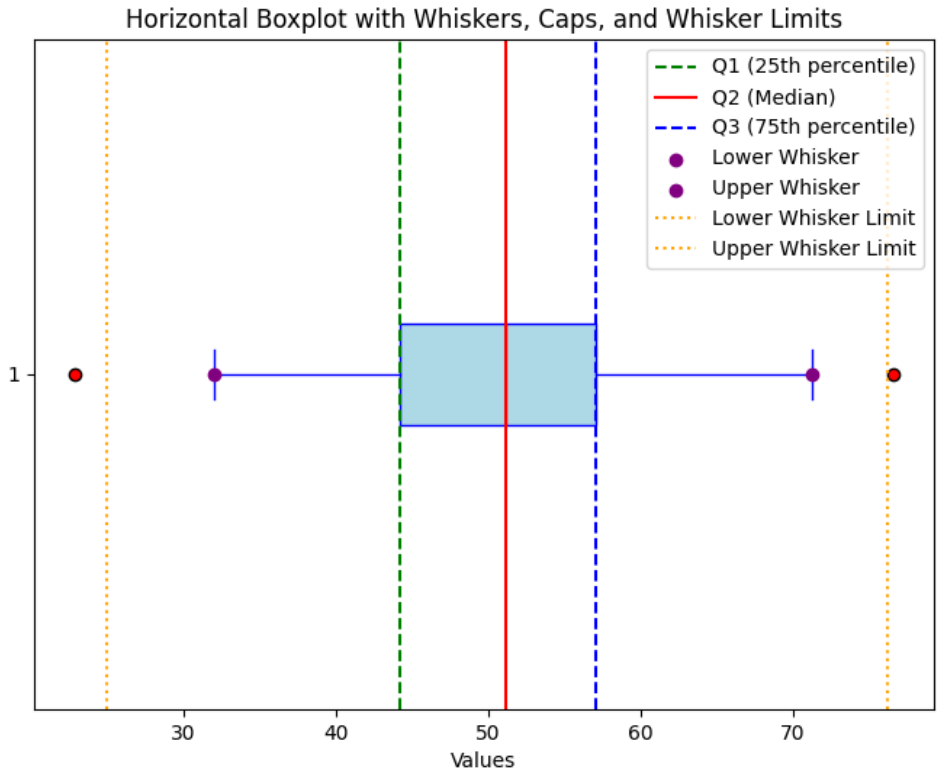
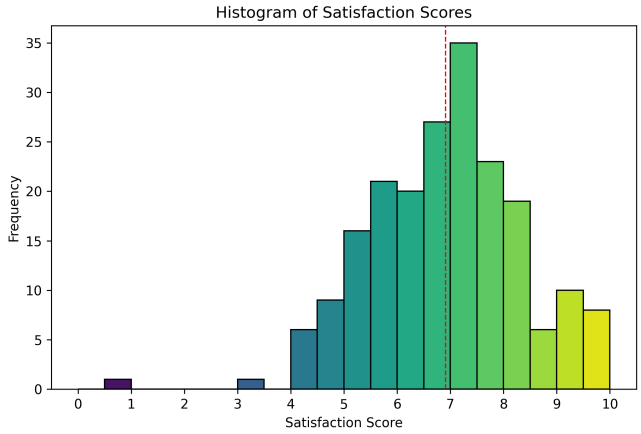
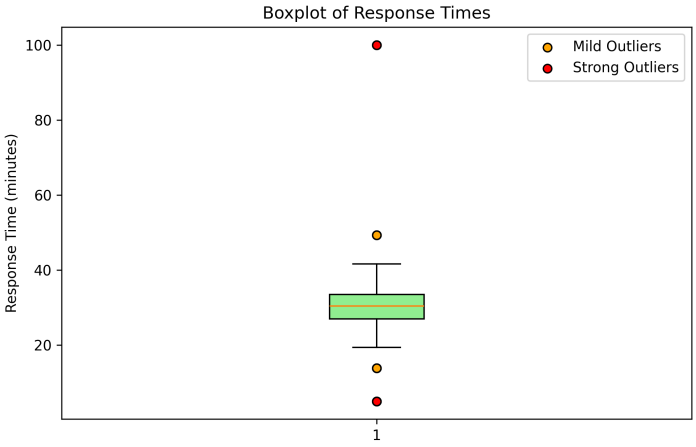


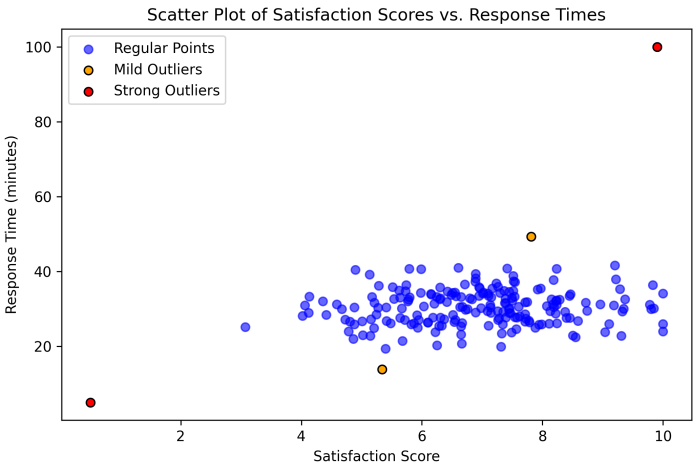

Figure 7: Horizontal Boxplot

**Exercise 5.** *Provide clear explanations of the definitions for mild and strong outliers as applied in the following code. Then, run the code to observe how the outliers are identified and explain the resulting well-designed output figures in detail.*

```
1 import numpy as np
```


Histogram of Satisfaction Scores
Frequency
Satisfaction Score
0
5
10
15
20
25
30
35
0
1
2
3
4
5
6
7
8
9
10


Boxplot of Response Times
Response Time (minutes)
Mild Outliers
Strong Outliers
20
40
60
80
100
1


Scatter Plot of Satisfaction Scores vs. Response Times
Response Time (minutes)
Satisfaction Score
Regular Points
Mild Outliers
Strong Outliers
20
40
60
80
100
2
4
6
8
10

## Statistical Techniques for Outlier Detection: A Concrete Example

The following code uses the same example as the visual techniques and computes the Z-score and the MAD-score for each of the features Satisfaction_Score and Response_Time to detect outliers.

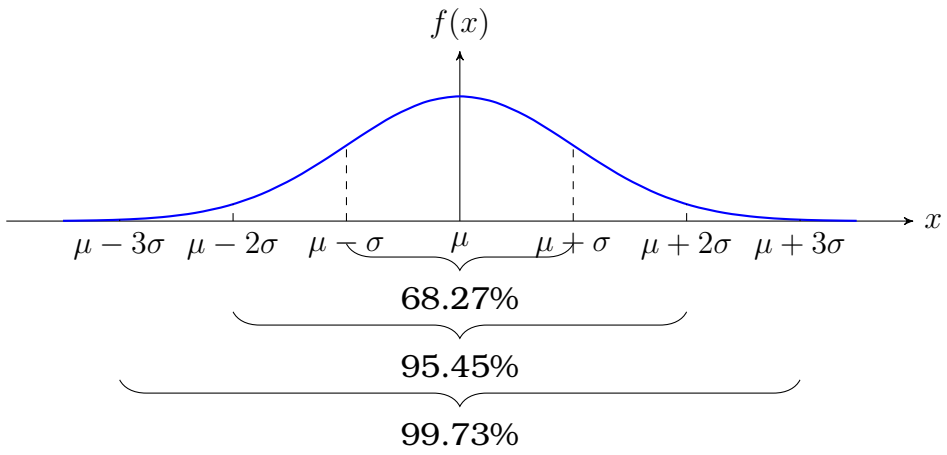



Figure 8: Probability Density of the Normal Distribution $N(\mu, \sigma)$

**Exercise 6.** *Note that a data point* $x_i$ *lies outside the interval* $[\mu - 3\sigma, \mu + 3\sigma]$ *if and only if its Standard Z-Score* $z_i = \frac{x_i - \mu}{\sigma}$ *falls outside the interval* $[-3, +3]$ *:*

$$x_i < \mu - 3\sigma \text{ or } x_i > \mu + 3\sigma \Leftrightarrow z_i = \frac{x_i - \mu}{\sigma} < -3 \text{ or } z_i = \frac{x_i - \mu}{\sigma} > 3 \Leftrightarrow |z_i| > 3 \, .$$

*Provide a clear explanation of how outliers are defined in the code below. Then, execute the code to identify the outliers and compare those detected using statistical methods with the outliers observed visually in the scatter plot.*

```
import numpy as np
import pandas as pd

# Set the global display format for floats to one decimal place; in {
    variable:.1f}, the : after the variable name tells Python that a
    format specification follows; in the specification .1f, the f
    stands for a floating-point number, and .1 specifies one decimal
```

```
      Satisfaction_Score  Response_Time
9                    7.8           49.3
62                   5.3           13.8
200                  0.5            5.0
201                  9.9          100.0

 All detected outliers:
      Satisfaction_Score  Response_Time
200                  0.5            5.0
201                  9.9          100.0
9                    7.8           49.3
62                   5.3           13.8
```

**Remark 3.** Less advanced readers may skip this remark on their first reading. In the code above, as well as in the scaling methods at the beginning of Section 4, we computed the MAD-score by dividing the MAD by $0.6745$. The reason for this division is as follows.

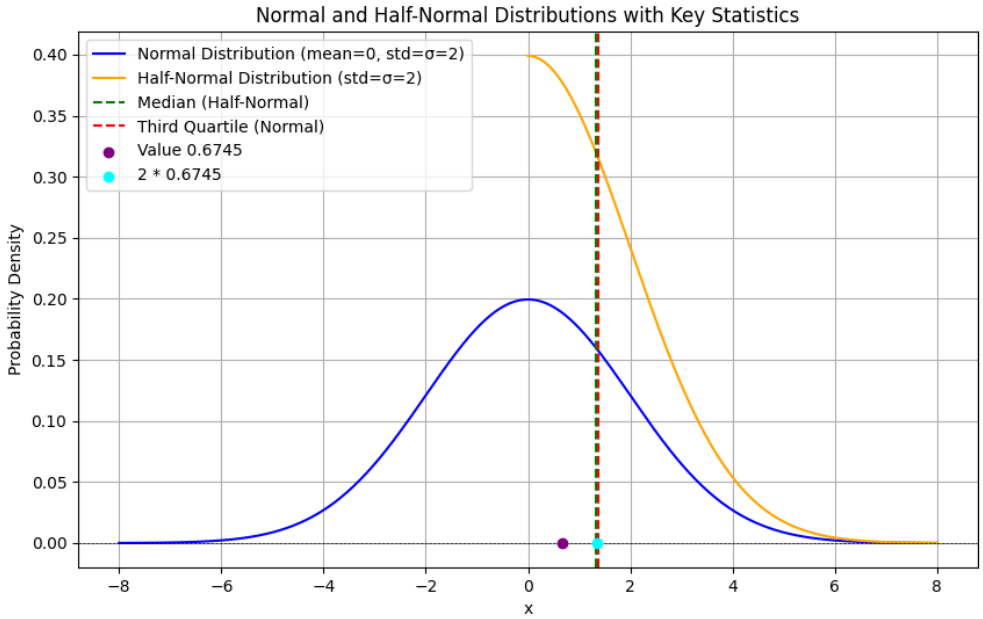
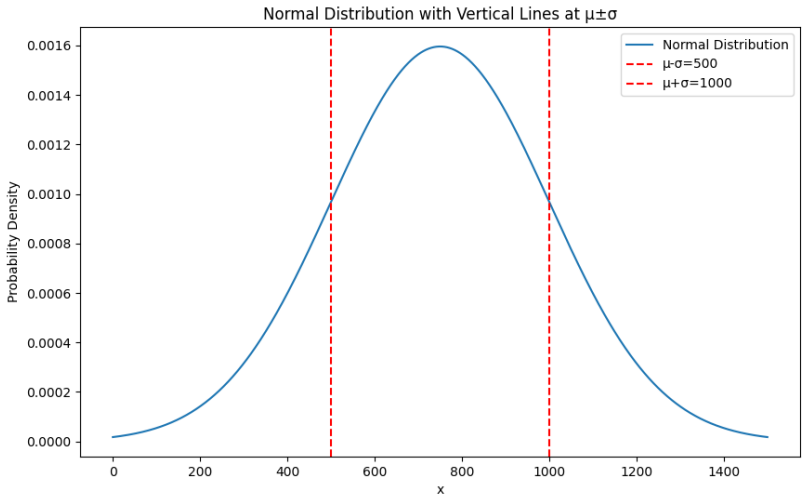
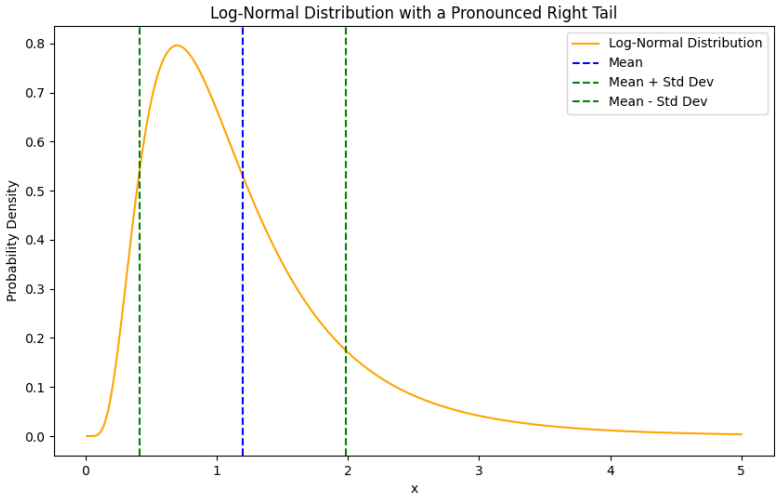



Figure 9: MAD and Standard Deviation




- **Machine Efficiency** (%): Computed as production divided by a maximum capacity (1100 kg) and scaled by 100.
- **Energy Consumption** (EC) (kWh): Computed as baseline consumption (BL) plus a term proportional to production (Prod), with added log-normal noise (LogNor) for variability:

$$\mathrm{EC} = \mathrm{BL} + 0.05 * \mathrm{Prod} + \mathrm{LogNor} \ .$$




The log-normal distribution is a natural choice for modeling energy

## 7.2 File Watcher and Automatic Data Preprocessor

The following Python code automates the merging, cleaning, and preparation of production data for analysis. It begins by combining an initial set of files matching the naming pattern `productionk_data.csv` (where $k \geq 1$) from the Downloads folder. Subsequently, it monitors the folder for new files matching the same pattern, automatically updating and preprocessing the combined dataset. The preprocessing pipeline includes:

- **Deduplication**: Ensures only the first occurrence of records with identical dates is retained.
- **Missing Value Imputation**: Applies KNN imputation for numeric columns.
- **Outlier Removal**: Uses the robust MAD rule with an equivalent of a 3 Sigma threshold to remove outliers.
- **Scaling**: Scales all numeric columns using MAD scaling to normalize values.

Continuous monitoring of the Downloads folder is achieved through Python's **watchdog** library, enabling real-time updates to the cleaned dataset whenever new files are added to the folder.

### Python Code

```
import pandas as pd
import os
import re
# Library used for performing operations with regular expressions,
    such as pattern matching, searching, and text manipulation

from sklearn.impute import KNNImputer
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler
import time

```

```
53        2025-01-16 21:15:43     668.00      55.67         54.29
54        2025-01-17 21:15:43     758.59      63.22         58.80
55        2025-01-18 21:15:43     200.00      58.56         31.02
56        2025-01-19 21:15:43     979.39      81.62         69.99
57        2025-01-20 21:15:43     949.88      79.16         68.59
58        2025-01-21 21:15:43     556.16      46.35         48.71
59        2025-01-22 21:15:43        NaN      59.81         20.85
60        2025-01-23 21:15:43     649.54      54.13         53.43

Production Data saved to production2_data.csv
```

**Exercise 7.** *Analyze the previous code and identify any outliers that have been implemented. Keep in mind that duplicates are not immediately visible unless you compare the December production data with the January production data. Upon closer inspection, you will notice that both tables contain a row corresponding to December 25, 2024.*

## 7.5 JupyterLab Output of Cleaned Data by Watcher and Preprocessor

```
New CSV detected: C:/Users/norbert.poncin/Downloads\production2_data.csv
Updated Combined DataFrame:
```

| prod._id | timestamp | prod. | machine_eff. | energy_cons. |
|---|---|---|---|---|
| 1 | 2024-11-26 20:55:58 | 695.91 | 63.26 | 55.69 |
| 2 | 2024-11-27 20:55:58 | 593.11 | 53.92 | 50.69 |
| 3 | 2024-11-28 20:55:58 | 538.08 | 48.92 | 47.93 |
| 4 | 2024-11-29 20:55:58 | 840.99 | 76.45 | 63.16 |
| 5 | 2024-11-30 20:55:58 | 514.12 | 46.74 | 46.65 |
| 6 | 2024-12-01 20:55:58 | 607.15 | 55.20 | 51.31 |
| 7 | 2024-12-02 20:55:58 | 664.62 | 60.42 | 54.00 |
| 8 | 2024-12-03 20:55:58 | 658.70 | 59.88 | 54.03 |
| 9 | 2024-12-04 20:55:58 | 798.15 | 72.56 | 60.87 |

| | | | | |
|---|---|---|---|---|
| 56 | 2025-01-19 21:15:43 | 1.83 | 1.59 | 1.81 |
| 57 | 2025-01-20 21:15:43 | 1.64 | 1.40 | 1.64 |
| 58 | 2025-01-21 21:15:43 | -0.87 | -1.07 | -0.78 |
| 60 | 2025-01-23 21:15:43 | -0.27 | -0.49 | -0.21 |

**Exercise 8.** *Curious about how we set up the automated data integration and real-time preprocessing pipeline? Explore key parts of the code for the file watcher and automatic data preprocessor, then run the examples to see it all in action! Compare the output in the command prompt with rows 31 to 60 of the tables in the JupyterLab Notebook.*

# 8 Self-Paced Challenge: Manufacturing Data Analysis

In this exercise, you will work with the dataset created by the following code and representing production data from a manufacturing process.

```
import pandas as pd
import numpy as np
import random

# Set a random seed for reproducibility
np.random.seed(42)

# Number of rows
n_rows = 100

# Feature generation
# 1. Production volume (normally distributed, Z-Score scaling applicable)
production_volume = np.random.normal(loc=1000, scale=200, size=n_rows)

# 2. Machine efficiency (requires MAD-Score scaling)
machine_efficiency = np.random.uniform(50, 100, size=n_rows)
machine_efficiency[[10, 20]] = [20, 120]  # Add weak and strong outliers
```

```
54
55 # Save the dataset to a CSV file
56 data.to_csv("C:/Users/norbert.poncin/Downloads/manufacturing_data.csv
       ", index=False)
57
58 # Replace the path "C:/Users/norbert.poncin/Downloads/" with the path to your
59 # Downloads folder or erase this absolute path
60 # If you erase the absolute path the file "manufacturing_data.csv"
       will be saved to the current working directory of your script or
       notebook. If you want to know this current directory, you can
       `import os' and `print(os.getcwd())' to display the current
       working directory
61
62 print("Manufacturing dataset generated and saved to '
       manufacturing_data.csv'.")
```

The dataset has been designed to include realistic challenges commonly faced in data analysis and machine learning:

- **Data Deduplication**: Identify and remove duplicate rows caused by system logging issues.
- **Outlier Handling**: Address weak and strong outliers in features like `Machine_Efficiency` and `Defect_Rate`.
- **Scaling**: Scale features for KNN analysis:
  - Use Z-Score scaling for approximately normally distributed features without significant outliers. To verify normality, apply a test such as Shapiro-Wilk or Anderson-Darling.
  - Use MAD-Score scaling for non-normally distributed features with outliers.
- **Imputation**: Handle missing values in `Energy_Consumption` using KNN imputation. Experiment with different $k$ values and distances (e.g., Manhattan distance).
- **Encoding**: Encode the categorical feature `Machine_Type` for further analysis.

- **Feature Engineering**:
  - Aggregate features (computing statistical metrics like mean, sum, median, standard deviation, or other summary statistics) such as `Production_Volume` and `Energy_Consumption`.
  - Create and join a new column of supplier data (e.g., `Non-Conformance Rate`, i.e., percentage of delivered goods that fail to meet quality standards).
- **Modeling**:
  - Split the data into training and testing sets.
  - Fit a linear regression model to predict `Defect_Rate` based on other features. Use, for instance, a correlation matrix to justify your feature selection.
  - Use the model to predict `Defect_Rate` for new observations.

By completing this exercise, you will gain hands-on experience in preparing and analyzing real-world manufacturing data.

Have fun!

# 9 Learning Outcomes

After working through this chapter, the reader should be able to:

- **Understand Data Warehousing and ETL Processes**: Explain the layers of the Python Data Ecosystem and their correspondence with warehouse layers, including the role of ETL in data transformation and integration.
- **Perform Data Cleaning and Preparation**: Describe and apply key data transformation processes such as deduplication, imputation, and scaling, while understanding their importance for data quality and analysis accuracy.
- **Apply Advanced Deduplication Techniques**: Use methods like fuzzy and phonetic matching to efficiently identify and merge duplicate records, while understanding the underlying encoding and logic behind these methods.
- **Implement Robust Imputation Techniques**: Explain and apply KNN and regression imputation methods to handle missing data, while recognizing different missing data mechanisms (MCAR, MAR, MNAR) and their impact on imputation strategies.
- **Use Scaling to Normalize Data**: Understand and apply scaling methods such as Z-Score and Median Absolute Deviation. Explain the importance of dispersion measures like standard deviation and MAD in scaling and managing data with varying distributions.
- **Detect and Handle Outliers**: Visualize outliers using boxplots and scatter plots, and apply statistical approaches like the 3-Sigma criterion or robust alternatives to remove outliers effectively.
- **Explain Additional Preprocessing Steps**: Describe key concepts of data preprocessing and data wrangling, including feature engineering and modeling preparation, while understanding their significance for improving data-driven models.

- **Automate Data Cleaning and Preparation**: Explain the use of truncated normal and log-normal distributions in simulations, and demonstrate the coding and implementation of automated data cleaning processes in real-world applications.
- **Analyze Production Data**: Apply learned preprocessing techniques on production data, interpret outputs from automated tools, and evaluate the results to ensure data quality and reliability for further analysis.
- **Develop Manufacturing Data Analysis Skills**: Complete self-paced exercises on manufacturing data analysis to reinforce data preparation, feature engineering, and modeling concepts.

YOUR SCIENCE

CREATING OPPORTUNITY

# About the Author

Norbert Poncin is a Luxembourgish mathematician, who was originally educated as a mathematical analyst and has worked extensively in partial differential equations (PDEs) at the University of Liège. His Master's thesis focused on the propagation of singularities in boundary value problems (BVPs) for dynamic hyperbolic systems. Applying the finite element method (FEM), his subsequent dissertation addressed BVPs for complex elliptic systems of PDEs. For his doctoral thesis, he explored mathematical quantization, while his post-doctoral education at the Polish Academy of Sciences strongly emphasized theoretical physics and its models.

Norbert has served as a Full Professor of Mathematics at the University of Luxembourg for more than 25 years and collaborated with more than 25 foreign professors and post-doctoral scholars. He has organized numerous academic events, notably approximately 10 international research meetings and over 20 research seminars focusing on theories, frameworks, concepts and models in Physics and Engineering. Beyond a substantial publication record in Differential Geometry, Algebraic Topology, and related disciplines, he has contributed roughly 25 papers to the fields of Mathematical Physics and Quantum Theory.

He was the leading instructor for over 20 university courses. Spanning a diverse spectrum of subjects, including mathematical analysis, probability theory, inferential statistics, point and solid dynamics, Lagrangian and Hamiltonian mechanics, mechanics of deformable solids, fluid dynamics, special relativity, quantum physics, geometric methods in mathematical physics, and supersymmetric models, his teaching portfolio underscores his extensive experience in applied aspects of mathematics.

In 2023, Norbert Poncin founded the mathematical consulting agency Your Science, where he currently serves as director. His primary interests include data science and artificial intelligence, along with mathematical modeling and computational science.