FIELD OF SCIENCE: Engineering and Technology

DISCIPLINE OF SCIENCE: Information and Communication Technology (ICT)

DOCTORAL DISSERTATION

Fixing data quality issues in CMDB in IT and OT by using machine learning algorithms. Forecasting for IT procurement

Szymon Niewiadomski

Suppervisor: dr hab. inż. Grzegorz Mzyk, University Professor.

Keywords: Data Cleaning; Configuration Management; CMDB; Kernel Density Estimation; Mutual Information; Rajski Distance; Jaccard Entropy; Outlier Detection; Procurement Forecasting; Genetic Algorithm

WROCŁAW, 2025

Contents

List of Principal Notations

Symbol	Description
$X^{(j)}$	jth feature (random variable); $X = \{X^{(1)}, \ldots, X^{(M)}\}$ is the feature set
D_{good}, D_{bad}	Subsets of records labeled (or trusted as) correct vs. erroneous (used for density comparisons)
$\hat{p}^{(j)}_{\text{good}}(x)$, $\hat{p}^{(j)}_{\text{bad}}(x)$	Smoothed (discrete/binary) class-conditional probability estimates for $X^{(j)}$ on D_{good} and D_{bad}
$\hat{f}^{(j)}_{\text{C}}(x)$, $\hat{f}^{(j)}_{\text{E}}(x)$	KDE class-conditional densities for continuous $X^{(j)}$ (classes C/E)
$K(\cdot)$	Kernel function (e.g., Gaussian) used in KDE
h, h_{C}, h_{E}	Bandwidth parameter(s) in KDE (Silverman/Scott rules as defaults)
$I(X;Y)$	Mutual information
$H(X)$, $H(Y)$	Marginal entropies
$H(X,Y)$	Joint entropy
$D_{\text{Rajski}}(X,Y)$	Rajski (probabilistic Jaccard) distance: $2H(X,Y) - H(X) - H(Y)$
σ, IQR	Standard deviation and interquartile range
δ_j	Discriminative score of feature j (divergence between class-conditional distributions)
J_{jk}	Pairwise distance between features j, k (Rajski/Jaccard-style)
Δ_j	Informativeness of feature j w.r.t. already selected features (marginal contribution)
α	Weight in the combined score $Score_j = \alpha\,\delta_j + (1-\alpha)\,\Delta_j$, $\alpha \in [0,1]$
τ_δ, τ_J	Thresholds for discarding weak features ($\delta_j < \tau_\delta$) and pruning redundancy ($J_{jk} < \tau_J$)
S_j	Support set of records where $X^{(j)}$ is active (for binary features)
N, N_{B}, N_{C}	Counts of all records, erroneous-class records, and correct-class records.
R, $\omega(R)$	A CMDB record and its (true) class label
(r_i, d_i)	A record and the verification decision assigned by experts
$G = (V, E, \varphi, \psi)$	CMDB as labeled, attributed, directed multigraph: vertices V, edges E, node-attribute map φ, edge-type/metadata map ψ
Σ	Schema (allowed relation types and cardinalities)
R_c, $a_{\text{name}}(v)$	Pattern family admissible for class c; identifier of node v (used in naming-rule checks)
$\{\tau_i\}_{i=0}^{n}$	Discrete decision times (e.g., weekly purchase opportunities)

$\Delta = \tau_i - \tau_{i-1}$	Time step between decisions
m_i, $\mathbf{m}$	Quantity purchased at τ_i; decision vector (purchasing strategy)
$p(t, m)$	Unit-price function (time/volume dependent)
$u(t)$	Demand function
$\xi(m)$	(Nonlinear) procurement process cost (e.g., threshold-driven overhead)
r	Interest rate (used for discounting when relevant)
L	Resource lifetime (time in service for a unit once delivered)
$s_i(t)$	Availability from order i: $s_i(t) = m_i\big(\mathbf{1}(t - \tau_i) - \mathbf{1}(t - \tau_i - L)\big)$
$\sum_i s_i(t) \geq u(t)$	Feasibility (capacity–demand) constraint for all t in the horizon
$Q(\mathbf{m})$	Objective (total procurement cost): $\sum_i \big(m_i p(\tau_i, m_i) + \xi(m_i)\big)$
$H = \lfloor L/\Delta \rfloor$	Number of decision periods a purchase remains effective (used in GA seeding; termed h in Ch. 3)
$m_{i,\min}$, $m_{i,\max}$	Bounds ensuring feasibility over $[\tau_{i+1}, \tau_{i+H}]$
$\hat{P}(q, t)$	Price estimator (poly in t with a $\log q$ term)
X, β, $\mathbf{p}$	Design matrix, coefficient vector (LS), and vector of observed historical prices
y_t, λ	AR(1) demand component: $y_t = \lambda y_{t-1} + \varepsilon_t$
Y_d, Y_m, Y_a	Dynamic baseline, multiplicative events, additive events; demand model $Y = Y_d Y_m + Y_a$
$q(\cdot)$	Fitness (selection) function used in parent selection
$Q_{\max}$	Max cost used for scaling in roulette-wheel selection
η	Stopping or ranking threshold (e.g., for top-K review or convergence)

List of Tables

List of Figures

Chapter 1

Introduction

This doctoral dissertation focuses on two IT Service Management processes that are particularly challenging to implement: configuration management and capacity management. Configuration Management Databases (CMDBs) underpin change, incident, and capacity management in complex environments, yet they are prone to data quality defects that degrade service reliability and cyber-resilience.

We present a transparent, modular, and on-premise machine learning framework that materially elevates CMDB quality while respecting stringent security, auditability, and staffing constraints typical of critical infrastructure operators.

The second contribution addresses the forecasting for IT procurement under demand, price, availability, and financing constraints. The dissertation formulates a cost-minimization problem with price and demand estimators and proposes a genetic algorithm with principled seeding, selection, crossover, and mutation to search feasible purchasing schedules.

1.1 Dissertation Thesis

This dissertation demonstrates that machine learning algorithms, specifically designed with full transparency, modular architecture, and analytically justified error detection mechanisms, can significantly improve the quality of data stored in Configuration Management Databases (CMDB) across both IT and OT domains. The proposed methods—including recursive kernel probability density function estimation, Jaccard entropy measures, and interpretable neural networks developed from first principles—do not rely on opaque heuristics or black-box libraries; instead, they are grounded in well-defined mathematical frameworks.

Theoretical contributions of this work include: (1) the formulation of novel recursive learning algorithms that incorporate forgetting and rollback mechanisms within a nonparametric probabilistic framework; (2) the analytical and experimental comparison of distance- and correlation-based methods for anomaly detection in heterogeneous configuration data; and (3) the development of criteria for validating model robustness and reliability in operational environments.

These innovations contribute not only to the understanding of data quality assurance in complex digital infrastructure systems but also offer practical value through industrial deployment in the energy sector, enabling more accurate IT resource forecasting and proactive maintenance planning.

1.2 The Industrial Doctorate Programme in Poland (Implementation PhD)

This dissertation has been conducted within the framework of the Polish *Industrial Doctorate Programme* (*Doktorat wdrożeniowy*), a national initiative launched by the Ministry of Science and Higher Education of the Republic of Poland in 2017. The primary objective of this programme is to strengthen collaboration between academia and industry by supporting doctoral research that is directly embedded within the context of an enterprise's real-world operations.

Unlike traditional doctoral studies, where the research is typically carried out exclusively at a university or research institute, the industrial doctorate model involves two parallel affiliations: the doctoral candidate is employed by a company while simultaneously enrolled in a doctoral programme at a higher education institution. The research undertaken must address a concrete problem identified by the industrial partner and should lead to both scientific advancement and practical implementation within the company.

Our project complies with the requirements of the ministerial programme and exhibits the following characteristics:

- A research problem defined jointly by the university and the enterprise, with relevance to both scientific inquiry and industrial application.
- Supervision provided by an academic advisor from the university and a mentor from the company, ensuring alignment of scientific rigor with operational needs.
- Implementation of the research outcomes in the business environment of the industrial partner, typically in the form of a technical, procedural, or organizational innovation.

Participation in the programme required the submission of a formal implementation plan that had been approved by both academic and industrial institutions. This doctoral dissertation presents a solution that is applicable in industry and significant for the effective management of IT services in enterprises.

The present doctoral research was conducted in cooperation with a company from the energy sector, focusing on the use of artificial intelligence methods to improve data quality in Configuration Management Databases (CMDBs) and to support forecasting in IT procurement processes. This industrial context has played a key role in shaping both the problem formulation and the methodological approach adopted in this work.

1.3 Business Context

1.3.1 Gaz System Company Information

GAZ-SYSTEM S.A. is a strategic company of key importance to Poland's national energy security. It is the operator of the national natural gas transmission system in Poland, responsible for the transport of natural gas and the management of the high-pressure gas pipeline infrastructure. The company plays a central role in ensuring the safe, reliable, and efficient supply of natural gas across the country, and in integrating the Polish gas system with the broader European market.

The core activities of GAZ-SYSTEM include the planning, development, operation, and maintenance of gas transmission infrastructure. This encompasses both domestic transmission pipelines and interconnections with neighboring countries, as well as critical infrastructure such as compressor stations, underground gas storage facilities (in cooperation with other operators), and gas hubs. The company is also responsible for the operation of the LNG terminal in Świnoujście, which significantly enhances Poland's capacity to diversify its sources of gas supply.

As part of its long-term strategy, GAZ-SYSTEM is actively engaged in projects that enhance energy transition and decarbonization. This includes participation in European initiatives promoting low-carbon fuels, such as hydrogen and biomethane. Moreover, the company contributes to the European Union's energy policy goals by investing in cross-border interconnectors and infrastructure expansion projects co-financed by the EU under programs such as Connecting Europe Facility (CEF).

From the organizational perspective, GAZ-SYSTEM is characterized by a complex and technologically advanced IT and OT (Operational Technology) environment. Its operations depend on a wide array of technical systems, including SCADA platforms, asset and maintenance management systems, geospatial databases, and enterprise resource planning (ERP) solutions. The integration and quality management of data within these systems are critical for ensuring continuity of operations, regulatory compliance, and effective asset lifecycle management.

The implementation project described in this dissertation is conducted in cooperation with GAZ-SYSTEM S.A. and focuses on the development and deployment of advanced methods for improving data quality in Configuration Management Databases (CMDBs) and forecasting models for IT procurement planning. The results of this research aim to provide scalable and explainable machine learning solutions aligned with the specific operational and strategic needs of energy sector enterprises.

1.3.2 Business Constraints in the Gaz-System Implementation Environment

The implementation of machine learning algorithms for Configuration Management Database (CMDB) data quality assurance at Gaz-System S.A. must operate under a specific set of business and technical constraints. These constraints arise from the company's strategic role as a national gas transmission operator, its regulatory obligations, and the critical nature of its operational technologies (OT). Accordingly, the solution proposed in this dissertation was explicitly designed to respect the following limitations:

On-Premise Data Processing Requirement

Due to strict regulatory and security policies, operational and configuration data are not permitted to leave the internal IT infrastructure of Gaz-System. All processing must be performed on-premises within the company's secure network. Consequently, it is not possible to use commercial cloud-based machine learning platforms, external APIs, or remote computing services for data storage or model execution.

This constraint has substantial implications for the implementation:

- All software components, including data preprocessing, model training, and inference pipelines, must be containerized or deployable on internal servers.

- Model updating and retraining must rely exclusively on local computing resources, necessitating efficient and computationally lightweight algorithms.
- No external telemetry or log forwarding is permitted, which requires full offline explainability and auditability of AI decision-making processes.

Limited Human Resources for Data Quality Assurance

At Gaz-System, the primary mission of the technical personnel is to ensure the secure and uninterrupted flow of natural gas through the national transmission network. As such, tasks related to inventory accuracy or CMDB data reconciliation are secondary responsibilities. The available staff time and expertise dedicated to data quality monitoring is limited, particularly for manual validation tasks.

This operational constraint justifies the design of a semi-automated, human-in-the-loop architecture, where:

- Machine learning models nominate the most suspicious records for human verification, maximizing the value of each manual check.
- Feedback from human inspections is used to adaptively tune the model over time, reducing the need for frequent manual interventions.
- Routine processes (e.g., monthly quality checks) are aligned with the organizational capacity to verify a maximum of 100 records per month.

High Security Requirements for IT and OT Systems

As Gaz-System operates critical infrastructure, the security of IT and OT environments is paramount. This includes protection from data leaks, cyberattacks, and unauthorized modifications to system configurations. The CMDB, as a foundational system for change and incident management, must conform to strict access control, logging, and compliance standards.

From a research and deployment perspective, this leads to several implementation constraints:

- The AI system must operate within the boundaries of the existing security policies and ITSM tools, without requiring elevated privileges or external integration.
- The model must not autonomously modify production data; all corrections must be verified and approved by designated personnel.
- Audit trails and decision rationales must be preserved for all model suggestions, enabling forensic analysis and supporting cybersecurity audits.

Implications for System Design

These business constraints necessitate a design philosophy centered around explainability, auditability, modularity, and resilience. Rather than relying on black-box solutions or scalable cloud infrastructure, this dissertation proposes a lightweight, mathematically grounded framework tailored to the realities of industrial deployment. The methodology ensures alignment with operational priorities and demonstrates that high-quality data governance can be achieved even under strict infrastructural and human resource limitations.

1.4 IT Service Management and OT

The Information Technology Infrastructure Library (ITIL) is a widely recognized framework for IT service management (ITSM), offering a structured set of best practices for delivering high-quality IT services aligned with business objectives. Developed initially by the UK Government's Central Computer and Telecommunications Agency (CCTA), ITIL has evolved through multiple versions and is currently maintained under the guidance of AXELOS. ITIL is structured around the service lifecycle, which comprises five core stages: Service Strategy, Service Design, Service Transition, Service Operation, and Continual Service Improvement.

Among the numerous processes defined within the ITIL framework, Configuration Management, Capacity Management, and Demand Management play a pivotal role in ensuring efficient and reliable IT service delivery. These processes are particularly critical in complex enterprise environments where system interdependencies, resource constraints, and fluctuating business requirements pose significant operational challenges.

1.4.1 Configuration Management

Configuration Management, as defined in ITIL, is the process responsible for maintaining information about Configuration Items (CIs) required to deliver an IT service, including their relationships and dependencies. This information is stored in the Configuration Management Database (CMDB), which serves as the authoritative source for understanding the infrastructure landscape. The accuracy and completeness of the CMDB are fundamental to effective change control, incident resolution, impact assessment, and compliance auditing. However, maintaining data quality in the CMDB is a well-documented challenge, often due to manual updates, integration inconsistencies, or lack of automated validation mechanisms. Errors in CMDB data can propagate across ITSM processes, leading to degraded service quality, increased downtime, and inefficiencies in resource allocation.

1.4.2 Capacity Management

Capacity Management ensures that IT infrastructure is adequately sized to meet current and future demands in a cost-effective and timely manner. It involves monitoring, analysis, and planning of system capacity across three sub-processes: business capacity management, service capacity management, and component capacity management. The core objective is to balance resource availability with performance requirements while optimizing expenditure. Accurate forecasting, trend analysis, and performance modeling are central to this process. In modern IT environments—particularly those involving hybrid infrastructure (cloud/on-premise), dynamic workloads, and regulatory constraints—Capacity Management requires the integration of predictive analytics and adaptive control mechanisms to remain effective.

1.4.3 Demand Management

Demand Management operates in conjunction with Capacity Management to anticipate and influence customer demand for services. By analyzing usage patterns, business cycles, and user behavior, this process aims to ensure that the supply of IT resources can be aligned with business priorities. Demand Management often involves profiling user demand, segmenting workloads, and applying techniques such as differential service levels or demand shaping to prevent over-utilization of critical resources. In data-driven enterprises, Demand

Management is increasingly supported by machine learning and statistical models capable of recognizing patterns and predicting future needs with greater accuracy.

Together, Configuration Management, Capacity Management, and Demand Management form a critical triad that underpins operational stability, strategic planning, and cost optimization in IT service delivery. The effectiveness of these processes, however, is contingent upon the availability of accurate, timely, and context-aware data—an area where the application of transparent and analytically grounded machine learning algorithms, as proposed in this dissertation, can deliver substantial improvements.

1.5 Common Data Model in Configuration Management Databases (CMDB)

In modern IT environments, the Configuration Management Database (CMDB) serves as a centralized repository of information about hardware, software, services, and their interrelationships. To ensure consistency, interoperability, and semantic clarity, a *Common Data Model* (CDM) is employed as the foundational schema for representing configuration items (CIs) and their dependencies. This section presents the role and benefits of CDM in CMDBs, and provides an example schema and conceptual visualization.

1.5.1 Purpose and Structure of a Common Data Model

A Common Data Model establishes a unified and standardized taxonomy for CIs, enabling diverse IT systems and processes to interpret and utilize configuration data consistently. It defines:

- **CI Classes:** Abstract categories such as Server, Application, Network Device, Business Service, or Virtual Machine.
- **Attributes:** Standardized properties for each CI class (e.g., hostname, operating system, status, serial number).
- **Relationships:** Canonical links such as `RunsOn`, `DependsOn`, `ConnectedTo`, enabling impact analysis and dependency mapping.

This standardization ensures that data collected from discovery tools, entered manually, or imported via integrations can be normalized, validated, and effectively queried.

1.5.2 Example Schema Definition

Table 1.1 illustrates an example schema based on a simplified CDM with three CI types.

1.5.3 Security, AI, and Interoperability Considerations

The CDM simplifies data integration by aligning third-party discovery tools, asset inventories, and ticketing systems under a unified structure. This improves data quality by enabling:

- **Validation rules**: Ensuring mandatory fields and permissible values.

CI Type	Attribute	Example Value
Server	Hostname	server01.corp.local
	Operating System	Ubuntu 22.04
	Status	Active
Application	Name	PayrollApp
	Version	3.5.1
	Runs On	server01.corp.local
Network Device	IP Address	192.168.0.1
	Device Type	Router
	Connected To	server01.corp.local

Table 1.1: Example schema fragment using a simplified Common Data Model.

- **Anomaly detection**: Allowing AI algorithms to work with predictable schemas.
- **Audit readiness**: Enhancing transparency of IT operations.

The proposed algorithm for data quality analysis (as described in Chapter 2.4) benefits directly from a CDM. By relying on predictable attribute structures and well-defined relationships, the algorithm can more efficiently generate features, detect anomalies, and present interpretable results.

1.5.4 CMDB Implementations by Leading Vendors

Several enterprise software providers offer CMDB platforms equipped with their own implementations of a Common Data Model. While each product has proprietary extensions and features, they are all grounded in ITIL-compliant data modeling principles. This section outlines selected vendor-specific approaches to CMDB architecture.

ServiceNow: Common Service Data Model (CSDM)

ServiceNow provides one of the most widely adopted CMDB platforms, built on top of its Now Platform. Its Common Service Data Model (CSDM) offers a standardized and recommended set of configuration item classes and relationships across four key domains:

- **Foundation**: Core elements such as users, locations, and organizational structures.
- **Design**: Logical models for applications and services.
- **Manage Technical Services**: Components that support technical service delivery.
- **Sell/Consume Business Services**: Business-level abstractions and service offerings.

The CSDM provides out-of-the-box support for integration with ServiceNow's ITSM, ITOM, and ITAM modules, and supports alignment with business capability maps, service portfolios, and event correlation tools. It is accompanied by governance documentation to guide implementation maturity.

BMC Helix CMDB and Atrium

BMC's CMDB offering, formerly known as Atrium, is part of the BMC Helix ITSM suite. It leverages a highly structured Common Data Model consisting of:

- **Base classes and subclasses** organized into domains such as System, Application, Network, and People.
- **Federated data model** allowing the inclusion of external data sources without direct duplication.
- **Reconciliation and normalization engines** to ensure consistency across discovery inputs.

BMC Helix CMDB supports integration with BMC Discovery and third-party data sources and includes visualization tools for dependency mapping. The CDM is tightly coupled with impact simulation, change tracking, and service modeling.

IBM Maximo and Control Desk

IBM Control Desk integrates CMDB functionalities based on the Maximo platform. It emphasizes asset management for IT and non-IT equipment, making it especially suited for operational technology (OT) environments. The Maximo data model:

- Covers both IT assets and enterprise physical assets.
- Supports lifecycle states, procurement history, and maintenance records.
- Leverages a flexible schema to adapt to vertical industry needs (e.g., utilities, transportation).

Integration with Watson AIOps allows for intelligent correlation and insights from CMDB data, enhancing predictive capabilities.

1.5.5 Comparative Summary

While each vendor's CDM implementation varies in architecture, terminology, and use cases, they all support the following shared goals:

- Providing a single source of truth for IT assets and services.
- Enabling automation and intelligent decision-making.
- Supporting ITIL-aligned processes for incident, change, and asset management.

Selecting an appropriate CMDB solution and its CDM implementation depends on organizational size, industry, security requirements, and integration needs. Regardless of the vendor, the Common Data Model remains a foundational element for effective IT governance and AI-supported data quality initiatives.

1.6 Configuration Management Database (CMDB) and its Graph Representation

1.6.1 Overview of CMDB

The **Configuration Management Database (CMDB)** is a central component of IT Service Management (ITSM), designed to store and manage information about the hardware, software, systems, and services that constitute an organization's IT and OT infrastructure. Its primary function is to maintain an accurate and up-to-date repository of all **Configuration Items (CIs)** and the relationships among them, thereby supporting change management, incident response, and impact analysis.

1.6.2 Configuration Items (CIs)

A **Configuration Item (CI)** is any component within the IT or OT ecosystem that must be managed to deliver an IT service. CIs typically include:

- **Hardware assets**: Servers, switches, routers, IoT devices, industrial controllers (e.g., PLCs).
- **Software components**: Operating systems, applications, databases, virtual machines.
- **Services**: Business or technical services composed of multiple hardware and software CIs.
- **Documentation**: Service level agreements (SLAs), policies, and operational procedures.

Each CI is characterized by a set of attributes such as:

- *Unique identifier* (e.g., asset tag, serial number)
- *Type and class* (e.g., "Server", "Database", "Application")
- *Status* (e.g., "In operation", "Under maintenance", "Retired")
- *Location*, *owner*, and *version information*

1.6.3 Relationships Between CIs

In addition to storing standalone attributes of CIs, the CMDB captures the **relationships** among them. These relationships are critical for understanding dependencies and system behavior. Common types of relationships include:

- **Depends on**: Indicates that a CI requires another CI to function properly.
- **Runs on**: Links applications or services to the infrastructure that supports them.
- **Connected to**: Captures physical or logical connectivity between components.

- **Hosted on**: Represents virtualization or containerization relationships.
- **Backed up by**, **Monitored by**, etc.: Denote auxiliary services or management tools.

These relationships form the structural backbone of the CMDB and are essential for impact analysis, root cause diagnostics, and change propagation assessments.

1.6.4 Graph Representation of CMDB

To enable analytical processing, especially for applications in machine learning and data mining, the CMDB can be modeled as a **labeled, attributed, and directed multigraph**, defined as:

$$G = (V, E, \phi, \psi)$$

Where:

- V is the set of vertices representing **Configuration Items (CIs)**.
- $E \subseteq V \times V$ is the set of directed edges representing **relationships** between CIs.
- $\phi : V \to \mathcal{A}_V$ maps each vertex to its **set of attributes** (e.g., type, status, location).
- $\psi : E \to \mathcal{A}_E$ maps each edge to its **relationship type and metadata**.

Properties of the CMDB Graph

- **Directedness**: Most relationships are inherently directional (e.g., "A runs on B" $\neq$ "B runs on A").
- **Multigraph**: Multiple relationship types can exist between the same pair of nodes (e.g., "depends on" and "connected to").
- **Heterogeneity**: The nodes and edges may carry heterogeneous types and attribute schemas.

Example

Consider a scenario where a web application (CI1) runs on a virtual server (CI2), which in turn is hosted on a physical server (CI3). The relationships can be modeled as:

$$CI_1 \xrightarrow{\text{runs on}} CI_2 \xrightarrow{\text{hosted on}} CI_3$$

This representation allows for formal reasoning and supports algorithmic operations such as:

- *Traversal algorithms* for dependency and impact analysis
- *Graph-based clustering* for identifying CI groups or service domains
- *Link prediction* to identify missing or likely relationships
- *Anomaly detection* for identifying unusual patterns or disconnected nodes

1.6.5 Motivation for Graph-Based Representation

Traditional relational or tabular representations of CMDB data are insufficient for capturing complex, many-to-many and hierarchical dependencies between components. The graph-based model enables:

- Better alignment with real-world infrastructure topologies.
- Native support for algorithms in graph theory and network science.
- Seamless integration with graph neural networks and other structure-aware machine learning methods.

This formalism forms the basis for the transformations and algorithms discussed in the following sections of this dissertation.

1.7 Business Goals

1.7.1 Configuration Management - Data Quality

The CMDB serves as a foundation that defines the scope of responsibilities carried out in other IT system maintenance processes, particularly those related to cybersecurity. For this reason, the quality of the database is a critical objective for teams responsible for system maintenance and operational continuity. For systems of high importance, it is assumed that CMDB quality should reach 99% accuracy in records. A key business constraint is the number of records that can be verified within a month, as this often requires field verification of the device. It is assumed that 100 records (out of 20,000) can be verified per month

1.7.2 Capacity Management - Procurement Strategy

The goal of optimizing the procurement strategy is to minimize total acquisition costs while ensuring the availability of necessary resources by selecting the most cost-effective purchasing plan over a defined time horizon, taking into account demand forecasts, pricing dynamics, and contractual constraints.

1.8 CMDB Data Quality Management Process

1.8.1 Dimensions of Quality Management

Effective quality management in the Configuration Management Database (CMDB) is a prerequisite for ensuring the reliability and usability of configuration data across the organization. In line with established information quality frameworks [16, 37], three primary dimensions of quality assurance are recognized in the context of CMDBs:

- **Completeness** – This dimension refers to the presence of all necessary information at each stage of the configuration item's (CI's) lifecycle. Incomplete data may prevent a CI from transitioning to the next lifecycle state. For instance, a server cannot be

promoted to the "production" status if the responsible administrator group has not been specified. Completeness is therefore essential for enabling automated workflows, compliance checks, and service accountability.

- **Accuracy Testing** – Accuracy pertains to the degree of alignment between the information stored in the CMDB and the actual state of the underlying IT infrastructure. Accuracy testing typically involves comparing CMDB records with trusted sources such as network scans, technical discovery tools, or authoritative hardware inventories. Discrepancies revealed during this process indicate data drift, integration issues, or misconfigurations.
- **Conformity** – This dimension addresses the syntactic and semantic adherence of data to predefined standards or formats. For example, IP addresses must conform to a specified regular expression pattern, and hostname conventions may be enforced to ensure consistency across systems. Conformity rules can be derived from organizational policies, regulatory frameworks, or industry best practices.

Among the three dimensions, conformity checking is often the most resource-intensive and operationally complex. In one illustrative case from the insurance sector, an organization succeeded in formalizing approximately 700 conformity rules. Over a three-year period, a dedicated team of eleven professionals worked exclusively on implementing these rules within the CMDB ecosystem. As a result, close to one million individual data fields were subjected to conformity validation in near real-time.

Despite this success, the effort required to maintain and extend such a rule set is substantial. Moreover, even 700 rules may not capture the full complexity of the IT environment, especially in large and heterogeneous infrastructures. Consequently, many organizations struggle to sustain such initiatives, which often leads to a degradation in CMDB reliability over time.

To address these challenges, recent research has proposed the use of algorithmic approaches [12] and machine learning techniques [28] to support or partially automate conformity checking. These methods offer the potential to significantly reduce the manual workload, enabling smaller teams to uphold data quality standards while redirecting human effort toward higher-value activities such as root cause analysis or infrastructure optimization.

1.8.2 Categories of Data Quality Errors in the CMDB

The Configuration Management Database (CMDB), due to its central role in IT operations and asset tracking, is particularly susceptible to data quality issues. These errors, if left unaddressed, can result in operational inefficiencies, erroneous changes, and security or compliance risks. Below, the most prevalent categories of data quality errors in CMDBs are described, along with typical causes and mitigation strategies.

- **Duplicate Entries** – This error occurs when multiple CMDB records refer to the same real-world configuration item (CI). It is commonly observed during the initial population of the database, particularly when multiple data sources are integrated. Although relatively straightforward to resolve using record linkage techniques, duplicates may persist if not addressed early. Scientific methods for resolving such errors are discussed in [17].
- **Lack of Unique Identifiers** – Among the most critical error categories, this issue arises when two or more distinct CIs are registered under identical names. Such duplication may result in severe consequences, including the unintended execution of

changes on incorrect assets (e.g., decommissioning the wrong server). The problem is often observed following mergers and acquisitions or during hardware refresh cycles. Middleware components installed with default names also contribute significantly to this issue (e.g., numerous instances of databases named "DEFAULT"). Mitigation strategies include enforcing naming conventions, utilizing compound identifiers (e.g., database name combined with host name), or implementing strict policies for unique naming.

- **Typographical Errors (Errata)** – Typing mistakes are ubiquitous in manually maintained datasets. These are especially common during data entry or reconciliation tasks. An illustrative example involves a pharmaceutical company where the initial match rate between CMDB data and manually collected inventory was 80%. After correcting for visual ambiguities (e.g., "0" vs. "O", "G" vs. "6"), the accuracy increased to 98%. While automation through barcode scanning and other data entry validation mechanisms is recommended, artificial intelligence methods—described in the subsequent sections—can also play a significant role in detecting and correcting such anomalies.

- **Field Misuse** – Field misuse occurs when users misunderstand the intended content of a particular attribute or mistakenly enter values in incorrect fields. For example, the "CPU Core Count" field may ambiguously refer to physical cores, logical cores, or cores per socket. Similarly, users may erroneously record part numbers in the serial number field. These inconsistencies can lead to incorrect reporting and inventory mismanagement. Clear field definitions and interface constraints are effective countermeasures.

- **Unfinished Runbooks** – This error class is specific to automated IT service provisioning. Complex workflows (e.g., deployment of clustered MS SQL servers with over 140 sequential tasks) may fail mid-process, leaving partially configured CIs erroneously registered as complete. A server may appear in the CMDB, while the associated service remains non-functional, leading to incorrect configuration statuses and potential cost leakage due to untracked resource consumption. Full transactional integrity in provisioning workflows remains a challenge.

- **Incorrect Lifecycle Status** – The lifecycle status of a CI (e.g., "planned", "production-ready", "production", "postproduction", "archived") is one of the most critical fields in the CMDB. An incorrect status may trigger legal, operational, or cybersecurity risks. For instance, CIs marked as "production-ready" should undergo licensing and security reviews. Errors in this category frequently stem from weak integration between the change management and configuration management processes.

Table 1.2: Error Categories and Corresponding Remediation Stages in the Process

Error Category	**Process Stage for Detection or Correction**
Duplicate Entries	Data Entry and Validation
Lack of Unique Identifiers	Data Entry and Validation
Typographical Errors (Errata)	Monthly Data Quality Check
Field Misuse	Data Entry and Validation
Unfinished Runbooks	Monthly Data Quality Check
Incorrect Lifecycle Status	Data Entry and Validation

1.9 AI Support for Data Quality Management Process

1.9.1 AI-Supported Data Entry and Validation Process

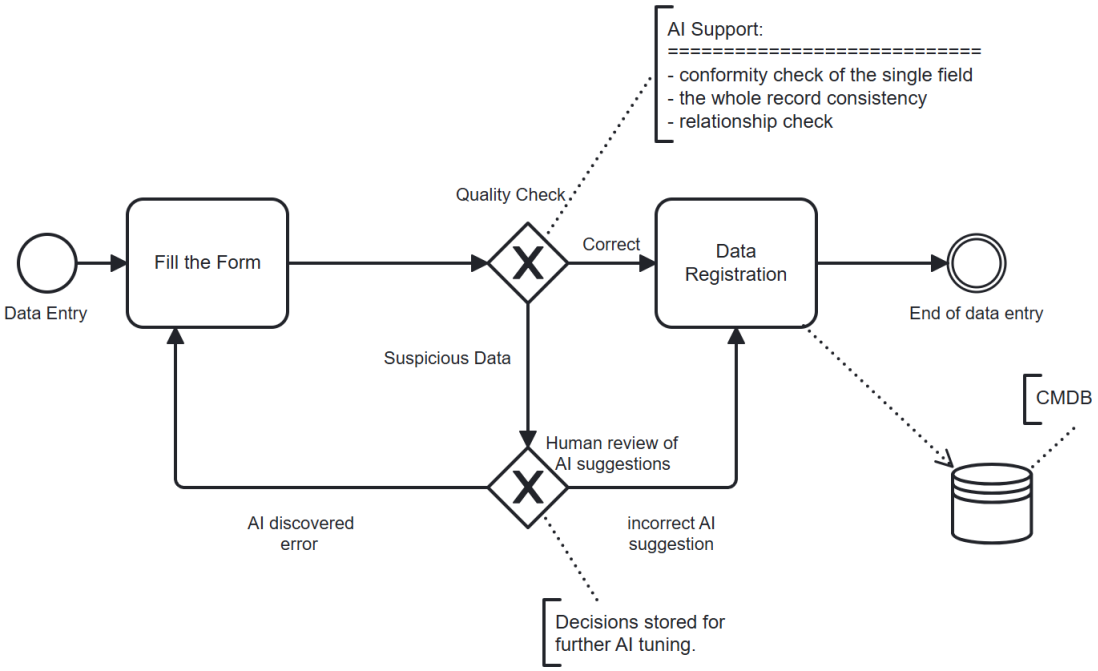


Figure 1.1: AI-Supported Data Entry and Validation Process.

The diagram in Figure 1.1 presents a semi-automated data entry process enhanced with artificial intelligence (AI) mechanisms to support data quality assurance. This process is designed to improve the integrity of Configuration Management Databases (CMDB), which serve as a foundational element in IT Service Management (ITSM) and are particularly critical in environments where cyber resilience and operational reliability are priorities.

Process Initiation – Data Entry. The process begins with a user-initiated *Data Entry* phase, in which a form is manually completed (*Fill the Form*). This input typically includes

multiple fields that describe configuration items (CIs) or assets, such as hardware identifiers, IP addresses, ownership, and relationships between components.

AI-Assisted Quality Check. Once the form is completed, the data undergoes an *AI-supported Quality Check*. This automated validation stage is augmented with three AI-driven verification layers:

- **Single Field Conformity Check** – Verifies whether each individual field complies with the expected format, taxonomy, or value range (e.g., valid IP format, known manufacturer names).
- **Whole Record Consistency** – Evaluates internal coherence of the data record (e.g., a server with a desktop classification would be flagged).
- **Relationship Validation** – Assesses the plausibility of relationships between CIs (e.g., whether the parent-child linkage is valid for given asset types).

This tri-layered AI inspection identifies entries that deviate from learned norms or defined business rules.

Decision Gate – Suspicious vs. Correct Data. The result of the AI-based quality check determines the path forward:

- If the data is deemed *Correct*, it proceeds directly to the *Data Registration* step and is subsequently persisted to the *CMDB*, marking the *end of the data entry process*.
- If the data is flagged as *Suspicious*, it is routed to a *Human Review* phase.

Human-in-the-Loop Validation. During the human review phase, users inspect AI suggestions and determine whether the identified issue is indeed a data quality error:

- If the *AI suggestion is correct*, the error is confirmed, and the data is returned to the *Fill the Form* step for correction.
- If the *AI suggestion is incorrect*, this decision is stored as feedback and used for further tuning of the AI models, improving their accuracy over time via continuous learning.

Learning and Feedback Integration. Feedback from incorrect AI suggestions is crucial to the evolution of the model. It supports semi-supervised learning or human-in-the-loop training pipelines, which enable the algorithm to adapt to domain-specific data anomalies and changing standards of data correctness.

Relevance to Dissertation. This process is a central component of the proposed methodology in this dissertation. It illustrates the integration of machine learning techniques within ITSM workflows, specifically in data quality assurance for CMDB systems. The iterative loop of AI analysis, human validation, and feedback-based model refinement exemplifies a hybrid intelligence system—combining statistical anomaly detection with expert oversight. The research explores not only the algorithmic foundations of the AI engine but also its operational impact on reducing data defects, improving trust in CMDB records, and lowering the cost of manual audits.

1.9.2 AI-Supported Monthly Data Quality Check Process

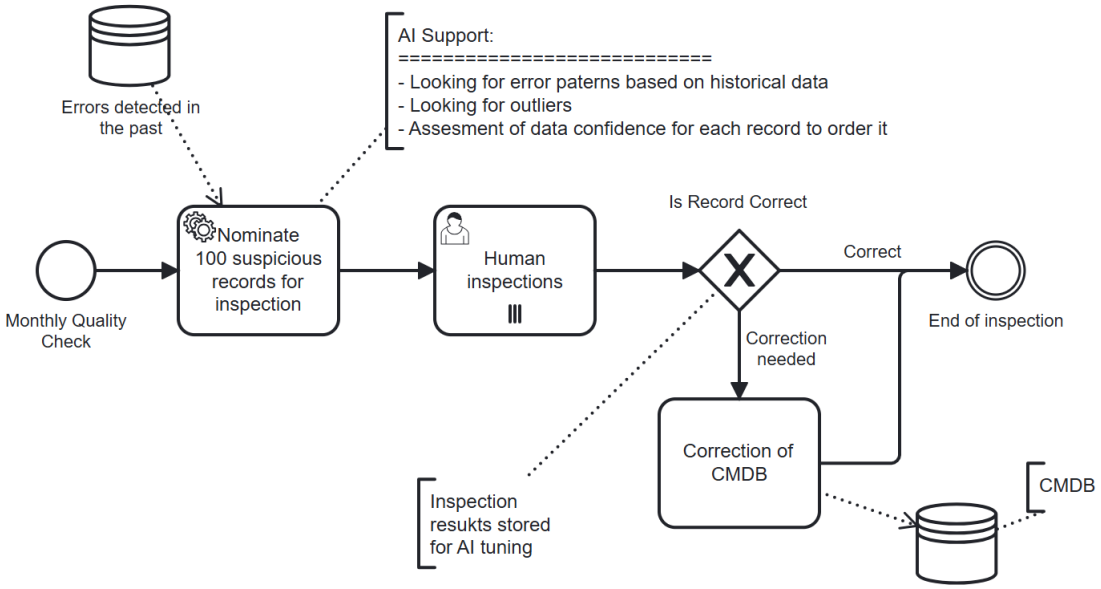


Figure 1.2: AI-Supported Monthly Data Quality Check Process

The diagram in Figure 1.2 illustrates a monthly data inspection process designed to enhance the quality and reliability of records stored in the Configuration Management Database (CMDB). The process incorporates artificial intelligence (AI) to assist in prioritizing and selecting records that are most likely to contain data inconsistencies or errors.

Initiation – Monthly Quality Check. The process is initiated as part of a scheduled *Monthly Quality Check*, during which a predefined number of records are selected for manual inspection. Due to business constraints and the effort required for verification (often involving physical asset review), the number of records is typically limited to 100 per month.

AI-Assisted Record Nomination. In the *Nominate 100 Suspicious Records for Inspection* step, AI support is leveraged to intelligently select the most suspicious records based on:

- Historical error patterns stored in the CMDB.
- Identification of statistical outliers across selected attributes.
- Assessment of data confidence metrics per record to prioritize review.

The AI model combines pattern recognition and probabilistic reasoning to suggest a ranked list of records for further human verification. This selection process is informed by past inspection results and dynamically tuned based on previous outcomes.

Human Inspections. The selected records undergo *Human Inspections*, where specialists manually assess the validity of each configuration item. The goal is to verify whether each record is accurate and complete, or whether it requires correction.

Decision Gate – Is the Record Correct? Based on the inspection, records are routed as follows:

- If the record is deemed *Correct*, it is approved and the process ends for that item.
- If *Correction is Needed*, the record is passed to the next phase.

Correction of CMDB. In cases where data inconsistencies are confirmed, the relevant records are updated during the *Correction of CMDB* phase. This step ensures the CMDB reflects accurate and current information, maintaining its reliability for dependent IT processes.

Feedback for AI Tuning. Both the outcomes of inspections and the corrections performed are stored and used to further refine the AI model. This closed feedback loop enables continuous improvement of the nomination algorithm, progressively increasing its precision in identifying records that require human review.

Relevance to Dissertation. This process supports the dissertation's broader research objectives by demonstrating how AI can be operationally integrated into configuration management practices to support long-term data quality assurance. By combining automated pattern detection with expert validation, the approach offers a scalable solution to improve the trustworthiness of large-scale CMDB systems while respecting resource limitations and auditability requirements.

1.9.3 Process of Handling Errors in the CMDB

The process of handling errors in the Configuration Management Database (CMDB) is depicted in Figure 1.3. This flow aims to ensure both the accuracy of CMDB entries and the continuous improvement of artificial intelligence (AI) models used for data quality enhancement. The process begins when an error is discovered within the CMDB. This discovery

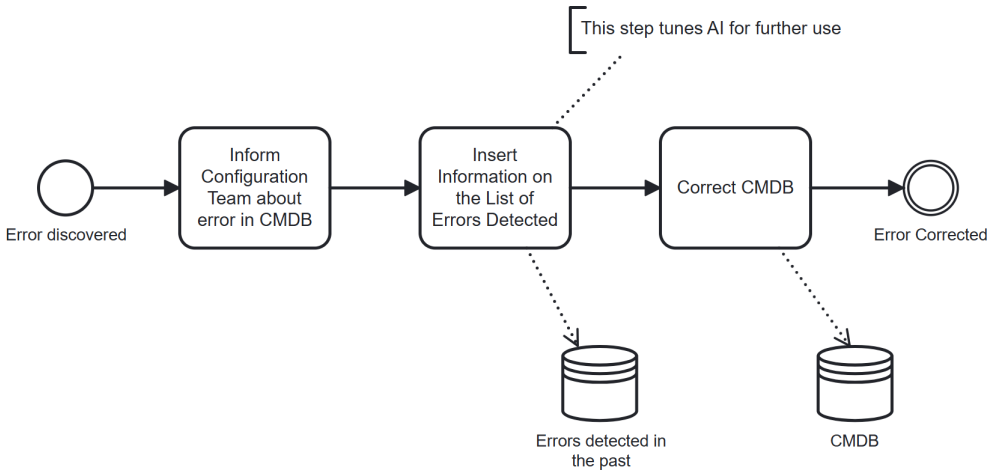


Figure 1.3: Process of Handling Errors in the CMDB

triggers a notification to the configuration team, which is responsible for verifying and processing the error. Once the error is acknowledged, relevant information is inserted into a

dedicated list of detected errors. This list not only serves as a historical repository of inconsistencies but also provides valuable input for further tuning of AI models that support error detection and correction.

Subsequently, the CMDB is corrected based on the validated error information. This correction is applied directly to the database and is also reflected in the record of past errors to prevent recurrence and improve future detection accuracy.

The step of inserting information into the list of detected errors is crucial, as it closes the feedback loop and allows the AI-based mechanisms to learn from past mistakes. Over time, this feedback improves the precision and robustness of AI components responsible for automated data quality assurance.

This workflow supports a semi-automated approach, where human expertise is complemented by machine learning techniques, ensuring a high standard of data integrity within the CMDB while maintaining adaptability to new error patterns.

1.10 Literature review

The field of data quality management in Configuration Management Databases (CMDBs) intersects with several research domains, including information systems, statistical learning, and machine learning-driven data cleaning. This section reviews key contributions that have shaped current methodologies, identifies open challenges, and positions this dissertation within the broader scholarly landscape.

1.10.1 Foundational Work on Data Quality Dimensions

The conceptual foundation for data quality management in information systems is rooted in the works of Wang and Strong [66], Eppler and Helfert [16], and Madnick et al. [38], who defined primary dimensions such as completeness, accuracy, consistency, and conformity. These dimensions have been widely adopted in industrial standards and frameworks such as ITIL, COBIT, and ISO/IEC 20000. In the context of CMDBs, these dimensions serve as the benchmark for evaluating the reliability and usability of configuration item records.

1.10.2 Data Cleaning and Rule-Based Approaches

Traditional approaches to improving CMDB quality have relied on rule-based systems, constraint validation, and manual auditing [17]. Techniques such as duplicate detection, syntactic validation (e.g., regex matching for IPs), and field normalization have been widely implemented in commercial ITSM tools. However, these methods are limited by their static nature and high cost of maintenance. The effort required to manually define and sustain hundreds of quality rules, as illustrated in large-scale industry cases, often renders these methods infeasible for dynamic or large-scale environments.

1.10.3 Machine Learning for Error Detection

Recent years have witnessed a shift toward using machine learning (ML) to support semi-automated data quality assessment. The work by Niewiadomski and Mzyk [49] introduced a statistically grounded methodology for detecting anomalies in CMDB records using kernel

density estimation, mutual information, and entropy-based metrics such as the Rajski and Jaccard distance. Their method emphasizes modularity, transparency, and adaptive learning, enabling models to evolve based on human-verified corrections.

The approach does not rely on predefined rules but instead constructs and evaluates millions of candidate features (e.g., substring counts, record lengths, character patterns) derived from string-type fields such as hostnames or serial numbers. Feature selection is performed based on two criteria: discriminative power for error detection and statistical independence from previously selected features. Recursive kernel density estimators and trust indicators are then used to prioritize records for manual inspection.

1.10.4 Hybrid Intelligence and Human-in-the-Loop Systems

The dissertation builds on the concept of hybrid intelligence, where ML models assist in data validation while final decisions remain under human supervision. This paradigm is especially important in regulated or safety-critical sectors like energy and utilities. The feedback loop between manual verification and model refinement, proposed in [49], is designed to adaptively enhance accuracy while preserving auditability and domain-specific control.

This is particularly relevant in industrial environments with strict data integrity requirements, where erroneous CMDB entries can lead to operational outages or cybersecurity risks. The semi-automated model allows organizations to scale their quality assurance processes without proportionally increasing human effort.

1.10.5 Gaps and Research Directions

Despite notable progress, several gaps remain:

- Most prior work either assumes clean training data or provides limited mechanisms for robustness against label noise. The recursive trust indicator methodology addresses this by incorporating uncertainty and history-aware learning.
- The integration of CMDB correction workflows with ML pipelines is rarely operationalized in existing research. This dissertation proposes an end-to-end pipeline from error nomination to post-correction feedback.
- Few studies explore the use of ML for IT procurement forecasting in tandem with CMDB analysis. This dissertation contributes novel solutions in that area, merging demand prediction with data quality metrics.

1.10.6 Related Work

A systematic overview of data error detection methods—including approaches based on outlier detection, duplication patterns, and rule-based constraints—can be found in [1]. Numerous works in this domain (e.g., [46]) focus on the construction of explicit data quality rules. In contrast, the core idea of the approach proposed in this dissertation is to avoid reliance on predefined, static conditions. Instead, our method emphasizes an automatic and adaptive generation of decision criteria, grounded in historical human validation outcomes.

A similar data-driven paradigm is adopted in [36], where constraint ranges are derived from multivariate data distributions rather than imposed manually. Another noteworthy contribution is the Raha framework [40], which integrates algorithm evaluation, hyperparameter

tuning, and result aggregation. Although highly general, the authors of Raha acknowledge that the predefined set of features may not always capture sufficient information to reliably identify errors.

Our approach differs fundamentally in its capacity to automatically generate a vast number of candidate features—potentially in the millions—and then filter and select an optimal subset for use in either traditional or neural network-based classifiers. This feature generation and selection process is dynamic and context-sensitive, enhancing detection accuracy and flexibility.

The HoloDetect method [27] introduces a neural network training framework augmented with synthetic data. However, its practical application in our context is limited due to insufficient labeled data for robust deep learning model training.

Importantly, the objective of our algorithm is not to make rigid binary decisions (i.e., error vs. no error), but rather to produce a ranked list of records prioritized for manual inspection. The subsequent feedback from this validation process is incorporated into the model iteratively, resulting in an adaptive learning mechanism. In this framework, the classification criteria are not fixed or rule-based but are learned and refined over time. Consequently, our solution does not require predefined vocabularies or hardcoded rules; instead, it evolves in response to real-world data and expert feedback.

1.11 Structure of the Dissertation

This dissertation separates two core contributions—CMDB data cleaning and procurement optimization—and adds implementation guidance and appendices that document governance and technical foundations.

Chapter 2: Data Cleaning. Chapter 2 develops an end-to-end pipeline for detecting and prioritizing CMDB data-quality issues. It defines the problem and monthly review workflow; details an algorithm combining automated feature construction (including n-gram tokenization), density-based discrimination, iterative feature filtering, and an interpretable confidence indicator; and evaluates estimators (KDE, bandwidth updates, single-record withdrawal). It then covers initial feature filtering and redundancy control (e.g., Jaccard/Rajski-style measures), a neural-network classifier, and experiments with results. Additional sections address distance-based detection, discovery of structural errors in CI relationships, graph transformations for ML, and a VIN case study, followed by a summary of advantages (on-prem, security-aware, modular, low-cost, explainable, adaptive) and robustness.

Chapter 3: Purchasing Strategy. Chapter 3 formulates the IT procurement optimization problem under demand/price/availability constraints and proposes an adaptive, data-driven approach. It specifies the overall algorithm, presents estimators for price (time polynomial with $\log q$) and demand (including AR(1)), discusses the role of the interest rate, and justifies the method selection. A genetic algorithm is described in detail (initial seeding, selection, crossover, mutation, termination) with experiment results, open problems, and practical implementation guidance (monitoring price trends, delivery-time prediction, evolving licensing models, cloud adoption), followed by a summary.

Chapter 4: Implementation. Chapter 4 focuses on deployment: transition management prerequisites, an internal implementation approach, service-oriented solution architecture,

and maintenance/support strategy.

Appendices: Appendix A provides a Trustworthy AI self-assessment; Appendix B documents the use of generative language models; Appendix C is a concise primer on algebra and mathematical statistics used in the methods; Appendix D lists functions from open-source ML libraries referenced in the thesis.

Chapter 2

Data Cleaning

2.1 AI Support for Monthly Data Quality Check

Initiation – Monthly Quality Check. The process is initiated as part of a scheduled *Monthly Quality Check*, during which a predefined number of records are selected for manual inspection. Due to business constraints and the effort required for verification (often involving physical asset review), the number of records is typically limited to 100 per month.

AI-Assisted Record Nomination. In the *Nominate 100 Suspicious Records for Inspection* step, AI support is leveraged to intelligently select the most suspicious records based on:

- Historical error patterns stored in the CMDB.
- Identification of statistical outliers across selected attributes.
- Assessment of data confidence metrics per record to prioritize review.

The AI model combines pattern recognition and probabilistic reasoning to suggest a ranked list of records for further human verification. This selection process is informed by past inspection results and dynamically tuned based on previous outcomes.

2.1.1 Problem Statement

We are given a dataset consisting of N tuples, denoted as $\{t_i\}_{i=1}^{N}$. Let c_i represent the true class of the element t_i, defined as follows:

$$c_i = \begin{cases} 1, & \text{if } t_i \text{ is erroneous,} \\ 0, & \text{if } t_i \text{ is correct.} \end{cases} \tag{2.1}$$

The true classes $\{c_i\}_{i=1}^{N}$ are unknown. The database consists of the following records:

$$\{(t_i, d_i, l_i)\}_{i=1}^{N}, \tag{2.2}$$

where d_i denotes the classification decision based on previous automated analyses and manual inspections, and $l_i \in \{0, 1\}$ is a flag indicating whether the tuple has been manually verified.

The a priori class probabilities, which reflect the level of data 'contamination', are assumed to be known:

$$\mathcal{P}\{c=0\}=p, \quad \mathcal{P}\{c=1\}=q, \quad p+q=1. \tag{2.3}$$

Our objective is to achieve a state in which $d_i = c_i$ for all i, i.e., all errors are correctly identified. The developed system nominates records with low confidence for manual verification. Tuples that have been verified are marked with $l_i = 1$. When an error is confirmed, d_i and l_i are both updated to 1, and a new record $\{t_{N+1}, 0, 1\}$, reflecting the corrected data, is inserted into the dataset.

Two types of potential errors may occur:

- (ERR1) $d_i = 0$ for $c_i = 1$ (false negative),
- (ERR2) $d_i = 1$ for $c_i = 0$ (false positive).

In this work, we focus on minimizing the number of (ERR1) cases. We assume that (ERR2)-type errors do not occur, as any decision with $d_i = 1$ is always confirmed through manual inspection. Once an error is detected and manually verified, the corresponding record is considered *permanently* classified and is labeled with $l_i = 1$.

The proposed algorithm operates on a large set of numerical features, indexed by $j = 1, ..., M$:

$$X^{(1)}(t_i),\ X^{(2)}(t_i),\ \ldots,\ X^{(j)}(t_i),\ \ldots,\ X^{(M)}(t_i), \tag{2.4}$$

which are generated automatically and may also include features proposed by domain experts.

2.1.2 Steps of Algorithm. Data Cleaning

This section presents a structured and modular algorithm designed for identifying and correcting erroneous records in Configuration Management Databases (CMDBs). The algorithm is based on the automatic extraction and selection of features from CMDB entries, and the use of machine learning methods—particularly neural networks—to support semi-automated verification workflows [45]. The approach is iterative and enables continuous improvement of data quality through human-in-the-loop feedback.

Notation. Let $\mathcal{X} = \{X^{(j)}\}_{j=1}^{M}$ denote the feature set constructed from CMDB records. We use indices j, k for features and avoid the letter f for features to prevent confusion with density functions (which are denoted by $\hat{p}$).

Below is a brief description of the algorithm's steps, which are discussed in greater detail in the subsequent sections of the document.

Step 1: Automatic Extraction of Features Using N-Gram Tokenization

The preprocessing stage begins with the automatic extraction of M features $\{X^{(j)}\}_{j=1}^{M}$ from the CMDB records. These features include both structural and content-based elements, extracted from categorical, textual, and potentially numerical fields.

In the case of textual fields, the content is first tokenized using an n-gram approach (commonly with $n = 2$ or $n = 3$). Tokenization converts each string into a sequence of overlapping substrings of length n, capturing local patterns and syntactic regularities. For example, the

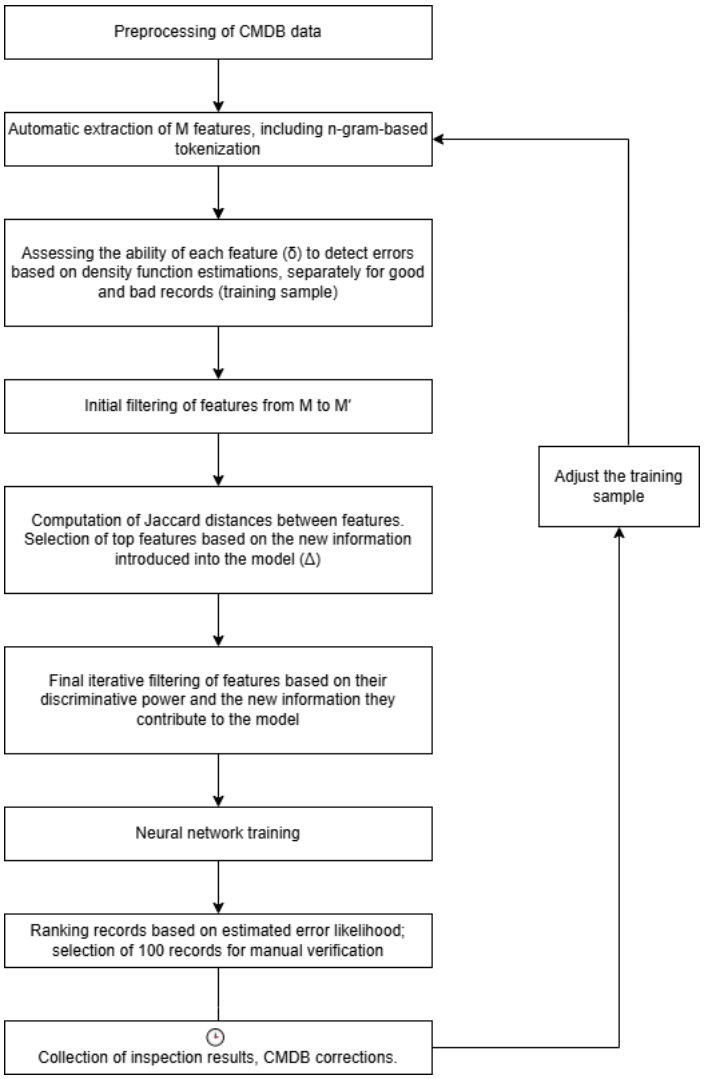


Figure 2.1: Data processing stages.

string ‘‘server_prod01’’ tokenized into bigrams yields the sequence: ‘‘se’’, ‘‘er’’, ‘‘rv’’, ‘‘ve’’, ‘‘er’’, etc. These n-grams are then counted or transformed into binary presence indicators to construct feature vectors for each record.

This step results in a high-dimensional representation of records, capturing subtle textual and lexical patterns that may be indicative of erroneous entries (e.g., typos, naming convention violations).

Step 2: Initial Filtering of Features (Reduction from M to M')

Due to the high dimensionality of the initial feature space, a preliminary filtering stage is applied to reduce noise and computational overhead. Features are excluded based on criteria such as:

- **Low variance**: features that exhibit near-constant values across the dataset are discarded;
- **Low frequency**: features (e.g., rare n-grams) that occur in very few records are unlikely to generalize and are therefore removed;

- **Missingness**: features with a high percentage of missing values or undefined states are also excluded.

The resulting feature subset of size M' (with $M' \leq M$) is carried forward to the next stage of discriminative analysis.

Step 3: Density-Based Evaluation of Feature Discriminative Ability (δ)

Each feature $X^{(j)} \in \mathcal{X}$ is evaluated for its ability to distinguish between correct and erroneous records. This is achieved by estimating the (univariate) probability density of the feature separately over two disjoint subsets:

- D_{good}: records known (or assumed with high confidence) to be correct;
- D_{bad}: records suspected or known to contain errors.

Density estimation is performed using non-parametric methods such as kernel density estimation (KDE). Let $\hat{p}^{(j)}_{\text{good}}$ and $\hat{p}^{(j)}_{\text{bad}}$ denote the estimated densities of $X^{(j)}$ on D_{good} and D_{bad}, respectively. The discriminative score for feature j is

$$\delta_j \;=\; \text{Div}\Big(\hat{p}^{(j)}_{\text{good}},\, \hat{p}^{(j)}_{\text{bad}}\Big)\,,$$

where $\text{Div}(\cdot,\cdot)$ may be, e.g., Jensen–Shannon divergence or the 1-Wasserstein distance. Features with $\delta_j < \tau_\delta$ are deemed non-informative and removed. A detailed proposal for this step is presented in Section 2.1.3.

Step 4: Computation of Jaccard Distances and Feature Informativeness (Δ)

To identify and eliminate redundant features, pairwise Jaccard distances are computed across the filtered set. For binary features, let S_j denote the set of records where $X^{(j)}$ is active. The Jaccard distance between features j and k is

$$J_{jk} \;=\; 1 - \frac{|S_j \cap S_k|}{|S_j \cup S_k|}.$$

In parallel, an informativeness score Δ_j quantifies the marginal contribution of $X^{(j)}$ relative to the features already selected. Features that are highly redundant (e.g., $J_{jk} < \tau_J$ for some selected k) and contribute minimal novel information ($\Delta_j \approx 0$) are excluded. Details regarding this step are presented in Section 2.1.7.

Step 5: Final Iterative Feature Filtering

A final refinement step combines both discriminative ability δ_j and informativeness Δ_j. A greedy forward-selection procedure may be used, or a hybrid scoring function:

$$\text{Score}_j \;=\; \alpha \cdot \delta_j \;+\; (1-\alpha)\cdot \Delta_j, \qquad \alpha \in [0,1].$$

The selected subset is then used as input for model training.

Step 6: Neural Network Training

A supervised model—specifically a feedforward neural network—is trained on the selected features. The network learns a mapping from feature vectors to a probability score indicating the likelihood that a given record is erroneous. The training data may consist of a mix of manually labeled records and pseudo-labeled data inferred from heuristics or rule-based systems. The network typically comprises an input layer of size equal to the selected feature subset, one or more hidden layers (e.g., ReLU activations), and a final sigmoid output.

A detailed proposal for this step is presented in Section 2.1.8.

Step 7: Ranking and Nomination of Records for Verification

Once the model is trained, it is applied to the full CMDB dataset. Each record is assigned an error-likelihood score. Records are ranked in descending order and the top-K records (e.g., $K = 100$) are nominated for expert verification, focusing effort on the most suspicious entries.

Step 8: Collection of Inspection Results and CMDB Corrections

The algorithm updates based on expert feedback. After each iteration, verified labels are incorporated, which improves subsequent detection and supports discovery of new error patterns not captured initially.

Step 9: Adjustment of the Training Sample

Manually verified records are added to the labeled training set. Corrected entries serve as confirmed positives (errors), while validated entries serve as confirmed negatives. The updated set is then used to retrain the neural network, incorporating the latest expert knowledge.

2.1.3 Kernel Estimation of Probability Density Functions

Let us denote the class of correct records as C and the class of incorrect records as E. Furthermore, let

$$f_C^{(j)}(x) \quad \text{and} \quad f_E^{(j)}(x) \tag{2.5}$$

be the probability density functions of the feature $X^{(j)}$ in classes C and E, respectively. The algorithm begins with non-parametric estimation of these densities using the kernel method:

$$\widehat{f}_C^{(j)}(x) = \frac{1}{N_C h_C} \sum_{i \in \Omega_C} \mathcal{K}\left(\frac{X^{(j)}(t_i) - x}{h_C}\right), \tag{2.6}$$

$$\widehat{f}_E^{(j)}(x) = \frac{1}{N_E h_E} \sum_{i \in \Omega_E} \mathcal{K}\left(\frac{X^{(j)}(t_i) - x}{h_E}\right), \tag{2.7}$$

where

$$\begin{aligned} \Omega_C &= \{i : d_i = 0 \wedge l_i = 0\}, \\ \Omega_E &= \{i : d_i = 1 \wedge l_i = 1\}, \end{aligned}$$

and

$$N_C = \#\Omega_C, \quad N_E = \#\Omega_E$$

represent the number of data records classified and labeled as correct and incorrect, respectively. The kernel function $\mathcal{K}(v)$ is defined as:

$$\mathcal{K}(v) = \begin{cases} 1, & \text{if } |v| \leq \frac{1}{2}, \\ 0, & \text{otherwise,} \end{cases} \tag{2.8}$$

which corresponds to the rectangular kernel function [15]. The parameters h_C and h_E are the kernel bandwidths. Guidance on tuning these smoothing parameters for finite sample sizes, as well as conditions for asymptotic consistency, can be found in [26]. A review of alternative kernel functions is presented in [24].

The estimated densities in Equations (2.6) and (2.7) are then used to compute the feature discriminative indicators:

$$\delta^{(j)} = \int_{\mathcal{D}} \left(\widehat{f}_C^{(j)}(x) - \widehat{f}_E^{(j)}(x) \right)^2 dx. \tag{2.9}$$

These indicators quantify the discriminative power of feature $X^{(j)}$. In the first stage, we reduce dimensionality—from $M \sim 10^4$ to $M' \sim 10^2$—by applying a δ-dependent threshold $\tau(\delta)$: we retain $X^{(j)}$ iff $\delta^{(j)} \geq \tau(\delta)$, where $\tau(\delta)$ is calibrated from the empirical distribution of $\{\delta^{(j)}\}_{j=1}^{M}$.

Remark 1 (Small Sample Sizes and Boosting***).*** Asymptotically, for $N_E \to \infty$, convergence of the kernel estimator to the true density has been formally proven. However, in practice, the system typically operates under transitional conditions with a limited number of observed errors ($N_E < \infty$). During this phase, full asymptotic properties may not be achieved. Several practical strategies can be considered:

- Bandwidth can be set automatically based on the sample variance [26], achieving a good bias-variance trade-off.
- Manual adjustment is also possible; for instance, increasing h_E when N_E is small so that the density $\widehat{f}_C^{(j)}(x)$ dominates.
- Boosting can be employed to emphasize rare or critical error cases.
- During the initial deployment phase, it may be beneficial to estimate $f_C^{(j)}(x)$ using both verified ($l_i = 1$) and unverified ($l_i = 0$) records. While this introduces statistical bias due to an unknown share of errors, it significantly reduces variance.

Remark 2 (Discrete and Binary Features***).*** For discrete features $X^{(j)}$, we propose the following smoothed estimators:

$$\widehat{p}_C^{(j)}(x) = \frac{1}{N_C} \sum_{i \in \Omega_C} \mathcal{K}\left(\frac{X^{(j)}(t_i) - x}{h_C} \right), \tag{2.10}$$

$$\widehat{p}_E^{(j)}(x) = \frac{1}{N_E} \sum_{i \in \Omega_E} \mathcal{K}\left(\frac{X^{(j)}(t_i) - x}{h_E} \right). \tag{2.11}$$

The discriminative index is then computed as:

$$\delta^{(j)} = \sum_x \left(\widehat{p}_C^{(j)}(x) - \widehat{p}_E^{(j)}(x) \right)^2. \tag{2.12}$$

2.1.4 Bandwidth selection and estimator updating

Bandwidth selection In our application, the bandwidth parameter smooths the estimation results of the discrete probability distribution. Since with a small number of N_B the results are not trustworthy, the use of a large h_B deactivates the influence of the historical error distribution on the performance of the classifier and sensitizes it to detect outliers. Many solutions found in the literature are either experimental based on the cross-validation technique or theoretical, but limited to special cases, e.g., assuming local smoothness of the estimated function. For our algorithm we recommend to use one of the following recomendations. Examples of such recommendations are

- Silverman's rule [62]
$$h = 0.9 \min \left(\widehat{\sigma}, \frac{IQR}{1.34} \right) N^{-1/5},$$
- Scott's rule [60]
$$h = 3.5 \widehat{\sigma} N^{-1/3},$$
where $\widehat{\sigma}$ and IQR denote standard deviation and interquartile range, respectively.

For large datasets typical in data warehouses, computing Equations (2.6) and (2.7) offline can be computationally expensive, so we consider online updating. For a selected attribute j and a point x, the offline kernel density estimator can be written as

$$\widehat{f}_C^{(j)}(x) = \frac{1}{N_C h_C} \sum_{i=1}^{N_C} \mathcal{K} \left(\frac{X^{(j)}(t_{[i]}) - x}{h_C} \right), \tag{2.13}$$

where $t_{[i]}$ is the (temporal) index of the i-th record classified as $d_{[i]} = C$, and N_C is their count. An analogous definition holds for $\widehat{f}_E^{(j)}(x)$.

Robustness to mislabels. In practice, some records may be mislabeled (e.g., $d_{[i]} = C$ while the true label is $\omega(t_{[i]}) = E$). The lemma below states sufficient conditions for consistency despite such errors.

Lemma 2.1.1. If

$$\lim_{N_C \to \infty} \mathcal{P}\big\{ c(t_{[i]}) = 1 \mid d_{[i]} = 0 \big\} = 0, \tag{2.14}$$

then for every continuity point x of $f_C^{(j)}(\cdot)$,

$$\widehat{f}_C^{(j)}(x) \xrightarrow{\mathbb{P}} f_C^{(j)}(x), \tag{2.15}$$

provided $h_C(N_C) \to 0$ and $N_C \, h_C(N_C) \to \infty$ as $N_C \to \infty$.

Proof. Let $S_C = \{ t_{[i]} : d_{[i]} = 0 \}$ be the index set of records classified as C, with $|S_C| = N_C$. Partition S_C into correctly and incorrectly labeled subsets:

$$T_C = \{ t_{[i]} \in S_C : \ c(t_{[i]}) = 0 \}, \qquad M_C = \{ t_{[i]} \in S_C : \ c(t_{[i]}) = 1 \},$$

and denote $N_T = |T_C| = N_C - N_E$, $N_E = |M_C|$. Write

$$\widehat{f}_C^{(j)}(x) = \frac{1}{N_C h_C} \sum_{t_{[i]} \in S_C} \mathcal{K}\Big(\frac{X^{(j)}(t_{[i]}) - x}{h_C}\Big) = \frac{1}{N_C h_C}\Big(\mathcal{K}_T + \mathcal{K}_M\Big),$$

where

$$\mathcal{K}_T = \sum_{t_{[i]} \in T_C} \mathcal{K}\Big(\frac{X^{(j)}(t_{[i]}) - x}{h_C}\Big), \qquad \mathcal{K}_M = \sum_{t_{[i]} \in M_C} \mathcal{K}\Big(\frac{X^{(j)}(t_{[i]}) - x}{h_C}\Big).$$

Factor the two contributions:

$$\widehat{f}_C^{(j)}(x) = \underbrace{\frac{N_T}{N_C}}_{\to 1} \underbrace{\frac{1}{N_T h_C}\mathcal{K}_T}_{=:\ \widetilde{f}_T(x)} + \underbrace{\frac{N_E}{N_C}}_{\to 0} \underbrace{\frac{1}{N_E h_C}\mathcal{K}_M}_{=:\ \widetilde{f}_M(x)}.$$

By assumption

$$\lim_{N_C \to \infty} \mathcal{P}\{\omega(t_{[i]}) = E \mid d_{[i]} = C\} = 0,$$

we have $\frac{N_E}{N_C} \xrightarrow{\mathbb{P}} 0$ and $\frac{N_T}{N_C} \xrightarrow{\mathbb{P}} 1$. Under the bandwidth conditions $h_C \to 0$ and $N_C h_C \to \infty$, we also have $N_T h_C \to \infty$ (since $N_T/N_C \to 1$), hence by standard KDE consistency (e.g., [25, 64, 42])

$$\widetilde{f}_T(x) = \frac{1}{N_T h_C}\mathcal{K}_T \ \xrightarrow{\mathbb{P}}\ f_C^{(j)}(x) \quad \text{for every continuity point } x.$$

For the mislabeled part, define

$$\widetilde{f}_M(x) = \frac{1}{N_E h_C}\mathcal{K}_M,$$

which is a kernel estimator of $f_E^{(j)}(x)$ based on the N_E mislabels (with bandwidth h_C). Under the usual kernel assumptions (integrable and bounded kernel, finite $f_E^{(j)}(x)$), $\widetilde{f}_M(x)$ is $O_{\mathbb{P}}(1)$. Therefore, by Slutsky's theorem,

$$\frac{N_E}{N_C}\widetilde{f}_M(x) \xrightarrow{\mathbb{P}} 0.$$

Combining the two parts yields

$$\widehat{f}_C^{(j)}(x) = \frac{N_T}{N_C}\widetilde{f}_T(x) + \frac{N_E}{N_C}\widetilde{f}_M(x) \xrightarrow{\mathbb{P}} f_C^{(j)}(x),$$

which proves the claim. □

Online updating with a local bandwidth. To avoid recomputing all kernels after each change of h, we apply the Wolverton–Wagner scheme [67] with individual bandwidths $h_C(i)$:

$$\widetilde{f}_C^{(j)}(x, N_C) = \frac{1}{N_C}\sum_{i=1}^{N_C} \frac{1}{h_C(i)}\mathcal{K}\Big(\frac{X^{(j)}(t_{[i]}) - x}{h_C(i)}\Big), \tag{2.16}$$

which yields the recursion

$$\widetilde{f}_C^{(j)}(x, N_C) = \frac{N_C - 1}{N_C}\widetilde{f}_C^{(j)}(x, N_C - 1) + \frac{1}{N_C}\frac{1}{h_C(N_C)}\mathcal{K}\Big(\frac{X^{(j)}(t_{[N_C]}) - x}{h_C(N_C)}\Big). \tag{2.17}$$

Lemma 2.1.2. If $h_C(N_C) \to 0$ and $\sum_{i=1}^{N_C} h_C(i) \to \infty$ as $N_C \to \infty$, then

$$\widetilde{f}_C^{(j)}(x, N_C) \xrightarrow{L^2} f_C^{(j)}(x) \tag{2.18}$$

at every continuity point of $f_C^{(j)}(\cdot)$, under standard kernel conditions (finite integral, boundedness, tail decay), cf. [23].

Proof. By the Parzen kernel assumptions in (2.8),

$$\int_{-\infty}^{\infty} |\mathcal{K}(v)|\, dv < \infty, \quad \sup_v |\mathcal{K}(v)| < \infty, \quad |v|\, \mathcal{K}(v) \to 0 \text{ as } |v| \to \infty.$$

Hence, for the Wolverton–Wagner estimator

$$\widetilde{f}_C^{(j)}(x, N_C) = \frac{1}{N_C} \sum_{i=1}^{N_C} \frac{1}{h_C(i)} \mathcal{K}\left(\frac{X^{(j)}(t_{[i]}) - x}{h_C(i)} \right),$$

we have

$$\lim_{N_C \to \infty} \mathbb{E}\, \widetilde{f}_C^{(j)}(x, N_C) = f_C^{(j)}(x) \int_{-\infty}^{\infty} \mathcal{K}(v)\, dv, \tag{2.19}$$

and

$$\lim_{N_C \to \infty} \frac{\operatorname{var}\big(\widetilde{f}_C^{(j)}(x, N_C)\big)}{\frac{1}{N_C^2} \sum_{i=1}^{N_C} \frac{1}{h_C(i)}} = f_C^{(j)}(x) \int_{-\infty}^{\infty} \mathcal{K}^2(v)\, dv. \tag{2.20}$$

Since in our case $\int \mathcal{K}(v)\, dv = 1$ (see (2.8)), (2.19) gives $\mathbb{E}\, \widetilde{f}_C^{(j)}(x, N_C) \to f_C^{(j)}(x)$. Moreover, under the bandwidth schedule $h_C(N_C) \to 0$ and $\sum_{i=1}^{N_C} h_C(i) \to \infty$, it follows (cf. Lemma 4.3 and Theorem 4.1 in [23]) that

$$\frac{1}{N_C^2} \sum_{i=1}^{N_C} \frac{1}{h_C(i)} \longrightarrow 0,$$

hence by (2.20) we have $\operatorname{var}\big(\widetilde{f}_C^{(j)}(x, N_C)\big) \to 0$. Therefore $\widetilde{f}_C^{(j)}(x, N_C) \to f_C^{(j)}(x)$ in L^2 at every continuity point of $f_C^{(j)}(\cdot)$, which proves (2.1.2). □

Adaptive version with a forgetting factor. In dynamic environments (changing error types and frequencies) we use *exponential* forgetting with $\lambda \in (0, 1)$:

$$\widetilde{f}_C^{(j)}(x, N_C) = \lambda\, \widetilde{f}_C^{(j)}(x, N_C - 1) + (1 - \lambda)\, \frac{1}{h_C(N_C)} \mathcal{K}\left(\frac{X^{(j)}(t_{[N_C]}) - x}{h_C(N_C)} \right), \tag{2.21}$$

$$\widetilde{f}_E^{(j)}(x, N_E) = \lambda\, \widetilde{f}_E^{(j)}(x, N_E - 1) + (1 - \lambda)\, \frac{1}{h_E(N_E)} \mathcal{K}\left(\frac{X^{(j)}(t_{[N_E]}) - x}{h_E(N_E)} \right). \tag{2.22}$$

The recursions (2.21)–(2.22) implement an *exponentially weighted* kernel sum; the effective memory length is approximately $N_{\text{eff}} \approx 1/(1 - \lambda)$. For a stationary setting and consistency goals one may take $\lambda_N = 1 - \frac{1}{N}$, which recovers equal weighting as in (2.16)–(2.17). For nonstationary settings, a fixed λ tracks changes at the price of a small asymptotic bias.

Remark 3 (Forecasting***).*** In cases where new data arrives rapidly and real-time density estimation is impractical, a forecasting technique akin to model predictive control can be applied. When computational efficiency outweighs estimation accuracy, a linear predictor of the form

$$\widetilde{f}_E^{(j)}(x, N_E) := \widetilde{f}_E^{(j)}(x, N_E - n_\alpha) + \frac{\widetilde{f}_E^{(j)}(x, N_E - n_\alpha) - \widetilde{f}_E^{(j)}(x, N_E - n_\beta)}{n_\beta - n_\alpha} n_\alpha \quad (2.23)$$

may be used. While such forecasts do not satisfy the formal properties of a probability density function, they are sufficient for computing $\delta^{(j)}$.

2.1.5 Correction when withdrawing a single record

Non-adaptive version (no forgetting) — explicit telescoping. If the operator confirms that record $t_{[i]}$ was erroneously used for class C, then

$$\begin{aligned}
\widetilde{f}_C^{(j)}(x, N_C - 1) &= \\
&= \widetilde{f}_C^{(j)}(x, N_C) - \frac{1}{i}\,\frac{1}{h_C(i)}\,\mathcal{K}\!\left(\frac{X^{(j)}\big(t_{[i]}\big) - x}{h_C(i)}\right)\,\frac{i}{i+1}\,\frac{i+1}{i+2}\cdot\ldots\cdot\frac{N_C-1}{N_C} \\
&= \widetilde{f}_C^{(j)}(x, N_C) - \frac{1}{N_C}\,\frac{1}{h_C(i)}\,\mathcal{K}\!\left(\frac{X^{(j)}\big(t_{[i]}\big) - x}{h_C(i)}\right),
\end{aligned}$$

and

$$N_C := N_C - 1\,.$$

The product $\frac{i}{i+1}\frac{i+1}{i+2}\cdots\frac{N_C-1}{N_C} = \frac{i}{N_C}$ telescopes, hence the factor $\frac{1}{N_C}$ after cancellation with the leading $\frac{1}{i}$. Since the bandwidth conditions of Lemma 2.1.2 (namely $h_C(N) \to 0$ and $\sum_{u=1}^{N} h_C(u) \to \infty$ as $N \to \infty$) remain unaffected by a single removal, the convergence claims still hold. Knowing the index i of the misprocessed record is therefore required, which motivates maintaining record indexing in the database. An analogous formula applies for class E.

Adaptive version (with forgetting). For (2.21) it is useful to make explicit that

$$\widetilde{f}_C^{(j)}(x, N_C) = \sum_{u=1}^{N_C} (1-\lambda)\,\lambda^{N_C-u}\,\underbrace{\frac{1}{h_C(u)}\,\mathcal{K}\!\left(\frac{X^{(j)}(t_{[u]}) - x}{h_C(u)}\right)}_{k_u(x)}.$$

Thus, the contribution of record $t_{[i]}$ to the current value equals $(1-\lambda)\lambda^{N_C-i}k_i(x)$. If we maintain the *unnormalized* EMA, a practical correction (without changing the counter N_C) is

$$\widetilde{f}_{C,\,\text{new}}^{(j)}(x, N_C) = \widetilde{f}_C^{(j)}(x, N_C) - (1-\lambda)\,\lambda^{N_C-i}\,k_i(x). \quad (2.24)$$

If a *generically normalized* version is used via the weight sum, i.e., $\overline{f}_C^{(j)}(x, N) = \widetilde{f}_C^{(j)}(x, N)/(1-\lambda^N)$ (assuming a zero start), then after withdrawing record $t_{[i]}$ and shortening the history to $N_C - 1$ we obtain

$$\overline{f}_C^{(j)}(x, N_C - 1) = \frac{(1-\lambda^{N_C})\,\overline{f}_C^{(j)}(x, N_C) \;-\; (1-\lambda)\,\lambda^{N_C-i}\,k_i(x)}{1-\lambda^{N_C-1}}. \quad (2.25)$$

Implementation remarks.

- In practice, removals of very old records ($N_C - i \gg N_{\text{eff}}$) can be ignored due to their negligible weights. For recent records it is beneficial to store $\{k_i(x), i\}$ or to recompute from the latest checkpoint.
- In both variants it is advantageous to index records and to buffer the contributions $k_i(x)$ for the currently evaluated points x (e.g., a grid), which enables $O(1)$ corrections without a full recomputation.

2.1.6 Initial Filtering of Features

The initial filtering step serves to remove clearly non-informative or poorly distributed features from the full set of extracted features. This phase operates before any supervised learning and is largely unsupervised, relying on general statistical properties of the feature vectors. The resulting a feature subset of size M' (with $M' \leq M$) is carried forward to the next stage of discriminative analysis.

The following criteria are applied:

1. **Low Variance Removal:** Features that do not vary significantly across records are unlikely to help in distinguishing between correct and erroneous entries. Mathematically, features with variance $\sigma_j^2 < \epsilon$ are excluded, where ϵ is a small, empirically chosen threshold.
2. **Sparsity Threshold:** Binary or categorical features (e.g., one-hot encoded n-grams) that occur in too few records are filtered out. Features with occurrence frequencies below a threshold τ (e.g., 0.5%) are discarded.
3. **Missing Value Rate:** Features that are undefined or missing in a significant portion of records (typically over 30–50%) are considered unreliable and removed.
4. **Redundancy Screening (Optional):** A fast, heuristic pre-check may be used to remove features that are linearly dependent or trivially duplicated versions of others (e.g., raw count and normalized count of the same token).

After this filtering, the feature set contains features that are at least statistically variable and potentially informative.

Remark 4 (Guidelines for feature cut-off***).*** For a one-hot indicator $Z \sim \text{Bern}(p)$ with prevalence p, the entropy is $H(Z) = h_2(p) = -p \log_2 p - (1-p) \log_2(1-p)$. Low-entropy (near-constant) columns inflate plug-in MI and destabilize normalized measures, hence we enforce a minimum-entropy (or minimum-prevalence) filter before scoring/selection. For text-scale vocabularies (e.g., n-grams), we combine an *expected-count* rule with an *entropy* rule and remove both extremely rare and extremely common substrings.

Rarity controls. (i) Expected-count rule: ensure the smallest cell in the $2 \times |Y|$ table exceeds ≈ 5 by setting $\text{min\_df} \geq 5/\pi_{\min}$ (or proportion $5/(N\pi_{\min})$), where $\pi_{\min}$ is the smallest target-class share. (ii) Entropy rule: keep columns with $H(Z) \geq H_{\min}$ (typ. 0.08–0.20 bit for large n-gram vocabularies).

Smoothing and bias correction. We estimate MI/entropy from contingency tables with light additive (Laplace/Lidstone) smoothing [34, 41, 31] and apply the Miller–Madow correction [43, 39] prior to computing d_R.

2.1.7 Feature Selection Using the Jaccard Similarity Index

At this stage of the algorithm, we operate on a significantly reduced set of features, denoted by M'. With this manageable number of dimensions, an analysis of similarity between features becomes computationally feasible. Features that encode similar or correlated information introduce redundancy, which unnecessarily increases dimensionality without improving classification performance.

Although correlation analysis is computationally efficient, it is not suitable in this context. As is well known, the correlation coefficient captures only linear dependencies. As a result, some strong but nonlinear or non-monotonic relationships may remain undetected.

To address this limitation, we propose a more general approach based on mutual information, an entropy-based measure of dependency between random variables. Specifically, we employ the Rajski (also known as Jaccard or probabilistic Jaccard) distance [51, 57, 58], defined as:

$$D_{j_1,j_2} = 1 - \frac{I\left(X^{(j_1)}, X^{(j_2)}\right)}{H\left(X^{(j_1)}, X^{(j_2)}\right)}, \tag{2.26}$$

where the mutual information $I(\cdot,\cdot)$ is given by

$$I\left(X^{(j_1)}, X^{(j_2)}\right) = \sum_{x_1}\sum_{x_2} \mathcal{P}\left(X^{(j_1)} = x_1 \wedge X^{(j_2)} = x_2\right) \tag{2.27}$$

$$\cdot \log \frac{\mathcal{P}\left(X^{(j_1)}=x_1 \wedge X^{(j_2)}=x_2\right)}{\mathcal{P}\left(X^{(j_1)}=x_1\right)\cdot\mathcal{P}\left(X^{(j_2)}=x_2\right)}, \tag{2.28}$$

and the joint entropy $H(\cdot,\cdot)$ is given by

$$H\left(X^{(j_1)}, X^{(j_2)}\right) = \sum_{x_1}\sum_{x_2} \mathcal{P}\left(X^{(j_1)} = x_1 \wedge X^{(j_2)} = x_2\right) \tag{2.29}$$

$$\cdot \log \frac{1}{\mathcal{P}\left(X^{(j_1)}=x_1 \wedge X^{(j_2)}=x_2\right)}. \tag{2.30}$$

The final selection of features is governed by a weighted criterion that accounts for both feature quality (their ability to discriminate between classes) and mutual independence.

Let I_C and I_S denote the sets of candidate and selected feature indices, respectively.

Initialization: Set

$$I_C = \{1, 2, \ldots, M'\}, \quad I_S = \{\}.$$

Step 1: For all candidate features $X^{(i)}$, where $i \in I_C$, compute the estimated minimum distance to the already selected features:

$$\widehat{\Delta}^{(i)} = \begin{cases} 0, & \text{if } I_S = \emptyset, \\ \min\limits_{j \in I_S} \widehat{D}_{i,j}, & \text{otherwise.} \end{cases} \tag{2.31}$$

Step 2: Select the feature $X^{(i^*)}$ for which the following criterion is maximized:

$$i^* = \arg\max_{i \in I_C} \left(\alpha \widehat{\delta}^{(i)} + (1-\alpha)\widehat{\Delta}^{(i)}\right). \tag{2.32}$$

Append i^* to the selected feature set:

$$I_S := I_S \cup \{i^*\}, \quad I_C := I_C \setminus \{i^*\}.$$

Repeat Steps 1 and 2 until the desired number of features m is selected, i.e., until $\#I_S = m$. The procedure then terminates.

The parameter $\alpha \in [0, 1]$ controls the balance in this convex combination: $\widehat{\delta}^{(i)}$ represents the individual discriminative power of the feature, while $\widehat{\Delta}^{(i)}$ promotes diversity by favoring features that are dissimilar to those already selected.

2.1.8 Neural Network–Based Classifier

Initially, a simple error–detection scheme was considered in which each record receives a confidence index equal to the product of m feature–wise indices, i.e.,

$$\begin{aligned} C\left(t_i\right) &\triangleq \mathcal{P}\left\{c_i = 0 | \bigcap_{j=1}^{m} X^{(j)}\left(t_i\right) = x^{(j)}\right\} \\ &= \prod_{j=1}^{m} \mathcal{P}\left\{c_i = 0 | X^{(j)}\left(t_i\right) = x^{(j)}\right\} = \prod_{j=1}^{m} C_j\left(x_i^{(j)}\right), \end{aligned} \tag{2.33}$$

where

$$\begin{aligned} C_j(x) &\triangleq \mathcal{P}\left\{c = 0 | X^{(j)}\left(t\right) = x\right\} = \lim_{dx \to 0} \frac{p f_C^{(j)}\left(x\right) dx}{q f_E^{(j)}\left(x\right) dx + p f_C^{(j)}\left(x\right) dx} \\ &= \frac{f_C^{(j)}\left(x\right)}{\frac{q}{p} f_E^{(j)}\left(x\right) + f_C^{(j)}\left(x\right)}. \end{aligned} \tag{2.34}$$

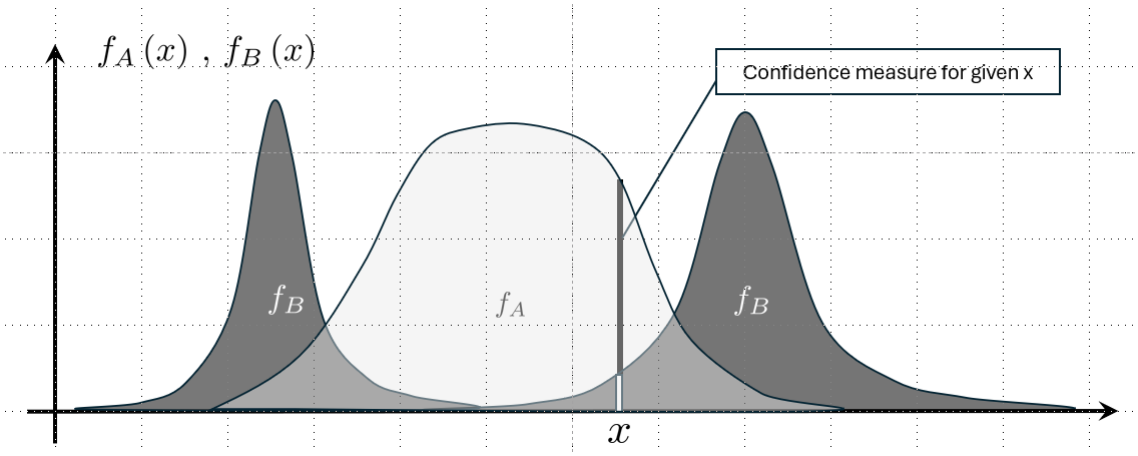


Figure 2.2: Confidence Indicator.

The equation (2.34) represents confidense index for one feature $X^{(j)}$. Graphical interpretation of this indicator is shown on Fig. 2.2. Equation (2.33) combines information among selected feature set. This construction assumes pairwise independence of features. However, reducing similarity via entropy–based criteria does not guarantee independence. In the absence of additional structural knowledge, we therefore employ a neural network to

model potential higher–order and nonlinear interactions among features. While this choice may reduce interpretability, it is expected to improve detection performance by capturing dependencies that the multiplicative model cannot represent.

The learning dataset consists of m–dimensional inputs

$$\left\{\left(x_i^{(1)}, x_i^{(2)}, ..., x_i^{(m)}\right)\right\}_{i=1}^{N},$$

and scalar outputs $\{y_i\}_{i=1}^{N}$, where $x_i^{(j)}$ denotes the value of the jth feature for the ith record, and

$$y_i = \begin{cases} 1, \text{ for an erroneous record} \\ 0, \text{ for a correct record} \end{cases}.$$

A four–layer feed–forward neural network was constructed for binary classification in the CMDB error–detection setting. The input layer contains m neurons, matching the number of selected features, thereby enabling the model to capture multi–feature nonlinear interactions, in contrast to the product–based confidence model (cf. [**?**]). Two hidden layers with four neurons each were used to ensure computational tractability on a standard PC; empirical tests indicated that increasing the hidden width did not materially improve performance. The network concludes with a single–neuron output layer. All activation functions were set to sigmoidal. Training was performed using *Keras* within the *TensorFlow* framework, with the *Adam* optimizer and the *MSE* objective; the training schedule was selected empirically. The output layer produces a scalar prediction for each tuple, which is interpreted as an error likelihood. The network architecture is shown in Figure 2.3.[1]

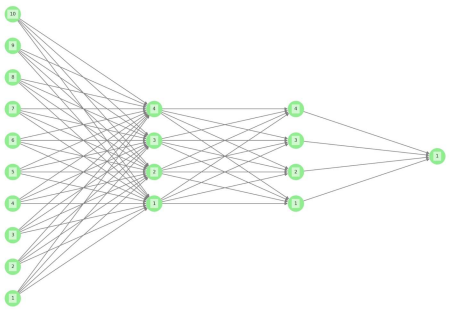


Figure 2.3: A four–layer neural network used for classification.

The network was trained on 15,000 records and evaluated on a held–out test set of 5,000 records. Both sets contained 10% erroneous tuples spanning diverse error types (e.g., extra or missing characters, digit transpositions, internal spaces). Test records were ranked in descending order by the predicted scores. For comparison, the confidence–index method was applied by ranking $C(t_i)$ values (see (20)). For both methods, the top η entries in each ranking were nominated for manual inspection. The resulting precision values are compared in the subsequent section.

2.1.9 Experiment

Structure of the Test Sample

The algorithm was tested on a synthetic dataset simulating serial numbers. The dataset comprises five distinct formats corresponding to valid serial numbers (representing different

[1]Figure prepared in Lucid.

manufacturers or models), along with three distinct types of erroneous formats.

- **Valid serial number structures:**

```
'ABC': '{:04d}{}ABC',
'DEF': 'DEF{}{}{}{}{}',
'GHI': 'GHI-{}{}{}{}{}{}',
'JKL': 'JKL{}{:02d}-{:02d}',
'MNO': 'MNO{}-{:02d}{}'
```

- **Erroneous serial number patterns:**

```
'ER1': '{}{}{}{}{}',
'ER2': '{}-TOLONG-{}{}SERIAL{}',
'ER3': 'ERR{}-{}'
```

Metaparameters and Hyperparameters

In the design and execution of machine learning algorithms for CMDB data quality improvement, both metaparameters and hyperparameters play a critical role in controlling model behavior, generalization, and computational efficiency. This section distinguishes between these two categories and outlines the specific parameters used in the experimental framework.

Metaparameters

Metaparameters refer to high-level configuration values that govern the structure of the data preprocessing pipeline and the design of the learning experiment. Unlike hyperparameters, which directly influence the behavior of the learning algorithm during training, metaparameters define the overall strategy or architecture of the solution. They are typically set manually and not optimized via training.

The primary metaparameters used in this research include:

- **Feature extraction strategy**: Choice of syntactic and semantic features, including whether to use n-gram tokenization (e.g., bigrams or trigrams), categorical encoding, or manually engineered indicators.
- **Number of features** M: The initial size of the feature space generated before filtering. This is determined based on domain knowledge and the complexity of the input schema.
- **Filtering thresholds**: Cutoff values used for removing low-variance features or infrequent tokens during initial feature selection.
- **Selection criteria for training data**: Rules used to define the initial labeled training set, e.g., based on expert confidence, heuristics, or prior correction history.
- **Ranking quota** N: Number of records selected per iteration for manual review and correction, typically set to $N = 100$ in the baseline scenario.

These metaparameters are typically held constant across experimental runs to ensure comparability and reproducibility of results.

Hyperparameters

Hyperparameters are values that govern the internal behavior of machine learning algorithms and must be set prior to training. Unlike model parameters (which are learned from the data), hyperparameters influence the training dynamics and model capacity.

The following hyperparameters were used in the core components of the pipeline:

- **Kernel bandwidth** h in density estimation: Controls the smoothness of the estimated probability densities for the distributions of features in correct vs. erroneous records. A small h leads to overfitting, while a large h oversmooths the data.
- **Jaccard distance threshold** τ_J: Defines the minimum distance between two features for them to be considered non-redundant. Features with pairwise Jaccard distances below this threshold are pruned.
- **Feature scoring weight** α: Balancing coefficient in the combined scoring function:

  $$\text{Score}(f_i) = \alpha \cdot \delta(f_i) + (1 - \alpha) \cdot \Delta(f_i)$$

  where $\delta(f_i)$ denotes the feature's discriminative ability, and $\Delta(f_i)$ its informativeness. The value of α is typically selected via grid search (e.g., $\alpha \in \{0.3, 0.5, 0.7\}$).
- **Neural network architecture**: The number of layers, number of neurons per layer, and activation functions (e.g., ReLU, sigmoid) are defined as part of the model configuration. For instance, a common setup may involve a 3-layer fully connected network with hidden sizes $[128, 64, 32]$.
- **Learning rate** η: The step size used in the optimizer (e.g., Adam or SGD). This parameter significantly impacts convergence and generalization.
- **Batch size and number of epochs**: Control how many samples are processed in each training iteration and how many full passes are made through the dataset. These parameters are tuned empirically.
- **Dropout rate** p: Regularization parameter used to prevent overfitting by randomly deactivating a fraction of neurons during training. Typical values are in the range $p \in [0.1, 0.5]$.

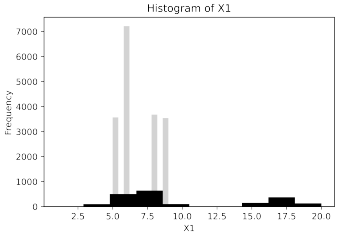
All hyperparameters were selected based on a combination of empirical tuning, cross-validation, and domain-specific heuristics. Where applicable, sensitivity analyses were conducted to assess the impact of hyperparameter variations on final model performance.

The simulation sample included 18,000 valid records and 2,000 erroneous entries. The placeholders in the format strings (denoted by curly braces) were filled using randomly generated values, respecting the expected type and format constraints.

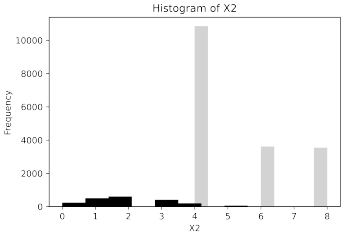
Feature Candidates

Out of many potential features, a set of seven simple features was selected for analysis. One of the features, $X^{(2)}$, was included twice under different labels ($X^{(2)}$ and $X^{(3)}$) to demonstrate the algorithm's behavior when redundant (i.e., fully dependent) features are included.

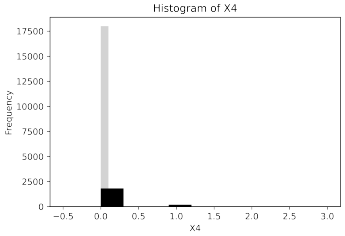
- $X^{(1)}$: Number of alphabetic characters



(a) $X^{(1)}$

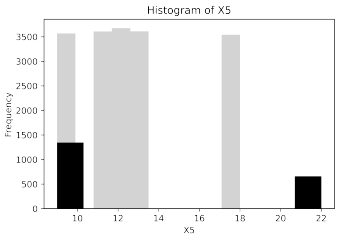


(b) $X^{(2)}$ and $X^{(3)}$

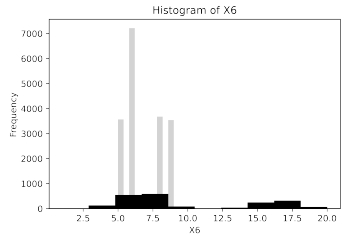


(c) $X^{(4)}$

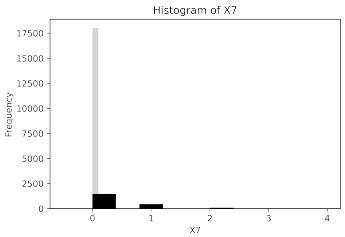
Figure 2.4: Histograms of $X^{(1)}$, $X^{(2)}$, $X^{(3)}$, $X^{(4)}$.



(a) $X^{(5)}$



(b) $X^{(6)}$



(c) $X^{(7)}$

Figure 2.5: Histograms of $X^{(5)}$, $X^{(6)}$, $X^{(7)}$.

- $X^{(2)}$: Number of numeric digits
- $X^{(3)}$: Number of numeric digits (same as $X^{(2)}$)
- $X^{(4)}$: Number of spaces
- $X^{(5)}$: Total string length
- $X^{(6)}$: Number of uppercase letters
- $X^{(7)}$: Number of lowercase letters

Figures 2.4 and 2.5 show histograms illustrating how each feature differentiates between valid and erroneous records. Gray bars represent correct records, while black bars represent errors.

Estimating $\widehat{\delta}^{(j)}$ and Selecting the Initial Feature

For each feature $X^{(j)}$, the value of $\widehat{\delta}^{(j)}$ was calculated using equation (**??**) to assess its effectiveness in error detection. A bandwidth of $h_A = h_B = 0.01$ was used for kernel density estimation.

- $X^{(1)}$: $\widehat{\delta}^{(1)} = 0.16$
- $X^{(2)}$: $\widehat{\delta}^{(2)} = 0.56$
- $X^{(3)}$: $\widehat{\delta}^{(3)} = 0.56$
- $X^{(4)}$: $\widehat{\delta}^{(4)} = 0.02$
- $X^{(5)}$: $\widehat{\delta}^{(5)} = 0.39$

- $X^{(6)}$: $\widehat{\delta}^{(6)} = 0.16$
- $X^{(7)}$: $\widehat{\delta}^{(7)} = 0.14$

The highest scores were obtained by $X^{(2)}$ and $X^{(3)}$, confirming their effectiveness. Since these two features are identical, including both offers no additional benefit. Based on $X^{(2)}$, the Confidence Indicator $C_2(x)$ (cf. Eq. 2.34) was computed, and records were sorted accordingly.

Rajski Distance Calculation

To evaluate the independence between features, Rajski distances were computed for all feature pairs. A higher value indicates a higher degree of informational independence. . For example, $X^{(2)}$ and $X^{(3)}$ show a distance of 0.00, as expected for identical features.

Performance Metric

To evaluate performance, the Top-K Precision metric (P@K) was used [30]. For a given K, it is defined as:

$$\text{P@K} = \frac{\text{Number of true errors in Top-K}}{K}$$

In the simulation, P@K was calculated for $K \in \{10, 100, 200, 300, 400, 500, 600, 700\}$.

Balancing Feature Quality and Independence

Using the previously calculated values of δ and Rajski distances, the parameter α was optimized for the experiment. The criterion was maximization of P@700. Results were compared for different values of α, including 0.55 and 0.1, and across different subsets of selected features.

The simulation confirmed that feature independence and detection quality must be jointly optimized, and that the value of α should be chosen depending on the number of features used.

Applying the Algorithm with N-Gram Encoding

To evaluate character-sequence-based detection, n-gram features of lengths 2 to 6 were extracted using `CountVectorizer` from the Scikit-learn library [55]. A vocabulary of 2000 most frequent tokens was created. Each record was represented as a sparse binary vector (bag-of-n-grams). These 2000 features were evaluated alongside the original 7 features. Feature selection was based on $\widehat{\delta}^{(j)}$ and mutual independence. Table 2.1 lists selected features.

2.1.10 Experimental Results

The experiment was conducted on anonymized data with a structure identical to that of the production database. The focus was placed on the analysis of text strings representing serial numbers. As a preprocessing step, all possible substrings of lengths ranging from 2 to 6 characters were generated. For each examined record, Boolean variables indicating the presence or absence of these substrings were defined and used as features. Additionally, a set of expert-defined features was incorporated into the feature pool.

Subsequently, using the methodology described in previous sections, the $m = 20$ most discriminative features were selected for classifier construction. Examples of selected features and their corresponding detection ability indicators $\widehat{\delta}^{(j)}$ are presented in Table 2.1.

Feature $X^{(j)}$	**Indicator** $\widehat{\delta}^{(j)}$
Number of digits in a string	0.31
Number of lowercase letters in a string	0.22
Occurrence of substring "-tolo"	0.16
Occurrence of substring "erial"	0.15
Occurrence of substring "olong"	0.15
Occurrence of substring "eri"	0.10
Number of spaces in a string	0.08

Table 2.1: Examples of automatically selected features along with their corresponding error detection ability indicators.

The resulting feature set included both expert-defined and non-intuitive, automatically discovered patterns, demonstrating the effectiveness of the selection procedure.

Figure 2.6 illustrates the superiority of the neural network approach in terms of detection precision. Notably, the drop in performance observed at the threshold $\eta = 500$ results from the exhaustion of detectable errors in the dataset. In contrast, the traditional rule-based method operates faster but with significantly lower detection efficiency.

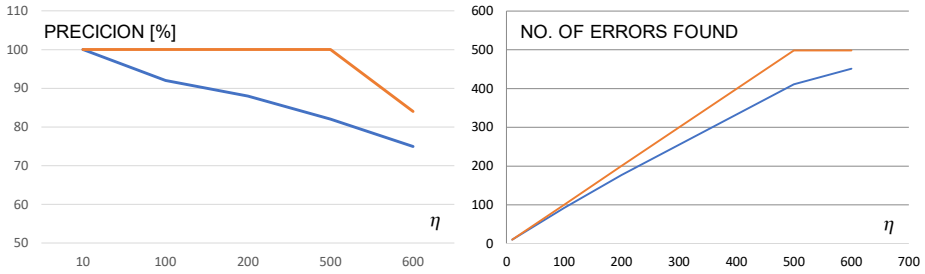


Figure 2.6: Precision and number of errors detected versus cut-off threshold η. Comparison between trust coefficient-based and neural network-based approaches.

Figure 2.7 presents the influence of the number of selected features m on detection precision at a fixed threshold $\eta = 500$. The results confirm that even a small subset of well-chosen features (e.g., $m = 10$) is sufficient to effectively detect a significant portion of errors. This observation suggests that the examined problem instance is relatively low in complexity, likely limited to a narrow class of anomalies. In practical applications, the number of selected features m will be increased appropriately to account for more complex scenarios.

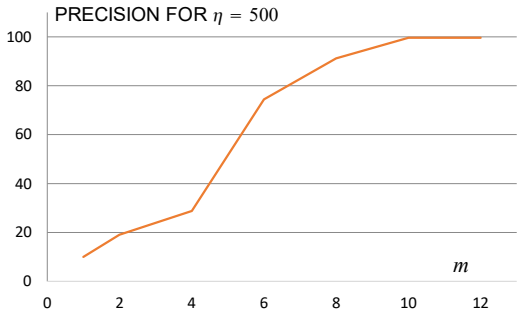


Figure 2.7: Precision [%] for $\eta = 500$ versus the number of selected features m.

Remark 5. When experimentally comparing two approaches, it is inherently challenging to ensure identical and fair conditions. The traditional method does not rely on any external libraries, which may be beneficial in environments with heightened security or regulatory constraints (e.g., critical infrastructure). It is also computationally lightweight and yields interpretable results. However, it is based on the strong assumption of feature independence, which may not hold in real-world data.

In contrast, the neural network approach incurs greater computational costs and requires careful tuning of hyperparameters. Nevertheless, it offers a key advantage: the ability to model interactions among features, leading to superior detection precision. This makes it particularly suitable for scenarios where error patterns are complex and highly dependent on feature combinations.

Economic Evaluation

Assuming a verification cost of \$100 per record, and a 10% error rate in a database of 5000 entries, random verification requires \$1000 to detect one error. In contrast, the proposed method can reduce the detection cost to approximately \$109. Over five months of inspecting 100 records monthly, this leads to an increase in CMDB quality from 90% to 98%, compared to only 91% with random inspections.

2.2 Distance–Based Detection of Suspicious Records

This section introduces a distance–based perspective on detecting suspicious records in the CMDB. The guiding assumption is a form of *neighborhood consistency*: objects that are similar in their semantic attributes should be close to one another in the feature space, and conversely, dissimilar objects should be far apart.

Unlike the problem in Section 2.1, no prior information regarding the errors is assumed. Let $\mathbf{x}_i \in \mathbb{R}^m$ denote the feature vector of record i, and let $d(\mathbf{x}_i, \mathbf{x}_j)$ be a distance function defined on the joint feature space (after appropriate scaling and encoding). Under the consistency assumption, a record $\mathbf{x}_i$ is deemed *suspect* if its pattern of proximities is contradictory—e.g., it is unusually close to some record(s) along one feature (or a small subset of features) while being simultaneously far from its peers in the remaining dimensions. Such cross–feature inconsistencies are symptomatic of typographical errors, malformed identifiers, misplaced values, or broken relationships.

Operationally, this viewpoint requires (i) a well–posed choice of distance(s) for heterogeneous data (numerical, categorical, and textual features), (ii) robust standardization or weighting to prevent a single feature from dominating the geometry, and (iii) decision rules that convert geometric evidence into nominations for human verification. While a single global metric such as the (weighted) Euclidean or Mahalanobis distance can be effective, local structure often matters: density variations across classes of configuration items, or sparse clusters induced by naming conventions, call for neighborhood–aware criteria (e.g., comparisons to a record's nearest neighbors).

Two complementary procedures are studied in detail. The first is an *all–pairs analysis*, which evaluates $d(\mathbf{x}_i, \mathbf{x}_j)$ for all $i \neq j$ and derives per–record anomaly indicators from the distribution of pairwise distances (e.g., quantiles or contrast measures across features). Although quadratic in the number of records, this approach provides a global baseline and exposes outliers that are isolated from the population. The second is a *nearest–neighbor method*, which replaces global comparisons with local consistency checks: each record is contrasted with its k closest peers, and nomination scores are computed from deviations between its observed neighborhood and the expected one. This local approach scales more favorably and adapts naturally to nonuniform densities. Together, these methods translate geometric inconsistencies into ranked lists of candidates for expert inspection, preserving the human–in–the–loop paradigm while exploiting the structure of the feature space.

2.2.1 Problem Statement

Let us consider a two-column dataset (not necessarily numeric)

$$\{(x_k, y_k)\}_{k=1}^{N} \tag{2.35}$$

where $x_k \in X$ are independent/explanatory variables (regressors), $y_k \in Y$ are dependent variables, and N denotes number of data records. Let us also introduce the vectors of numeric features

$$F(x_k) = (f_1(x_k), ..., f_n(x_k))^T \tag{2.36}$$
$$G(y_k) = (g_1(y_k), ..., g_m(y_k))^T \tag{2.37}$$

where

$$f_i(x_k) \in \mathcal{R},\ g_j(y_k) \in \mathcal{R}, \quad i = 1, 2, ..., n, \quad j = 1, 2, ..., m, \quad k = 1, 2, ..., N \tag{2.38}$$

and n, m denote number of considered features of x_k and y_k, respectively. We introduce the following distance metrics

$$D_x\left(x_{k_1}, x_{k_2}\right) = \left\|F\left(x_{k_1}\right) - F\left(x_{k_2}\right)\right\| \triangleq w_0 + \sum_{i=1}^{n} w_i d_x(f_i\left(x_{k_1}\right), f_i\left(x_{k_2}\right)), \tag{2.39}$$

$$D_y\left(y_{k_1}, y_{k_2}\right) = \left\|G\left(y_{k_1}\right) - G\left(y_{k_2}\right)\right\| \triangleq \sum_{j=1}^{m} v_j d_y(g_j\left(y_{k_1}\right), g_j\left(y_{k_2}\right)), \tag{2.40}$$

where w_i and v_j denote the weights determining sensitivity of D_x and D_y on particular features. Symbols $d_x()$ and $d_y()$ represent measure of distance in the feature spaces (2.36) and (2.37), respectively, e.g., square or absolute function. When a feature determines the object is belonging to a given class and assumes countable values, the indication function can also be used.

Let x be the string variable representing the name of a device, and let y be its serial number. As examples of the extracted features $f_i\left(x\right)$, $g_j\left(y\right)$ the following can be given: total number of symbols, number of numeric symbols, number of spaces, and index of substring found in a given/external dictionary.

The aim of the research is to develop a method that will detect cases in which the distance $D_y\left(y_{k_1}, y_{k_2}\right)$ in the set Y will be large for a relatively close to each other elements $\left(x_{k_1}, x_{k_2}\right)$ in X. Obviously, this step is preliminary as not all detected cases will be manually classified as errors. In order to optimize the operation of this stage, after the final decision is made, it is important to tune the weights

$$w = \left(w_0, w_1, w_2, ..., w_n\right)^T \text{ and } v = \left(v_1, v_2, ..., v_n\right)^T \tag{2.41}$$

in such a way as to maximize the effectiveness of automatic detection.

Below, we show three ideas on how to accomplish the above task. We start with the so-called complete search, which is the most computationally expensive. Next, we propose two faster methods, based on the nearest neighbor approach and kernel selection technique.

2.2.2 Solution

To solve the problem stated in this chapter—namely, to nominate records that are likely to be erroneous without relying on prior error labels —- we adopt a unified distance–based strategy grounded in the principle of neighborhood consistency[47]. Objects that are semantically similar should be close in the feature space, whereas records that violate this expectation (e.g., being close along one attribute but far along others) are flagged for manual verification. The solution comprises three complementary operators that can be deployed under different computational budgets and latency constraints: (i) a global *All–Pairs* ranking, (ii) a local *Nearest–Neighbor* (NN) test, and (iii) a nonparametric *Kernel Regression* consistency check. All three rely on the distances $D_x(\cdot,\cdot)$ and $D_y(\cdot,\cdot)$ defined earlier and can be optionally combined with categorical gating (e.g., brand/model) to avoid cross–class comparisons. In production, the system selects one operator or a cascade thereof, trading off detection power and throughput.

All–Pairs operator (global). For each pairs $\left\{\left(x_{k_1}, x_{k_2}\right)\right\}_{k_1,k_2=1,k_1\neq k_2}^{N}$ compute

$$\varkappa_{k_1,k_2} = \frac{D_y\left(y_{k_1}, y_{k_2}\right)}{D_x\left(x_{k_1}, x_{k_2}\right)} \tag{2.42}$$

Pairs (k_1, k_2) with the largest $\varkappa$ are deemed the most inconsistent (large change in y unsupported by x–space proximity) and are nominated for inspection. This operator is fully unsupervised and provides a global view of inconsistencies; its computational cost scales with the number of unique pairs, of order $N^2/2$, and can thus be time–consuming for very large N.

Nearest–Neighbor operator (local). As a scalable alternative, we restrict attention to each record's nearest neighbor in D_x. For each $k = 1, 2, \ldots, N$ the nearest neighbor k' is

$$k' = k'(k) = \arg\min_{k2} D_x\left(x_k, x_{k_2}\right), \tag{2.43}$$

and we compute the local inconsistency index

$$\chi_k = \frac{D_y\left(y_k, y_{k'}\right)}{D_x\left(x_k, x_{k'}\right)} \tag{2.44}$$

Records with the highest χ_k are nominated. Compared to (2.42), this operator reduces the number of distance evaluations (here of the order $N^2/4$ under a naive implementation, and lower with indexing), making it suitable for larger datasets while emphasizing local geometric coherence.

Kernel Regression operator (nonparametric consistency). A smooth, model–free consistency score is obtained by regressing $G(y)$ on x via kernel weights:

$$\widehat{R}\left(x\right) = \frac{\sum_{k=1}^{N} G\left(y_k\right) K\left(\frac{D_x(x_k,x)}{h}\right)}{\sum_{k=1}^{N} K\left(\frac{D_x(x_k,x)}{h}\right)}, \tag{2.45}$$

where $K(\cdot)$ is a kernel and h is the bandwidth. For a given x_k we compare the observed target to its local expectation,

$$\gamma_k = \left\| G\left(y_k\right) - \widehat{R}\left(x_k\right) \right\|. \tag{2.46}$$

Large γ_k indicates lack of support from nearby records in x–space and triggers nomination. This operator is particularly attractive for online screening of newly arriving data, where $\widehat{R}(\cdot)$ enables rapid, single–query evaluation.

Operational use. The three operators constitute a robust solution to the problem statement: each converts distance evidence into a scalar nomination score and produces a ranked list of candidates for expert review. The All–Pairs operator offers maximal global sensitivity; the NN operator provides a computationally lighter, locality–aware alternative; and the Kernel Regression operator yields smooth, nonparametric consistency checks amenable to streaming scenarios. Method comparision is shown in 2.2. In practice, thresholds on $\varkappa_{k_1,k_2}$, χ_k, or γ_k (or quotas on the top–ranked records) define the daily inspection budget, and adaptive retraining integrates verified outcomes to refine future nominations.

2.2.3 All–Pairs Distance Experiment

This section presents an unsupervised, distance–based procedure for nominating suspicious records in a dataset of vehicle identification numbers (VINs). The central premise is *neighborhood consistency*: records that are semantically similar (e.g., belong to the same brand)

Table 2.2: Summary of distance–based methods considered in this work.

Method	**Inputs**	**Key parameters**	**Complexity (baseline)**	**Primary output**
All–Pairs	Feature vectors $\mathbf{x}_i$, distance $d(\cdot,\cdot)$, optional gating	Feature weights; gating rule (e.g., same brand)	$O(N^2 m)$	Ranked pairs; per–record occurrence counts
Nearest Neighbors	As above, plus neighbor index	k (neighbors); feature weights; gating	$O(N \log N)$ (with index)	Per–record inconsistency scores; ranked records
Kernel Regression	As above; optional labels or pseudo–labels	Bandwidth h; kernel $K(\cdot)$; feature weights	$O(N^2)$ (naive); reduced with approximation	Continuous anomaly score per record; ranked records

should be close in the selected feature space, whereas erroneous entries tend to be isolated or form inconsistent neighborhoods. Accordingly, we rank *pairs* of records by a distance functional and identify records that repeatedly appear in the top portion of the ranking as prime candidates for manual verification. In contrast to supervised settings, the proposed ranking does not rely on prior labels during scoring; labels are used only for ex–post evaluation.

Objective and Setting

The goal is to determine how many injected (synthetic) errors can be detected before the first *false positive* pair (i.e., a pair consisting of two correct records) appears in the ranking. To this end, we (i) compute distances for *all* ordered pairs of records; (ii) suppress cross–brand comparisons via a brand–consistency gate; (iii) sort pairs in descending order of the gated distance; and (iv) scan the ranking until the first pair with two correct records is encountered. Let N denote the number of records. In the reported experiment $N = 1000$, and 10 VINs were manually manipulated to create distinct errors.[2]

Brand Gate and Distance Functional

The brand (class) variable is discrete. To avoid conflating inter–brand heterogeneity with error–driven anomalies, we introduce an indicator I_y that *enables* distance accumulation only for pairs from the same brand:

$$I_y = \begin{cases} 1 & \text{for the same brand,} \\ 0 & \text{for different brands.} \end{cases} \tag{2.47}$$

Let D_x denote the feature–space distance as defined in (2.39), computed on the filtered feature set and scaled by a nonnegative weight vector w. In this experiment we used

$$w^\top = [0.0001,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1],$$

and instantiated the feature mapping $F(\cdot)$ to include simple, interpretable text–derived counts (letters, digits, spaces, non–ASCII characters, total length, uppercase letters, lowercase letters, and punctuation markers). Intuitively, D_x measures differences in length and character–class composition between two VINs. The final score for a pair (i, j) is the gated distance $D_x(\mathbf{x}_i, \mathbf{x}_j)\, I_y(i, j)$.

[2]The manipulations include typical VIN defects such as character insertions/deletions, transpositions, and spacing anomalies. Exact edit types are immaterial to the unsupervised scoring but are used for qualitative discussion.

Table 2.3: All–pairs experiment: parameters and design choices.

Dataset size	$N = 1000$ records (VINs).
Injected errors	10 manually manipulated VINs (used only for ex–post evaluation).
Pairs evaluated	$N(N-1) = 1000 \times 999$ ordered pairs (gating can zero out cross–brand pairs).
Brand gate	I_y from (2.47); distances across different brands are ignored.
Distance D_x	As in (2.39); computed on $F(\cdot)$ after standardization, using weights $w^\top = [0.0001, 1, 1, 1, 1, 1, 1, 1]$.
Feature map $F(\cdot)$	Counts of: letters, digits, spaces, non–ASCII characters, total length, uppercase letters, lowercase letters, punctuation markers.
Ranking	Pairs sorted in *descending* order of $D_x I_y$.
Cut–off rule	Scan the ranking until the *first* pair (i, j) with both records correct appears (first false positive).
Outcome measures	(i) number of distinct erroneous VINs encountered before the first false positive; (ii) per–error frequency of appearance among the top pairs.

Experiment Parameters

Table 2.3 summarizes the parameters and design choices.

Procedure

1. **Preprocessing:** Apply the feature filtering pipeline to obtain a reduced set for $F(\cdot)$; standardize components to comparable scales; fix the weight vector w.
2. **Pairwise scoring:** For each ordered pair (i, j) with $i \neq j$, compute $I_y(i, j)$ and $D_x(\mathbf{x}_i, \mathbf{x}_j)$ per (2.39); set $S_{ij} = D_x(\mathbf{x}_i, \mathbf{x}_j)\, I_y(i, j)$.
3. **Ranking:** Sort all pairs by S_{ij} in descending order. Maintain a per–record counter of how often a record appears in the top of the list.
4. **Stopping rule:** Traverse the ranking until the first pair (i, j) is encountered such that both records are correct (first false positive). Let $K^\star$ denote this rank.
5. **Evaluation:** Report (a) the number of distinct injected errors encountered among the top $K^\star$ pairs; (b) for each injected error, the number of occurrences among the top $K^\star$ pairs.

Computational Considerations

The all–pairs approach is quadratic in N (specifically $O(N^2 m)$ for m features). For $N = 1000$ this yields 999,000 ordered pairs; with efficient vectorization and gating, this is tractable on a workstation. However, for $N = 10^4$ the pair count grows to $\approx 10^8$, making the approach time–consuming without parallelization or early–pruning strategies. In practice, I_y eliminates many cross–brand pairs by setting $S_{ij} = 0$, and approximate nearest–neighbor prefilters can be used to shortlist candidate pairs before exact scoring.

Table 2.4: Per–error frequencies among the top $K^\star = 1127$ pairs. "Record ID (row)" uniquely identifies the manipulated VIN in the dataset. "Occurrences" counts how many of the top $K^\star$ pairs contained that erroneous record. Two injected errors did not appear before the first false positive and are therefore absent.

Error label	**Record ID (row)**	**Occurrences in top $K^\star$ pairs**
E1	8	15
E2	32	295
E3	67	75
E4	229	228
E5	296	14
E6	320	233
E7	363	171
E8	781	101

Results

With $N = 1000$ and 10 injected errors, the ranking was traversed until the first false positive pair occurred at $K^\star = 1127$. By this point, 8 out of the 10 injected errors had already appeared in at least one high–distance pair. Table 2.4 reports, for each identified erroneous record, its row index in the dataset and the number of times it appeared among the top $K^\star = 1127$ pairs.

Interpretation. High occurrence counts (e.g., E2, E4, E6) indicate records that are consistently distant from their brand–peers across multiple partners, i.e., objects that form numerous large–distance pairs and are thus strong error candidates. Lower counts (e.g., E1, E5) are consistent with milder perturbations (small edits) or with errors that are close to dense clusters and therefore generate fewer extreme distances. The two injected errors not present among the top $K^\star$ pairs are plausibly weak perturbations or occur in locally sparse regions where cross–brand gating reduces the number of admissible comparisons.

Discussion and Practical Guidance

Sensitivity to weights. The choice of w directly shapes D_x. Emphasizing length and digit counts, for example, improves detection of truncations, insertions, and transpositions, whereas larger weights on non–ASCII and spacing highlight encoding and formatting anomalies. In operational deployments, w can be tuned via small validation studies or set adaptively based on feature–wise variance.

Role of the brand gate. The indicator I_y ensures that distances reflect *within–brand* consistency, rather than inter–brand heterogeneity. If cross–brand anomalies are of interest, one can introduce a softened gate (e.g., a penalty instead of hard zeroing) or a hierarchical scheme (brand $\rightarrow$ model $\rightarrow$ series).

When to stop scanning. The cut–off at the first false positive $K^\star$ provides a conservative measure of how many errors can be surfaced with high precision. In practice, a fixed budget (e.g., top 100 pairs) may be more appropriate; the same reporting (per–error occurrence counts) applies.

Comparison to local methods. While the all–pairs ranking offers a global baseline, its quadratic cost motivates the nearest–neighbors method, which replaces global comparisons with local consistency checks. The two approaches are complementary: all–pairs excels at surfacing globally isolated outliers; nearest–neighbors is better suited to nonuniform densities and scales more favorably.

Threats to Validity

The evaluation relies on synthetically injected errors whose distribution may differ from operational defects. Moreover, the brand gate presupposes correct brand labels. Finally, the specific feature map $F(\cdot)$ emphasizes string–level statistics; augmenting it with token n–grams or graph–based attributes may further increase sensitivity.

Reproducibility

All parameters needed to reproduce the experiment are listed in Table 2.3. The procedure is deterministic given $F(\cdot)$, w, and the brand labels. The resulting ranked list and per–error occurrence counts are stable under repeated runs with the same configuration.

2.3 Discovering Errors in an inner-relationships

Sections 2.3 and 2.4 are primarily implementation-oriented. They document engineering choices—data ingestion and preprocessing steps, parameterisation details, system interfaces, and runtime considerations—that are necessary to build a robust, reproducible pipeline from the theoretical framework presented earlier. Their main function is to enable faithful re-implementation and reliable deployment in operational settings, rather than to advance new theoretical results.

From a scientific standpoint, the novelty and methodological contributions reside elsewhere (in the problem formulation, modelling assumptions, and analytical results). By contrast, Sections 2.3–2.4 focus on translating those ideas into practice under real-world constraints such as legacy system compatibility, data quality variability, performance targets, and maintainability. The design decisions recorded there (e.g., specific thresholds, batching schemes, logging granularity, or fallback heuristics) are grounded in pragmatics and may vary across organisations or technology stacks without altering the central claims of the work.

Including these sections is nonetheless important. First, they strengthen reproducibility by making explicit the versions, configurations, and resource assumptions under which the reported results were obtained. Second, they support technology transfer: practitioners can trace how theoretical components are orchestrated within an end-to-end solution, facilitating auditability and integration. Third, they surface limitations and trade-offs that would be invisible in a purely theoretical exposition, thereby clarifying the scope of applicability.

Readers primarily interested in conceptual or theoretical advances may skim Sections 2.3–2.4 for context. Practitioners and evaluators concerned with deployment readiness, robustness, and compliance will find them essential. Future work may formalise some of the implementation heuristics presented there (e.g., through ablation studies or complexity analyses), but such extensions are beyond the scientific remit of these sections.

2.3.1 Problem Statement: Structural Error Discovery in CMDB Data

Configuration Management Databases (CMDBs) encode heterogeneous assets and their inter–relationships. We model a CMDB as a labeled graph $G = (V, E, \mathcal{A})$ with nodes V (configuration items, CIs), edges $E \subseteq V \times \mathcal{R} \times V$ (typed relations $\mathcal{R}$), and attribute maps $\mathcal{A}$ assigning string, categorical, numerical, and temporal fields to nodes and edges. In operational settings, G is affected by data defects that impair analytics and service reliability. This section defines the problem of *discovering structural errors* in G—specifically: (i) **missing nodes or edges**, (ii) **violations of naming conventions** in identifiers and labels, and (iii) **incorrect inner–relationships** (edges that contradict the intended dependency structure).

Error categories.

1. **Missing elements:** Absent but required edges or nodes implied by design rules or by observed patterns. Let Σ denote a schema comprising allowed relation types and cardinality constraints (e.g., each application server must *depend_on* exactly one database). A node v violates completeness if it lacks required incident edges under Σ, or if a mandatory counterpart node type is not present in its neighborhood. Such violations indicate missing $e \in E$ or, when no suitable counterpart exists, a missing $v' \in V$.

2. **Naming–convention errors:** Attribute strings that should conform to class–specific patterns (regular expressions, dictionaries, or templates) but do not. Let $\mathcal{R}_c$ be the admissible pattern family for class c and $a_{\text{name}}(v)$ the identifier of v; a defect occurs when $a_{\text{name}}(v) \notin \mathcal{L}(\mathcal{R}_{c(v)})$ or when cross–field consistency is broken (e.g., vendor inferred from the identifier disagrees with the declared vendor).

3. **Incorrect inner–relationships:** Edges that violate Σ (domain/range, multiplicity), contradict operational evidence (e.g., observed communication), or create cyclic/forbidden motifs (e.g., *depends_on* cycles across tiers). Formally, an edge $e = (u, r, v)$ is defective if $(r, \text{dom}(r), \text{ran}(r)) \notin \Sigma$ or if it causes a constraint breach (cardinality, acyclicity, separation of concerns).

Inputs and constraints. The system receives (i) the current graph G, (ii) a (possibly partial) schema Σ and convention catalog $\{\mathcal{R}_c\}$, and (iii) optional operational signals (e.g., topology discovery, logs) that provide weak evidence for relationships. Ground–truth labels are scarce and obtained through costly manual inspection; therefore, the solution must operate *with limited supervision*, support batch and streaming updates, and respect an inspection budget B per cycle.

Outputs. The objective is to produce *ranked nominations* for expert verification across the three error categories, together with proposed corrective actions:

- *Add–edge / add–node proposals* (candidate endpoints and relation types) for suspected missing structure;
- *Rename / normalize suggestions* for identifiers that violate conventions, including candidate canonical forms;
- *Unlink / relink proposals* for edges that breach schema or create forbidden motifs, with compliant alternatives when inferable.

Nominations must include confidence scores to support prioritization under budget B.

Success criteria. Effectiveness is measured by Top–K precision/recall of nominated defects, reduction of schema and convention violations after remediation, and stability under incremental updates. Secondary criteria include interpretability of nominations (explicit rule or evidence trace), scalability on large $|V|, |E|$, and robustness to missing or noisy attributes.

Scope. This problem statement is method–agnostic: it specifies *what* must be detected and *how* success is evaluated, independent of any particular detection technique. Subsequent sections instantiate concrete detection and ranking procedures consistent with this specification.

2.3.2 Discovering Errors in a inner-relationships - Algorithm

Purpose. This subsection formalizes and details an algorithm that quantifies the extent to which the *Names* (identifier strings) field *predicts* other attributes present in a CMDB table. The working hypothesis is that naming conventions implicitly encode business–relevant

attributes (e.g., brand, environment, datacenter, owner), and that a high predictive fit indicates a strong cross–field dependency which can be exploited to (i) validate attributes and edges, (ii) discover naming–convention violations, and (iii) propose corrections for missing or incorrect structure in the CMDB graph. We instantiate the predictor with a *Complement Naive Bayes* (CNB) model trained on character n–gram features extracted from *Names*.

Inputs and outputs. Let a CMDB table be given with N rows (configuration items) and a designated string column `Names`. Let $\mathcal{C}$ denote the set of other candidate columns (attributes) to be tested for correlation with `Names`. The algorithm returns, for each $c \in \mathcal{C}$, (i) a *CNB fit score* S_c (cross–validated predictive performance of `Names`$\rightarrow c$), (ii) a baseline–adjusted effect size Δ_c, (iii) an optional significance assessment (permutation p–value with multiple–testing control), and (iv) if S_c (or Δ_c) exceeds a threshold, a *correlated* flag together with interpretable token–level evidence (top n–grams by class–conditional weight).

Notation and preprocessing. For each row $i \in \{1, \ldots, N\}$, let s_i be the `Names` string and $x_i = \phi(s_i) \in \mathbb{R}^d$ its sparse feature vector obtained via a character n–gram vectorizer with range $[n_{\min}, n_{\max}]$ and vocabulary size d (maximum features). The label for column c is $y_i^{(c)} \in \mathcal{Y}_c$. Categorical columns are encoded as class labels; high–cardinality tails may be grouped into `OTHER`. Continuous columns may be discretized into quantile bins if included. Missing labels are handled by excluding the corresponding rows for c (or by adding a dedicated `MISSING` category, depending on the use case). Unless specified otherwise, vectorization preserves digits and punctuation and applies a consistent casing policy.

Model and score. For a fixed column c, CNB is trained to estimate $p(y \mid x)$ for $y \in \mathcal{Y}_c$ using additive smoothing parameter $\alpha > 0$. To obtain a robust *fit score* S_c, we perform K–fold stratified cross–validation and average a class–imbalance–robust metric, e.g., macro–F_1 (default), balanced accuracy, or (for one–vs–rest reductions) macro–AUC. To contextualize S_c, we compute a *baseline* S_c^{base} using a majority–class classifier or via label permutation, and define an effect size

$$\Delta_c \;=\; S_c \;-\; S_c^{\text{base}}.$$

When significance is desired, we estimate a permutation p–value by comparing S_c to the null distribution obtained from B random relabelings (typically $B \in [100, 1000]$) and control the false discovery rate across all $c \in \mathcal{C}$ using Benjamini–Hochberg.

Decision rule and interpretability. A column c is flagged as *correlates with naming* if

$$\Delta_c \geq \tau_\Delta \quad \text{and/or} \quad S_c \geq \tau_S,$$

with optional requirement $q_c \leq \alpha_{\text{FDR}}$ when using p–values (where q_c is the BH–adjusted p–value). For interpretability, we extract per–class token weights from CNB to expose which n–grams in `Names` drive the association with c; these are valuable for policy and convention refinement.

Hyperparameters and recommended defaults.

- *n–gram range*: $[1, 4]$ or $[1, 6]$ for general identifiers; expand upper bound if longer patterns are known to be informative. In the algorithm listing it is represented as n_{min}, n_{max}

- *Vocabulary size* d: start at $d = 3200$; increase to 6400 for large, diverse namespaces (cf. Section 2.3.4).
- *CV folds* K: 5 or 10 (stratified).
- *Smoothing* α: 1.0 (Laplace) or 0.1; tune on a subset if needed.
- *Score* S_c: macro–F_1 (default) or balanced accuracy for severe imbalance; report baseline.
- *Thresholds*: τ_S in $[0.80, 0.95]$ depending on risk appetite; τ_Δ in $[0.10, 0.25]$ above baseline; $\alpha_{\mathrm{FDR}} = 0.05$ if significance filtering is used.

Algorithm listing. We elaborate the high–level pseudocode from Algorithm 1 into the following procedure.

Algorithm 1: CNB–based cross–field correlation discovery from `Names`

input : CMDB table with `Names`, candidate columns $\mathcal{C}$, vectorizer params $(n_{\min}, n_{\max}, d)$, CV folds K, smoothing α, thresholds (τ_S, τ_Δ), optional permutations B, FDR level α_{FDR}

output: For each $c \in \mathcal{C}$: fit score S_c, baseline S_c^{base}, effect Δ_c, (optional) p–value/ q–value, correlated flag, and top tokens per class

$X \leftarrow \mathrm{Vectorize}\big(\texttt{Names};\, n_{\min},\, n_{\max},\, d\big)$;
for $c \in \mathcal{C}$ **do**
  $\big(\mathbf{y}^{(c)},\, \mathcal{Y}_c\big) \leftarrow \mathrm{EncodeLabels}\big(\text{column } c\big)$;
  $\big(S_c,\, S_c^{\mathrm{base}}\big) \leftarrow \mathrm{CrossValidatedScores}\big(\mathrm{CNB}_\alpha,\, X,\, \mathbf{y}^{(c)},\, K\big)$;
  $\Delta_c \leftarrow S_c - S_c^{\mathrm{base}}$;
  if *permutation test enabled* **then**
    $p_c \leftarrow \mathrm{PermutationPvalue}\big(\mathrm{CNB}_\alpha,\, X,\, \mathbf{y}^{(c)},\, K,\, B\big)$;
if *permutation test enabled* **then**
  $\big(q_c\big)_{c\in\mathcal{C}} \leftarrow \mathrm{BenjaminiHochberg}\big(\,(p_c)_{c\in\mathcal{C}},\, \alpha_{\mathrm{FDR}}\,\big)$;
for $c \in \mathcal{C}$ **do**
  if $\big(\Delta_c \geq \tau_\Delta$ ***or*** $S_c \geq \tau_S\big)$ ***and*** $\big(q_c \leq \alpha_{\mathrm{FDR}}$ *if available*$\big)$ **then**
    $\mathsf{flag}[c] \leftarrow \texttt{true}$;
    $\mathsf{evidence}[c] \leftarrow \mathrm{TopClassTokens}\big(\mathrm{CNB}_\alpha\big)$;
return $\big\{\, \big(c,\, S_c,\, S_c^{\mathrm{base}},\, \Delta_c,\, p_c,\, q_c,\, \mathsf{flag}[c],\, \mathsf{evidence}[c]\big) \;\big|\; c \in \mathcal{C} \big\}$;

We estimate cross-field dependence from identifier text using character n-grams [11] and Complement Naive Bayes [59] implemented via scikit-learn [55], assessing significance with permutation tests [22] and controlling multiplicity by the Benjamini–Hochberg procedure [6].

Complexity and scalability. Let $\mathrm{nnz}(X)$ denote the number of nonzero features. Training CNB is $O(\mathrm{nnz}(X))$ per fold; scoring is linear in $\mathrm{nnz}(X)$. The total cost scales as $O(|\mathcal{C}| \cdot K \cdot \mathrm{nnz}(X))$ (plus permutations if enabled). Vectorization is the dominant one–time cost. The procedure parallelizes trivially across $c \in \mathcal{C}$ and across CV folds.

Interpretation and downstream use. Columns flagged as *correlates with naming* indicate that `Names` encodes substantial information about those attributes. This has three immediate consequences aligned with the structural–error problem statement: (i) *Policy*: token evidence reveals the de facto naming convention and supports standardization. (ii) *Validation*: for a given item, the CNB posterior induces an expected value for c against which the current entry can be checked (naming–convention violations). (iii) *Graph corrections*: when c is a relational attribute (e.g., brand), posteriors can propose *add–edge/unlink/relink* actions in the CMDB graph when attribute–edge inconsistencies are observed.

Robustness and safeguards. To mitigate spurious findings: (i) remove exact duplicates and trivial echoes (e.g., when `Names` is a concatenation of the target column); (ii) cap category cardinality or use target–grouping for rare labels; (iii) optionally include a *holdout period* split to assess temporal stability; (iv) prefer conservative thresholds and FDR control under multiple testing; (v) add syntactic guards to ensure inputs conform to the expected identifier manifold before scoring.

Variants. While CNB offers speed and strong baselines on sparse text–like features, alternatives (logistic regression with ℓ_2 regularization, linear SVMs, calibrated tree ensembles) may be substituted if higher accuracy or different calibration properties are required. The scoring and decision protocol remains unchanged.

Reproducibility log. Report $(n_{\min}, n_{\max}, d)$, tokenization rules (case, allowed character set), missing–data policy, label encodings, K, α, chosen metric, baselines, (τ_S, τ_Δ), and (if used) $(B, \alpha_{\mathrm{FDR}})$. Persist vectorizer and CNB parameters to enable identical re–runs and consistent downstream validation.

Summary. The proposed algorithm turns the `Names` field into a predictive instrument that quantitatively measures its information content with respect to other CMDB attributes. High fit scores identify attributes that are effectively *encoded* by naming conventions; these dependencies are then leveraged to detect violations, propose graph corrections, and prioritize expert inspections under budget constraints.

2.3.3 Transforming CMDB Graph for Machine Learning Algorithms

In modern IT and OT environments, the Configuration Management Database (CMDB) serves as the central repository for configuration items (CIs), relationships between them, and their metadata. The graph structure of a CMDB, which captures the relationships between these entities, plays a crucial role in providing insights into system dependencies, configurations, and operational status. However, for machine learning (ML) algorithms to process and derive valuable patterns from the CMDB data, it is necessary to transform the CMDB graph into a format that is suitable for model training and inference. In this section, we describe the necessary steps to transform the CMDB graph into a machine learning-friendly format, aligning with the algorithms discussed in this dissertation.

Understanding the CMDB Graph Structure

The CMDB graph consists of nodes and edges:

- **Nodes:** Represent the configuration items (CIs) such as servers, network devices, software, and services.
- **Edges:** Represent the relationships between these CIs, such as dependencies, network connections, and service relationships.

The raw CMDB graph may include diverse data types such as categorical (e.g., software version), numerical (e.g., resource utilization), and relational data (e.g., communication between services). For machine learning algorithms to process this graph, we need to convert the data into a format that captures both the structure and semantics of the graph.

Steps to Transform the CMDB Graph

The transformation process involves several steps:

1. Graph Representation The first step is to convert the CMDB graph into a suitable format that can be used for machine learning. Two common representations of graphs are:

- **Adjacency Matrix:** A square matrix where each element (i, j) indicates the presence or strength of an edge between nodes i and j. This representation is particularly useful for algorithms that operate on fixed-size data inputs.
- **Edge List:** A list of edges with associated weights or attributes. Each entry in the list contains two nodes and an edge weight, representing the relationship between them. This representation is often used for algorithms that handle sparse graphs.

For example, if we want to use graph neural networks (GNNs) or other graph-based ML algorithms, the adjacency matrix or edge list provides the necessary structure for capturing graph-based dependencies.

2. Feature Extraction For each node and edge in the graph, we need to extract relevant features that will be used as input for machine learning algorithms. The features should capture both the inherent properties of the nodes (CIs) and the relationships between them. Feature extraction can be divided into two categories:

- **Node Features:** Each node in the graph represents a CI, and its features can include:
  - *Categorical features:* CI type (e.g., server, router, service), CI status (e.g., active, inactive).
  - *Numerical features:* Resource utilization, uptime, number of dependent services.
  - *Temporal features:* Last update time, maintenance window, historical performance.
- **Edge Features:** Each edge represents a relationship between two CIs, and its features can include:
  - *Relationship type:* Dependency, communication, service linkage.
  - *Relationship strength:* Frequency of interaction, connection bandwidth, service load.

These features should be pre-processed to handle missing data (e.g., using imputation methods described in this dissertation) and standardized if necessary.

3. Handling Missing Data CMDBs often suffer from incomplete or inconsistent data, which can adversely affect machine learning model performance. The following approaches can be employed to handle missing data:

- **Missing Node Data:** For missing node attributes, imputation techniques such as KNN imputation, regression-based imputation, or the use of domain-specific knowledge can be employed to estimate the missing values.
- **Missing Edge Data:** Missing edges or relationships can be inferred using graph completion techniques, such as link prediction algorithms (e.g., Common Neighbors, Jaccard Similarity).

4. Data Normalization and Transformation To ensure that the machine learning model can efficiently process the extracted features, the following transformations can be applied:

- **Normalization:** Numerical features should be normalized or standardized (e.g., Min-Max scaling or Z-score normalization) to ensure that all features are on a similar scale, particularly when applying distance-based algorithms.
- **Encoding Categorical Features:** Categorical features such as CI types or service statuses can be transformed into numerical representations using techniques such as one-hot encoding or label encoding.
- **Dimensionality Reduction:** If the number of features is large, dimensionality reduction techniques such as Principal Component Analysis (PCA) or t-SNE can be applied to reduce the feature space while retaining essential information.

5. Graph-based Machine Learning Techniques Once the CMDB graph is transformed into a machine learning-compatible format, various algorithms can be applied, including:

- **Graph Neural Networks (GNNs):** GNNs operate directly on graph structures and learn node or edge embeddings based on the graph's topology and node/edge features. These methods are highly effective for capturing dependencies between CIs and predicting system behavior or detecting anomalies.
- **Clustering Algorithms:** Clustering techniques such as K-Means or DBSCAN can be applied to identify similar groups of CIs based on their attributes and relationships, helping in anomaly detection and classification.
- **Classification and Regression Models:** After transforming the graph data into feature vectors, traditional machine learning models such as Decision Trees, Random Forests, Support Vector Machines (SVM), and Neural Networks can be applied to predict outcomes such as failure likelihood, procurement requirements, or risk levels.

Conclusion

Transforming the CMDB graph for machine learning algorithms requires a series of preprocessing steps that ensure the graph's structure and node/edge attributes are compatible with ML models. Feature extraction, missing data handling, and normalization are essential

steps to prepare the data for efficient model training. Once the graph is appropriately transformed, it can be leveraged for a wide variety of machine learning tasks, such as anomaly detection, forecasting, and risk prediction, as described in the dissertation.

2.3.4 Experiment — Identifier–Driven Discovery of Structural Errors (VIN Case Study)

Objective and link to the problem statement. We evaluate whether the intrinsic structure of Vehicle Identification Numbers (VINs) can be leveraged to *discover structural errors* in a CMDB: (i) violations of naming conventions in CI identifiers, (ii) *missing* edges that should connect a vehicle node to its manufacturer (brand) node, and (iii) *incorrect inner–relationships* (e.g., a vehicle linked to a wrong brand). The central hypothesis is that VINs encode brand–specific substrings; thus, a robust identifier–to–brand model provides an independent source of evidence to validate node attributes and incident edges.

Data and graph context. We use the *US Cars* dataset (VIN, brand). In the CMDB graph G, a vehicle CI v has: (a) an identifier attribute $a_{\text{vin}}(v)$; (b) a declared brand attribute $a_{\text{brand}}(v)$; and (c) a (possibly missing or incorrect) relationship $e = (v, \texttt{has\_brand}, b)$ to a brand node b. Our procedure produces, for each vehicle, a probabilistic prediction $\hat{b}$ from its VIN and then checks *consistency* between $\hat{b}$ and both the attribute $a_{\text{brand}}(v)$ and the edge e.

Method overview. We model VIN strings with character n–grams and train a Complement Naive Bayes (CNB) classifier to estimate posteriors $p(y \mid s)$ over brands $y \in \mathcal{Y}$ for a VIN s. Two feature–extraction hyperparameters govern the representation: *(i)* the n–gram range (e.g., 1–4, 1–6, 2–7, 3–8), and *(ii)* the maximum number of retained tokens (vocabulary size). The sparse bag–of–n–grams matrix feeds CNB with additive smoothing (default α). Training and inference are linear in the number of nonzero features.

From predictions to structural–error nominations. For each vehicle node v with VIN s:

1. **Identifier check (naming convention).** Let $\hat{b}(s) = \arg\max_y p(y \mid s)$ and $p_{\max}(s) = \max_y p(y \mid s)$. A high–confidence mismatch between $\hat{b}(s)$ and the declared $a_{\text{brand}}(v)$ signals a naming–convention defect in either the identifier or the attribute. We nominate v if $p_{\max}(s) \geq \tau_{\text{id}}$ and $\hat{b}(s) \neq a_{\text{brand}}(v)$.

2. **Missing edge detection.** If $\hat{b}(s) = a_{\text{brand}}(v)$ with sufficient confidence and G lacks an edge $(v, \texttt{has\_brand}, b)$ to the corresponding brand node b, we nominate an *add–edge* correction.

3. **Incorrect edge detection.** If G contains $(v, \texttt{has\_brand}, b')$ with $b' \neq \hat{b}(s)$ and $p_{\max}(s) \geq \tau_{\text{edge}}$, we nominate an *unlink/relink* from b' to $\hat{b}(s)$.

Confidence thresholds τ_{id} and τ_{edge} are tuned to the inspection budget and desired precision. For triage we rank nominations by $p_{\max}(s)$ (descending) and, where useful, by margin $p_{\max}(s) - p_{2\text{nd}}(s)$.

Table 2.5: Identifier model accuracy (VIN→brand) as a function of n–gram range and vocabulary size. High accuracy at moderate token budgets supports reliable structural checks.

Tokens	1–4	1–6	2–5	2–7	3–6	3–8
50	0.88	0.88	0.85	0.84	0.54	0.54
100	0.89	0.89	0.88	0.88	0.72	0.72
200	0.90	0.90	0.90	0.90	0.85	0.83
400	0.93	0.93	0.93	0.93	0.89	0.89
800	0.93	0.93	0.93	0.93	0.92	0.91
1600	0.95	0.95	0.94	0.94	0.94	0.94
3200	0.96	0.96	0.96	0.96	0.96	0.95
6400	0.97	0.97	0.97	0.97	0.97	0.97

Table 2.6: Posterior probabilities (in %) from the CNB identifier model for selected inputs (rows). These posteriors drive naming–convention checks and edge corrections in the CMDB.

input string	aud	bmw	bui	cad	che	dod	for	gmc	hon	hyu	jee	kia	lex	mer.	nis
`1f`	0	0	0	0	0	0	94	0	0	0	0	0	0	0	0
`test`	2	2	2	2	29	1	45	2	2	2	2	2	2	2	1
`jn1by1ar`	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
`t`	6	6	6	6	6	6	12	6	6	6	6	6	6	6	6
`zakopane`	4	4	8	4	16	25	1	4	4	4	4	4	4	4	9
`weather`	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0

Experimental protocol and parameters. We systematically vary the n–gram range and the vocabulary size while fixing the classifier (CNB). Accuracy (held–out) under these configurations is reported in Table 2.5 to demonstrate that the identifier model is sufficiently accurate to support structural inference. For CMDB use, we additionally record: (i) the rate at which high–confidence mismatches produce valid naming–convention corrections; (ii) precision of add–edge proposals; and (iii) precision of unlink/relink proposals. In production, we deploy conservative thresholds to favor precision in nominations.

Illustrative outputs and CMDB actions. Table 2.6 shows posteriors (in %) for selected strings. A prefix `1f` strongly indicates Ford; thus, for a vehicle with $a_{\text{brand}} = \text{Ford}$ but *no* `has_brand` edge, we propose an *add–edge*. Conversely, if a vehicle is linked to a non–Ford brand, we propose *unlink/relink*. Arbitrary non–VIN strings (e.g., `weather`) highlight the need for syntactic guards (length/charset) before invoking the identifier model.

Outcomes and alignment with the problem statement. High identifier accuracy (Table 2.5) enables reliable nomination of: (a) naming–convention violations (high–confidence mismatches), (b) *missing* `has_brand` edges (confident match with absent edge), and (c) *incorrect* `has_brand` edges (confident mismatch with present edge). Thus, the VIN–based model functions as an automated, data–driven oracle to expose structural errors in the CMDB and to propose concrete corrective actions (rename, add–edge, relink), directly addressing the problem statement on structural error discovery.

2.4 Advantages of the Proposed Algorithms

The proposed algorithm has been designed with practical deployment constraints and enterprise IT governance principles in mind. Its architecture is purposefully lightweight, explainable, and secure—meeting the operational, security, and compliance requirements of organizations managing sensitive infrastructure, including IT and OT systems.

2.4.1 On-Premise Processing without External Dependencies

One of the most significant advantages of the algorithm is that it can be executed entirely *on-premise*. Unlike many modern data quality and anomaly detection tools, it does not require the transfer of sensitive configuration data to external servers or cloud-based environments. This is particularly important in industries with strict regulatory or cybersecurity policies (e.g., energy, finance, defense), where outbound data flow may be restricted by internal policy or legal frameworks.

The algorithm requires no internet connection or cloud computing services. As a result, organizations retain full control over their data and infrastructure, ensuring compliance with data sovereignty and security regulations such as GDPR or internal ISO 27001-aligned standards.

2.4.2 Security-Conscious Library Design

To further enhance security and transparency, the algorithm relies solely on standard components of the Python ecosystem—most notably, the `math` module and the widely adopted `pandas` library. No unverified third-party libraries, proprietary function packages, or machine learning frameworks that may obscure internal processes (e.g., TensorFlow or PyTorch) are required for basic operation.

This design choice not only minimizes the attack surface and mitigates supply chain risks, but also facilitates code audits and verifiability—critical in contexts where code transparency is essential. This is especially important when the algorithm is used in the management of critical assets or cyber-physical infrastructure.

2.4.3 Modularity and Continuous Improvement

The solution is architected as a set of loosely coupled, highly cohesive modules (e.g., data ingestion and validation, feature screening and selection, density estimation and scoring, decision logic, and monitoring/feedback). Each module exposes a stable interface contract (input/output schema, preconditions, postconditions) that decouples implementation details from system integration. This separation of concerns enables targeted refinement: a single step can be replaced or improved—such as substituting a new selector or estimator—without requiring disruptive changes to adjacent components. In practice, this lowers integration risk and shortens iteration cycles, allowing the team to improve *one step at a time*.

Modularity directly enhances testability and scientific rigor of engineering choices. Because modules are isolated behind explicit contracts, we can attach unit tests and module-level benchmarks, run ablation studies, and perform sensitivity analyses that quantify the marginal effect of a change in a single stage. Versioned interfaces permit side-by-side (A/B) comparisons under identical upstream conditions, while integration tests ensure end-to-end

consistency. This design supports reproducibility (via pinned versions and configuration manifests) and auditability (via per-module logs and artifacts), so that observed gains can be attributed to specific modifications rather than confounded by pipeline-wide variability.

Finally, modularity operationalizes continuous improvement. With monitoring hooks at module boundaries—covering data drift, concept drift, latency, and accuracy budgets—the system can trigger targeted retraining or hot-swaps of components, followed by canary releases and safe rollbacks if needed. This aligns naturally with CI/CD practices in MLOps, enabling frequent, incremental deployments that deliver measurable improvements while controlling operational risk. Over time, the architecture accumulates validated enhancements across modules, yielding a robust, evolvable system whose performance improves steadily without jeopardizing stability.

2.4.4 Minimal Computational Requirements

The proposed approach is computationally efficient and designed to operate on modest hardware. It does not require GPU acceleration, distributed clusters, or cloud-native platforms for effective performance. In typical use cases (e.g., monthly data scans or real-time form validation), the algorithm can be executed on a standard workstation or a mid-range server with several gigabytes of memory.

This low hardware requirement makes the algorithm highly deployable in small and medium-sized enterprises (SMEs), as well as in edge environments where resources are constrained. It also reduces total cost of ownership (TCO), which is a key consideration in infrastructure management.

2.4.5 Explainability of Results

In contrast to many black-box machine learning methods, the algorithm provides *interpretable outputs*. Features used in record evaluation are explicitly named (e.g., number of digits in a string, presence of a known pattern), and their contribution to the final score can be quantified using formally defined importance measures (e.g., $\widehat{\delta}^{(j)}$).

This level of transparency supports informed decision-making, as data quality specialists and system administrators can understand and verify why specific records are flagged. Explainability also facilitates the communication of model behavior to stakeholders who may not have technical backgrounds, thereby increasing trust and acceptance of automated suggestions.

2.4.6 Adaptive and Feedback-Driven Learning

The algorithm is inherently adaptive. It incorporates user feedback collected during the manual verification phase, using it to dynamically update its internal model. Over time, it improves its understanding of what constitutes a data quality error in a specific environment. In addition to feedback-driven updates, the algorithm accommodates both periodic (seasonal) and steady (trend-like) changes in the data-generating process: seasonality-aware baselines and detectors (e.g., moving-window normalization with EWMA/CUSUM tests) separate cyclical fluctuations from structural shifts, while calibrated online updates adjust class priors, decision thresholds, and model parameters as gradual trends emerge.

Unlike static rule-based systems, which require manual updates and rule engineering, the pro-

posed algorithm continuously evolves. This learning process makes it especially suitable for dynamic environments, where systems, naming conventions, or operational contexts change over time. Beyond mere reactivity, a forecasting component produces short- and medium-horizon predictions of key quantities (e.g., expected error prevalence, workload, or feature distribution shifts). These forecasts enable *predictive control*: the system selects operating points and allocates verification effort using a receding-horizon policy that anticipates forthcoming conditions, with safeguards such as canary evaluation and rollback criteria. As a result, the pipeline maintains high performance under both seasonal variability and gradual drift, and it proactively prepares for impending changes rather than responding only after performance degrades.

2.4.7 Knowledge transfer across similar databases.

The proposed architecture supports knowledge transfer between databases of similar nature (e.g., comparable schema/ontology and overlapping feature semantics). After a light-weight normalization step that maps source and target attributes via a schema alignment, we quantify cross-domain similarity by comparing marginal/conditional distributions (e.g., using divergence measures or task-specific distances) and proceed when the discrepancy is below a preset tolerance. In this setting, several artefacts are portable with minimal risk: (i) the feature screening results (the ranking by $\delta^{(j)}$ and the associated threshold $\tau(\delta)$), (ii) class-prior estimates and calibrated decision thresholds, and (iii) initial class-conditional density estimates $\widehat{f}_C^{(j)}, \widehat{f}_E^{(j)}$ or prototype sets for kNN. This warm start reduces cold-start delay, labeling effort, and computational cost, while preserving safety: online monitors and drift tests remain active, triggering incremental adaptation or rollback if the target distribution departs from the source. Consequently, prior knowledge accelerates deployment on new but related databases, and the system converges more rapidly to high accuracy than starting from scratch.

2.4.8 Extensibility to Other Domains

Although the algorithm was initially designed for CMDB (Configuration Management Database) validation, its architecture is sufficiently general to be applied to other structured inventories. For example, it can be extended to support:

- **Enterprise Asset Management (EAM)** databases, used in industrial and utility sectors,
- **Hardware or software inventory systems** not covered by ITIL-based processes,
- **Non-IT domains**, such as HR records, logistics registries, or biomedical datasets.

This broad applicability stems from its data-agnostic feature generation approach and its ability to derive pattern-based quality indicators directly from data—without relying on predefined domain-specific taxonomies.

2.4.9 Summary

The proposed algorithm is tailored to the dissertation's scope: detecting and correcting data–quality defects in enterprise CMDBs spanning IT and OT. Its modular and transparent design—grounded in formally justified criteria (e.g., correlation and mutual-information

screening, Rajski/Scott measures, kernel density estimation, kNN scoring, and δ-based thresholds)—enables stepwise improvement, rigorous auditing, and reproducible results. The pipeline integrates natively with CMDB maintenance and change-management workflows by leveraging operator feedback, adapting to both periodic and steady distribution shifts, and exposing forecasts of error prevalence and parts demand for capacity planning. These characteristics make the solution practical and scalable specifically for CMDB quality assurance and IT/OT asset governance, aligning directly with the thesis objectives rather than serving as a generic data-quality tool.

Chapter 3

Purchasing Strategy

The procurement of information technology (IT) resources in large enterprises, particularly within the energy sector, presents a complex and multidimensional challenge. Organizations must ensure the timely availability of critical resources such as computing power, storage capacity, and network infrastructure, while simultaneously adhering to financial constraints and regulatory requirements. The dynamic nature of the IT market, combined with uncertainties in demand forecasting and supply chain variability, necessitates advanced planning methodologies capable of balancing cost efficiency with operational reliability.

In this context, the central problem addressed in this work involves the development of a long-term procurement strategy under conditions of uncertainty. Historical data on resource utilization, together with insights from previous tender processes, serve as the basis for constructing a budget and designing a strategic purchasing plan. However, the scale and heterogeneity of resources, coupled with the interplay of numerous economic and organizational factors, render traditional optimization approaches insufficient. The objective, therefore, is not to compute an exact global optimum but to design an algorithmic framework that supports decision-making and adapts to evolving conditions.

Several practical aspects make this problem particularly challenging:

- **Resource lifespan:** IT assets such as servers, storage devices, and computational resources are subject to limited operational lifetimes and require phased decommissioning.
- **Procurement lead time:** The time from order placement to resource delivery is stochastic and influenced by global supply chain conditions.
- **Cost dependencies:** Procurement costs exhibit a nonlinear relationship with order volumes. For organizations governed by public procurement regulations, cost thresholds may impose additional procedural requirements.
- **Demand uncertainty:** Forecasting future demand involves significant uncertainty, which grows with the length of the planning horizon.
- **Demand dynamics:** Demand functions are often composite, combining long-term growth trends with short-term spikes driven by factors such as new projects, capital expansions, or market-driven initiatives.
- **Financial considerations:** Long-term planning must incorporate the time value of money, as future costs and savings are subject to discounting.

The combination of these factors results in a problem characterized by nonlinearity, probabilistic uncertainty, and high-dimensional constraints. Traditional deterministic models offer limited applicability under such conditions. This research draws upon principles from resource-constrained project scheduling and related optimization frameworks [10], while extending these approaches to accommodate the economic and operational realities of enterprise-scale IT procurement. The importance of this problem extends beyond IT; similar challenges arise in the acquisition of routine goods and services as categorized by the Kraljic Portfolio Matrix [54].

Given the inherent unpredictability of technological trends and global market conditions, a static, long-term procurement plan is unrealistic. Continuous adaptation is essential, as new information—ranging from technological innovations to geopolitical disruptions—can rapidly invalidate prior assumptions. To address this, the proposed approach emphasizes adaptivity in both the algorithmic framework and the organizational processes supporting procurement [56].

Two contrasting procurement strategies illustrate the strategic trade-offs involved. The first, a *"big order"* approach, consolidates demand into a single, large purchase, thereby leveraging economies of scale and mitigating short-term risks. The second, an *"as late as possible"* strategy, prioritizes frequent, smaller acquisitions to capitalize on potential price declines and defer commitment. While each strategy offers advantages under specific conditions, neither is universally optimal [63]. The solution developed in this work aims to identify intermediate strategies that balance these competing considerations.

The algorithm presented herein is designed for the optimization of IT storage procurement but can be generalized to other contexts, including computational resources, subscription-based services, and long-term service contracts. From a methodological perspective, the problem is broad and exhibits no guarantees of convexity, with nonlinear constraints and stochastic elements introduced by uncertain demand forecasts. Consequently, heuristic and metaheuristic approaches are explored to deliver practical, implementable solutions.

3.1 Problem Statement

The problem is finding the optimal purchasing strategy. Let:

- $\tau_0, \ldots, \tau_n$ – discrete decision times (e.g., each Friday) at which a purchase may be made;
- m_i – quantity purchased at decision time τ_i, $i \in \{1, \ldots, n\}$;
- $\mathbf{m} = [\, m_0, m_1, \ldots, m_n \,]$ – decision vector (purchasing strategy);
- $p(t, m)$ – unit price function (continuous time t and order volume m);
- $u(t)$ – demand function (continuous time t);
- $\xi(m)$ – transaction/process cost; due to public-procurement thresholds $\xi(\cdot)$ is nonlinear.

The purchasing strategy is represented by the decision vector $\mathbf{m}$ that must satisfy the problem constraints and minimize the optimisation criterion $\mathcal{Q}(\mathbf{m})$.

Given:

- historical utilization samples $\{(y_g, \bar{t}_g)\}$, $g \in \{1, \ldots, M\}$ (used to estimate u), where $\bar{t}$ denotes past (historical) time; here y_g are *utilization* observations;
- historical price/volume records $\{(p_j, q_j, \bar{t}_j)\}$, $j \in \{1, \ldots, P\}$;
- process cost ξ depends nonlinearly on anticipated transaction price $\bar{p}$: $\xi = \xi(\bar{p})$, due to threshold-induced workflow steps;
- delivery-time observations $\{(d_k, \bar{t}_k)\}$, $k \in \{1, \ldots, D\}$, where d_k is delivery time;
- given interest rate r;
- inventory at τ_0 and sunset dates of components (e.g., arrays contributing to total storage capacity) are known.

Constraints

- Resource availability must meet demand at all times.
- Each order yields availability for a finite lifetime L (assumed common across orders). The availability function attributable to the ith order is

$$s_i(t) = m_i \Big\{ \mathbf{1}(t - \tau_i) - \mathbf{1}(t - \tau_i - L) \Big\}. \tag{3.1}$$

  The boundary condition is

$$\sum_{i=1}^{n} s_i(t) \;\geq\; u(t)\,, \qquad \forall\, t \in [0, T]\,. \tag{3.2}$$

Optimization Criterion

$$\mathcal{Q}(\mathbf{m}) \;=\; \sum_{i=1}^{n} \Big(m_i\, p(\tau_i, m_i) + \xi(m_i) \Big) \;\longrightarrow\; \min_{\mathbf{m}}\,. \tag{3.3}$$

3.2 Algorithm

We have decision points $\{\tau_i\}_{i=1}^{n}$. The solution starts at τ_0. The idea is to find an optimal plan over horizon (3.1), execute the purchase at τ_0, advance to τ_1 to collect new data and update forecasts, and re-run the optimisation. Thus the plan is iteratively updated and the decision at τ_1 becomes ready for execution. Optimisation at each step is carried out using forecast models. The general form appears in the Appendix (Section 3.8.5), and a special case is given below.

The first step is to estimate price and demand functions. A related problem assumes a Wiener process with Poisson demand [7]. Here, we propose a heuristic based on a Genetic Algorithm to find an optimal purchasing plan. We analysed alternative methods such as Bellman dynamic programming [5], convex optimisation ([9], [44]), stochastic-gradient neural models (e.g., Adam [32]), ant colony search [14], tabu search ([20], [52]), and simulated annealing ([33]). Results are reported in [48].

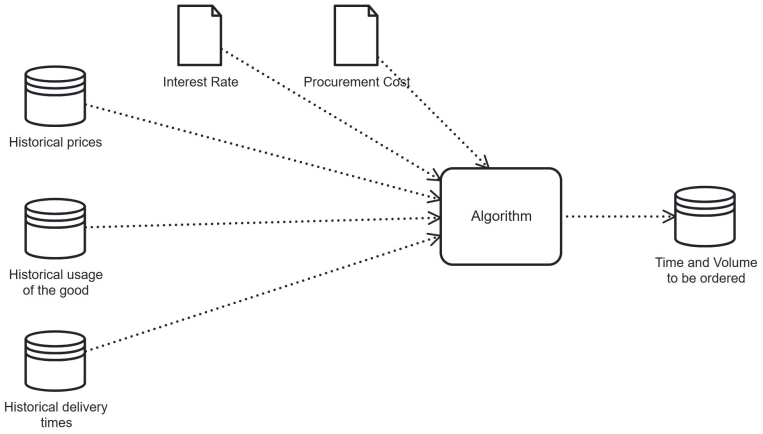


Figure 3.1: General idea of the Algorithm.

3.3 Market and Financial Input

The first step is to estimate price and demand functions.

3.3.1 Price Function Estimators

We use a model with the following key features:

- polynomial in time: trend captured by a 4th-degree polynomial in t (allows non-linear growth/decline/inflection);
- logarithmic in volume: price reductions vs. order volume follow a logarithmic relationship.

Examples are shown in Fig. 3.2 and Fig. 3.3. The estimator of the expected price is:

$$\hat{P}(q,t) = \hat{\beta}_0 + \hat{\beta}_1 t + \hat{\beta}_2 t^2 + \hat{\beta}_3 t^3 + \hat{\beta}_4 t^4 + \hat{\beta}_5 \log(q),$$

with least-squares estimate

$$\hat{\beta} = (X^\top X)^{-1} X^\top \mathbf{p},$$

where $\hat{\beta} = [\hat{\beta}_0, \ldots, \hat{\beta}_5]^\top$, the design matrix X is

$$X = \begin{bmatrix} 1 & \bar{t}_1 & \bar{t}_1^2 & \bar{t}_1^3 & \bar{t}_1^4 & \log(q_1) \\ 1 & \bar{t}_2 & \bar{t}_2^2 & \bar{t}_2^3 & \bar{t}_2^4 & \log(q_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \bar{t}_P & \bar{t}_P^2 & \bar{t}_P^3 & \bar{t}_P^4 & \log(q_P) \end{bmatrix},$$

and the vector of historical prices is

$$\mathbf{p} = (p_1, p_2, \ldots, p_P)^\top . \tag{3.4}$$

An approach to modelling discount policies is presented in [13].

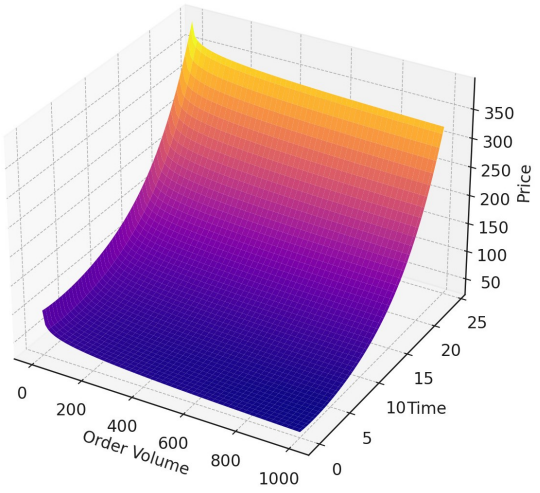


Figure 3.2: First example of price function.

Figure 3.3: Second example of price function.

3.3.2 Demand Function Estimator

To forecast demand, we use historical data and select a model that reflects the organisation's natural growth, then augment it with known upcoming events. The model combines three components (uppercase Y distinguishes these structural components from raw utilisation samples y_g):

- Y_1: baseline dynamics,
- Y_2: multiplicative event factors,
- Y_3: additive one-off events.

$$Y \;=\; Y_1 Y_2 + Y_3.$$

Here Y_2 and Y_3 are specified by domain input, while Y_1 is estimated from utilization samples $\{(y_g, \bar{t}_g)\}_{g=1}^{M}$.

We consider a simple AR(1) model for the utilisation series:

$$y_\ell = \lambda\, y_{\ell-1} + \epsilon_\ell,$$

with $\mathbb{E}[\epsilon_\ell] = 0$, $\mathrm{Var}(\epsilon_\ell) = \sigma^2$. The OLS estimator is

$$\hat{\lambda} = \frac{\sum_{\ell=2}^{M}(y_\ell - \bar{y})(y_{\ell-1} - \bar{y})}{\sum_{\ell=2}^{M}(y_{\ell-1} - \bar{y})^2},$$

where $\bar{y}$ is the sample mean. An alternative approach is discussed in [69].

3.3.3 Interest Rate: Effects on Pricing and Purchasing

The interest rate r captures the time value of money and the opportunity cost of capital. A future cash flow C_t has present value $\mathrm{PV} = C_t/(1+r)^t$. As r rises, PV falls: projected future savings (e.g., from bulk buys) contribute less to today's business case; when r is low, earlier and larger purchases become relatively more attractive. Suppliers pass financing costs into quotes and terms; higher r tends to increase prices for long lead times, while making early-payment discounts more valuable. Policies thus trade off timing, quantity, and contract structure to minimize discounted total cost under scenarios for r, prices, and demand.

3.4 Method selection

The first idea was exhaustive search. Even with coarse quantisation of m_i, this is computationally infeasible. Dynamic programming [5] is challenging due to path dependence and memory demands. Since convexity is not guaranteed, convex optimisation ([9], [44]) is not directly applicable. The objective is not even differentiable because of discontinuities in $\xi(\cdot)$; stochastic-gradient neural training (e.g., Adam [32]) is thus problematic. The situation is further complicated by nonlinear constraint (3.2) and many hyperparameters. The problem structure is not naturally graph-based, limiting ant-colony search [14]. Tabu search ([20], [52]) entails many tunables. Given these limitations, we compare a Genetic Algorithm ([2], [4]) and simulated annealing ([33]). Constraint satisfaction is enforced via a penalty-function coordination scheme [18]. Future work includes particle-swarm optimisation [65] and branch-and-bound [35]. The pre-implementation system will also include a forecaster for $u(t)$ with uncertainty bands [8].

3.5 Genetic Algorithm

3.5.1 Initial Population Seed

Let $\Delta \coloneqq \tau_i - \tau_{i-1} = \frac{T}{n}$ be the spacing between consecutive decision times (constant for all i). Assuming $L \geq \Delta$, purchase m_i affects availability during $\tau_i, \tau_{i+1}, \ldots, \tau_{i+h}$, where the horizon length is

$$h = \left\lfloor \frac{L}{\Delta} \right\rfloor. \tag{3.5}$$

The components m_i of each new individual (candidate solution) are generated successively ($i = 1, 2, \ldots, n$) from a uniform generator

$$m_i \sim \mathcal{U}[m_{i,\min},\, m_{i,\max}]\,, \tag{3.6}$$

where $m_{i,\min}$ ensures demand satisfaction up to the next decision time,

$$m_{i,\min} = u(\tau_{i+1}) - \sum_{j<i} s_j(\tau_{i+1}) \,, \tag{3.7}$$

whereas $m_{i,\max}$ guarantees sufficiency up to τ_{i+h},

$$m_{i,\max} = u(\tau_{i+h}) - \sum_{j<i} s_j(\tau_{i+h}) \,. \tag{3.8}$$

Our crossover approach tends to gravitate toward the mean, so we seed a substantial share of individuals near the extremes of feasibility: 20% as single large early orders, 20% as many small purchases, and the remaining 60% random. The initial seed is shown in Fig. 3.4.

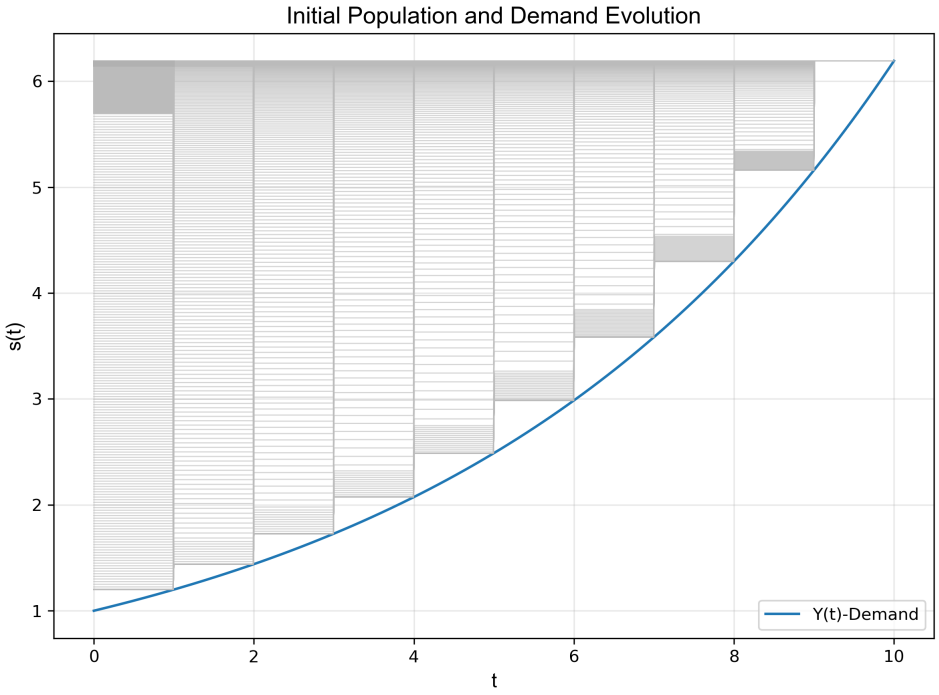


Figure 3.4: Initial population

[50] [48]

3.5.2 Parent Selection

We use fitness-proportionate (roulette-wheel) selection. Lower cost implies higher selection probability. For a population $\{\mathbf{m}^{(\kappa)}\}_{\kappa=1}^{N}$ the probability of selecting individual k is

$$P\big(\mathbf{m}^{(k)} \text{ is selected}\big) = \frac{q\big(\mathbf{m}^{(k)}\big)}{\sum_{\kappa=1}^{N} q\big(\mathbf{m}^{(\kappa)}\big)}, \tag{3.9}$$

with adaptation function, e.g.,

$$q\big(\mathbf{m}^{(\kappa)}\big) = Q_{\max} - Q\big(\mathbf{m}^{(\kappa)}\big) \,, \tag{3.10}$$

and

$$Q_{\max} := \max_{\kappa=1,\dots,N} Q\big(\mathbf{m}^{(\kappa)}\big). \tag{3.11}$$

3.5.3 Crossover

Children must satisfy constraints, i.e., represent valid strategies. The admissible region after exchanging genes between parents A and B is shown in Fig. 3.4. Our offspring generation randomly selects a switching time and splices strategies (Fig. 3.5), optionally in both directions (Fig. 3.6). An alternative crossover averages entries m_i coordinatewise.

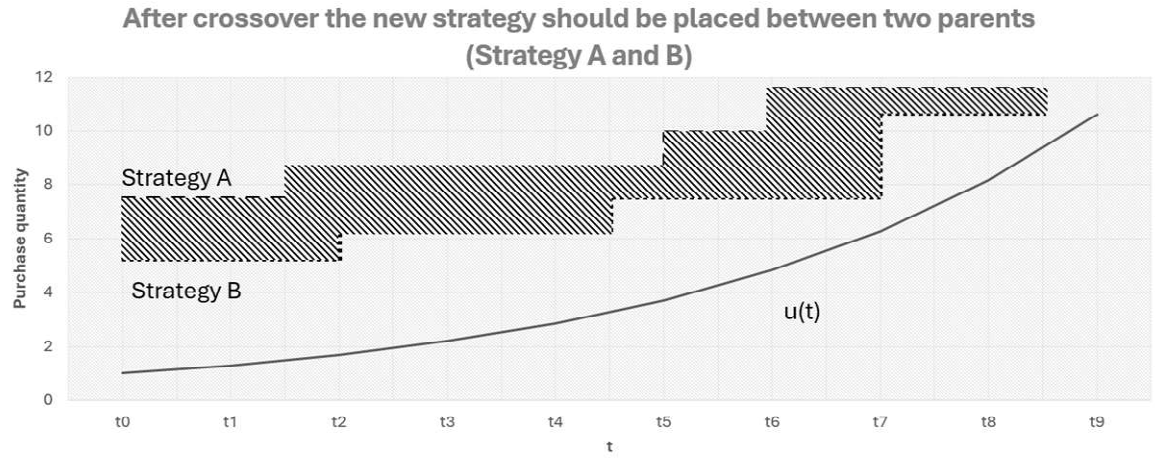


Figure 3.5: Valid crossover area

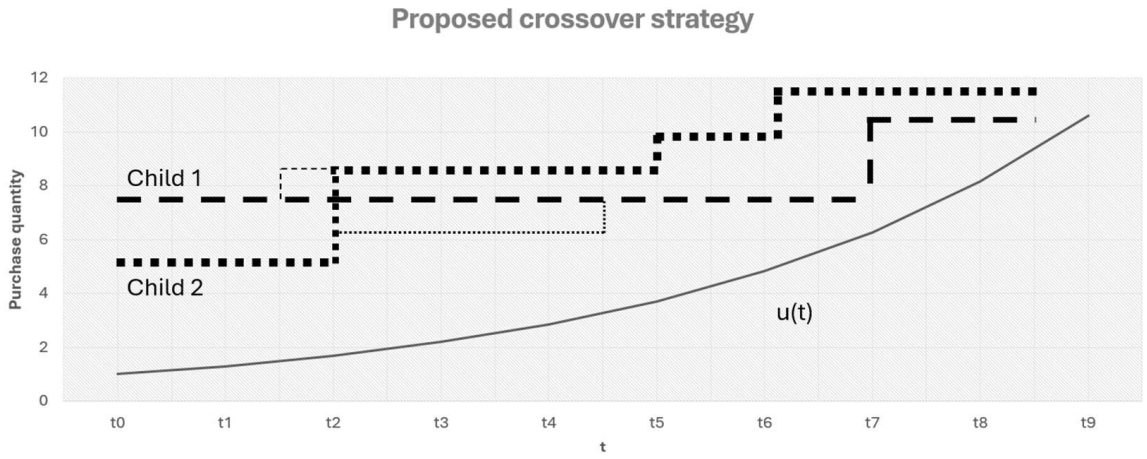


Figure 3.6: Crossover: children that are valid strategies

3.5.4 Mutation

At each generation, mutations relocate an order from a random decision time to an adjacent one (preceding or succeeding), maintaining validity.

3.5.5 Termination Condition

Stop when the mean cost over the population changes by less than ϵ for three consecutive generations (parameterised per resource scale).

3.6 Experiment Results

The algorithm was assessed in edge cases where the solutions are clearly defined. In the first test we checked algorithm under following parameters:

- constant product price over time;
- constant procurement expenses;
- significant discounts for bulk purchases.

In this case, the best plan is to make the largest single order. Solution was found by our simulation. Results are presented on Fig. 3.7. Then, we've considered another case:

- no discounts;
- decreasing product price over time;

The optimal tactic is to postpone the purchase for as long as possible. The behavior of the population within the genetic algorithm is depicted in Fig. 3.8.

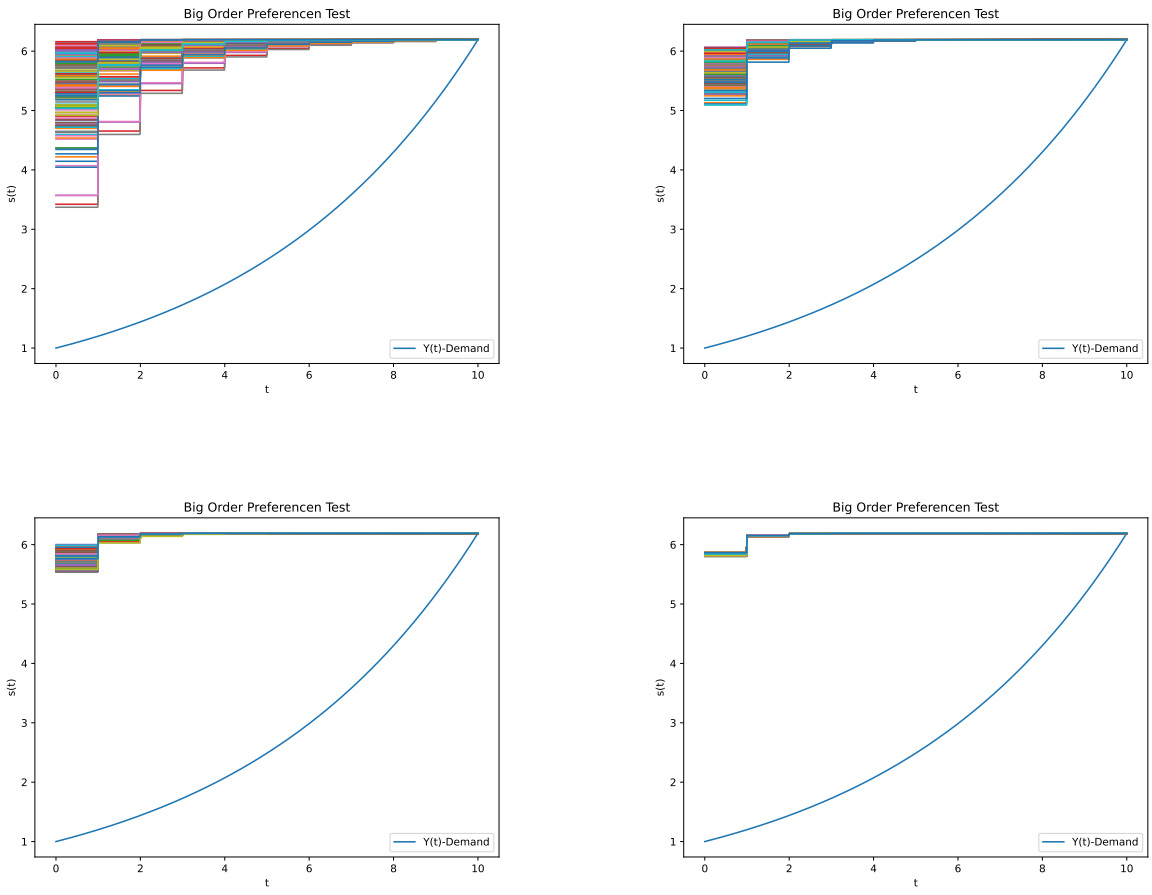


Figure 3.7: Preference for big orders. Generations 10,20,40,80.

The proposed solution is capable of identifying non-obvious outcomes when we introduce a limited lifespan for resources within the organization. An example of an interesting strategy discovered by the algorithm is presented in Fig.3.9. To evaluate the effectiveness of

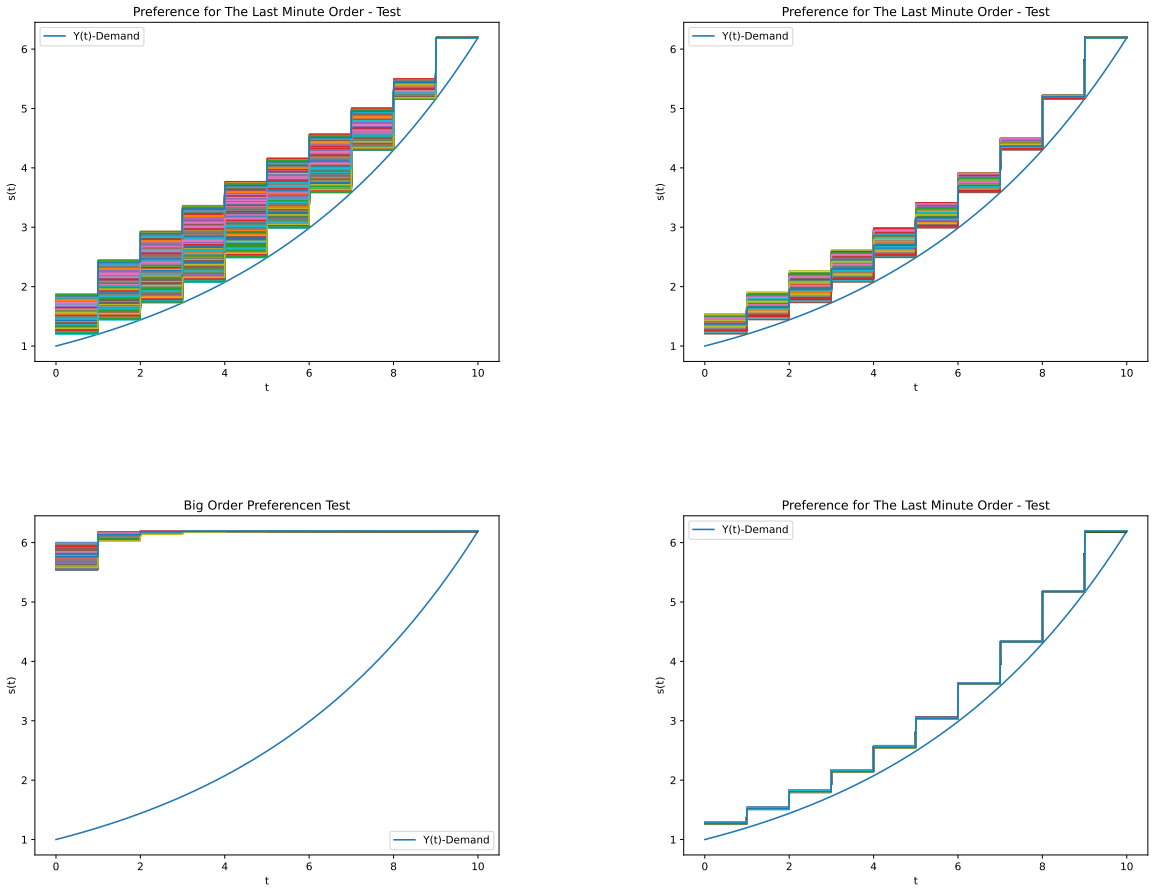


Figure 3.8: Preference for the last minute orders. Generations 10,20,40,80.

the solution, the strategy found by the algorithm was compared to several standard strategies—namely, the regular replenishment strategy and the edge strategies of making a large purchase or delaying the order as much as possible. The algorithm was able to find strategies that resulted in savings of 5 to 20 percent.

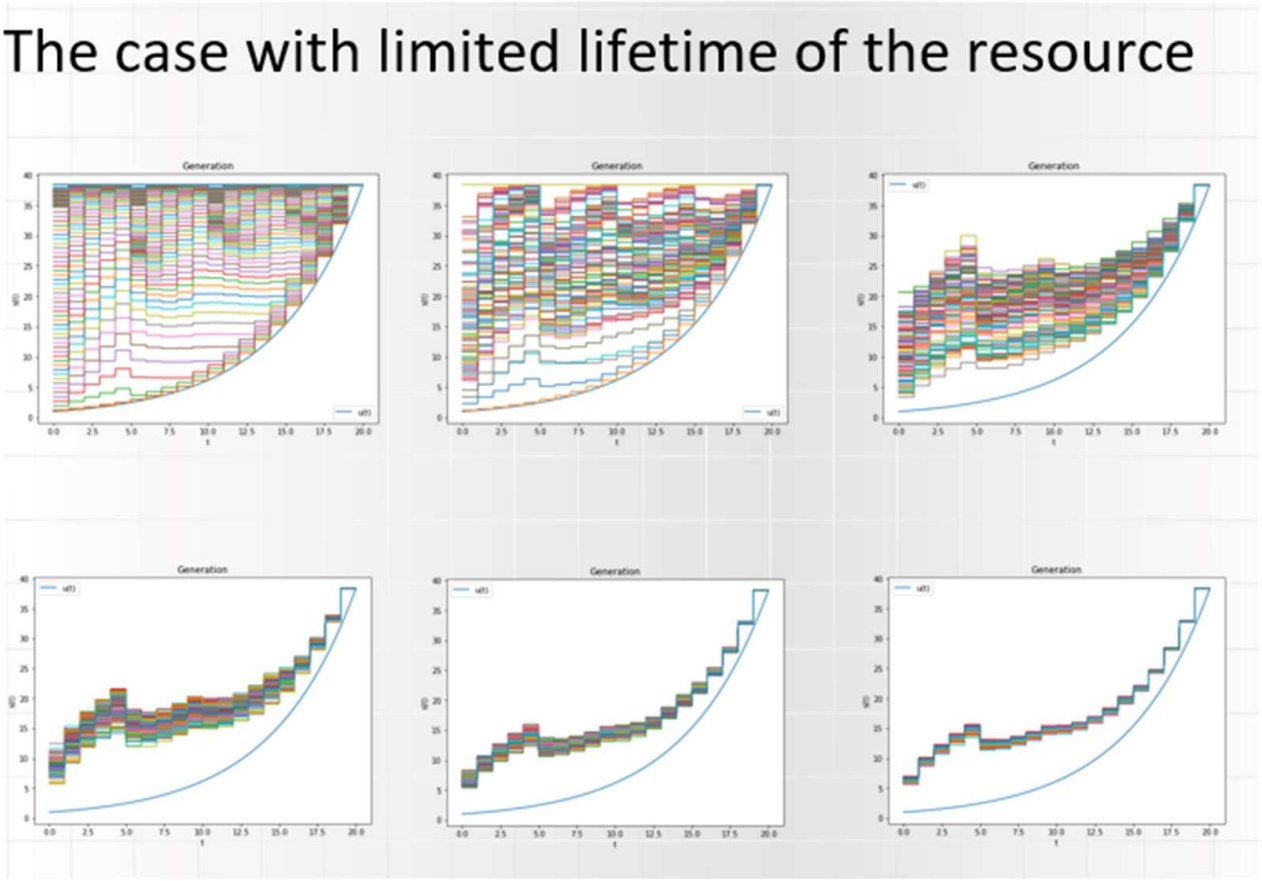


Figure 3.9: Exploring a genetic algorithm approach to address a problem involving restricted time availability of resources. (Generations: 1, 5, 10, 15, 20, 25).

3.7 Open problems

New ideas for crossover

An extension of this algorithm could involve the development of alternative methods for blending features. The region of valid solutions presented in the article can be expanded, which may mitigate the algorithm's tendency to average successive generations and search for niche solutions. However, it is crucial to ensure the correctness of strategies in subsequent generations.

Introduction of delay between order and delivery. Initial conditions for the problem.

The practical application of the algorithm may require the introduction of a time delay between the resource ordering time and its delivery, along with its implementation. The

time delay, along with the risks associated with delivery delays, could be an additional, very interesting extension of the presented problem. Another practical aspect is the possibility of introducing the initial state of existing resources along with information about the dates of their withdrawal from the infrastructure. The information about the initial state does not affect the algorithm but can offer much greater utility to teams planning purchases in the enterprise

Introduction of resource degradation in time

Some resources that we aim to provide in enterprises may experience a decline in performance, leading to degradation of their capacity and usability. In the algorithm, a binary degree of resource availability was adopted, but the algorithm can be extended to a solution that takes into account functions defining the manner of resource degradation. A good example could be photovoltaic panels, and the analysis of purchasing strategies may consider the decline in the efficiency of devices.

3.8 Implementation of Algorithm – Practical Aspects

The development of a long-term procurement strategy in the context of IT and OT assets involves multiple dynamic and interrelated factors. While theoretical models of optimization, forecasting, and risk minimization provide valuable insight, the practical implementation of such strategies in real-world conditions poses significant challenges. This section outlines the key operational and environmental considerations, focusing on historical price modeling, delivery time prediction, and the influence of evolving licensing and service delivery models.

3.8.1 Monitoring and Modeling of Price Trends

A fundamental component of procurement optimization is the ability to predict future prices based on historical trends. In the public sector and regulated industries, many purchasing decisions are informed by public procurement data, which offers a valuable source of historical pricing information [53]. However, several steps are necessary to operationalize this information effectively:

- **Data Acquisition:** Data must be systematically collected from public procurement portals, tender announcements, and procurement archives. Where available, APIs or web-scraping tools may be used to automate the collection process [68].
- **Data Normalization:** Due to variability in product naming, specification formatting, and supplier terminology, preprocessing is required to ensure comparability between records. This may include text normalization, categorization, and matching using natural language processing techniques [3].
- **Data Structuring and Storage:** Cleaned data should be stored in a structured database that supports query and filtering by parameters such as vendor, product type, contract date, delivery volume, and pricing conditions.
- **Function Estimation:** Historical prices can be modeled as a function of time and quantity. Polynomial regression or spline-based models may be applied to estimate the

price trend $p(t)$, possibly modified by order volume through logarithmic or piecewise modifiers [61].

3.8.2 Prediction of Delivery Times

A second critical variable in procurement planning is the expected delivery time, especially for critical infrastructure components. Delivery times are increasingly volatile due to global disruptions such as the COVID-19 pandemic, semiconductor shortages, and geopolitical tensions [29]. Forecasting such delays is extremely difficult and often lacks historical traceability.

From a methodological perspective, delivery time is treated as a stochastic variable. Forecasting models must incorporate conservative assumptions, Monte Carlo simulations, or robust planning buffers to accommodate volatility.

3.8.3 Dynamic Evolution of Licensing Models

One of the most disruptive trends in IT procurement is the shift from traditional perpetual licensing toward subscription-based and consumption-based billing. This has direct implications for procurement strategy design [19]:

- **Budgeting complexity increases**, as predictable capital expenditure (CAPEX) is replaced by variable operational expenditure (OPEX).
- **Renewal cycles shorten**, introducing more frequent procurement cycles and negotiations.
- **Licensing terms become more fragmented**, including metering, user-based pricing, and usage thresholds.

In many cases, the same software or hardware asset is offered with a range of licensing schemes, requiring detailed cost-benefit analysis and risk assessment of vendor lock-in.

3.8.4 Impact of Cloud Adoption

Although the focus of this dissertation remains on on-premise asset procurement, broader enterprise trends—particularly the migration of workloads to public and hybrid clouds—fundamentally shift the nature of what is being procured [21]. Instead of purchasing discrete hardware or software units, organizations increasingly acquire:

- Virtualized compute, storage, and network capacity,
- SaaS-based business applications,
- Managed services and DevOps automation platforms.

These changes transcend traditional procurement strategy models and necessitate the integration of IT financial management (ITFM), service-based accounting, and continuous vendor management frameworks.

3.8.5 Summary

In conclusion, the practical implementation of a procurement strategy for IT/OT infrastructure must address both data-driven and strategic challenges. Monitoring public procurement data enables the development of pricing models; however, delivery time forecasting remains uncertain due to external risks. Simultaneously, the rapid evolution of licensing and service delivery models requires procurement teams to stay informed and agile. While this dissertation focuses on algorithmic and optimization aspects, these real-world constraints and trends must inform any long-term strategy design.

Chapter 4

Implementation

4.1 Transition Management as a Prerequisite for Deployment

The effective deployment of the proposed solution was contingent upon the prior establishment of core transition processes within the organization, specifically **Change Management** and **Configuration Management**. These processes are fundamental to IT Service Management (ITSM) and must be operational before any data-driven automation or machine learning integration can take place.

In the case of Gaz-System, both Change Management and Configuration Management were fully implemented across the **Information Technology (IT)** and **Operational Technology (OT)** domains. Their implementation ensured that infrastructure changes are properly controlled, documented, and auditable, and that configuration items (CIs) are registered in a consistent and structured manner.

As part of this preparatory effort, previously fragmented repositories maintained independently by various technical teams were successfully **consolidated into a single, unified CMDB**. This integrated configuration management database now serves as the authoritative source of truth for infrastructure records and relationships.

The availability of this unified dataset was a necessary condition for applying the algorithms developed in this dissertation. It provided the consistent structure required for feature extraction, error detection, and confidence scoring. Moreover, the integrity of the configuration data ensured that outputs of the machine learning models could be interpreted reliably within the operational context and acted upon with trust by subject-matter experts.

4.2 Internal Implementation Approach

The deployment of the proposed solution is being carried out internally by in-house experts and software developers employed at Gaz-System. The implementation follows the organization's established procedures for the development of internal IT applications.

Specifically, the work is conducted in alignment with Gaz-System's formal **software development lifecycle (SDLC)** standards, which cover areas such as requirements analysis, design documentation, coding practices, version control, security auditing, and deployment governance. This ensures that the integration of machine learning components into the Configuration Management Database (CMDB) environment meets the company's technical,

operational, and compliance requirements.

The project is managed collaboratively between the internal development team and domain experts from IT and OT departments, ensuring that both functional correctness and domain-specific constraints are observed. The use of internal resources further supports knowledge retention within the organization and facilitates ongoing maintenance and evolution of the system without dependency on external vendors.

This implementation strategy aligns with Gaz-System's broader policy of technological sovereignty, data protection, and sustainable innovation in critical infrastructure systems.

4.3 Solution Architecture as a Service-Oriented Component

The architecture of the proposed system has been designed as a service-oriented solution, allowing external applications to interact with the machine learning engine through clearly defined interfaces. This architectural model enables seamless integration of data quality functionalities into existing workflows, without requiring tight coupling between the core algorithmic logic and specific client systems.

At its core, the solution exposes a RESTful API, which acts as an intermediary layer between the underlying machine learning components and external enterprise applications. These applications—such as CMDB tools, ITSM platforms, or custom asset management dashboards—can submit structured requests to the service, which then processes the input data, applies the relevant validation or anomaly detection algorithms, and returns the results for further action.

Typical elements of the architecture include:

- **API Gateway**: Responsible for managing access to the service, including authentication, rate limiting, logging, and routing of requests to the appropriate processing module.
- **Processing Engine**: Implements the core algorithmic logic, including feature extraction, vectorization, kernel-based density estimation, confidence scoring, and record ranking. The engine is stateless, modular, and designed for scalability within the limits of on-premise deployment.
- **Data Access Layer**: Provides secure read-only access to the CMDB and auxiliary metadata repositories. This layer abstracts direct database interaction and ensures compliance with data governance policies.
- **Feedback and Learning Module**: Stores validation outcomes (e.g., confirmed or rejected anomalies) and uses them to update internal model parameters or feature selection criteria. This enables incremental learning and adaptive refinement of the system over time.
- **Monitoring and Audit Module**: Tracks usage, records inference outcomes, and ensures that all transactions are logged for traceability and compliance with cybersecurity requirements.

This modular architecture promotes interoperability and facilitates future extensions of the system. For example, additional endpoints may be introduced to support bulk validation

tasks, trigger re-training cycles, or provide visual diagnostics to users. By abstracting the ML logic behind a well-documented interface, the solution aligns with modern principles of composability and reusability in enterprise software engineering.

Moreover, the architecture supports deployment behind corporate firewalls and within segmented network zones, in accordance with Gaz-System's security policies. All data exchanged between components is encrypted in transit, and sensitive inference outputs are exposed only to authenticated users with the appropriate privileges.

This service-oriented approach ensures that data quality functions can be accessed on demand by any authorized system, fostering broader adoption and operational integration across the organization.

4.4 Maintenance and Support Strategy

Following deployment, the long-term maintenance and operational support of the proposed solution will be carried out internally by the Application Maintenance Team at Gaz-System. This team is responsible for managing the lifecycle of internal IT applications, including issue tracking, updates, incident resolution, and user support.

The support model follows established internal procedures and includes:

- **Routine monitoring** of system health and inference quality metrics,
- **Incident response** for any deviations or failures in the service pipeline,
- **Version control and deployment** of updated models or algorithmic components,
- **Documentation and knowledge base** maintenance to support internal users and administrators.

The internalization of maintenance responsibilities ensures both technical continuity and alignment with Gaz-System's cybersecurity and data protection policies. It also allows for agile adaptation of the solution in response to evolving business requirements or changes in the configuration data landscape.

Compliance with Trustworthy AI Principles

In addition to satisfying internal operational standards, the system has been designed to conform to the core principles of **Trustworthy Artificial Intelligence**, as outlined in the European Commission's High-Level Expert Group on AI. In particular, the following dimensions are addressed:

- **Human agency and oversight** – All ML-based recommendations (e.g., suspicious records) are subject to human validation. The system does not autonomously modify CMDB entries.
- **Technical robustness and safety** – The architecture is fault-tolerant, modular, and operates in a controlled environment without external dependencies. Data is processed deterministically, and fallback mechanisms are in place.

- **Privacy and data governance** – The solution processes data on-premise only, adheres to the principle of least privilege, and maintains full auditability of every inference.
- **Transparency** – Model logic is interpretable, with documented feature selection and scoring mechanisms. All suggestions can be traced back to statistically defined indicators.
- **Accountability and auditability** – All interactions are logged, and justification for algorithmic decisions is preserved. This supports both operational debugging and external compliance reviews.
- **Societal and environmental well-being** – By improving data quality without increasing manual workload, the system contributes to operational efficiency while minimizing unnecessary resource consumption.

Collectively, these features ensure that the proposed system is not only technically effective, but also aligned with ethical and regulatory expectations for responsible AI use in critical infrastructure environments.

Appendix A

Trustworthy AI - Self Assessment

In its Communication of 25 April 2018 and 7 December 2018, the European Commission set out its vision for artificial intelligence (AI), which supports "ethical, secure and cutting-edge AI made in Europe".5 Three pillars underpin the Commission's vision: (i) increasing public and private investments in AI to boost its uptake, (ii) preparing for socio-economic changes, and (iii) ensuring an appropriate ethical and legal framework to strengthen European values. To support the implementation of this vision, the Commission established the High-Level Expert Group on Artificial Intelligence (AI HLEG), an independent group mandated with the drafting of two deliverables:

- AI Ethics Guidelines,
- Policy and Investment Recommendations.

The following assessment of the solution is carried out in the areas indicated by the EU recommendation for artificial intelligence systems. The assessment is addressed to the Management Board of the company where the solution is being implemented, the IT Division and Human Capital Management leadership, as well as to the Trade Unions.

1. Human Agency and Oversight

The AI system under evaluation is developed to enhance the quality of data serving as a foundational element for various security-related processes within the organization. The system plays a strictly supportive role, where its core function is to recommend potential corrections or enhancements to the data repository (such as asset metadata, firmware versions, and location attributes). Crucially, all AI-generated outputs are reviewed by designated human experts prior to implementation. This reflects a clear "human-in-the-loop" model, which ensures that human autonomy and decision-making remain central.

The operational environment is structured such that no automated actions are carried out by the system without prior human approval. This design choice is intentional to maintain full human control, allowing for the detection of errors or misjudgments in AI recommendations. The users — typically IT administrators or data stewards — are well-informed about the scope and behavior of the AI system, and receive contextual explanations for every recommendation. The system does not override human judgment and is integrated in a way that enhances the capacity of professionals to make informed and timely decisions, rather than replacing them.

2. Technical Robustness and Safety

The system demonstrates robustness by operating in a tightly controlled environment, where each data correction is reviewed manually. This ensures that under both normal and unexpected circumstances, the system cannot introduce harmful modifications or fail silently. From a design perspective, the model incorporates deterministic algorithms — primarily statistical methods — which are inherently stable and interpretable.

The AI is not exposed to open networks or dynamic inputs that could trigger adversarial behavior. All data inputs originate from known, authenticated, and audited sources. This minimizes the likelihood of adversarial inputs or poisoned data. Additionally, since the model does not operate autonomously, any erroneous output is intercepted during the manual review stage.

While fallback mechanisms such as automated rollbacks are not necessary due to the lack of autonomous execution, a comprehensive logging system ensures that all decisions and recommendations can be traced, explained, and reversed if necessary. The system operates within a high-security zone, which further insulates it from external disruptions or failures.

3. Privacy and Data Governance

The data used and processed by the AI system includes both technical metadata (e.g., device types, firmware versions, IP ranges) and selected categories of personal data relevant for asset ownership and access management. All processing is conducted exclusively on-premises, within a secured network zone that is compliant with internal data protection policies and aligned with general principles of data minimization and access control.

Although anonymization or pseudonymization is not applied — due to the operational need to maintain data linkability for updates and auditability — stringent technical and organizational measures are in place. These include multi-factor authentication (2FA), identity federation and authorization via Red Hat Single Sign-On (SSO), and segmented access rights. Access to data is limited to authorized personnel, and all access events are logged.

The system architecture was reviewed in light of GDPR requirements and organizational data governance frameworks. The explicit decision not to anonymize data is justified by the operational context, where traceability and accountability for specific assets and their history are critical for regulatory compliance and security audits.

4. Transparency

The AI system is designed with transparency as a core principle. It employs statistical algorithms that are both mathematically straightforward and fully documented. The underlying logic, including all decision rules and thresholds, is openly available to internal stakeholders, including end-users, technical reviewers, and compliance officers.

To maximize interpretability, the system provides detailed contextual information with each recommendation — for instance, highlighting the specific record it proposes to change, the rationale for the suggestion (e.g., mismatch between reported and expected firmware), and the confidence level of the recommendation. Because the system is built using well-known, well-supported Python libraries, the implementation inherits the transparency and auditability of these open-source tools.

The role and limitations of the AI system are clearly communicated across the organization through internal documentation, user training sessions, and operational guidelines. Staff are aware that the AI is not a decision-maker, but rather a support mechanism. The internal communication strategy reinforces responsible use and expectations, and invites feedback for continuous improvement.

5. Diversity, Non-Discrimination and Fairness

The system operates on technical datasets that do not contain sensitive personal attributes such as gender, ethnicity, age, or socioeconomic status. As such, it is inherently insulated from many of the risks typically associated with algorithmic bias or discrimination. The inputs used by the AI model include data such as device serial numbers, software versions, network topology metadata, and physical location identifiers — none of which correlate with protected demographic categories.

Given this scope, the system does not engage with vulnerable populations or marginalized groups. However, due diligence was exercised during the design phase to confirm that no unintended inferences or correlations could lead to indirect discrimination or unfair treatment. Stakeholder review processes and periodic audits are in place to monitor this.

Moreover, because humans validate all AI outputs, there exists a natural safeguard against any systemic errors that might otherwise go unchecked. Diversity in team composition, in terms of disciplinary background and technical specialization, contributes to robust review and validation of system behavior.

6. Societal and Environmental Well-being

The AI system has no direct societal impact, as it functions solely as an internal support tool for maintaining the accuracy and integrity of an enterprise configuration management database (CMDB). Its influence is limited to technical domains such as IT asset management, security compliance, and procurement forecasting.

In terms of environmental impact, the AI system is highly efficient. The training process is optimized through aggressive feature filtering and selection techniques, reducing computational overhead and associated energy consumption. The model does not rely on high-capacity GPU training or large-scale data streaming. Instead, it runs on standard enterprise hardware within the organization's internal infrastructure, contributing to overall resource efficiency.

While broader environmental concerns are not the primary focus of this system, its low resource demands and limited footprint position it as a sustainable solution, particularly compared to larger-scale, data-hungry machine learning models.

7. Accountability

A clear accountability framework has been defined for the system's development, deployment, and ongoing operation. Technical decisions are documented within the software development lifecycle (SDLC) artifacts, and every recommendation made by the system is recorded, including the input context, the proposed correction, and the human decision taken.

Logging and audit trails ensure that all interactions with the system are traceable. This includes system-level operations, user decisions, and administrative overrides. These logs are stored in secure, immutable formats and are reviewed periodically by internal auditors.

Responsibility for the technical aspects of the system lies with the AI development team and platform maintainers. Legal accountability, including compliance with internal policies and external regulations, is assigned to the data governance and legal teams. This separation of roles ensures that expertise is applied appropriately and that there is clarity in escalation paths if issues arise.

Appendix B

Use of Generative Language Models in Dissertation Development

Recent advancements in generative language models, particularly those based on the Transformer architecture such as OpenAI's GPT (Generative Pretrained Transformer), have opened new possibilities in scientific writing and technical assistance. In the context of this dissertation, GPT models were employed as intelligent assistants in several capacities to enhance both the quality and productivity of the research process.

Natural Language Editing and Translation Support

GPT was used to assist in refining the clarity, fluency, and grammatical correctness of English-language sections of the dissertation. This included:

- Paraphrasing technical descriptions to improve academic tone,
- Ensuring terminological consistency across sections,
- Translating fragments written originally in Polish into formal academic English.

This allowed for a more efficient drafting process and reduced the time spent on stylistic revisions.

Technical Writing and LaTeX Structuring

The model was employed to help generate well-structured LaTeX code for various components of the dissertation, including:

- Mathematical formulations and equations,
- Algorithm descriptions and pseudocode formatting,
- Figure captions and table summaries,
- Structuring multi-level sections in a coherent academic style.

Clarification of Machine Learning Concepts

GPT provided on-demand explanations and reformulations of technical concepts relevant to this dissertation, including:

- Feature selection metrics such as mutual information, Rajski distance, and discriminative indices,
- Kernel Density Estimation (KDE) and divergence measures,
- Hyperparameter optimization and semi-supervised learning workflows.

Assistance in Simulation Design and Experimental Protocols

While all algorithmic ideas and implementations used in this dissertation were original and developed independently by the author, GPT was used to help describe simulation procedures, parameter justification, and performance metric interpretations in formal written form.

Limitations and Ethical Considerations

The use of GPT was limited strictly to non-creative, assistive functions. No scientific claims, proofs, or algorithmic innovations were generated by the model. The author retained full control and intellectual responsibility for all content. In line with academic integrity standards, this section acknowledges the auxiliary role that large language models played in improving the expression and documentation of the research.

Conclusion

GPT served as a valuable tool for scientific writing support, helping to bridge the gap between technical innovation and its effective communication. The integration of such tools, when used responsibly and transparently, can significantly enhance the quality and accessibility of academic research.

Appendix C

Basic Concepts from Algebra and Mathematical Statistics Used in This Dissertation

- Least Squares Method
- Correlation Method
- Rajski Distance
- Scott Method
- Mutual Information
- Nonparametric (kernel) estimation of the probability density function (recursive)
- Nearest neighbor method (k-nearest neighbors)
- Jaccard Entropy Measure/Index
- Neural Network

Least Squares Method

The *Least Squares Method* (LSM) is a foundational approach in regression analysis and parameter estimation. It minimizes the sum of the squares of the differences (residuals) between observed values and those predicted by a linear model. This technique assumes that the errors are independent and identically distributed with zero mean and constant variance. Formally, given a linear model

$$y = X\beta + \varepsilon,$$

where $y \in \mathbb{R}^n$ is the vector of observed values, $X \in \mathbb{R}^{n \times p}$ is the design matrix, $\beta \in \mathbb{R}^p$ is the vector of unknown parameters, and $\varepsilon \in \mathbb{R}^n$ is the error vector, the LSM solution is obtained by solving:

$$\hat{\beta} = \arg\min_{\beta} \|y - X\beta\|^2.$$

The analytical solution, under the assumption that $X^\top X$ is invertible, is given by:

$$\hat{\beta} = (X^\top X)^{-1} X^\top y.$$

This method is widely used in curve fitting, time series forecasting, and system identification problems.

Correlation Method

The *Correlation Method* evaluates the strength and direction of a linear relationship between two random variables. The most common metric is the Pearson correlation coefficient:

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y},$$

where $\operatorname{cov}(X,Y)$ is the covariance between X and Y, and σ_X, σ_Y are their standard deviations. Values of ρ close to 1 or -1 indicate a strong linear relationship, while values near 0 suggest no linear correlation. This method is instrumental in feature selection and understanding dependencies in multivariate data.

Rajski Distance

The *Rajski Distance* is an entropy-based measure of dissimilarity between two random variables or distributions. Unlike traditional metrics, it incorporates both mutual information and entropy, offering a nuanced view of statistical dependence. Given two discrete random variables X and Y, the Rajski distance is defined as:

$$D_{\text{Rajski}}(X,Y) = 2H(X,Y) - H(X) - H(Y),$$

where $H(X)$ and $H(Y)$ are the Shannon entropies, and $H(X,Y)$ is the joint entropy. The Rajski distance is always non-negative and equals zero if and only if X and Y are perfectly dependent. It is particularly useful in clustering and information-theoretic learning.

Scott Method

The *Scott Method* provides an empirical rule for selecting the bin width h when constructing histograms to estimate a probability density function (PDF). The bin width is given by:

$$h = 3.5 \cdot \sigma \cdot n^{-1/3},$$

where σ is the standard deviation of the sample and n is the sample size. This rule assumes that the underlying distribution is approximately normal and aims to minimize the integrated mean squared error (IMSE). Scott's rule is preferred for its simplicity and effectiveness, especially when the data are unimodal and symmetric.

Mutual Information

Mutual Information (MI) quantifies the amount of information shared between two random variables, revealing both linear and non-linear dependencies. For discrete random variables X and Y with joint distribution $p(x,y)$ and marginal distributions $p(x)$ and $p(y)$, the mutual information is defined as:

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right).$$

In continuous cases, integrals replace summations. Mutual information is symmetric, always non-negative, and equals zero if and only if X and Y are independent. It is widely used in feature selection, clustering, and information theory.

Nonparametric (Kernel) Estimation of the Probability Density Function (Recursive)

Kernel Density Estimation (KDE) is a nonparametric way to estimate the probability density function of a random variable. The recursive version allows online updating of the estimate as new data arrives. The general KDE formula is:

$$\hat{f}_n(x) = \frac{1}{nh_n} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h_n}\right),$$

where K is the kernel function (e.g., Gaussian, Epanechnikov), h_n is the bandwidth, and X_i are the data points. Recursive forms may use exponential weighting or adaptive bandwidths, making them suitable for time-varying distributions and streaming data.

Nearest Neighbor Method (k-Nearest Neighbors)

The *k-Nearest Neighbors* (k-NN) method is a simple, instance-based learning algorithm used for classification and regression. Given a query instance x, the algorithm:

1. Computes the distance between x and all training points.
2. Selects the k closest points.
3. Outputs the majority class (for classification) or average value (for regression).

Common distance metrics include Euclidean, Manhattan, and cosine similarity. The method is nonparametric and makes no assumptions about the data distribution, which makes it flexible but computationally expensive for large datasets.

Jaccard Entropy Measure/Index

The *Jaccard Entropy Measure* extends the classical Jaccard index, which quantifies similarity between two sets A and B:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

To incorporate uncertainty and distributional characteristics, the entropy-enhanced version is:

$$J_{\text{entropy}}(A, B) = J(A, B) \cdot \left(1 - \frac{H(A \cup B)}{\log |A \cup B|}\right),$$

where $H(A \cup B)$ is the entropy of the union. This index is valuable in comparing categorical or binary datasets with imbalanced distributions.

Neural Network

A *Neural Network* is a machine learning model inspired by the structure of the human brain, consisting of layers of interconnected neurons. A typical feedforward network comprises an input layer, one or more hidden layers, and an output layer. Each neuron performs a weighted sum of its inputs followed by a nonlinear activation function:

$$\hat{y} = f\left(\sum_{j=1}^{m} w_j^{(2)} \cdot \phi\left(\sum_{i=1}^{n} w_{ji}^{(1)} x_i + b_j^{(1)}\right) + b^{(2)}\right),$$

where ϕ is an activation function (e.g., ReLU, sigmoid), w are the weights, and b are the biases. Neural networks are trained using backpropagation and gradient descent to minimize a loss function. They are particularly powerful in modeling complex, nonlinear relationships and are widely used in fields such as computer vision, natural language processing, and time series forecasting.

Appendix D

Functions from ML Open Source Libraries used in This Disertation

D.1 Machine Learning Functions Utilized in the Implementation

The implementation phase of this dissertation employed selected functions from prominent open-source machine learning libraries, namely `scikit-learn` and `Keras/TensorFlow`. These functions were integrated into the data quality assurance and forecasting pipelines in support of feature engineering, model training, and performance evaluation. This section provides a brief overview of their theoretical basis and implementation role.

SKLEARN: `Vectorize`

The `vectorize` functionality, typically implemented through classes such as `CountVectorizer` or `TfidfVectorizer` in the `scikit-learn` library, is used to transform textual or categorical data into numerical vectors suitable for machine learning models.

In the context of CMDB record preprocessing, vectorization enables the encoding of textual features (e.g., hostnames, serial numbers) into high-dimensional sparse vectors that capture token frequency or importance. This transformation is a prerequisite for applying distance metrics, entropy-based analysis, or classification algorithms to nominal data. The choice between binary, count, or TF-IDF weighting schemes is based on the discriminative needs of the model and the interpretability of feature importance.

Mathematically, vectorization transforms a corpus $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ into a matrix $X \in \mathbb{R}^{n \times m}$, where each row corresponds to a document (e.g., a CMDB entry) and each column corresponds to a token or n-gram from the vocabulary.

SKLEARN: `train_test_split`

The `train_test_split` function in `scikit-learn` is a utility for randomly partitioning datasets into training and testing subsets. This function supports stratified sampling and reproducibility via random seed control.

It is defined as:

$$(X_{\text{train}}, X_{\text{test}}, y_{\text{train}}, y_{\text{test}}) = \texttt{train\_test\_split}(X, y, \text{test\_size} = \alpha, \text{random\_state} = s)$$

where $\alpha \in (0, 1)$ is the proportion of data reserved for testing and s is the random seed.

In this dissertation, `train_test_split` is used in the validation of supervised learning models trained on error-labeled CMDB records. This split ensures that model performance metrics (e.g., accuracy, precision, recall) are computed on unseen data, thereby reducing the risk of overfitting and enhancing generalizability.

KERAS: TensorFlow Backend

Keras is a high-level neural network API that operates atop lower-level machine learning frameworks, most notably TensorFlow. In this dissertation, Keras was employed for prototyping and deploying interpretable neural network models aimed at anomaly detection and confidence estimation within configuration records.

Key components include:

- **Model architecture definition** via the `Sequential` or `Functional` API.
- **Compilation** using optimizers (e.g., `Adam`, `SGD`) and loss functions (e.g., binary cross-entropy).
- **Training** with gradient-based optimization routines and support for callbacks (e.g., early stopping).

The use of Keras enabled the construction of lightweight, fully transparent neural classifiers with shallow architectures tailored to small and medium CMDB datasets. These models contributed to decision support in error nomination and record validation.

The TensorFlow backend provides the computational engine for tensor operations, automatic differentiation, and GPU acceleration, thus ensuring scalability and numerical stability during training.

Bibliography

[1] Ziawasch Abedjan, Xu Chu, Mourad Ouzzani, Nan Tang, Sattam Alsubaihin Thirumuruganathan, and Gerhard Weikum. Detecting data errors: Where are we and what needs to be done? *Proceedings of the VLDB Endowment*, 9(12):993–1004, 2016.

[2] D. Ashlock. *Evolutionary computation for modeling and optimization*, volume 571. Springer, 2006.

[3] Arash Asudeh, X. Zhang, and H. V. Jagadish. Data integration through deep learning. *IEEE Data Engineering Bulletin*, 42(3):66–78, 2019.

[4] R. J. Bauer. *Genetic algorithms and investment strategies*, volume 19. John Wiley & Sons, 1994.

[5] R. Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[6] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):289–300, 1995.

[7] Peter Berling and Zhixue Xie. Approximation algorithms for optimal purchase/inventory policy when purchase price and demand are stochastic. *OR spectrum*, 36:1077–1095, 2014.

[8] G. EP. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[9] S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[10] P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112(1):3–41, 1999.

[11] William B. Cavnar and John M. Trenkle. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, 1994.

[12] Fei Chiang and Renée J Miller. Discovering data quality rules. *Proceedings of the VLDB Endowment*, 1(1):1166–1177, 2008.

[13] Yves Crama, A Torres, et al. Optimal procurement decisions in the presence of total quantity discounts and alternative product recipes. *European Journal of Operational Research*, 159(2):364–378, 2004.

[14] M. Dorigo and T. Stützle. *Ant colony optimization: overview and recent advances.* Springer, 2019.

[15] P. Duda, L. Rutkowski, M. Jaworski, and D. Rutkowska. On the Parzen kernel-based probability density function learning procedures over time-varying streaming data with applications to pattern classification. *IEEE Transactions on Cybernetics*, 50(4):1683–1696, 2018.

[16] Martin J Eppler. Information quality problems and current approaches. *Managing Information Quality*, pages 15–49, 2003.

[17] Wenfei Fan, Shuai Ma, Nan Tang, and Wenyuan Yu. Interaction between record matching and data repairing. *Journal of Data and Information Quality (JDIQ)*, 4(4):1–38, 2014.

[18] W. Findeisen, F. N. Bailey, M. Brdyś, K. Malinowski, P. Tatjewski, and A. Wozniak. *Control and coordination in hierarchical systems.* John Wiley & Sons, 1980.

[19] John M. Gallaugher and David Wang. Software as a service and the new pricing paradigm. *MIS Quarterly Executive*, 17(3), 2018.

[20] F. Glover and M. Laguna. *Tabu search.* Springer, 1998.

[21] Raul Gonzalez, Krishna Chari, and Niharika Rana. Cloud computing and procurement: Modernizing the acquisition of it services. *Journal of Strategic Contracting and Negotiation*, 4(1-2):90–105, 2018.

[22] Philip Good. *Permutation Tests: A Practical Guide to Resampling Methods for Testing Hypotheses.* Springer, 2 edition, 2000.

[23] Włodzimierz Greblicki and Mirosław Pawlak. *Nonparametric system identification*, volume 1. Cambridge University Press Cambridge, 2008.

[24] L. Györfi, M. Kohler, A. Krzyżak, and H. Walk. *A Distribution-Free Theory of Nonparametric Regression.* Springer Verlag, New York, 2002.

[25] Wolfgang Härdle. *Applied nonparametric regression.* Number 19. Cambridge university press, 1990.

[26] Wolfgang Härdle, Marlene Müller, Stefan Sperlich, Axel Werwatz, et al. *Nonparametric and semiparametric models*, volume 1. Springer, 2004.

[27] Atefeh Heidari, Baharan Bahmani, and Ziawasch Abedjan. Holodetect: Few-shot learning for error detection. In *Proceedings of the 2019 International Conference on Information and Knowledge Management (CIKM)*, pages 1913–1922. ACM, 2019.

[28] Ihab F Ilyas and Theodoros Rekatsinas. Machine learning and data cleaning: Which serves the other? *ACM Journal of Data and Information Quality (JDIQ)*, 14(3):1–11, 2022.

[29] Dmitry Ivanov. Viable supply chain model: Integrating agility, resilience and sustainability perspectives—lessons from and thinking beyond the covid-19 pandemic. *Annals of Operations Research*, 312:1–21, 2020.

[30] Kalervo Järvelin and Jaana Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. *Information Processing & Management*, 39(1):285–302, 2003.

[31] Dan Jurafsky and James H. Martin. *Speech and Language Processing*. 3rd edition, 2024. Draft, Chapter 3 (Add-one/Laplace and Lidstone smoothing).

[32] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[33] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[34] Pierre-Simon Laplace. *Théorie Analytique des Probabilités*. Courcier, Paris, 1812.

[35] E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966.

[36] Lu Liu, Xu Chu, et al. Rapidash: Statistical error detection via dynamic constraint discovery. In *Proceedings of the 2024 International Conference on Management of Data (SIGMOD)*. ACM, 2024.

[37] Stuart E Madnick, Richard Y Wang, Yang W Lee, and Hongwei Zhu. Overview and framework for data and information quality research. *Journal of data and information quality (JDIQ)*, 1(1):1–22, 2009.

[38] Stuart E Madnick, Richard Y Wang, Yang W Lee, and Hongwei Zhu. Overview and framework for data and information quality research. *Journal of Data and Information Quality (JDIQ)*, 1(1):1–22, 2009.

[39] William G. Madow. On the limiting distributions of estimates based on samples from finite universes. *Annals of Mathematical Statistics*, 19(4):535–545, 1948.

[40] Mohsen Mahdavi and Ziawasch Abedjan. Raha: A configuration-free error detection system. In *Proceedings of the 2019 International Conference on Management of Data (SIGMOD)*, pages 865–882. ACM, 2019.

[41] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, 2008.

[42] Elias Masry. Probability density estimation from sampled data. *IEEE Transactions on Information Theory*, 29(5):696–709, 2003.

[43] George A. Miller. Note on the bias of information estimates. In Henry Quastler, editor, *Information Theory in Psychology: Problems and Methods, II-B*, pages 95–100. Free Press, Glencoe, IL, 1955.

[44] J. J. Moré. The Levenberg-Marquardt algorithm: implementation and theory. In *Numerical analysis: proceedings of the biennial Conference, Dundee, June 28–July 1, 1977*, pages 105–116. Springer, 2006.

[45] Grzegorz Mzyk and Szymon Niewiadomski. Entropic metrics in feature selection for neural network-based data error detection in cmdb. In *International Conference on Artificial Intelligence and Soft Computing*, page ACCPETED. Springer, 2025.

[46] Xiaolan Ni, Xu Chu, et al. Automatic data quality rule discovery. *Proceedings of the VLDB Endowment*, 16(5):1046–1058, 2023.

[47] Szymon Niewiadomski and Grzegorz Mzyk. Ml support for conformity checks in cmdb-like databases. In *International Conference on Artificial Intelligence and Soft Computing*, pages 366–376. Springer, 2023.

[48] Szymon Niewiadomski and Grzegorz Mzyk. Optimization of procurement strategy supported by simulated annealing and genetic algorithm. In *International Conference on Dependability of Computer Systems*, pages 196–205. Springer, 2024.

[49] Szymon Niewiadomski and Grzegorz Mzyk. Semi-automatic error detection using kernel estimators and entropy-based metrics. *ACM Journal of Data and Information Quality*, 2024. Accepted, in press.

[50] Szymon Niewiadomski and Grzegorz Mzyk. Optimization of purchasing plans based on forecasted demand for resources. *International Journal of Electronics and Telecommunication*, 71(2):419–425, 2025.

[51] Suphakit Niwattanakul, Jatsada Singthongchai, Ekkachai Naenudorn, and Supachanun Wanapu. Using of Jaccard coefficient for keywords similarity. In *Proceedings of the international multiconference of engineers and computer scientists*, volume 1, pages 380–384, 2013.

[52] E. Nowicki and Cz. Smutnicki. A fast taboo search algorithm for the job shop problem. *Management Science*, 42(6):797–813, 1996.

[53] Open Contracting Partnership. Open procurement and tendering data standards. `https://www.open-contracting.org/`, 2019. Accessed July 2025.

[54] Sidhartha S Padhi, Stephan M Wagner, and Vijay Aggarwal. Positioning of commodities using the kraljic portfolio matrix. *Journal of Purchasing and Supply Management*, 18(1):1–8, 2012.

[55] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[56] Mike Prince, J Cole Smith, and Joseph Geunes. A three-stage procurement optimization problem under uncertainty. *Naval Research Logistics (NRL)*, 60(5):395–412, 2013.

[57] C. Rajski. A metric space of discrete probability distributions. *Information and Control*, 4(4):371–377, 1961.

[58] Raimundo Real and Juan M Vargas. The probabilistic basis of Jaccard's index of similarity. *Systematic biology*, 45(3):380–385, 1996.

[59] Jason D Rennie, Lawrence Shih, Jaime Teevan, and David R Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 616–623, 2003.

[60] David W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization.* John Wiley & Sons, New York, 1992.

[61] Edward A. Silver, David F. Pyke, and Rein Peterson Thomas. *Inventory and Production Management in Supply Chains*. CRC Press, 4th edition, 2016.

[62] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London, 1986.

[63] RP Sundarraj and Srinivas Talluri. A multi-period optimization model for the procurement of component-based enterprise information technologies. *European Journal of Operational Research*, 146(2):339–351, 2003.

[64] Matt P Wand and M Chris Jones. *Kernel smoothing*. CRC press, 1994.

[65] D. Wang, D. Tan, and L. Liu. Particle swarm optimization algorithm: an overview. *Soft Computing*, 22:387–408, 2018.

[66] Richard Y Wang and Diane M Strong. Beyond accuracy: What data quality means to data consumers. *Journal of management information systems*, 12(4):5–33, 1996.

[67] Charles T Wolverton and Terry J Wagner. Recursive estimates of probability densities. *IEEE Transactions on Systems Science and Cybernetics*, 5(3):246–247, 1969.

[68] Peng Xu, Jingrui Chen, and Rui Zheng. Automated analysis of public procurement data: Challenges and opportunities. In *Proceedings of the International Conference on e-Procurement Analytics*, 2021.

[69] Lu Zhen. Optimization models for production and procurement decisions under uncertainty. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(3):370–383, 2015.