Metadata-Centric Root-Cause Modeling of Machine-Learning Performance Degradation Using Synthetic Data Corruption

Master's Thesis of

Rony Munnik (2674125)

submitted in fulfillment of the requirements for the Master of Science degree in Business Analytics at the

Vrije Universiteit Amsterdam

Supervisor & First Reader: Prof. Dr. Sandjai Bhulai
External Supervisor (Capgemini): Mr. Drs. Hilke Schuurmans
Second Reader: Dr. Kevin S. Luck

Utrecht, 17.05.2026

Abstract

Machine-learning pipelines increasingly rely on automated data monitoring, yet most existing approaches remain data-centric: they can detect that incoming data has changed, but not which data-quality issues are responsible when model performance degrades. This thesis investigates how batch-level metadata can be used to learn a supervised model for root-cause diagnosis of machine-learning performance degradation.

To address the scarcity of labeled real-world incidents, a synthetic data generator is used to create controlled batch-level degradation scenarios, including both single-cause and multi-cause cases, under varying perturbation rates and degradation intensities. A Quality Feature Builder then transforms each batch into a fixed-dimensional representation based on metadata profiles, outlier signals, and drift indicators, after which supervised multi-label models are trained to predict the degradation types present in a batch.

The empirical evaluation shows that both learned models substantially outperform a prior baseline based on label prevalence. On the test set, the FFNN achieves a macro-F2 of 0.9331 and XGBoost one-vs-rest achieves 0.9496, compared with 0.4988 for the baseline. Detection is strongest for Missingness and Distributional shift/Drift, while Noise/Measurement error and Outliers/Extreme values remain more difficult, especially at low perturbation rates and under higher co-occurrence.

Overall, the findings support the viability of metadata-driven root-cause diagnosis in simulated monitoring settings, while further research is needed to validate the approach on real operational incidents before practical deployment.

Acknowledgments

I would like to express my sincere gratitude to Capgemini for providing me with the opportunity and the resources to carry out this research. Conducting this thesis in a professional environment, with access to relevant expertise and support, has been a valuable and enriching experience. I would also like to thank everyone at Capgemini for their openness, approachability, and willingness to help. The supportive atmosphere and the readiness of colleagues to think along with me made this process especially pleasant and motivating.

I would like to personally thank Maaike Willekes for helping to place this research in its business context. Her assistance by reviewing parts of this thesis in light of her experience and expertise was greatly appreciated and helped to enhance its practical relevance.

In addition, I would also like to thank Dr. Kevin S. Luck for serving as second reader and for the time and attention devoted to reviewing this thesis.

I am particularly grateful to Hilke Schuurmans for her guidance, critical perspective, and continuous support throughout this project. Her constructive feedback, practical insights, and commitment have been of great value to the development of this research. I especially appreciated her ability to look beyond the academic objectives of this thesis and to keep in mind the practical implementation of the model. Her way of challenging my ideas in a constructive and encouraging manner, combined with her willingness to share her knowledge and expertise, has greatly contributed to both the direction and the quality of this thesis. Beyond her guidance on the content of this research, I also greatly appreciated the way she helped me become familiar with the organization, made me feel welcome within the team, and connected me with the right people whenever needed.

Finally, I would like to sincerely thank Prof. Dr. Sandjai Bhulai for his academic guidance and supervision. His thoughtful feedback, critical questions, and broad scientific perspective have helped to strengthen the rigor and quality of this thesis. I especially appreciated his quick responses and his ability to bring a strong balance between academic support and practical applicability. His clarity in feedback, calm and reliable supervision, and encouragement of independent thinking helped me to approach this research with both rigor and confidence. His guidance has been invaluable throughout the research process and has played an important role in bringing this thesis to completion.

This thesis would not have been possible without their support and feedback, and I am truly grateful to all who contributed to it.

Contents

1 Introduction

1.1 Capgemini Insights & Data

Capgemini is one of the world's leading consulting and technology firms, operating across more than 50 countries. It describes itself as a global leader in partnering with companies to transform and manage their business by harnessing the power of technology, which reflects its strong focus on digital transformation and IT services [18]. Spread over multiple business units, professionals focus on strategic consulting, cloud and infrastructure services, software engineering, and the development and deployment of data-driven solutions. The current workforce exceeds 420.000 globally and generated revenue of €22.5 billion in 2025 [20]. As a result of its scale and achievements, Capgemini consistently ranks among the leading global service providers in annual industry evaluations published by Gartner, Forrester, IDC, and Everest Group [19].

Data-driven solutions are crafted primarily by or in collaboration with the Insights & Data (I&D) team. This team, consisting of specialists in data engineering, data analytics, machine-learning, AI, business analytics and data (quality) management, supports organizations in utilizing data to its fullest potential. Among others, this is achieved through designing and deploying data pipelines.

After deployment, managing these pipelines is the main task of operators. This framework, often referred to as data governance, focuses on managing data assets, establishing policies, and ensuring compliance of the pipeline. In practice, data management and data monitoring operate as complementary disciplines within a broader data governance framework: data management establishes the processes and standards governing data pipelines, while data monitoring operationalizes these policies through continuous oversight to ensure pipeline integrity, observability, and predictability in production.

As organizations increasingly pivot to real-time analytical systems and machine-learning models, these governance practices extend further into data quality management and data quality monitoring. Ensuring that data remains accurate, complete, consistent, and stable over time is critical for maintaining trust in data-driven decision-making.

Therefore, data-oriented teams increasingly explore mechanisms that safeguard data quality, which directly provides the foundation for this thesis. Consequently, this thesis has been carried out in collaboration with the Insights & Data team of Capgemini Netherlands.

1.2 Background: ML and Data Quality

Over the past decade, organizations have witnessed an unprecedented expansion in the volume, velocity, and variety of data generated across digital ecosystems. Advances in large-scale data collection, the introduction of large language models (LLMs), the adoption of artificial intelligence (AI), the digitization of systems, the proliferation of IoT devices, and user-generated content have created new data sources. Where for a long time, data deemed non-crucial/relevant was omitted due to limited computational resources and storage capacity, significant improvements in affordability and accessibility of high-performance GPUs, data storage and scalable cloud architectures

have made it possible to train and deploy machine learning (ML) models on these underexplored data environments.

As organizations are able to process these large and diverse datasets, they are motivated to extract as much valuable information as possible from these assets to gain a competitive advantage and improve their decision-making. This strategic imperative has intensified the demand for data-driven automation, predictive analytics, and intelligent systems capable of uncovering patterns that were previously inaccessible. Whereas data analysis was traditionally performed on static, periodically updated datasets, organizations now require timely insights to get ahead of competitors. This has resulted in a shift towards continuous data-processing and machine learning utilization, defined as pipelines. These pipelines integrate data ingestion, transformation, validation, and model inference into automated, end-to-end workflows that operate with minimal human intervention. Consequently, ML pipelines are now widely deployed across domains including finance, healthcare, logistics, retail, and public services.

However, this shift from isolated model development to real-time deployment introduces a fundamental challenge: the performance and reliability of ML systems are inseparably tied to the quality of the data that feeds them. Research conducted in data and information quality [7, 49, 77] demonstrates that real-world data is frequently imperfect, containing inconsistencies and inaccuracies. These imperfections become especially problematic when models continuously consume new incoming data whose properties may diverge significantly from the data the initial model is developed on. An empirical study supports this inherent dependency between data quality and the performance of ML models, where various dimensions of data quality, such as noise, missingness, redundancy, and inconsistency, can systematically affect predictive accuracy and model stability [12, 50].

In relation to the importance of data quality assessment within these pipelines, research has been conducted on the data lineage. The data lineage is the process of tracking how data propagates through a pipeline. For example, the origin of the data and what transformations were applied. The research around this topic shows that quality issues often originate within data engineering and data quality management, long before the data reaches the ML model [31].

1.3 Data Quality Assessment in Practice

To address these quality issues, several frameworks of data quality assessment and quality awareness have been described to systematically evaluate across all stages of the data lineage [10, 54]. The practical implementation of these frameworks in modern machine-learning pipelines is centered around a variety of tools, checks, and monitoring practices. Organizations currently utilize various data quality tools to assess and monitor their data quality within their pipelines.

A large portion of these tools is centered around rule-based or expectation-driven validation frameworks. Tools such as TensorFlow Data Validation, built in collaboration with Google for their TFX platform [22], Deequ, built in collaboration with Amazon Web Services [66], and direct commercial solutions like Informatica [39], Collibra [26] and Soda [69] allow for constraints on schema structure, value distributions, missingness, and other quality dimensions. These tools enforce data constraints and checks that ensure new batches adhere to predefined value ranges, thereby preventing abnormal data from entering ML pipelines.

Since these approaches are static, there have emerged multiple data observability platforms, which allow for real-time automated data monitoring, such as Monte Carlo [21], Soda Cloud [69], and Informatica Data Management Cloud [39]. These tools extend beyond static rules by incorporating algorithms such as anomaly and drift detection. Rather than relying solely on predefined constraints, they aim to detect unexpected behavior of the incoming data stream by leveraging statistical monitoring and automated alerting mechanisms.

Across these tools and platforms, datasets are monitored primarily through metadata: structured information describing the properties of a data asset, including schema-level structure and system-derived indicators such as sparsity, cardinality, and value distributions [5]. Metadata creates an overview of distributional characteristics, missingness patterns, outlier behavior and drift indicators without requiring inspection of the full underlying data. This abstraction is essential for monitoring in production environments, where real-time inspection of incoming data streams is impractical for human interpretation. As a result, metadata has become a common and scalable representation for dataset characteristics in production environments [22, 28, 66].

1.4 Limitations of existing practice

Although these tools are valuable for monitoring and assessing incoming data, their assessments are data-centric rather than model-centric. By comparing incoming data with historical data characteristics, they evaluate whether this incoming data differs substantially. However, such deviation-based monitoring introduces several limitations in production ML pipelines.

First, these approaches are primarily sensitive to abrupt or threshold-exceeding changes, meaning that slow, incremental drift over time may remain undetected even as the underlying data gradually diverges from the initial training distribution. Second, these systems flag deviations independently of their impact on model behavior. A feature distribution may shift substantially without affecting predictive performance, while small but influential changes may go unnoticed.

Recent research attempts to bridge this gap by estimating expected model performance on incoming data [64] or by evaluating models under controlled, corrupted conditions [65]. While such methods offer valuable insights into potential performance degradation, they still do not provide interpretable insights between specific data quality issues and their influence on model behavior. Studies focused on anomaly detection and data characterization likewise identify unusual patterns [56, 78] in model performance, but do not connect model under-performance to particular root causes in the underlying data.

Taken together, these limitations highlight a central challenge: current tools are effective at detecting that data has changed (monitoring, data-centric) or if the model degrades in accuracy (expected model performance, model-centric), but they do not determine what in the underlying data has changed to cause this degradation in performance. This becomes especially problematic when model performance degrades unexpectedly, yet the available monitoring signals show no incoming data-quality issues.

1.5 Gap and Objective

The limitations identified in the previous section reveal a fundamental gap between current monitoring practices and the needs of production ML systems. Data-centric approaches focus pri-

marily on detecting deviations in incoming data, such as shifts in distributions, schema changes, or anomalies, but provide no mechanism for determining whether these deviations meaningfully affect model performance. In contrast, model-centric approaches such as the score-prediction model [64] can signal that a performance drop is likely or occurring, but they do not reveal which specific changes in the data have caused this degradation. As a result, practitioners can observe a degradation in predictive accuracy without the ability to identify its underlying root cause.

Bridging this gap requires moving beyond deviation-based monitoring and black-box performance estimation toward an approach that can attribute observed model degradation to concrete root causes in the underlying data.

However, this is unachievable by just combining existent data-centric and model-centric approaches for two main reasons. First, the relationship between changes in input data and downstream model performance is often non-linear and model-dependent. Seemingly minor shifts in marginal distributions, missingness patterns, or rare inconsistencies can trigger disproportionate performance drops once they interact with feature engineering, decision thresholds, or learned non-linear feature interactions (Section 2.2). Conversely, large distributional changes that are easily detectable by drift metrics may have little effect on predictive quality if they occur in low-importance regions of the feature space or in features that the model effectively ignores. In addition, checks can be passed successfully while the model's performance deteriorates. As a result, deviation of known distributions is not a reliable proxy for data-quality and purely data-centric approaches are insufficient.

Second, degradation is difficult to detect and diagnose in operational pipelines because the ground-truth labels are typically unavailable at the moment a problem emerges. The labels are often delayed, sparse, or entirely unavailable, and performance changes may only be identified much later. Even when performance monitoring is possible, identification remains challenging because multiple degradation mechanisms often co-occur within the same batch (Section 2.1).

An actionable root-cause approach must therefore operate at batch level, support multi-label explanations where multiple degradation types can co-exist, and explicitly model the link between data characteristics and performance-relevant behavior rather than treating all deviations as equally important. Because real degradation events are relatively rare and costly to identify, a practical way to obtain supervised training signals is to generate controlled synthetic perturbations. Synthetic degradations can be designed to reflect recurring technical defect types, vary perturbation strength and rate and co-occurrence patterns, and produce labeled examples that connect specific changes in data characteristics to measurable performance effects and their corresponding root causes.

To achieve this, this thesis extends the score-prediction model towards a supervised root-cause prediction model for performance degradation in ML pipelines building on concepts from score-prediction [64] and robustness analyses such as JENGA [65].

Concretely, the score-prediction model [64] serves as an external trigger in the online phase of the proposed model (Section 3.1), while this thesis contributes the diagnostic stage that follows. Motivated by the sensitivity analysis approach of JENGA [65], which demonstrates that controlled data corruptions can reveal model vulnerability to specific quality dimensions, this thesis adopts a similar synthetic corruption strategy.

The proposed approach adopts synthetic data degradations to construct labeled degradation scenarios. By learning the relationship between metadata features and specific degradation causes, the framework aims to identify the data quality factor(s) responsible for the observed degradation of the model. This results in a model-centric diagnostic framework for data-quality that provides insight into the root cause of the degradation in model performance.

1.6 Research Question & Contributions

To achieve the objective discussed above, this thesis investigates the following research question:

How can batch-level metadata be used to learn a supervised model that identifies and explains the root causes of machine-learning performance degradation in production pipelines, using synthetic data degradations?

To answer this research question, we introduce the following sub-questions:

- How can synthetic degradation be designed to generate realistic, controllable, and labeled single-cause and multi-cause data-quality incidents at batch level?
- Which metadata features (e.g., distributional statistics, missingness/outlier signals, drift measures) are most predictive for distinguishing degradation types, and how should metadata be aggregated to represent batch-level change?
- To what extent can supervised multi-label models predict the correct degradation type(s) for unseen batches, and how does performance vary between single-cause and multi-cause scenarios and across perturbation ranges?

To address these research questions, this thesis makes four primary contributions:

First, it introduces a monitoring framework that moves beyond simple performance estimation and is designed to support batch-level predictions around model degradation. This extension establishes the conceptual foundation for linking performance deviations to insights of the incoming data.

Second, the thesis introduces a (feature-importance–aware) synthetic degradation pipeline that generates controlled single- and multi-cause data quality perturbations. This pipeline enables the creation of labeled degradation scenarios that would be rare, costly, or difficult to observe in real operational environments.

Third, it proposes a metadata extraction pipeline capable of capturing dataset- and batch-level characteristics relevant to data quality drift and outlier behavior. The pipeline integrates multiple metadata dimensions, including distributional and profiling statistics, drift measures, and outlier signals, allowing the framework to learn relationships between metadata features and degradation outcomes.

Finally, the thesis designs and evaluates a supervised root-cause model that learns to identify the data quality factors most likely responsible for observed performance degradation. The model is assessed on its ability to distinguish between single- and multi-cause scenarios for synthetically created degradation scenarios.

Together, these contributions advance a model-centric diagnostic approach to data quality, complementing existing monitoring tools and supporting more transparent and reliable operation of machine-learning systems.

1.7 Outline

The remainder of this thesis is organized into five chapters. Chapter 2 presents an in-depth literature review, focusing on key concepts in data quality, metadata, and machine-learning pipeline behavior. In addition, the literature review will position the thesis within current research on data quality assessment, model monitoring, and performance degradation. Chapter 3 outlines the methodological framework, detailing the construction of metadata representations, the generation of degradation scenarios, and the design of the supervised root-cause explanation model. Chapter 4 reports the empirical results, evaluating the model's ability to identify degradation causes under both single- and multi-cause conditions. Thereafter, chapter 5 discusses the evaluation, limitations, and directions for future research. Finally, chapter 6 concludes the thesis.

2 Literature Review

The methodological framework introduced in Chapter 1 draws upon concepts and techniques from several research areas, including data and information quality and its influence on machine-learning performance, metadata-based validation, anomaly and drift detection, and synthetic corruption. Although the different literature addresses complementary components of data-quality management in machine-learning pipelines, it has largely evolved in parallel. Data-centric approaches primarily focus on detecting deviations in incoming data, whereas model-centric approaches aim to estimate performance under distributional change. Yet, as discussed in the introduction, existing work rarely provides an integrated mechanism for attributing observed performance degradation to specific, interpretable data-quality causes once such an event has occurred, particularly when multiple degradations co-occur.

Figure 1: High-level schematic of the design of a machine-learning pipeline.

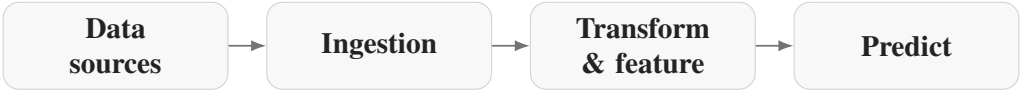


Figure 2: High-level schematic of a real-time operating machine-learning pipeline.

This chapter examines the bodies of literature that inform the proposed model. For guidance purposes, a high-level overview of a machine-learning pipeline (Figure 1 during initial development, Figure 2 while in operation) is presented throughout the literature review to show where methods and approaches generally reside within these pipelines. This literature review begins by reviewing core definitions, dimensions, and taxonomies of data-quality, followed by evidence on how quality degradation affects machine-learning performance. It then examines metadata as a scalable representation of dataset characteristics and reviews how data-validation and observability tools operate in modern machine-learning pipelines. Subsequent sections discuss anomaly and drift detection methods, as well as research on synthetic data corruptions used to study model robustness. Finally, the chapter surveys model families suitable for metadata-driven root-cause prediction.

2.1 Data-Quality: Definitions, Concepts, and Dimensions

A precise understanding of data and information quality is essential for assessing how data degradations affect machine-learning pipelines. Before discussing technical mechanisms for detection, monitoring, or diagnosis of data-quality, it is important to identify what is meant by data-quality. Accordingly, the literature first explores which dimensions are commonly observed and how these dimensions are formalized. To further understand how these dimensions are present in datasets, studies are consulted that identify at which structural levels data quality problems arise.

Early work in data-quality frequently equated quality with accuracy, adopting a system-centric perspective in which data was considered high quality if it represented the real-world values they

were based on [32]. Wang and Strong challenge this narrow view by arguing that data-quality cannot be defined solely from the perspective of system- or database-engineers. Instead, data-quality must be understood from the perspective of data consumers, namely those who rely on data to perform tasks or make decisions [77].

Based on empirical studies of data consumers, Wang and Strong propose a hierarchical framework that organizes data-quality dimensions into four broad categories (Figure 3). Intrinsic data-quality captures properties that data should possess in their own right, such as accuracy, objectivity, believability, and reputation, reflecting whether data is fundamentally correct and trustworthy. Contextual data-quality emphasizes that quality must be evaluated relative to a specific task or use case, including dimensions such as relevance, timeliness, completeness, and appropriate amount of data. Representational data-quality concerns how data is structured and presented, focusing on clarity, consistency of representation, and interpretability, all of which affect how easily users can understand and correctly interpret data. Accessibility data-quality addresses whether data can be obtained and used when needed, encompassing availability, ease of access, and access security.

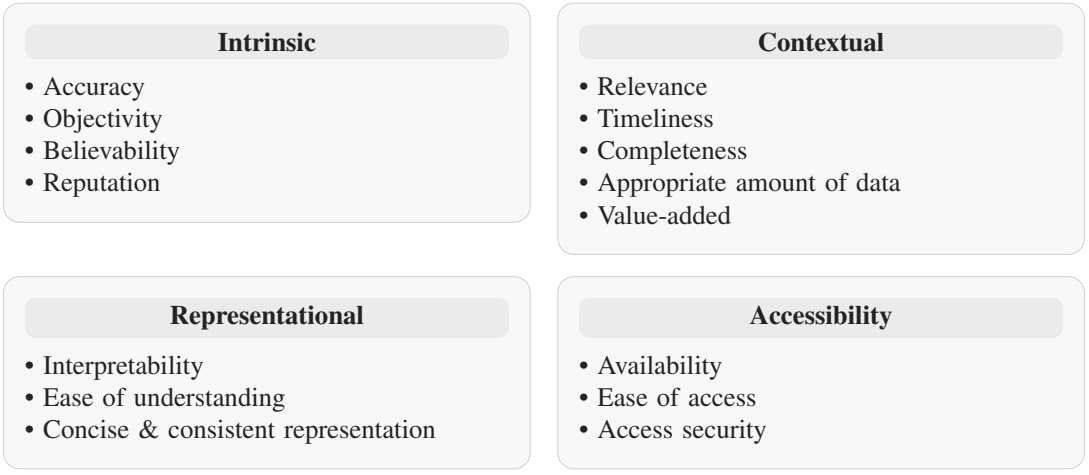


Figure 3: Wang & Strong's consumer-oriented data-quality categories and dimensions.

From this viewpoint, data-quality is inherently contextual and purpose-dependent: data that is technically correct may still be unfit for use if it is incomplete, outdated, difficult to interpret, or inaccessible in a given usage context. Most importantly, it provides a conceptual bridge between abstract quality dimensions and their practical consequences: data-quality issues do not always violate formal constraints, but can directly undermine the usefulness, reliability, and trustworthiness of the data and model.

Pipino et al. provide empirical support for Wang and Strong's argument that data-quality is "fitness for use" and therefore stakeholder-dependent: they show that database custodians and data consumers can evaluate the same dataset differently, and that these disagreements are not trivial [54]. In machine-learning pipelines, this perspective is particularly relevant, as models act as data consumers whose predictive behavior depends on whether incoming data satisfies the quality expectations they are developed on.

While these studies provide a consumer-oriented foundation, subsequent work has emphasized the need to formalize data-quality dimensions in a technically precise manner to avoid ambiguity [7, 8, 51]. This shows that data-quality can be categorized into two partially overlapping

but conceptually different perspectives: a management-oriented perspective, focusing on frameworks which safeguard the consumer-oriented quality of the data, and a technical perspective, which focuses on the technical formalization of data dimensions. This distinction is central for understanding not only how data-quality problems arise in operational systems, but also how to formalize, categorize, and detect them.

2.1.1 Technical and management-oriented perspectives on data-quality

Management-oriented research focuses on organizational structures, governance processes, and strategic alignment around data assets. Work in this field examines how roles, responsibilities, incentives, and decision-making processes influence data-quality across an organization by examining common problems in data-quality management, such as organizational barriers and coordination challenges [34, 49, 62, 68]. For example, Ryu et al. describe how organizations evolve from ad hoc and reactive data-quality practices toward systematically managed and continuously improved processes [62]. Similarly, Silvola et al. and Haug and Arlbjørn identify recurring organizational challenges, including unclear data ownership, fragmented responsibility, and misaligned incentives, which hinder data-quality [34, 68]. In addition, multiple studies emphasize that technical data-quality defects frequently originate from managerial and organizational issues, such as poorly defined data standards, inconsistent data collection processes, insufficient governance, or undocumented changes in upstream systems [31, 49].

Bridging from this management-oriented perspective to technicalities, Batini and Scannapieco synthesize prior research on data and information quality and move beyond defining dimensions. Instead, they explicitly distinguish between quality dimensions, quality metrics, and assessment methodologies [7, 8]. Furthermore, they argue that data-quality must be evaluated systematically across the full data life-cycle and propose a set of core technical dimensions, including accuracy, completeness, consistency, timeliness, uniqueness, and integrity. Supporting this approach, Pipino et al. argue that data-quality dimensions are only actionable if they can be translated into measurable indicators that reflect both technical properties of the data and its fitness for use [54]. Their work provides a critical link between conceptual quality dimensions and the computable metrics required for automated assessment, which is particularly relevant in machine-learning pipelines. Oliveira et al. further refine this view and argue that formal definitions are necessary to avoid ambiguity and overlap in the classification of data-quality problems [51].

While the preceding discussion has framed data-quality primarily in terms of technical dimensions and conceptual taxonomies, these perspectives only become meaningful once we define the scope of data-quality issues. In practice, data-quality issues are not exceptional edge cases but widespread in real-world datasets. The next section therefore examines how frequently quality problems occur and at what scale.

2.1.2 The extent of data-quality problems in practice

The importance of systematic data-quality frameworks is reinforced by extensive empirical evidence demonstrating that data-quality problems are widespread and multifaceted in real-world systems. These studies consistently report that operational datasets are rarely free of errors.

Abedjan et al. provide a comprehensive survey of data error types and detection techniques, synthesizing findings from a broad range of real datasets [1]. They show that missing values occur

in nearly all examined datasets, often following systematic rather than random patterns. Inconsistencies are particularly prevalent in enterprise and open data settings, while duplicates and near-duplicates frequently arise in large-scale repositories and integration scenarios. Outliers and noisy observations are common in sensor data, user-generated content, and transactional logs. Crucially, these error types rarely occur in isolation; datasets typically exhibit multiple, interacting data-quality problems simultaneously, which significantly complicates both detection and remediation.

Operational studies of data-warehouse and analytics projects reinforce these findings. Empirical analyses of large production datasets report double-digit percentages of unusable or low-quality data [27]. Furthermore, studies of data pipelines and analytics initiatives show that a substantial fraction of projects underperform or fail due to unaddressed data-quality issues [1, 31].

More recent discussions also note emerging challenges in modern AI workflows, such as annotation inconsistencies and sampling biases in deep-learning data collection, and the increasing need for automated mechanisms to monitor data-quality at scale [40, 75]. Model sensitivity to such data-quality degradation has been observed across application domains. Pol et al. show that data-quality certification is essential for maintaining reliable model behavior in large-scale scientific pipelines [55]. Swazinna et al. study the effects of data-quality degradation and show that deficiencies in offline reinforcement-learning datasets can induce systematic bias in policy evaluation [72]. Collectively, these findings suggest that technical data-quality issues affect performance beyond standard classification and regression benchmarks, leading to downstream, real-world impact, and motivate a shift from ad hoc fixes toward systematic approaches that characterize and monitor quality degradation.

2.1.3 Granularity levels of technical data-quality

Since Foidl et al. and Abedjan et al. show that defects observed in machine-learning pipelines often originate upstream and then propagate across processing stages, the location where a quality issue first arises is frequently different from where its effects become visible [1, 31]. Therefore, understanding the granularity at which quality problems arise provides a foundation for formalizing technical data-quality dimensions. Rahm and Do stated that rather than treating quality defects as uniform, problems arise at distinct structural levels, each associated with different error types and different implications for downstream systems [58]. They incorporated this by separating single-source problems from multi-source integration problems, where heterogeneity across systems and the merging process itself can introduce additional defects. Oliveira et al. refine the single-source case further by distinguishing single-relation issues (e.g., duplicates, constraint or dependency violations within a table) from multi-relation issues that emerge when considering relationships between relations [51].

Figure 4 summarizes these levels. At the finest granularity, defects affect individual attribute values or tuples, such as missing entries, typographical errors, or outlier measurements (section 2.4). At higher levels, issues increasingly arise from interactions—among tuples within a relation, among relations within a source, or across sources during integration, so that errors can be transformed or amplified as the pipeline aggregates, joins, and engineers features.

Granularity therefore clarifies where technical defects originate, but it does not yet specify how such defects should be represented in a way that enables automated diagnosis. In particular, the

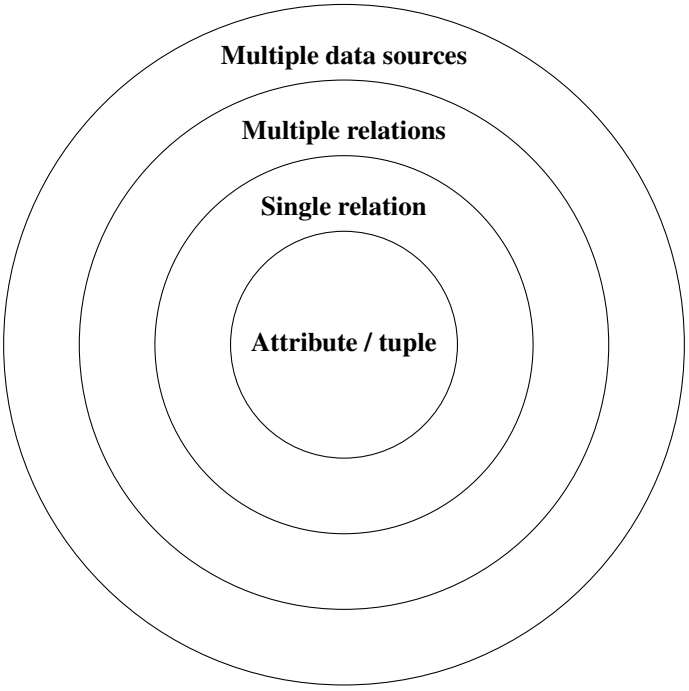


Figure 4: Granularity levels of technical data-quality problems following Oliveira et al. [51].

same abstract quality dimension can manifest through different defect mechanisms depending on whether the issue occurs at the attribute/tuple level, within a relation, across relations, or during multi-source integration [51].

For the purposes of this thesis, this implies that technical data-quality must be defined as a set of computable indicators whose definition is explicitly formalized [51] and comes from a set of computable indicators for data-quality [1, 54].

2.1.4 Formalizing technical data-quality dimensions and indicators

Following Batini and Scannapieco, we adopt a set of technical dimensions, namely accuracy, completeness, consistency, timeliness, uniqueness, and integrity as the basis for quality assessment [7, 8]. These dimensions are sufficiently general to cover common issues, yet concrete enough to be mapped to measurable indicators. Different types of degradation for detection are sourced from Abedjan et al. [1]. In addition, Oliveira et al.'s formalization of data-quality problems is adopted to create these mappings [51]. These metric families are batch-computable, so they can be logged as metadata and used as model inputs regardless of the dimensions of the datasets. Table 1 summarizes how each adopted dimension is translated into a family of computable metrics. Figure 5 shows how these dimensions and computable metrics relate to the different granularity levels. The exact metric construction and the feature extraction pipeline are specified in Chapter 3.

Degradation type	Quality dimensions	Operational definition
Missingness	Completeness	Required attribute values or records are absent (nulls, omitted entries, incomplete), including missing values introduced by schema evolution, joins, or mapping rules.
Noise / Measurement error	Accuracy	Values are imprecise or incorrect due to faulty transformations, erroneous entry, sensor noise, or inconsistent update operations; manifests as attribute-level distortions that violate expected value properties.
Outliers / Extreme values	Accuracy; integrity (domain conformance)	Rare or aberrant values that violate domain/range expectations or implicit distributional assumptions. Erroneous updates, measurement faults, or invalid assignments.
Redundancy / Duplication	Uniqueness; consistency	Duplicate or near-duplicate entity representations under a given identity definition, introduced by repeated ingestion, insufficient de-duplication, unresolved entity matching, or inconsistent merge rules.
Distributional shift / Drift	Timeliness (as change over time); accuracy proxies	Systematic changes in univariate or multivariate feature distributions or co-occurrence patterns across batches relative to a historical reference, detectable via drift indicators and associated with potential performance degradation.

Table 1: Taxonomy of technical data-quality degradation types and their mapping to quality dimensions [1, 6–8, 51].

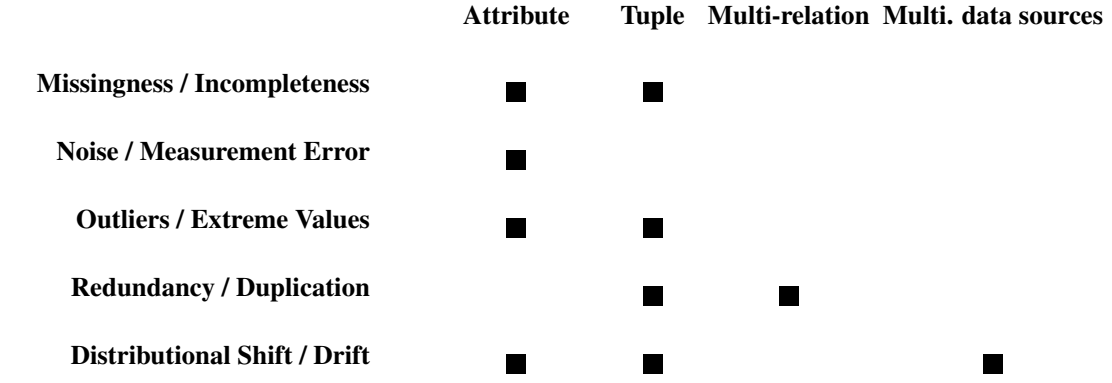


Figure 5: Degradation types by granularity level. Filled circles indicate granularity levels at which a degradation type typically manifests. *Attribute* = single feature value; *Tuple* = individual record (row); *Multi-relation* = issues arising from relationships among tables within the same source; *Multi. data sources* = final, model-consumed batch after preprocessing and joining.

2.2 Data-Quality and Its Effect on Machine-Learning Performance

2.2.1 From technical data-quality defects to performance degradation

While Section 2.1 established that data-quality problems are pervasive and structurally embedded in data pipelines, their relevance for machine-learning pipelines becomes most evident when considering how such degradations affect predictive performance. Machine-learning models implicitly assume that training and deployment data are drawn from similar distributions and satisfy certain statistical and semantic properties [38]. Violations of these assumptions, introduced through technical data-quality defects, can therefore have a direct and often non-linear impact on model behavior. The degradation mechanisms studied in this literature can be interpreted through the technical degradation categories and granularity levels introduced in Section 2.1, including missingness and noise at the attribute level, inconsistencies at the relation level, and distributional changes at batch or dataset level.

2.2.2 Empirical evidence on the effects of data-quality issues in supervised learning

A substantial body of empirical work demonstrates that data-quality issues such as missingness, noise, outliers, inconsistencies, and redundancy systematically degrade machine-learning performance. Blake and Mangiameli provide one of the earliest empirical analyses of this relationship, showing that classification accuracy deteriorates as data-quality declines and that the magnitude of this effect depends on both the type of degradation and the complexity of the learning task [12]. Their results indicate that certain defects, particularly noise and missing values, interact with model complexity in non-trivial ways, making performance degradation difficult to predict using simple heuristics or isolated quality indicators.

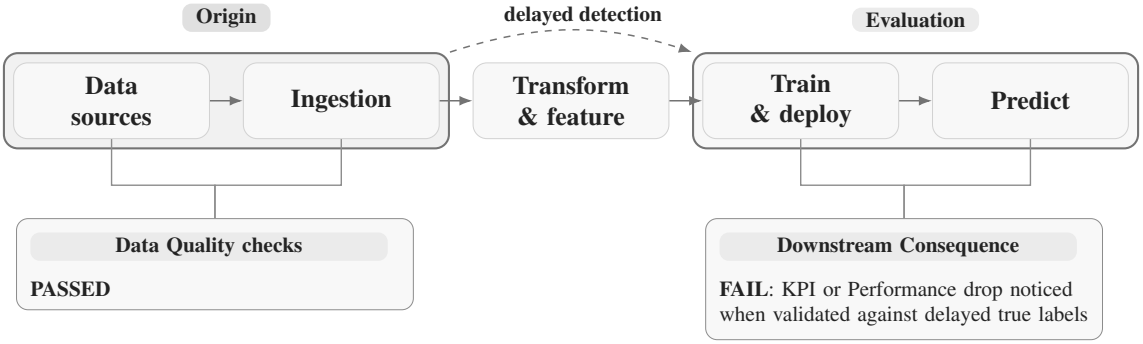


Figure 6: Data-quality issues may exist upstream, but checks can still be passed and these issues are exposed much later when model is validated against true labels.

More recent studies extend these findings to modern machine-learning settings and larger datasets. Budach et al. investigate the impact of multiple data-quality dimensions on supervised learning models and show that different degradation types affect models in qualitatively different ways [16]. For example, systematic missingness and label noise lead to consistent performance deterioration across model families, whereas outliers and redundancy exhibit more model-dependent effects. Mohammed et al. similarly demonstrate that technical data-quality issues in tabular datasets, including noise, missing values, and inconsistent encodings, can significantly affect

both predictive accuracy and model stability, even when traditional data-centric validation is satisfied [50]. Together, these studies reinforce the conclusion that data-quality defects are a primary source of performance variation in deployed machine-learning pipelines and that performance degradation can occur when the underlying data passes data-centric validation (Figure 6).

2.2.3 Sensitivity benchmarking through controlled perturbations: the JENGA framework

An intuitive approach operationalizes data-quality dimensions as controllable degradation factors in order to study model sensitivity. JENGA introduces a benchmarking framework that evaluates machine-learning models under systematic data corruptions targeting specific quality dimensions, such as missingness, noise, outliers, and distributional shift [65].

Rather than focusing on adversarial robustness or model defenses, JENGA adopts a diagnostic perspective by measuring how model performance degrades when individual data-quality dimensions are perturbed in isolation. This limitation of individual dimension degradation results in maximum interpretability, simplicity, and usability for routine robustness checks, rather than evaluating multiple data-quality defects, often seen in real-world data. In addition, JENGA assumes that the type and severity of degradation are known in advance. This assumption never holds in production, where labels are delayed, sparse, or entirely unavailable at inference time.

Nevertheless, the framework provides empirical support by showing that models exhibit highly heterogeneous sensitivity profiles, with certain degradations consistently leading to substantial performance loss and others producing more subtle or model-dependent effects. Importantly, JENGA demonstrates that no single model is uniformly robust across all data-quality dimensions, and that sensitivity depends jointly on the degradation type, its severity, and the learning algorithm. As such, JENGA establishes the importance of model-centric data-quality validation, but does not qualify for adoption in pipelines.

2.2.4 Monitoring model-centric validation through metadata-driven score prediction

Since performance degradation is only observable after ground-truth labels become available or after downstream business impact has occurred, new approaches have been studied. Schelter et al. propose a score-prediction framework that estimates expected model performance on incoming data batches using metadata extracted from the data itself [64]. Rather than directly measuring accuracy, their approach learns a mapping between dataset-level characteristics, such as distributional statistics and other metadata features, and observed model performance during training and evaluation. This enables the prediction of performance degradation under distributional change or corrupted input data, even in the absence of labels at inference time. Empirical results show that metadata-driven score prediction can successfully anticipate performance loss across a range of degradation scenarios, including synthetic corruptions and dataset shift.

However, the score-prediction approach remains limited. While it can indicate that a model is likely to underperform, it does not explain why the degradation occurs. The predicted score is a summary that does not attribute performance loss to specific data-quality defects or combinations

thereof. Consequently, practitioners are alerted to potential issues but lack actionable insight into which aspects of the data have changed in ways that harm the model.

2.2.5 From performance-aware monitoring to root-cause explanation

Taken together, the literature reviewed in this section reveals a gap between performance-aware monitoring and actionable root-cause explanation. Empirical studies and sensitivity benchmarks demonstrate that technical data-quality degradations have structured and often substantial effects on machine-learning performance. Score-prediction models further enable early warning of potential performance loss without requiring immediate labels through the use of metadata representation. However, these approaches do not directly identify which specific data-quality factors, or combinations thereof, are responsible for an observed performance drop in production settings. This motivates a supervised, metadata-driven approach that explicitly maps batch-level changes to interpretable degradation causes and can support root-cause analysis under multi-cause conditions.

2.3 Metadata-driven Data-Quality Assessment

Sections 2.1 and 2.2 established two key premises for operational machine-learning systems. First, technical data-quality defects are pervasive in real pipelines and arise at multiple granularity levels, from individual attribute values to source integration. Second, these defects have structured and often non-linear effects on model performance, while performance itself is frequently hard to measure online due to delayed or missing labels. In this setting, continuously inspecting raw data at instance level becomes increasingly impractical as data volumes, heterogeneity, and pipeline complexity increase, both from a computational and a human interpretability perspective [1, 31, 49], and even if these checks are satisfied, a model can still deteriorate in accuracy [50]. Modern ML pipelines ingest, transform, and materialize data into batches at high velocity and scale, which makes full manual inspection infeasible and complicates reasoning about how data properties evolve across pipeline stages.

To address these constraints, data-quality research and practice have adopted metadata as a scalable abstraction for describing, monitoring, and reasoning about dataset characteristics. Metadata captures structural and statistical properties of data, such as schema information, value distributions, missingness patterns, constraint violations, and drift indicators, without requiring persistent access to all underlying records. By summarizing datasets at batch level, metadata enables continuous quality assessment in large-scale systems and provides a representation through which validation, monitoring, and observability can be operationalized.

2.3.1 Metadata as a representation of data-quality

In the context of data-quality, metadata denotes complementary information of datasets and their evolution over time. This includes but is not limited to schema-level metadata, such as attribute names, data types, and constraints, statistical metadata, such as distributions, moments, sparsity patterns, value ranges, and higher-order indicators derived from comparisons across batches, such as drift metrics or anomaly scores. Rather than replacing raw data, metadata serves as a

compact and tractable representation that supports scalable reasoning about data properties and their change over time.

Several studies position metadata as a foundational mechanism for expressing and evaluating data-quality conditions. A broad perspective is provided by Ulrich et al., who synthesize the roles of metadata across data-intensive systems and emphasize that metadata is not only descriptive, but also analytical and operational, enabling validation and system-level reasoning as data evolves [74]. Aljumaili et al. support this claim more specific to data-quality by describing metadata as a primary metric for representing technical quality dimensions, showing how properties such as completeness, consistency, and validity can be represented through metadata-derived indicators [4]. Visengeriyeva and Abedjan similarly demonstrate that deviations in statistical and structural metadata can be used to detect data errors without record-level inspection, which is particularly important in large repositories where exhaustive checks are computationally expensive [76].

2.3.2 Categories for Data-Quality assessment

Because metadata is a broad concept, it is important to distinguish which types are most informative for technical data-quality dimensions in ML pipelines and how they relate to the quality dimensions and granularity levels introduced in Section 2.1. Structural metadata specifies expected schema and constraints at the attribute/tuple and single-relation level, and relationships at the multiple-relations level, for example keys and joins, making it closely linked to representational and consistency failures such as type mismatches, schema drift with missing or extra features, and integrity-constraint violations. Statistical metadata summarizes empirical properties such as distributions, missingness rates, correlations, and value ranges, computed at the attribute/tuple level and aggregated for a single relation; it can also be extended across multiple relations and multiple data sources to monitor completeness, validity, and distribution shift or drift. Semantic metadata captures the meaning and intended interpretation of data, for example feature and label semantics, units, and domain concepts, enabling detection of inconsistent encodings and implausible values at the attribute/tuple level and improving coherence across relations and sources when combined with structural constraints. Operational metadata characterizes how data is produced and transported through pipelines, including batch size, freshness, latency, lineage or run status, and change indicators, which is most salient at the multiple data sources level and connects directly to timeliness and reliability concerns in production settings [31, 74].

For technical data-quality in ML pipelines, structural and statistical metadata are typically the most central, because they can be computed repeatedly on evolving batches and map naturally to common degradation types such as missingness, noise, outliers, inconsistencies, redundancy, and drift as introduced in Section 2.1 [1, 4, 76]. The effectiveness of metadata-based monitoring depends on metadata-quality itself. For example, Rousidis et al. show that incomplete, inconsistent, or unreliable metadata can significantly limit downstream validation and analysis in large repositories. [60]. In this context, 'metadata-quality' focuses semantic and descriptive metadata (e.g., business meaning, provenance, labels) rather than technical metadata derived directly from the data itself, which cannot be of low-quality. This dependence has motivated automatic detection methods. For example, Lorenzini et al. use supervised text classification to identify low-quality descriptive metadata [47].

Recent work also extends metadata beyond static summaries toward learning-oriented representations. Elouataoui et al. propose an active metadata perspective in which metadata features are continuously updated and analyzed using machine-learning models to support quality management in large-scale environments, reflecting a shift toward treating metadata as an input signal for automated decision-making rather than only documentation [29].

2.3.3 Metadata-driven validation frameworks

Metadata is not only used for descriptive profiling, but also as the basis for automated data-quality validation. In constraint-based validation, quality expectations are expressed over metadata properties, such as acceptable value ranges, schema consistency, uniqueness, completeness, or distributional stability. Influential frameworks in large-scale ML environments are Deequ, Great Expectations and TensorFlow Data Validation (TFDV), which formalize validation as metadata profiling plus anomaly detection relative to expected distributions, predefined constraints and learned baselines.

Deequ provides a declarative framework for defining and verifying data-quality constraints at scale, designed for integration with distributed processing settings [66]. The core idea is to compute dataset and batch-level metrics, for example completeness rates, uniqueness ratios, approximate distributions, and constraint violation counts, and to validate these metrics against expected ranges. Expectations can be specified manually or derived from historical profiling, which enables detection of deviations such as missingness spikes, shifts in categorical value sets, unexpected range expansions, or changes in distributional shape. The Deequ papers emphasize that this approach supports continuous verification in production pipelines by treating each incoming batch as an object to be assessed through metadata rather than inspected record by record [66].

TFDV adopts a similar metadata-centric philosophy within the TensorFlow Extended ecosystem [22]. It profiles datasets to infer schemas and statistical baselines, then validates new data against these metadata profiles. This enables detection of anomalies such as missing features, type mismatches, schema violations, unexpected categorical values, and distributional changes. Like Deequ, TFDV operationalizes technical data-quality primarily as conformance between an observed metadata profile and an expected metadata profile, which is a natural fit for batch-based ML pipelines where quality must be checked repeatedly as new data arrives.

Great Expectations provides a related paradigm that emphasizes human-readable, test-oriented expectations, expressed as executable checks over metadata summaries [30]. While often used in analytics settings, it is also used to validate training and inference data in ML pipelines, especially when teams want explicit, auditable quality assertions. Across these frameworks, a common pattern emerges: metadata enables scalable validation of many technical dimensions defined in Section 2.1, including completeness, validity, consistency, uniqueness, and distributional stability, by translating these concepts into computable batch-level statistics and constraint checks.

2.3.4 Automating validation specification and adaptation

A central limitation of constraint-based validation in practice is the specification burden. Defining and maintaining high-quality rule sets is labor-intensive, brittle under changing pipelines, and

difficult to scale across many datasets. This motivates a line of research that automates the discovery, adaptation, and maintenance of validation constraints by learning normal patterns from data, from history, or from broader repositories.

Auto-Validate proposes unsupervised techniques to infer data-domain patterns and generate validation constraints from large repositories, aiming to reduce manual rule authoring by learning what typically holds for similar tables and attributes [70]. Auto-Validate-by-History extends this idea by leveraging historical pipeline executions to identify recurring properties and derive validation rules that are stable over time within a given pipeline context [73]. Shah et al. propose an AI-powered monitoring perspective that adapts validation logic over time using learned baselines, reflecting the broader trend toward using machine learning to maintain and refine data-quality monitoring under non-stationarity [40].

More recently, large language models have been explored as a mechanism to lower the barrier for specifying validation requirements. Abughazala and Muccini propose approaches that translate high-level requirements into executable checks, for example into Great Expectations specifications, which can improve usability and accelerate the development of quality test suites [2]. Across these automation-oriented approaches, metadata remains the central representation, either as the object on which constraints are defined or as the learned signal that characterizes normal behavior.

2.3.5 Enterprise Data Validation Platforms

In industrial practice, metadata-driven validation and monitoring principles are embodied in enterprise data observability platforms such as Collibra [26], Informatica [39], Monte Carlo [21], and Soda [69]. These systems continuously compute metadata profiles over incoming data batches and monitor properties such as schema changes, freshness and latency, volume anomalies, distributional shifts, and rule violations. In contrast to many academic prototypes, they place strong emphasis on operational concerns, including alerting workflows, integration with data stack components, and broad coverage across many pipelines and datasets. Conceptually, they operationalize the same core idea as the validation frameworks above: data incidents are detected through deviations in metadata relative to expected behavior.

2.3.6 Data-centric versus model-centric validation

Across both research and practice, the dominant use of metadata is data-centric (Section 2.1). Metadata is used to validate or monitor incoming batches relative to constraints or historical norms, with the objective of detecting deviations in the data itself. These approaches provide strong coverage for many technical issues, but they typically do not determine whether a detected deviation is consequential for a downstream machine-learning model.

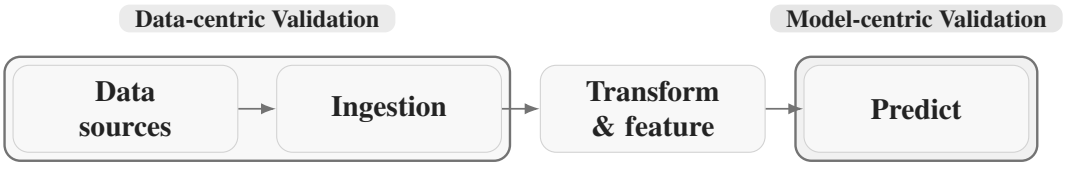


Figure 7: Data-centric and model-centric methods within real-time ML-pipeline.

Model-centric approaches instead aim to reason about model behavior under changing data conditions (Section 2.2. A prominent example is the score-prediction framework of Schelter et al., which estimates expected model performance from dataset-level metadata when labels are unavailable [64]. However, model-centric monitoring approaches do not yield an explanation for the observed degradation, and therefore do not identify which technical degradation types, or combinations of types, are responsible for underperformance.

This separation between deviation detection and performance attribution exposes an important gap for operational ML systems. Data-centric tools can indicate that incoming data differs from expected baselines, and model-centric methods can indicate that performance may degrade, but neither provides a mechanism that links metadata changes to interpretable explanations of model underperformance. This gap motivates the methodological direction of this thesis, which treats metadata not only as a detection instrument, but also as a structured signal for root-cause explanation.

The next section examines anomaly and drift detection methods. These methods, integrated into the tools and frameworks discussed, provide models and approaches for identifying unusual patterns and distributional changes in datasets, and they additionally inspect dataset characteristics which cannot be concluded from current derived statistics.

2.4 Anomaly & Drift Detection for Data-Quality

Building on the idea of representing data dimensions through metrics, a large body of work studies how unexpected deviations in incoming data can be detected automatically. In the literature, this problem is generally addressed through anomaly detection. Smaller deviations accumulating over time are generally measured using drift or shift detection. This section reviews detection methods and algorithms, with an emphasis on how they relate to technical data-quality degradations (Section 2.1 and 2.2).

2.4.1 From technical data-quality degradations to detectable signals

Technical data-quality problems such as missingness, noise, outliers, inconsistencies, redundancy, and distributional change often manifest as deviations in batch-level summaries, for example shifts in missing-value rates, changes in value ranges, altered category frequencies, unusual correlation structure, or increasing constraint-violation counts [1, 4, 76].

Using pure metadata derived statistics as in Section 2.3 is possible by the following approach: given metadata snapshots extracted from historical batches that were considered acceptable, determine whether the metadata of a new batch is statistically atypical or violates expected patterns. However, the literature also stresses that deviations are heterogeneous. Some correspond to data-quality defects (for example systematic missingness or domain-violating values), while others reflect genuine changes in the underlying process (for example seasonality or shifting user populations). This ambiguity motivates robust detection methods and careful interpretation of alerts, particularly in operational settings where distributions are non-stationary [63]. Additional methods of detecting shifts and outliers can provide valuable complementary information to prevent misinterpretation.

2.4.2 Anomaly detection as a mechanism for data-quality control

Anomaly detection provides methods to flag atypical observations or batches without requiring labeled error instances. Chandola et al. provide a foundational survey, distinguishing between point anomalies, contextual anomalies, and collective anomalies, and highlighting that anomaly definitions are application dependent [23]. In data-quality contexts, anomalies may occur at multiple granularity-levels: at attribute level (unexpected value ranges, missingness), at tuple level (aberrant records), and at batch level (unusual global statistics).

Classical approaches rely on distributional assumptions or distance-based criteria. In pipeline settings, these methods often operate on feature-wise statistics or on learned representations of batches, thereby aligning naturally with metadata extraction pipelines [4, 76]. More scalable and widely adopted methods include tree-based techniques such as Isolation Forest [46], as well as copula-based [43] and multivariate techniques [61] that aim to handle complex dependencies among attributes. In practice, these methods are frequently used to detect outliers and rare patterns that can indicate sensor malfunctions, logging errors, or upstream transformation issues. Empirical studies in applied domains, including medical and scientific data pipelines, show that multivariate outlier detection can support data-quality evaluation when raw inspection is infeasible [55, 71].

In the data-quality literature, a recurring theme is that anomaly detection is most useful when paired with domain constraints or expected invariants. For example, recent work explores anomaly-based metrics for semantic consistency and probability-based assessment of consistency violations, reflecting a trend toward combining statistical evidence with semantic validity [35, 41]. This line of work is relevant for ML pipelines because many impactful defects are not extreme values but subtle inconsistencies, such as logically incompatible attribute combinations.

2.4.3 Drift and dataset shift detection

While anomaly detection often targets rare or extreme deviations, drift detection focuses on systematic changes over time. In machine learning, drift is frequently operationalized as dataset shift, meaning that the joint distribution of inputs, labels, or both changes between training and deployment [63]. The literature distinguishes several common shift types that are often adopted for data-quality monitoring.

Covariate shift refers to changes in the input distribution while the conditional label distribution remains stable. Discriminative learning under covariate shift and related work show that such shifts can be detected and sometimes corrected using reweighting or density-ratio estimation, including leveraging unlabeled target data [11, 36].

Label shift refers to changes in the label marginal distribution, and methods exist to detect and correct for it using black-box predictors [45]. From a data-quality perspective, shifts in feature distributions can arise from real-world process change but also from technical degradations such as upstream encoding changes, systematic missingness, or changes in data collection instrumentation, which again highlights that drift detection is not equivalent to defect diagnosis. Recent work also explores embedding-based drift detection, for example Drift Lens, which evaluates shifts in

per-label embedding distributions in an unsupervised manner [33]. Such approaches are particularly compatible with metadata pipelines when embeddings or latent features are included as part of monitoring data.

A major study by Rabanser et al., which evaluates practical methods for detecting dataset shift, emphasizes that detection reliability varies strongly by method, data modality, and shift type [57]. Their findings motivate the use of multiple complementary signals rather than a single drift statistic, which supports the implementation of multiple batch-wise drift detection algorithms in combination with metadata derived statistics.

2.4.4 Pipeline-aware and slice-based detection in ML operations

Instead of applying these methods at one stage in the data life-cycle, pipeline-aware detection monitors assess the data through multiple stages and changes are tracked as first-class objects [7, 8]. More recently, work such as CM-Explorer targets ingestion problems specifically, supporting the view that operational failures often reflect stage-specific breakdowns rather than purely statistical shift [17]. In addition, work on change exploration in data pipelines formalizes change as an analyzable signal and provides conceptual tools for systematically dissecting how datasets evolve through pipelines [13].

A further operational challenge is that drift and anomalies are often localized. dataset-level aggregates can hide that a specific subgroup, feature slice, or regime has shifted. Slice-based approaches attempt to detect drift or failures on subsets of the data. Ackerman et al. propose detecting model drift via weak data slices, which operationalizes the intuition that changes concentrated in certain slices may be early indicators of broader degradation [3]. While such methods are often framed as model drift detection, the underlying mechanism remains data-centric in the sense that it detects localized distribution changes, and it does not by itself provide root-cause labels tied to technical degradation types.

2.4.5 Limitations for model-centric diagnosis

Despite strong progress in anomaly and drift detection, the literature indicates two limitations that are central for this thesis. First, most methods are data-centric and largely unsupervised. They identify that incoming data deviates from historical baselines, but they do not attribute deviations to specific technical degradation types, nor do they resolve multi-cause scenarios where several issues co-occur [23, 57]. Second, deviation does not necessarily imply harm. A batch can drift in ways that are benign for a particular model, while small and localized changes can materially affect performance. As a result, alerts based on distributional deviation or outlier scores alone are difficult to prioritize and often remain non-actionable in operational pipelines.

These limitations imply that detection, performance awareness, and explanation are distinct tasks. Detection methods provide candidate signals that something has changed, but they do not provide a controlled basis for learning which kinds of changes are harmful, nor a label space for explaining which data-quality factors are responsible when degradation is observed.

2.4.6 Remaining gaps for root-cause diagnosis

In summary, anomaly and drift detection methods offer scalable mechanisms for flagging unexpected changes in data batches using statistical and metadata-based signals. They are therefore

essential components of data-quality monitoring and observability. However, their outputs are typically alerts rather than explanations: they do not provide ground-truth labels about the underlying degradation mechanisms, and they do not isolate causal factors when multiple defects co-occur.

This motivates the use of synthetic data corruption as a complementary research strategy. By injecting controlled, feature- and severity-specific perturbations into otherwise clean data, synthetic corruption enables the creation of labeled degradation scenarios that are rare or hard to observe in production. Such scenarios make it possible to quantify model sensitivity to specific technical data-quality dimensions and to learn supervised mappings from metadata changes to degradation causes. The next section therefore reviews literature on synthetic corruptions and data error simulation, focusing on how these techniques are used to study robustness and to generate controlled degradation sets.

2.5 Synthetic Data Corruption & Data Error Simulation

The previous sections show that anomaly and drift detection methods indicate that incoming data deviates from historical behavior in addition to metadata summaries, because these are not optimized for outlier or drift detection. However, there exists a problem with testing these features applicability within root-cause analysis.

In operational machine-learning pipelines, incident data is rarely curated in a way that supports supervised diagnosis, since failures may be infrequent, context-specific, and only partially observed [49]. This motivates the use of synthetic data corruption and data error simulation as a methodological instrument to create controlled, repeatable degradation events with known causes.

Synthetic corruptions are widely used in both research and practice to study how models behave under degraded input conditions. The central idea is to apply parameterized perturbations that instantiate technical data-quality defects, often focused on data quality metrics such as missingness, noise, outliers, inconsistencies, redundancy, or distributional shift as introduced in Section 2.1. These perturbations are utilized to measure how predictive performance changes as a function of the defect type and its severity [12, 16, 50]. In contrast to purely observational studies, synthetic simulation provides experimental control: the degradation mechanism is explicitly defined, its intensity can be varied, and the resulting performance impact can be evaluated consistently across models and datasets.

2.5.1 Corruption-based sensitivity analysis & robustness evaluation

JENGA, discussed in Section 2.2 is a prominent example of corruption-based evaluation. The framework provides an approach to study the impact of data errors on machine-learning predictions by applying corruptions that target a specific data-quality dimension [65]. The goal is not to propose defensive techniques, but to quantify sensitivity. JENGA operationalizes common technical defects as controllable interventions, enabling comparative evaluation of different model families under dimension-specific stress tests.

At the same time, these corruption-based benchmarks highlight an important gap between controlled evaluation and operational diagnosis. Evaluation frameworks such as JENGA assume that

the single degradation type and its severity are known [65]. As discussed in Section 2.1, in practice degradations are not labeled, may emerge gradually, and frequently co-occur, which makes it difficult to infer the underlying cause from performance behavior alone. This limitation is present in JENGA, which provides evidence that technical defects matter for performance, but that a separate mechanism is required to infer which defects have occurred when performance degradation is observed.

2.5.2 Corruption operators aligned with technical data-quality dimensions

To produce meaningful insights, corruption operators must be aligned with technical data-quality dimensions and can be related back to the taxonomy introduced in Section 2.1. Missingness injection operationalizes defects in completeness by removing values at the attribute or tuple level. Noise injection operationalizes defects in accuracy by perturbing values through additive, multiplicative, or categorical replacement mechanisms. Outlier injection operationalizes accuracy issues by creating rare or extreme values that violate expected ranges or statistical regularities. Duplication and redundancy injection operationalize uniqueness defects by inserting identical duplicate records.

Distributional shift simulation operationalizes dataset-level degradations by changing feature values which will alter distributions across batches, either through covariate shift or through more structured changes in joint distributions. This connects directly to the dataset shift literature and to empirical studies showing that models can fail silently under distributional change [57, 63]. In addition, relation-level inconsistencies can be simulated by generating constraint violations, such as invalid combinations of categorical values or broken integrity relationships in relational settings, which aligns with the constraint-based view of validation widely adopted in metadata-driven frameworks [22, 66].

This mapping between corruption operators and technical dimensions is closely related to a broader line of work that treats data errors as actionable objects for machine-learning pipeline improvement. For example, BoostClean frames error detection and repair as a way to improve downstream model performance by systematically searching over cleaning operations and selecting those that yield measurable performance gains [42]. Similarly, work on data evaluation and enhancement for machine learning emphasizes that dataset quality can be improved through systematic assessment and targeted transformations, with downstream performance as a guiding signal [24]. While these approaches focus on improving data and models, they reinforce the methodological premise that structured, controllable data transformations can be used to study and manage the relationship between data-quality and performance.

2.5.3 Design space of synthetic corruptions

Synthetic corruption does not consist of a single technique. Instead, there exists a design space of choices that determine whether simulated degradations are informative and realistic.

The first key choice concerns the granularity at which corruptions are applied. As discussed in Section 2.1, technical data-quality problems can manifest at multiple structural levels, ranging from attribute and tuple level defects to relation-level issues and dataset-level shifts. Corruption operators should therefore be designed and interpreted with respect to these levels.

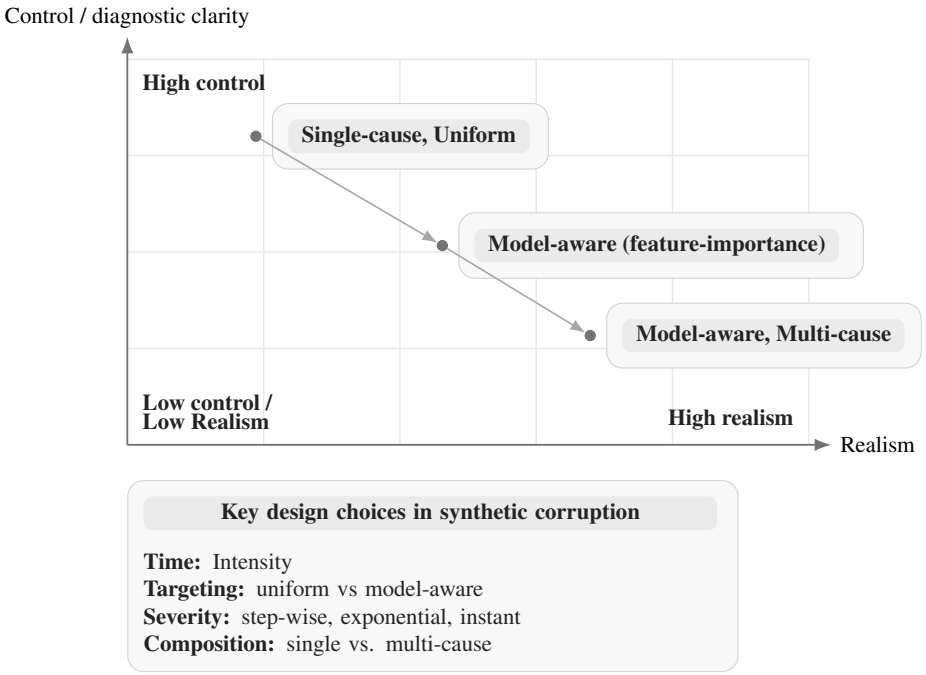


Figure 8: Design space of synthetic corruptions, illustrating trade-offs between realism and diagnostic control, and core design choices (granularity, severity/temporal structure, composition, and targeting).

A second design choice concerns severity parameterization and temporal structure. Severity controls the intensity of a degradation, such as the missingness rate, the magnitude of additive noise, the fraction of duplicated records, or the distance between historical and current distributions. Temporal structure determines how corruptions unfold across batches, for example as abrupt step changes, gradual drift, or intermittent spikes.

A third design choice concerns composition. The data-quality literature discussed in Section 2.1, emphasizes that error types rarely occur in isolation and often interact [1,31]. In machine-learning pipelines, combined defects may amplify performance degradation beyond what is implied by individual defects. This makes multi-cause simulation particularly important for root-cause modeling, since it enables training and evaluation under the realistic assumption that multiple degradation mechanisms can be present simultaneously.

Finally, corruption design must consider targeting. A common approach applies corruptions uniformly at random across records or features, which is a shortcoming regarding model-centric monitoring. When models use regularization, altering features which are not used by the model can give the illusion that a model is robust with respect to a certain data-quality dimension. To prevent this, one can adopt targeted corruption strategies, where features that matter for the machine-learning model are more likely to be perturbed. In this thesis, this consideration directly supports the design choice of a feature-importance aware synthetic degradation pipeline, which allows controlled generation of both single- and multi-cause scenarios within the model's view. The design space of the synthetic corruption and its mechanics of simulating realistic scenarios are discussed in Figure 8.

2.5.4 Limitations of synthetic simulation & implications for this thesis

Synthetic corruption is inherently an approximation of real-world incidents. If corruptions ignore feature dependencies, domain semantics, or unrealistic transformations, they may produce artifacts that are not representative of real failures [31, 49]. To combat this, the approach should not violate these easy to detect defects, since data-centric detectors within the pipeline would already have raised alerts.

Moreover, simplistic random perturbations may not capture structured incidents that affect specific subpopulations, specific source systems, or specific processing steps. These limitations do not invalidate synthetic simulation, but they motivate careful design choices that balance control with realism. In the context of this thesis, this motivates a pipeline-centric, feature importance aware corruption strategy that supports both targeted and multi-cause scenario generation, and that aligns with the technical degradation taxonomy established in Section 2.1.

2.5.5 Positioning and bridge to root-cause modeling

In summary, the literature supports synthetic corruption as a practical and widely used method for studying how technical data-quality defects affect machine-learning models under controlled conditions [12, 16, 65]. Synthetic simulation is particularly valuable in pipeline settings because real degradation events are often rare, heterogeneous, and unlabeled, while simulated corruptions provide known causes, tunable severity, and repeatability [1, 31]. At the same time, prior corruption-based work is primarily evaluative: it quantifies sensitivity to specific degradation mechanisms, but it does not by itself provide an operational mechanism for attributing observed performance loss to concrete, interpretable causes when multiple degradation types co-occur.

This thesis builds on these insights by using synthetic degradation to generate labeled single-cause and multi-cause scenarios that reflect the technical degradation taxonomy established in Section 2.1. This setup naturally induces a multi-label prediction problem at the batch level, since multiple degradation types may be present simultaneously within the same incoming dataset. The root-cause model therefore must learn to map metadata signatures to a set of likely degradation labels, rather than selecting a single class. The next section reviews candidate model families for this task, with particular emphasis on approaches that can handle non-linear interactions among features, support multi-label outputs, and remain sufficiently interpretable to produce actionable explanations.

2.6 Supervised Models for Metadata-driven Root-Cause Prediction

The preceding sections established three requirements that shape the learning problem addressed in this thesis. First, technical data-quality degradations are pervasive, arise at multiple granularity levels, and frequently co-occur in realistic pipelines (Section 2.1). Second, these degradations have structured and often non-linear effects on predictive performance (Section 2.2). Finally, metadata-driven monitoring and validation frameworks can detect anomalies in incoming batches at scale, but typically remain data-centric and do not attribute model underperformance to concrete degradation types (Section 2.3). These observations motivate a supervised learning approach that takes batch-level metadata as input and outputs probabilistic explanations over a set of degradation labels.

2.6.1 Problem formulation: multi-class versus multi-label root-cause prediction

A central modeling decision concerns whether degradation diagnosis should be framed as multi-class or multi-label prediction. A multi-class formulation assumes that exactly one degradation type is responsible for a detected degradation event. This assumption can be appropriate in highly controlled settings where failures are isolated or where a dominant cause is known to exist. However, empirical evidence showed that error types often appear simultaneously within the same dataset or batch, and that these error types already in isolation influence machine-learning performance in non-linear ways (Section 2.2). Therefore, considering the literature, a single-label assumption becomes restrictive and unrealistic.

It is therefore more appropriate to treat root-cause identification as a multi-label problem in which a batch may be associated with one or multiple degradation types. This aligns with the objective of predicting co-occurring degradations rather than single-class outcomes. Instead of only predicting a single label, the model can output a probability for each degradation label, enabling a thresholding method to binary classify a degradation. This probabilistic framing is consistent with the diagnostic goal of the thesis, which is to determine whether multiple degradation types are present in an observed batch rather than to provide a hard single-cause prediction.

2.6.2 Metadata as supervised input features

Metadata representations discussed in Section 2.3 provide a practical feature space for supervised learning. Prior work demonstrates that dataset- and batch-level statistics encode meaningful signals about data state and can be used for automated assessment. The viability of this representation for model-centric monitoring is supported by score-prediction research, which shows that metadata can be predictive of downstream performance under shift and corruption [64]. These findings justify treating metadata not only as a validation artifact but also as a structured input space for root-cause learning.

2.6.3 Candidate Model Families

The metadata feature space in this thesis is tabular, heterogeneous, and composed of interacting statistics, which makes it well suited to model families that handle non-linearity and feature interactions without requiring extensive manual feature engineering. Deep learning is a popular approach to learn rather complicated interactions between features and the output variables. However, research conducted comparing tree ensemble models, such as XGBoost to deep learning models on tabular datasets has shown that deep-learning models generally underperform [67].

Therefore, from a methodological perspective, the first class of candidates are tree-based ensemble methods. These methods consist of random forests and gradient-boosted decision trees. Random forests are robust, handle mixed-scale features well, and provide a strong baseline for tabular learning [14]. Gradient boosting often achieves better performance on structured data by iteratively fitting weak learners to residual error and capturing complex interaction effects [25]. These properties align with the need to learn non-linear relationships between metadata and degradation outcomes.

The second class of candidates are neural networks for tabular learning, such as multi-layer perceptrons. Neural models can learn flexible decision boundaries and may benefit from large feature

sets or higher-order interactions. Schwartz-Ziv et al. showed that deep learning models can outperform tree-ensemble models, but require generally more tuning [67].

2.6.4 Explanation & Interpretability Methods

A supervised root-cause model is only operationally valuable if its predictions can be translated into actionable insight. Beyond identifying one or more likely degradation types for a batch, practitioners need to understand which metadata changes the model considers most indicative of each predicted degradation. This motivates post hoc explanation methods that attribute model outputs to input features and thereby support diagnostic reasoning. A common explanatory method is SHAP [48].

LIME (Local Interpretable Model-Agnostic Explanations) [59] explains an individual prediction by sampling perturbations around the instance of interest and fitting a simple surrogate model locally, using the surrogate's coefficients as an explanation. This model-agnostic strategy is intuitive, but a direct limitation of this is its locality, which can reduce stability and comparability across explanations in monitoring scenarios.

SHAP is based on Shapley-value feature attributions and explains a prediction through additive decomposition into a baseline value and per-feature contributions. This additive structure yields explanations that are directly comparable across batches and across predicted outcomes.

For the setting in this thesis, the explanation method is primarily used as a diagnostic aid rather than as a causal claim. Feature attributions describe which features the model relies on to produce its predictions, but they do not establish the true causal mechanism behind a degradation event, since this cannot be interpreted from the data and model alone. This limitation is relevant in multi-cause settings, where correlated metadata changes may arise from shared upstream processes. Consequently, explanation methods must be interpreted as evidence about the model's decision logic and therefore, should be evaluated empirically in controlled settings.

Given that the proposed root-cause model operates on tabular features and may predict multiple degradation types, SHAP is a reasonable choice to support explanations in a structured and comparable way, with efficient methods available for common model families used on tabular data. For example, there exist versions of SHAP optimized for tree-based models [79], which can speed up the explanation process.

2.6.5 Positioning within this thesis

In summary, the literature motivates framing root-cause identification as a multi-label, probabilistic prediction problem over degradation labels, using batch-level metadata as supervised input features. Tree-based ensembles and neural networks emerge as the most relevant model families for this setting due to their ability to learn non-linear relationships in tabular data [14]. Interpretability methods such as SHAP support predicted degradations by linking predicted causes to specific metadata signals [48]. Building on this literature, the next section details the concrete model framework design, training choices, and the evaluation metrics used, including how multi-label predictions are produced and how interpretability methods are utilized for the strongest-performing model.

3 Methodology

After discussing the design rationale and literature, this chapter describes the methodological design of the metadata-driven root-cause model. The goal is to learn a mapping from batch-level quality signals to the degradation types present in that data batch. In contrast to model-centric or data-centric monitoring, the evaluation in this thesis does not focus on predicting model performance or data-quality issues before prediction. Instead, it evaluates whether batch-level metadata and derived quality signals are sufficient to detect which degradation types are present in a batch after a model-centric performance alarm is raised by an external trigger (adopted from the score-prediction approach of Schelter et al. [64]).

First, this chapter introduces formal notation and the overall model design. It then formulates the problem as a supervised multi-label classification task. Next, it details the two main modules that compute model inputs and generate training data:

1. The Quality Feature Builder, which transforms each batch into a fixed-dimensional feature representation based on content-level metadata profiles, outlier signals, and drift indicators;
2. The Synthetic Data Generator, which produces labeled degraded batches by applying programmable combinations of degradation operators.

Thereafter, the chapter describes the datasets and how they are partitioned to create the reference snapshot and incoming batches. In addition, it specifies the experimental setup, including the model families considered. Finally, reproducibility factors and evaluation metrics are discussed.

3.1 Model Design

To introduce the framework, this section presents the top-level view of the model design, by discussing the setting, formal notation and data flow through the model.

3.1.1 Setting & Formal Notation

The model assumes a real-time machine learning pipeline that materializes data at continuous time, but performs monitoring and (if needed) root-cause analysis at discrete time steps. This is necessary because assessment is performed at batch level. Let B_t denote the incoming batch at time t, consisting of n data records. Different values of n are considered (discussed in Section 3.5.7).

Some quality signals require a baseline, while others can be computed purely from the batch. Let R_t denote a reference dataset available at time t, intended to represent accepted and stable data under normal pipeline operation. In this thesis, R_t is constructed from the data used to develop the initial machine-learning model in the pipeline. In production, accepted batches could be incorporated into R_t over time, which is discussed in Section 5.

To obtain labeled training data for root-cause prediction, the framework generates degraded batch instances from the reference snapshot. Let $D_{t,k}$ denote the k-th degraded batch instance at time t. Each degraded instance is derived from R_t by sampling n tuples with replacement and applying a set of degradation operators.

In addition, a distinction is made between perturbation rate and degradation strength. Perturbation rate denotes the proportion of the data that is affected by a degradation, whereas degradation strength denotes the intensity of the degradation applied to the affected values or observations. Throughout this thesis, perturbation rate and degradation rate are used as synonyms; perturbation rate is the preferred formal term. The sampling scheme and degradation operators are formalized in Section 3.3.

For any batch X and reference snapshot R_t, the Quality Feature Builder (QFB) computes a fixed-dimensional representation

$$\phi(X, R_t) \in \mathbb{R}^d,$$

based on batch-internal profiling signals as well as baseline-dependent outlier and drift signals. The feature construction is specified in Section 3.2.

3.1.2 Prediction Formulation

The prediction task is formulated as supervised multi-label classification. Let $\mathscr{L}$ denote the set of degradation labels (Section 2.1) excluding Redundancy/Duplication, as explained in Appendix G. Each constructed degraded batch $D_{t,k}$ is associated with a binary label vector

$$\mathbf{y}_{t,k} \in \{0,1\}^{|\mathscr{L}|},$$

where $y_{t,k}^{(\ell)} = 1$ indicates that degradation type $\ell \in \mathscr{L}$ is present in $D_{t,k}$.

For each batch, the Quality Feature Builder computes

$$\mathbf{x}_{t,k} = \phi(D_{t,k}, R_t) \in \mathbb{R}^d.$$

The root-cause model is a classifier f_θ parameterized by θ that is trained on $\mathbf{x}_{t,k}$ to predict $\mathbf{y}_{t,k}$. Thereafter, the trained model f_θ predicts the degradation set of an incoming batch:

$$\hat{\mathbf{y}}_t = f_\theta(\phi(B_t, R_t)).$$

The central evaluation question of this thesis is whether batch-level quality signals of B_t are sufficient to reliably recover the true degradation label set under multi-cause degradation scenarios. Evaluation metrics are specified in Section 3.5.6.

3.1.3 Module Interfaces & Flow

Figure 9 summarizes the end-to-end flow of the proposed model to achieve predictions. The design separates an offline phase, in which labeled training data is generated and a root-cause predictor is developed, from an online phase, in which incoming batches are checked and assessed if necessary.

In the offline phase (Algorithm 1), the Synthetic Data Generator produces a collection of degraded batch instances. This is achieved by sampling from the reference snapshot and applying programmable combinations of degradation operators. Each degraded batch is associated with a multi-label target vector that indicates which degradation types are present. The Quality Feature

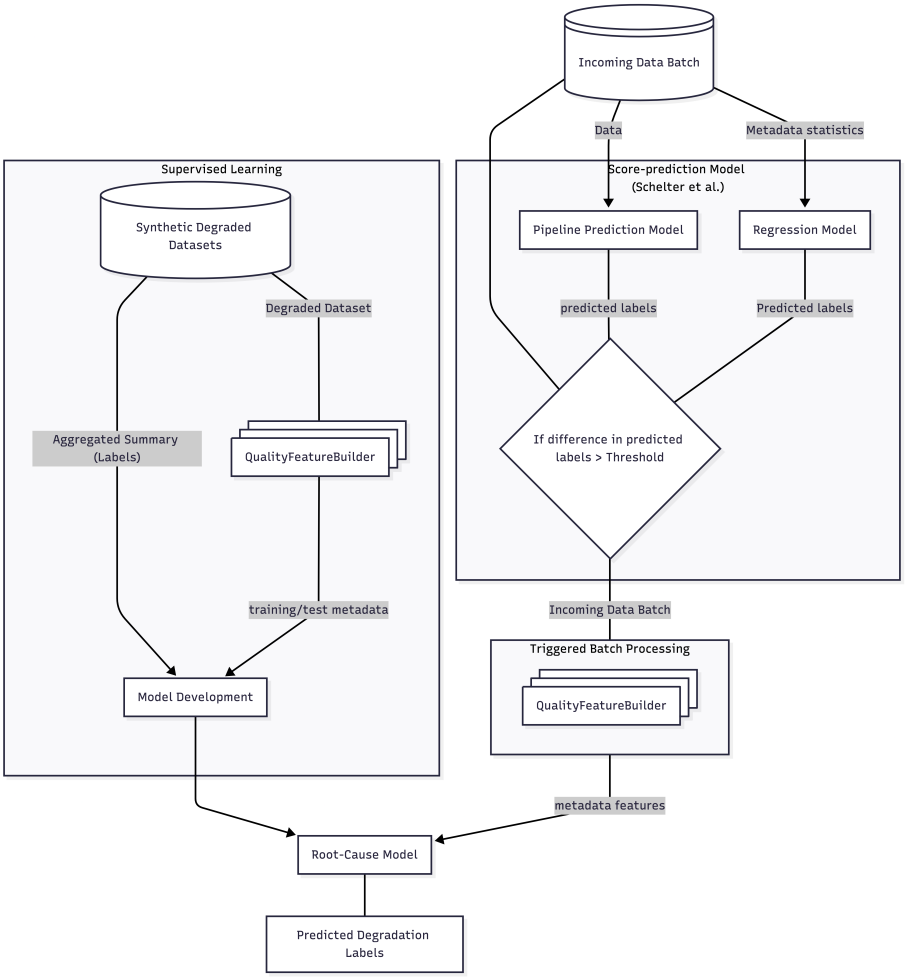


Figure 9: The overall model design and data flow, showing the score-prediction model and placement of the root-cause model within the framework.

Builder is then applied to each generated batch to compute the feature vector of this degraded batch (Section 3.2). The resulting feature and label pairs form the supervised training set.

Algorithm 1 Offline training: synthetic generation and supervised learning

Require: Reference snapshot R_t, batch size n, label set $\mathscr{L}$, Synthetic Data Generator Γ, number of instances K

Ensure: Trained root-cause model f_θ

$\mathscr{T} \leftarrow \emptyset$
for $k \leftarrow 1$ **to** K **do**
  $S_{t,k} \sim \text{SampleWithReplacement}(R_t, n)$
  Sample configuration $\Theta_{t,k}$ (strategy, strengths, perturbation rates)
  $D_{t,k} \leftarrow \Gamma(S_{t,k}, \Theta_{t,k})$
  $\mathbf{y}_{t,k} \leftarrow \text{Labels}(\Theta_{t,k}) \in \{0,1\}^{|\mathscr{L}|}$
  $\mathbf{x}_{t,k} \leftarrow \phi(D_{t,k}, R_t)$
  $\mathscr{T} \leftarrow \mathscr{T} \cup \{(\mathbf{x}_{t,k}, \mathbf{y}_{t,k})\}$
end for
Train multi-label classifier f_θ on $\mathscr{T}$
return f_θ

In the online phase (Algorithm 2), the incoming batch is first run through the external model-

centric trigger (score-prediction model [64]). This trigger estimates whether predictive performance on the unseen batch is likely to deviate from expected behavior based on monitoring signals. If the trigger fires, the batch is routed to the diagnostic stage: QFB computes $\phi(B_t, R_t)$ and the trained root-cause model outputs a set of predicted degradation labels. Thus, the proposed model acts as a diagnostic extension to the score-prediction model, since after the trigger, the root-cause model determines what degradation mechanisms are likely present. The trigger is treated as an adopted mechanism rather than an evaluated module. Therefore, the experimental focus is on the diagnostic stage, namely whether batch-level quality signals computed by QFB are sufficient to predict degradation labels in a multi-cause setting. To understand the feature space, the QFB is discussed next.

Algorithm 2 Online operation: model-centric triggering and root-cause prediction

Require: Incoming batch B_t, reference snapshot R_t, trigger $h(\cdot)$, score-prediction threshold δ, trained multi-label classifier f_θ
Ensure: Predicted degradation labels $\hat{\mathbf{y}}_t$ if triggered; otherwise no output

$\hat{S}_{dev} \leftarrow h(B_t)$
if $\hat{S}_{dev} > \delta$ **then**
    $\mathbf{x}_t \leftarrow \phi(B_t, R_t)$
    $\hat{\mathbf{y}}_t \leftarrow f_\theta(\mathbf{x}_t)$
    return $\hat{\mathbf{y}}_t$
else
    return null
end if

3.2 Quality Feature Builder

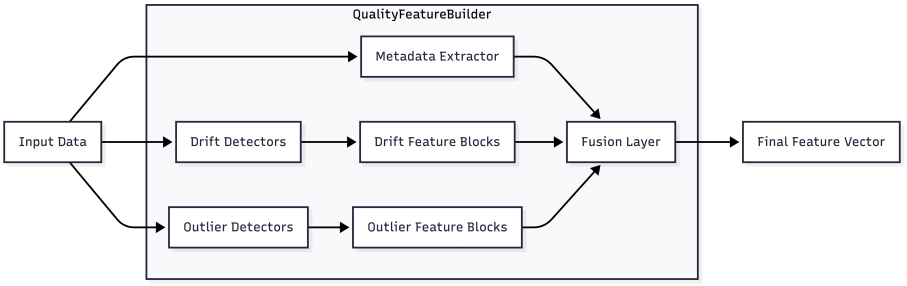


Figure 10: The design of the QFB, showing the different modules it consists of.

The Quality Feature Builder (QFB) (Figure 10) transforms each batch into a fixed-dimensional vector of batch-level quality signals. In the implementation used in this thesis, QFB is realized as a fusion layer (`QFBFusionLayer`) that concatenates three feature blocks:

- **Metadata features** computed by `MetadataExtractor`: profiling signals derived within the batch and summarized across columns.
- **Outlier features** computed by `OutlierFeatureBlock`: record-level anomaly scores summarized at batch level.
- **Drift features** computed by `DriftFeatureBlock`: reference-relative distribution change computed per column and summarized at batch level.

The root-cause model is positioned after the monitoring and prediction stages. Therefore, QFB intentionally excludes data-centric validation, such as type mismatches and constraint validation. Instead, QFB focuses on the construction of features that remain informative after data-centric validation has passed. This design is consistent with the literature and practices discussed in Section 2, where quality dimensions such as completeness and extreme values are utilized to assess data quality. Moreover, the overall architecture is aligned with metadata-driven monitoring approaches that utilize repeatedly computed profiling statistics as signals for technical data-quality degradation (Section 2.3).

3.2.1 Interface & Representation

Let X denote a batch (either an incoming batch B_t or a synthetic degraded batch $D_{t,k}$). QFB produces a feature vector

$$\phi(X, R_t) \in \mathbb{R}^d.$$

The metadata block is computed directly from X. The outlier and drift blocks require a reference snapshot as baseline, in this case R_t. To avoid repeated baseline preparation, QFB separates a one-time reference fitting step from per-batch feature computation:

- **Fit:** `QFB.fit`(R_t) prepares baseline-dependent objects. It fits the outlier detectors on R_t and calibrates detector-specific thresholds. In addition, it fits the drift detectors on R_t to define baseline distributions (for example, bin edges, reference moments, category supports, and classifier baselines).
- **Transform:** `QFB.build_features_for_batch`(X) computes and concatenates all block outputs into a single vector.

The exact emitted feature keys and batch-level summaries are documented in Appendix B.6–D.8. Therefore, if a certain feature block specifies (batch-level) summary features, these are specified in the appendix. Under a fixed QFB configuration and schema, the feature dimensionality d is fixed for all batches.

3.2.2 Metadata Block (`MetadataExtractor`)

The metadata block implements a statistical profiling approach. This means that for each statistic family, it first computes the statistic for every relevant column and retains those column-level values. In addition, it computes batch-level summaries by aggregating the column-level values across columns (for example, mean, maximum, or high-percentiles). A small number of statistics are computed directly at batch level (for example, row-level missingness rates). This design follows metadata-driven monitoring practices in which profiling statistics serve as indicators for degradation categories such as missingness, noise, outliers, inconsistencies, redundancy, and drift. The different features computed and their relation to the literature are presented below.

Completeness through missingness

The extractor computes missingness fractions per column, $m_j = \#\mathrm{NA}/n$, and retains these column-level values. It also aggregates $\{m_j\}$ across columns to produce batch-level completeness summaries (for example, mean, max, and a high percentile). In addition to column-wise missingness,

it computes row-level missingness rates. Missingness is a proxy indicator of incompleteness in data-quality taxonomies and is also a standard metric of monitoring approaches.

Lightweight numeric profiling

For numeric columns, the extractor computes per-column dispersion statistics such as standard deviation and range ($\max - \min$). In addition, it aggregates these values across numeric columns to form batch-level summaries. These low-order descriptors are simple and act as inexpensive proxies for changes in numeric spread that can be associated with degradations, such as noise injection, measurement changes, truncation, or outlier inflation

Categorical representation structure

For categorical columns, the extractor computes per-column statistics including distinct ratio (divided by n) and entropy. In addition, the extractor computes a batch-level dominance indicator as the fraction of categorical columns whose modal probability exceeds a high threshold.

Rare-category mass (feature enrichment)

Beyond entropy and dominance, the extractor computes a rare-category mass per categorical column. With $p_j(v) = \frac{n_j(v)}{n}$ and $\tau_{\text{rare}} \in [0,1]$, the rare-category mass is computed by

$$\text{RareMass}_j = \sum_v p_j(v)\,\mathbb{I}[p_j(v) < \tau_{\text{rare}}].$$

Since we want a rare-category, τ usually is a small value (for example, 0.03). The Metadata Extractor retains the resulting values per column, and then aggregates them across categorical columns into batch-level summaries. This statistic is included as a diagnostic add-on to capture cases where rare categories are more prevalent within the data. It is motivated by interpretability and in multi-cause settings. Together, these statistics characterize concentration, diversity, and irregularities.

Datetime profiling

For datetime columns that are successfully parsed, the extractor computes a per-column span in days ($\max - \min$), retains these column-level values, and aggregates spans across datetime columns. These features align with timeliness-oriented discussions in data-quality work, but are implemented here in a lightweight form compatible with batch-level monitoring.

3.2.3 Outlier feature block (`OutlierFeatureBlock`)

The outlier block captures how unusual each record is by combining the results of an ensemble of complementary detection methods. This ensemble design is motivated by robustness considerations: different detectors capture different deviation notions (marginal extremes, local density deviations, and multivariate structure), and combining them mitigates false detection under heterogeneous corruption mechanisms [57]. The detectors are fit on R_t and applied to the numeric projection of batch X.

Each configured outlier detector d produces an outlier score per record i:

$$s_d(i;X) \in \mathbb{R}.$$

where higher values indicate that a record is more likely to be an outlier. Since some detectors output scores where lower values indicate stronger outliers (e.g., LOF and Isolation Forest), we apply a monotone sign inversion so that for every detector d, higher scores consistently correspond to more outlier-like records.

For each detector, QFB aggregates the score distribution over records into batch-level summaries using mean, median, standard deviation, minimum, maximum, and selected quantiles.

Detector set

The detector set of outlier detectors combines marginal, density-based, and multivariate structure detectors.

First, Z-score and IQR-based scoring act as univariate baselines that primarily react to heavy-tailed deviations.

For a batch X, the Z-score for feature j is computed as $z_{ij} = \frac{x_{ij}-\mu_j}{\sigma_j}$, with a magnitude-based anomaly score $s_{ij}^{(Z)} = |z_{ij}|$. Missing feature values are ignored by assigning them zero contribution to the final detector score.

For IQR-based scoring, let Q_{1j} and Q_{3j} denote the first and third quartiles of feature j, define $\mathrm{IQR}_j = Q_{3j} - Q_{1j}$, and $k = 1.5$ as

$$L_j = Q_{1j} - k\,\mathrm{IQR}_j, \qquad U_j = Q_{3j} + k\,\mathrm{IQR}_j.$$

A value is traditionally flagged as anomalous if $x_{ij} < L_j$ or $x_{ij} > U_j$. Instead of a binary flag, we use a *soft exceedance* score that retains how far outside the fences a value falls. For each record i and feature j, define the normalized exceedance

$$e_{ij} = \frac{\max\{0,\, L_j - x_{ij},\, x_{ij} - U_j\}}{\mathrm{IQR}_j + \varepsilon},$$

where $\varepsilon > 0$ is a small constant to avoid division by zero when IQR_j is (near) zero. This score satisfies $e_{ij} = 0$ for inlier values ($L_j \le x_{ij} \le U_j$) and increases linearly with the distance to the nearest fence outside the admissible range, normalized by the feature scale. Missing feature values are excluded from the mean aggregation, and records with no observed numeric features receive score 0.

To obtain a batch-level signal, we aggregate exceedances per feature using the mean exceedance. For a batch $X = \{x_i\}_{i=1}^n$, the per-feature batch statistic is

$$\bar{e}_j(X) \;=\; \frac{1}{n}\sum_{i=1}^{n} e_{ij}.$$

Large values of $\bar{e}_j(X)$ indicate that feature j contains frequent and/or strong tail deviations relative to the reference distribution captured by R_t, while remaining comparable across features due to

the IQR normalization.

In addition, a robust covariance-based distance using the Minimum Covariance Determinant (MCD) [37] estimator is adopted. It approximates a robust Mahalanobis distance, which is sensitive to anomalous multivariate combinations under an approximately elliptical reference structure and has shown to be rather effective in outlier detection. The MCD works as follows:

Let $(\hat{\mu}_{\mathrm{MCD}}, \hat{\Sigma}_{\mathrm{MCD}})$ denote the robust location and scatter estimates obtained via MCD. The resulting robust distance is

$$D_{\mathrm{MCD}}(x) = \sqrt{(x-\hat{\mu}_{\mathrm{MCD}})^{\top}\hat{\Sigma}_{\mathrm{MCD}}^{-1}(x-\hat{\mu}_{\mathrm{MCD}})}.$$

Furthermore, Isolation Forest [46] provides a scalable multivariate detector that flags points that are easy to isolate under random partitioning, which is useful when corruption affects multiple numeric attributes jointly.

Finally, the Local Outlier Factor (LOF) [15] complements the detector set by focusing on local density deviations, capturing cases where records are atypical relative to their neighborhood within the vector space rather than globally extreme. Together, these detectors cover different outlier mechanisms and therefore provide a robust feature set.

Missing-value handling

Outlier detection is applied only to columns designated as numeric. Before detector-specific processing, these columns are coerced to numeric format. Values that cannot be parsed numerically are treated as missing. Missing-value handling is then detector-specific. For the univariate detectors, missing entries do not contribute to the anomaly score. Therefore, in the mean-aggregated IQR-based score only observed numeric features are included in the denominator. For the multivariate detectors that require complete inputs, missing values are imputed using medians.

Tail-shape & detector-agreement (feature enrichment)

Beyond standard score summaries, the outlier block derives tail-shape indicators such as a tail gap $(q99 - q95)$ and a tail ratio $q95/(\mathrm{med} + \varepsilon)$. Here, $\varepsilon > 0$ is a small constant included for numerical stability to avoid division by zero when the median score is close to zero.

In addition, It derives detector-agreement indicators by computing pairwise correlation between detector score vectors and summarizing the resulting agreement values. These diagnostics are included to support interpretability in multi-cause settings. Tail shape separates regimes with a few extreme anomalies from regimes with many moderate anomalies, while detector disagreement can indicate that different anomaly notions are being activated by different degradation mechanisms. These are treated as feature enrichment rather than standard data-quality indicators.

3.2.4 Drift feature block (`DriftFeatureBlock`)

The drift block quantifies how strongly X differs from R_t by computing drift scores per column and aggregating them to create batch-level features. Instead of relying on a single detector, the drift set utilizes a combination of drift detectors to improve robustness across drift mechanisms, which is a common approach in drift-detection practices [57].

Each configured drift detector m produces a drift score per column j:

$$d_j^{(m)}(X, R_t) \in \mathbb{R}.$$

Detector set & multivariate proxy

To obtain a robust variety of drift signals while keeping computation lightweight, the implementation compares each incoming batch B_t against the reference snapshot R_t using marginal hypothesis tests, divergence-based effect sizes, and a multivariate covariate-shift detector.

First, KS two-sample testing [9] and χ^2 testing [53] act as univariate baselines that flag distributional changes at the feature level for numeric and categorical/boolean variables, respectively.

Let $x^{(j)}$ denote feature j. For numeric features, with empirical CDFs $\hat{F}_{R_t}^{(j)}$ and $\hat{F}_{B_t}^{(j)}$, the KS statistic is

$$D_{\mathrm{KS}}^{(j)} = \sup_x \left| \hat{F}_{R_t}^{(j)}(x) - \hat{F}_{B_t}^{(j)}(x) \right|.$$

For categorical features with category set $\mathscr{C}_j$, let $p_{R_t}^{(j)}(c)$ be the baseline proportion in R_t let $n_{B_t}^{(j)}(c)$ be the count in B_t, and n_{B_t} be the batch length. A goodness-of-fit χ^2 statistic against the baseline is

$$\chi^{2(j)} = \sum_{c \in \mathscr{C}_j} \frac{\left(n_{B_t}^{(j)}(c) - n_{B_t}\, p_{R_t}^{(j)}(c) \right)^2}{n_{B_t}\, p_{R_t}^{(j)}(c)}.$$

In addition, effect-size measures quantify the magnitude of drift rather than only its statistical significance: Wasserstein distance is used for numeric features [52], while Jensen-Shannon divergence is used for categorical features [44].

For numeric features, the Wasserstein distance between the distributions can be written via the quantile functions $\hat{Q}_{R_t}^{(j)}$ and $\hat{Q}_{B_t}^{(j)}$ as

$$W_1^{(j)} = \int_0^1 \left| \hat{Q}_{R_t}^{(j)}(u) - \hat{Q}_{B_t}^{(j)}(u) \right| du.$$

For categorical/boolean features, let $p_{R_t}^{(j)}$ and $p_{B_t}^{(j)}$ be the empirical probability vectors over $\mathscr{C}_j$, and let $m^{(j)} = \frac{1}{2}(p_{R_t}^{(j)} + p_{B_t}^{(j)})$.

The Jensen–Shannon divergence is then calculated as

$$\mathrm{JS}^{(j)} = \frac{1}{2}\mathrm{KL}\left(p_{R_t}^{(j)} \,\|\, m^{(j)} \right) + \frac{1}{2}\mathrm{KL}\left(p_{B_t}^{(j)} \,\|\, m^{(j)} \right), \qquad \mathrm{KL}(p\|q) = \sum_{c \in \mathscr{C}_j} p(c) \log \frac{p(c)}{q(c)}.$$

Together, these detectors cover complementary drift mechanisms while remaining efficient to compute.

Aggregations over features

For each drift detector m, QFB computes nonnegative per-feature drift scores and aggregates them separately over numeric and categorical/boolean features. Let $s_m^{\mathrm{num}} = (s_{m,1}, \ldots, s_{m,d_{\mathrm{num}}})$ and

$s_m^{\text{cat}} = (s_{m,1}, \ldots, s_{m,d_{\text{cat}}})$ denote the resulting score vectors, where larger values indicate stronger drift. Within each vector, QFB uses a small set of complementary summaries that capture different drift patterns. The median $Q_{0.50}(s_m^{\text{num}})$ and $Q_{0.50}(s_m^{\text{cat}})$ reflect typical drift levels and are sensitive to diffuse, low-level shifts affecting many features. A high quantile, such as $Q_{0.90}(\cdot)$ (or $Q_{0.95}(\cdot)$), captures tail intensity and highlights localized strong drift without being dominated by a single extreme value.

Finally, to measure whether drift is concentrated in a small subset of features or spread across many, QFB computes a concentration index based on normalized scores $\tilde{s}_{m,j} = s_{m,j} / \left(\sum_\ell s_{m,\ell} + \varepsilon\right)$ as

$$\text{Conc}_m^{\text{num}} = \sum_{j=1}^{d_{\text{num}}} (\tilde{s}_{m,j})^2, \qquad \text{Conc}_m^{\text{cat}} = \sum_{j=1}^{d_{\text{cat}}} (\tilde{s}_{m,j})^2.$$

Together, these summaries separate diffuse drift from localized drift and retain diagnostic information on whether drift is driven primarily by numeric features or by categorical/boolean features, without requiring cross-detector score calibration.

3.2.5 Fusion Layer

Given the three blocks described above, the final QFB representation is the concatenation

$$\phi(X, R_t) = \left[\phi_{\text{meta}}(X),\ \phi_{\text{out}}(X; R_t),\ \phi_{\text{drift}}(X; R_t)\right],$$

where ϕ_{meta} is computed from X only, and ϕ_{out} and ϕ_{drift} use baseline fitting and/or reference comparison to R_t.

3.3 Synthetic Data Generator

The Synthetic Degradation Generator (SDG) is used to create controlled, labeled scenarios of data quality degradation for supervised model development and evaluation. Its purpose is to approximate realistic quality incidents in a reproducible manner by applying a set of predefined degradation operators. Given an input dataset (or batch) and a configuration that specifies which operators may be applied, the SDG outputs a degraded dataset $D_{t,k}$ together with a log that indicates which degradation types are applied.

3.3.1 Sampling perturbed cells

The perturbation of cells in a data batch works as follows. Let $S_{t,k}$ denote the sampled base batch drawn from R_t, with n rows and p columns. The SDG first computes column sampling weights $w_1, \ldots, w_p$:

$$w_j = \begin{cases} \frac{\text{FI}_j}{\sum_{j'} \text{FI}_{j'}}, & \text{if feature importance is provided and enabled,} \\ \frac{1}{p}, & \text{otherwise,} \end{cases}$$

where $\text{FI}_j \geq 0$ denotes the provided feature-importance weight for column j. Columns with zero weight are treated as inactive and are not selected for corruption. The feature importance focused perturbations make the SDG degradations dependent on the original pipeline model. This model

is more dependent on features with higher feature importance and therefore focusing on these cells results in more focused perturbations.

The generator then samples a set of cell indices without replacement. Let p_{active} be the number of active columns under a certain degradation strategy and $n_{\text{cells}} = n \cdot p_{\text{active}}$ the number of eligible cells. p_{active} prevents more cells than accessible to be perturbed. With target cell-perturbation rate $\rho \in (0,1]$, the number of perturbed cells is

$$n_{\text{target}} = \min\left(n_{\text{cells}}, \max(1, \text{round}(\rho \cdot n_{\text{cells}}))\right).$$

Cell-level sampling probabilities are defined by spreading each column's weight uniformly across its rows. Therefore, each cell in column j has probability proportional to w_j/n. The sampled set is converted into a mapping per column to apply corruptions efficiently.

3.3.2 Degradation operators and strategies

Each selected cell is modified according to a degradation strategy. The set of strategies is configurable and contains of the following modes (`strategy_mode`):

- `fixed`**:** a single configured strategy is used for all perturbed cells.
- `per_column_random`**:** for each corrupted column, a single strategy is assigned and the corrupted cells of that column are solely perturbed using that strategy.
- `per_cell_random`**:** for each corrupted cell, a strategy is sampled independently from the valid set of that column.

The degradation operators implemented in the SDG are:

Missingness (`missing`)

Replaces the cell value by a true missing value (NaN/NaT).

Placeholder missingness (`placeholder_missing`)

Replaces the value by a type-dependent sentinel (non-NaN): numeric → `missing_numeric_placeholder`, categorical → `missing_categorical_placeholder`, datetime → `missing_datetime_placeholder` (parsed to datetime). These are often observed in practice for manual entries of data values where the real value is unknown and chosen so they clearly differentiate from values of the distribution of the data.

Permutation within column (`permute`)

Replaces the value by another value sampled uniformly from the same column's observed values. This preserves the marginal value support but disrupts record-level consistency and feature relationships.

Gaussian noise (`gaussian_noise`, numeric only)

Adds noise $\varepsilon \sim \mathcal{N}(0, (\alpha\sigma_j)^2)$ to a numeric value, where σ_j is the column standard deviation and $\alpha =$ `noise_strength`:

$$x' = x + \varepsilon$$

Shift-and-scale distortion (`shift_scale`, numeric only)

Applies a combined scaling around the column mean and an additive shift proportional to the column standard deviation. For a value x in numeric column j, with column mean μ_j, column standard deviation σ_j, and sampled strength $s = \texttt{shift\_strength}$, the value-level transformation is

$$x' = (x - \mu_j)(1 + s) + \mu_j + s\,\sigma_j.$$

In the dataset-generation procedure, this operator is applied as a localized range distortion rather than a strictly single-cell perturbation: after selecting a primary cell, a local window width is randomly sampled from the interval specified by `shift_range_width` (expressed as a fraction of dataset length), and the same transformation is then propagated to nearby rows within that window in the same column.

Outlier injection (`outlier`, numeric only)

Moves a numeric value away from the column mean by sampling a large deviation whose scale is proportional to the column standard deviation σ_j. The scale is controlled by `outlier_strength`. To obtain a more robust variety of extreme values, the deviation is generated by one of three heavy-tailed mechanisms chosen at random:

- Gamma distribution with shape 2.0
- Gamma distribution with shape 0.5
- Student-t distribution with 3 degrees of freedom.

This produces both moderate and more extreme outliers on either side of the mean.

Category swap (`category_swap`, categorical only)

Replaces a category by a different category sampled from the same column's observed categories.

Datetime shift (`datetime_shift`, datetime only)

Applies a large day offset of magnitude `datetime_shift_days` with random sign:

$$t' = t \pm \Delta, \quad \Delta = \texttt{datetime\_shift\_days} \text{ days}$$

Datetime jitter (`datetime_jitter`, datetime only)

Applies small random temporal noise in days:

$$t' = t + \delta, \quad \delta \sim \mathcal{N}(0, \sigma_\delta^2),\ \sigma_\delta = \texttt{datetime\_jitter\_days\_std}$$

The degradation mechanisms were applied with predefined strength ranges to control the intensity of the injected corruptions. These ranges determine, for example, the magnitude of numeric noise, numeric shifts, injected outliers, and datetime perturbations. The final strength ranges used in the experiments are reported in Appendix F.

3.3.3 Label construction and reproducibility

Since each degraded batch is accompanied by a summary table (*column*, *strategy*, *# changed cells*, *affected-row fraction in that column*), the taxonomy labels are derived from the set of operators that are present in the summary table through a mapping from operators to quality types.

Let $\mathscr{L} = \{\text{Missingness}, \text{Noise/Measurement error}, \text{Outliers/Extreme values}, \text{Redundancy/Duplication}, \text{Distributional shift/Drift}\}$ denote the degradation label set (Table 1). We define a fixed mapping from SDG strategies to labels (Appendix Table G.11):

- `missing` and `placeholder_missing` $\mapsto$**Missingness**;
- `gaussian_noise`, and `datetime_jitter` $\mapsto$ **Noise / Measurement error**;
- `outlier` $\mapsto$ **Outliers / Extreme values**;
- `shift_scale`, `permute`, `category_swap`, and `datetime_shift` $\mapsto$ **Distributional shift / Drift**.

The label **Redundancy / Duplication** is part of the taxonomy, but is not activated by SDG because no row-level duplication operator is implemented, since this generally is spotted by upstream data-centric quality checks.

Given an instance with applied strategy set $\mathscr{S}_{t,k}$ (extracted from the summary), the multi-label target vector is constructed as

$$y_{t,k}^{(\ell)} = \begin{cases} 1, & \text{if } \exists s \in \mathscr{S}_{t,k} \text{ such that } \text{map}(s) = \ell, \\ 0, & \text{otherwise,} \end{cases} \qquad \ell \in \mathscr{L}.$$

All randomness (cell sampling, strategy sampling, and perturbation magnitudes) is controlled by `random_state`, ensuring reproducibility across runs when configuration and seed are fixed.

3.4 Data

This section describes the dataset used for evaluation and how it is partitioned into both a baseline period and a runtime period. The baseline period defines the reference snapshot and the runtime period provides the incoming batches.

3.4.1 Datasets & Preprocessing

For reproducibility purposes, the experiment utilizes a tabular dataset from a public source. The dataset is selected to reflect realistic, mixed-type schemas that occur in operational pipelines, including numerical, categorical, and date-time attributes. The dataset chosen for this experiment is the **Enterprise Fraud Detection Dataset** (downloaded from `https://www.kaggle.com/datasets/mohamedasak/enterprise-fraud-detection-dataset` on March 8, 2026). The dataset contains $200,000$ rows and 43 columns. After dropping derived variables (`hour, day, weekday, is_weekend, is_night`), the final dataset contains 38 columns. Usage is permitted under the dataset's stated license, Apache 2.0. A snippet of the dataset and its final dimensions

can be found in Appendix H. Every batch after application of the SDG and QFB transforms into a feature vector consisting of 210 features.

A key design choice is that the dataset is in complete form. Concretely, the dataset is chosen such that it does not contain missing values and does not require cleaning or filtering. This reflects the thesis assumption that the baseline data used to develop the original machine-learning pipeline has already undergone data preparation and acceptance checks, and is therefore suitable to serve as a stable reference snapshot. As a result, no dataset-specific cleaning, repair, or manual correction is performed prior to experimentation.

3.4.2 Batch Sampling & Reference Construction

Figure 11 illustrates the dataset partitioning strategy used throughout the experiments. For each dataset, the original data is split into a development or baseline period and a runtime period. The baseline period is used to construct the reference snapshot, while the runtime period is split up to simulate a stream of incoming batches.

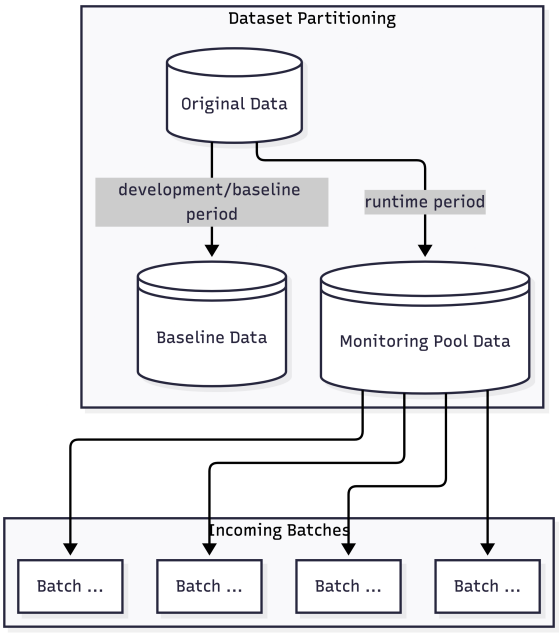


Figure 11: The Partitioning of the datasets.

Let R_t denote the reference dataset snapshot available at time t, as defined in Chapter 3 (80% of data). In the experimental setting, the reference is constructed from the baseline period and is treated as fixed for the duration of a run. The implications of updating R_t online, for example by incorporating accepted batches over time, are discussed later in Chapter 5.

Let $\{B_t\}_{t=1}^{T}$ denote the sequence of incoming batches drawn from the runtime period (20% of data). Each incoming batch B_t is formed by sampling n tuples from the runtime pool, where n is a fixed batch size of 128. Since the Quality Feature Builder computes batch-level ratios and normalized summary statistics, the extracted features are designed to be comparable across batches independent of the number of tuples. The schema, however, must remain consistent: incoming batches and the reference snapshot share the same set of columns so that per-column metrics and their aggregations are well-defined.

3.5 Experimental Setup

This section specifies the experimental protocol used to train and evaluate the proposed root-cause models. The setup is designed to support reproducible experiments, reduce unintended overlap between training and test instances, and evaluate performance for different factors.

3.5.1 Reproducibility Factors

All experiments are executed under controlled randomness. For each experimental run, the `random_state` determines:

- the batch sampling from the runtime pool,
- the sampling with replacement from the reference snapshot for synthetic generation,
- the degradation configuration randomness (for example, which cells or columns are perturbed),
- model initialization and training procedures where applicable.

To ensure there exists no overlap between training, validation, and test instances, the experiments use disjoint splits at two levels. First, the complete dataset is partitioned into the reference snapshot R_t (80% of data) and runtime pool (20% of data, X_{test}), as discussed in Section 3.4.2.

Thereafter, the reference snapshot R_t is partitioned into disjoint pools R_{train} and R_{val} using a fixed split ratio (80/20). Synthetic degraded batches used for training (X_{train}) and validating (X_{val}) are then generated by sampling disjoint batches from the corresponding pool. This ensures that identical base tuples are unlikely to appear across splits.

For each split, the Synthetic Degradation Generator is driven by disjoint seed sets. This is implemented, because degradations are randomized (for example, which cells or columns are perturbed and which strategy parameters are sampled). Additionally, disjoint seed sets further reduce the likelihood of identical degraded batches across splits, thereby reducing the chance of overfitting. The exact split specification, the seeds used for seed set randomization per setup, and the strength ranges of all detectors are recorded to enable full reproducibility (Appendix I).

Finally, Figure H.1 shows the distribution of label combinations in the training set after degradation simulation. The plot provides additional context by showing the distribution of the label combinations. The validation and test set showed similar distributions.

3.5.2 Model Families Development

As mentioned before, the root-cause predictor is trained as a supervised multi-label classifier that maps batch-level quality features to a set of degradation labels. Concretely, each incoming batch B_t is represented by a fixed-dimensional feature vector $x_t \in \mathbb{R}^d$ produced by the Quality Feature Builder. The learning target is a multi-label vector $y_t \in \{0,1\}^{|\mathscr{L}|}$ over the degradation taxonomy $\mathscr{L}$. This formulation supports the presence of multiple simultaneous degradations within the same batch and yields independent per-label probabilities in $[0,1]$. During inference, these probabilities are converted to binary predictions using a thresholding procedure described in Section 3.5.5.

Two model families are evaluated to reflect common trade-offs in tabular learning and classification. Gradient-Boosted Decision Trees (GBDTs) provide a strong baseline due to their ability to capture non-linear feature interactions and their strong empirical performance on heterogeneous tabular data. Furthermore, a Feed-Forward neural Network (FFNN) is utilized as a more flexible model that can learn higher-order decision boundaries on the same feature space.

For the FFNN, the final layer produces a vector of real-valued logits $z_t \in \mathbb{R}^{|\mathscr{L}|}$. These logits are mapped to per-label probabilities using an element-wise sigmoid function, given by

$$\hat{y}_{t,\ell} = \sigma(z_{t,\ell}) = \frac{1}{1+\exp(-z_{t,\ell})}, \quad \ell \in \mathscr{L}.$$

The FFNN is trained on X_{train} using a label-wise binary cross-entropy objective. During implementation, this is realized as binary cross-entropy on logits for stability. The label-wise binary cross-entropy loss on logits is formulated as

$$\mathscr{L}_{\text{BCEWithLogits}}(y_t, z_t) = \frac{1}{|\mathscr{L}|} \sum_{\ell \in \mathscr{L}} \Big(\max(z_{t,\ell}, 0) - z_{t,\ell}\, y_{t,\ell} + \log\big(1+\exp(-|z_{t,\ell}|)\big) \Big).$$

The weighted version to adapt to class imbalance of the BCE is not utilized, since we want to compare to the GBDT.

For the GBDT family, the multi-label problem is decomposed into $|\mathscr{L}|$ one-vs-rest binary classifiers, where each classifier is trained on X_{train} with a logistic objective. Each classifier outputs a probability score $\hat{y}_{t,\ell} \in [0,1]$ for its corresponding label, which is interpreted as the estimated likelihood that degradation type ℓ is present in batch B_t. The binary cross entropy loss objective for the GBDT is formulated as

$$\mathscr{L}_{\text{GBDT}}(y_t, \hat{\mathbf{p}}_t) = \frac{1}{|\mathscr{L}|} \sum_{\ell \in \mathscr{L}} -\Big(y_{t,\ell} \log(\hat{p}_{t,\ell}) + (1-y_{t,\ell}) \log(1-\hat{p}_{t,\ell}) \Big),$$

3.5.3 Hyperparameter Optimization and Model Selection

Hyperparameter tuning is performed separately for each model family using a fixed budget of randomized trials. In each trial, a candidate hyperparameter configuration is sampled from a predefined search space and the model is trained on X_{train}.

FFNN search space and regularization

For the FFNN family, each trial is trained for a maximum of 50 epochs. Model capacity is controlled by the number of hidden layers and hidden width, while generalization is controlled by dropout and weight decay. Dropout is included because the QFB feature representation contains correlated, partially redundant signals. Dropout reduces co-adaptation between hidden units and encourages the network to rely on distributed evidence across the quality feature space rather than memorizing specific feature combinations. The dropout rate p_{drop} is therefore treated as a tunable regularization strength and is sampled during random search. Weight decay is tuned alongside the learning rate to provide an additional smoothness constraint on the network parameters.

GBDT search space and regularization

For the GBDT family, XGBoost [25] is used as the representative implementation. The randomized search includes both capacity parameters and explicit regularization parameters. Tree depth (`max_depth`) directly controls interaction complexity. To prevent specific splits that fit to predictive noise (not the label), the search additionally tunes the minimum child weight (`min_child_weight`) and the minimum split gain (`gamma`).

Regularization is introduced via row subsampling (`subsample`) and feature subsampling (`colsample_bytree`), which reduces variance and improves robustness when the QFB representation contains correlated feature groups. Finally, L2 and L1 penalties on leaf weights (`reg_lambda` and `reg_alpha`) are tuned to shrink leaf values and reduce sensitivity to idiosyncratic training realizations. Learning rate (`eta`) and the number of boosting rounds (`n_estimators`) are tuned jointly to control the bias-variance trade-off, where smaller learning rates generally require more boosting rounds and often yield smoother decision functions.

Table 2: Random-search hyperparameter space used for model selection.

Model family	**Hyperparameters sampled per trial**
FFNN	Hidden layers $\in \{1,2,3\}$; hidden width $\in \{64,128,256,512\}$; dropout $p_{\text{drop}} \in [0,0.3]$; learning rate $\in [10^{-4},10^{-2}]$ (log-uniform); weight decay $\in [10^{-6},10^{-2}]$ (log-uniform); batch size $\in \{128,256,512\}$; max epochs $= 50$
XGBoost	`max_depth` $\in \{3,4,5,6,8,10\}$; `eta` $\in [0.01,0.3]$ (log-uniform); `n_estimators` $\in \{200,500,1000,2000\}$; `min_child_weight` $\in [1,50]$ (log-uniform); `gamma` $\in [0,10]$; `subsample` $\in [0.5,1.0]$; `colsample_bytree` $\in [0.5,1.0]$; `reg_lambda` $\in [10^{-6},10^{1}]$ (log-uniform); `reg_alpha` $\in [10^{-6},10^{1}]$ (log-uniform).

Both model families are trained using a probabilistic logistic loss for each label: the FFNN minimizes binary cross-entropy on logits, while XGBoost minimizes the equivalent Bernoulli log-loss via its logistic objective. Thereafter, the trained model is evaluated on X_{eval} using a selection criterion that considers the preference to penalize false negatives more than false positives. Therefore, the primary selection criterion is the validation macro-F_2 score, as it weights degradation labels equally and places more emphasis on recall than precision. This design choice makes that if a degradation type is not retrieved, this is seen more harmful than if a degradation type is marked as present, but is not actually present within the data batch.

3.5.4 Prevalence-Matched Random Baseline

To compare model performance to a guessing scheme, a prevalence-matched random baseline was constructed based on the marginal label distribution in the test set. For each label ℓ, the probability π_ℓ was estimated as its empirical frequency in the test set. Predictions were subsequently generated independently for each instance i and label ℓ according to

$$\hat{Y}_{i\ell} \sim \text{Bernoulli}(\pi_\ell). \tag{1}$$

Since the label probabilities were estimated from the test set, this baseline does not constitute a realistic predictive benchmark. Rather, it serves as a reference point for random guessing under the label prevalence. This makes it suitable as a comparative metric across models in this synthetic

approach, but not as a baseline for practical deployment. Finally, performance of this baseline is approximated using Monte Carlo simulation with $K = 2000$ to generate a confidence interval.

3.5.5 Multi-label Inference and Thresholding

The decision threshold used to convert predicted per-label scores into binary label decisions is treated as part of the selection pipeline.

Instead of a global threshold, label-specific thresholds are utilized. These label-specific thresholds $\{\tau_\ell\}_{\ell \in \mathscr{L}}$ account for differences in label prevalence and separability:

$$\hat{y}_i^{(\ell)} = \mathbb{I}\big[s_i^{(\ell)} \geq \tau_\ell\big], \qquad \forall \ell \in \mathscr{L}.$$

The search space, the number of trials, and the selected configurations are reported in Appendix J.

3.5.6 Evaluation Metrics

Performance is evaluated using multi-label classification metrics that capture multiple aspects of predictive quality. In the considered diagnostic setting, false negatives correspond to missed degradation types, which can delay remediation, whereas false positives correspond to spurious predicted degradation types, which can increase effort by directing investigation toward incorrect causes. The selected metrics therefore jointly quantify the trade-off between under-diagnosis and over-diagnosis, as well as exact versus partial recovery of the degradation label set.

Micro- and macro-averaged precision, recall, and F2

Let $\mathscr{L}$ denote the set of degradation labels. For each label $\ell \in \mathscr{L}$, let TP_ℓ, FP_ℓ, and FN_ℓ denote the number of true positives, false positives, and false negatives, respectively, accumulated over all evaluated batches.

Reporting both micro- and macro-averaged metrics is useful because label imbalance is expected in multi-label degradation diagnosis. Micro-averaging pools decisions over all $(\text{instance}, \text{label})$ pairs and therefore reflects aggregate performance under the observed label prevalence, meaning it is dominated by frequent degradation types and is informative for the expected operational mix. Macro-averaging computes performance per degradation type and then averages across labels so that each label contributes equally, ensuring that rare degradations are not masked by strong performance on common ones (in contrast to weighted macro-averaging, which weights labels by their support).

Micro-averaging

Micro-averaged precision and recall are computed by pooling counts across labels:

$$P_{\text{micro}} = \frac{\sum_{\ell \in \mathscr{L}} \mathrm{TP}_\ell}{\sum_{\ell \in \mathscr{L}} (\mathrm{TP}_\ell + \mathrm{FP}_\ell)}, \qquad R_{\text{micro}} = \frac{\sum_{\ell \in \mathscr{L}} \mathrm{TP}_\ell}{\sum_{\ell \in \mathscr{L}} (\mathrm{TP}_\ell + \mathrm{FN}_\ell)}.$$

The micro-averaged F2 score is then:

$$F2_{\text{micro}} = \frac{5\, P_{\text{micro}} R_{\text{micro}}}{4P_{\text{micro}} + R_{\text{micro}}}.$$

Macro-averaging

For macro-averaging, precision and recall are first computed per label:

$$P_\ell = \frac{\mathrm{TP}_\ell}{\mathrm{TP}_\ell + \mathrm{FP}_\ell}, \qquad R_\ell = \frac{\mathrm{TP}_\ell}{\mathrm{TP}_\ell + \mathrm{FN}_\ell},$$

and the per-label F2 score is:

$$F2_\ell = \frac{5 P_\ell R_\ell}{4P_\ell + R_\ell}.$$

Macro-averaged metrics are obtained by averaging over labels:

$$P_{\text{macro}} = \frac{1}{|\mathscr{L}|} \sum_{\ell \in \mathscr{L}} P_\ell, \qquad R_{\text{macro}} = \frac{1}{|\mathscr{L}|} \sum_{\ell \in \mathscr{L}} R_\ell, \qquad F2_{\text{macro}} = \frac{1}{|\mathscr{L}|} \sum_{\ell \in \mathscr{L}} F2_\ell.$$

Subset Accuracy

To assess per-batch correctness, subset accuracy is reported, defined as the fraction of instances for which the predicted label set exactly matches the ground-truth label set:

$$\text{SubsetAcc} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\big[\hat{\mathbf{y}}_i = \mathbf{y}_i\big].$$

Subset accuracy directly measures whether the model identifies the full set of present degradations for a batch. It is intentionally strict and typically decreases as the number of co-occurring labels increases, but it provides a clear indicator of fully correct, actionable diagnosis.

Hamming Loss

Hamming loss measures the average fraction of misclassified labels per instance:

$$\text{HammingLoss} = \frac{1}{N|\mathscr{L}|} \sum_{i=1}^{N} \sum_{\ell \in \mathscr{L}} \mathbb{I}\big[\hat{y}_i^{(\ell)} \neq y_i^{(\ell)}\big].$$

Unlike subset accuracy, Hamming loss provides partial credit when some, but not all, degradation labels are correctly predicted and can be interpreted as a label-wise error rate over $(\text{instance}, \text{label})$ decisions.

Mean Jaccard Similarity

To quantify partial set recovery at the instance level while penalizing spurious labels, we also report the mean Jaccard similarity (intersection-over-union) between the predicted and true label sets:

$$\text{Jaccard} = \frac{1}{N} \sum_{i=1}^{N} \frac{\big|\hat{\mathscr{Y}}_i \cap \mathscr{Y}_i\big|}{\big|\hat{\mathscr{Y}}_i \cup \mathscr{Y}_i\big|},$$

where $\mathscr{Y}_i = \{\ell \in \mathscr{L} : y_i^{(\ell)} = 1\}$ and $\hat{\mathscr{Y}}_i = \{\ell \in \mathscr{L} : \hat{y}_i^{(\ell)} = 1\}$. Mean Jaccard provides an interpretable middle ground between strict subset accuracy and per-label error rates by jointly penalizing missed labels and over-predicted labels in the recovered degradation set.

3.5.7 Experimental factors

Metrics are computed under different experimental settings to study how diagnostic performance changes under controlled scenario variations. The following experimental factors are varied using predefined value sets:

- **Cell perturbation technique**: per-cell random perturbations versus per-column random perturbations.
- **Perturbation range**: the percentage of the data on which degradation operator(s) are applied.
- **Number of co-occurring degradation labels**: the maximum number of degradation types injected simultaneously into a batch.

All other components of the pipeline are kept fixed. A fixed batch size of 128 rows is chosen for fast computation of degradation and a decent learning, validation and test space. The number of co-occurring degradation types per batch corresponds to the multi-cause scenarios discussed in Section 2.6. These terms are used interchangeably throughout this thesis.

To minimize overlap between generated batches used for training and validation, the Synthetic Degradation Generator is driven by disjoint seed sets for the training, validation and test splits, so that different realizations of degradations are observed in each split. Randomness within each split (for example, sampling of affected rows/columns and operator parameterization) is determined by these disjoint seed sets which are produced using a random number generator with master seed 20.

The concrete values used for each factor are listed in Table 3.

Table 3: Experimental factor grid used to define evaluation scenarios.

Factor	**Values**
Cell perturbation technique	{per_cell_random, per_column_random}
Max. co-occurring labels	$\{1, 2, 3\}$
Perturbation range	$\{(0, 0.5), (0.05, 0.3)\}$

3.5.8 Assumptions & limitations

In conclusion, this thesis evaluates an online, batch-based diagnostic framework for data quality issues. The experimental pipeline assumes that the incoming data stream can be represented as discrete batches B_t and that a reference snapshot R_t is available to represent accepted and stable data. In the experiments, this snapshot is constructed from a baseline period and is treated as clean and fixed within a run. To ensure that all quality features remain well-defined, the methodology assumes a consistent schema between R_t and each B_t, meaning that the same set of columns is available and can be processed using a combination of per-column quality signals and aggregated batch-level summaries.

A key methodological choice is that degradation labels are generated synthetically rather than obtained from historical incident logs. The framework therefore assumes that realistic training and evaluation can be approximated by injecting controlled corruptions into a clean dataset, after

which multi-label targets are derived deterministically from the applied corruption strategies. This enables systematic experimentation under known ground truth, but it also limits external validity. Injected corruptions may not fully capture existing degradation types that occur in production, and performance may differ when the baseline data is noisier, when labels are uncertain, or when failures are batch specific.

In addition, the triggering mechanism that determines whether diagnosis should be executed is treated as an adopted external step rather than a module evaluated in this thesis, meaning that the experiments start at the point where a batch is already flagged for further inspection.

Finally, since R_t is fixed within each run, the methodology does not study strategies for updating the reference snapshot over time as new accepted batches arrive, which would be an important consideration in long-running production deployments.

4 Results

This chapter evaluates whether the constructed metadata feature space contains sufficient signal for degradation-label classification. The analysis first compares the FFNN and XGBoost OvR models against the prevalence-matched baseline. It then examines how performance changes across perturbation rates, degradation labels, and co-occurrence levels. Finally, two robustness checks assess whether the main patterns depend on the perturbation range or the degradation application mode.

Unless stated otherwise, macro-F2 is used as the primary evaluation metric. The assessment of label-specific detection performance is performed using the F2-score.

4.1 Global overview of model performance

The evaluation begins at the broadest level by establishing whether the proposed feature representation yields meaningful predictive power at all. Thereafter, the training diagnostics and global label-wise results are considered in order to clarify how reliably the models predict certain degradation types before turning to the perturbation-based analyses.

4.1.1 Overall predictive performance and baseline comparison

Table 4 reports the overall results using macro-F2 as the primary aggregate evaluation metric. In addition, micro-F2, Hamming Loss, Mean Jaccard Similarity, and subset accuracy are reported to provide a complete view of multi-label predictive performance. The interpretation of these metrics can be found in Section 3.5.6

The results show that both models substantially outperform the prior baseline across all reported test metrics. On the primary evaluation metric, the prior baseline achieves a test macro-F2 of 0.498756, whereas the FFNN reaches 0.933116 and XGBoost OvR reaches 0.949628. A similar pattern is observed for micro-F2, where the prior baseline obtains 0.563327 compared with 0.950478 for the FFNN and 0.964292 for XGBoost OvR. The same pattern is reflected in Mean Jaccard Similarity and subset accuracy.

Between the two models, XGBoost OvR performs best on both the validation and test sets. In particular, it achieves the highest score across all metrics, while also yielding the lowest Hamming Loss. On the test set, the Hamming Loss decreases from 0.435622 for the prior baseline to 0.089850 for the FFNN and further to 0.065011 for XGBoost OvR. These results provide a clear first indication that the proposed feature representation yields meaningful predictive power, with XGBoost OvR showing superior performance.

These aggregate results establish that the method works in principle and that XGBoost OvR provides the strongest overall fit. The next question is whether this advantage is supported by equally reliable optimization behavior during training.

Table 4: Overall predictive performance of the prior baseline, FFNN, and XGBoost OvR models.

Metric		Prior baseline	FFNN	XGBoost OvR
Macro-F2	Val	–	0.934068	0.950608
	Test	0.498756	0.933116	0.949628
Micro-F2	Val	–	0.950871	0.964890
	Test	0.563327	0.950478	0.964292
Subset Accuracy	Val	–	0.707733	0.784600
	Test	0.098904	0.705021	0.781143
Hamming Loss	Val	–	0.089683	0.063900
	Test	0.435622	0.089850	0.065011
Mean Jaccard Similarity	Val	–	0.892017	0.922439
	Test	0.409908	0.890759	0.920989

4.1.2 Learning curves

Training diagnostics did not indicate equally favorable optimization behavior for both models. The learning curve figures are reported in Appendix J.2.

For the FFNN with the highest macro-F2 score, the learning curves suggest that training was not fully stable. The training loss continued to decrease, while the validation loss fluctuated substantially and showed no stable convergence pattern. This points to limited generalization during training and overfitting within the selected FFNN configuration. By contrast, the XGBoost OvR model showed a more regular optimization pattern, with training and validation log-loss decreasing smoothly and stabilizing over the boosting rounds, suggesting that the selected configuration achieved a more reliable fit.

In addition to the learning curves, computation times were recorded to provide a practical indication of the computational cost of the evaluated models. Here, XGBoost OvR shows a substantially higher learning time than the FFNN, but results in a higher macro-F2 score. Since these diagnostics are not within the primary scope of this thesis, the runtime information is reported in Appendix J.18.

This suggests that the stronger performance of XGBoost OvR is accompanied by a more stable optimization pattern. With this global comparison in place, the next subsection turns to how predictive performance is distributed across the individual degradation labels.

4.1.3 Global label-wise performance

This subsection presents the global label-wise predictive performance of the FFNN and XGBoost OvR models. The purpose of this overview is to identify which degradation labels are generally easier or harder to detect before turning to the more detailed analyses.

Figure 12 summarizes the label-wise F2 scores for both models. These results show how predictive performance is distributed across the degradation labels, and whether the models perform relatively consistently across labels or show stronger variation in detectability.

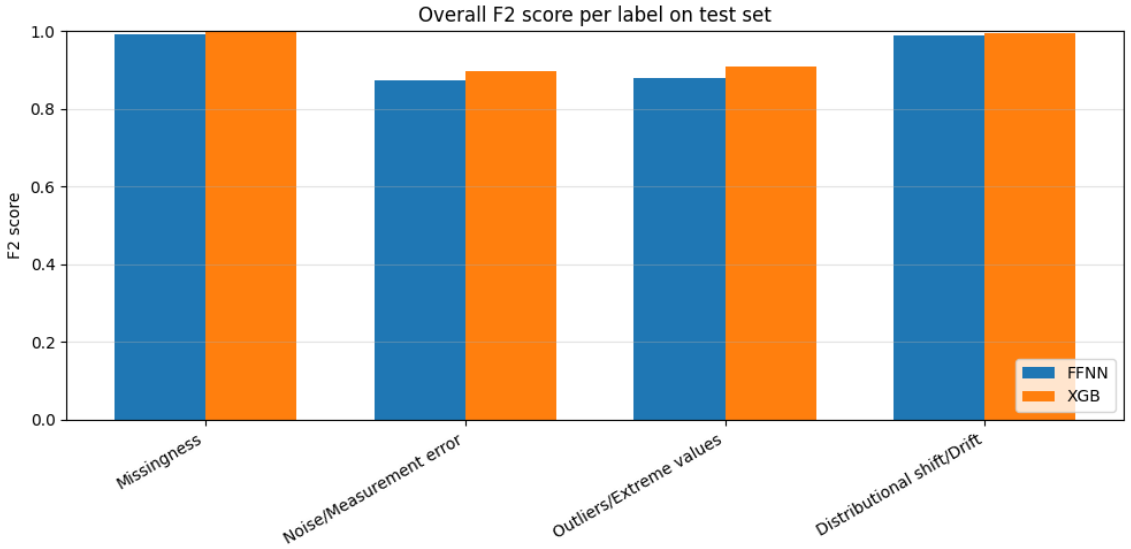


Figure 12: Overall label-wise F2 scores on the test set for the FFNN and XGBoost one-vs-rest models.

At a global level, the label-wise results show a consistent ordering across both models. For the FFNN, the strongest label-wise performance is observed for `Missingness` and `Distributional shift/Drift`, with F2 scores of 0.993419 and 0.988425, respectively. The weakest performance is observed for `Noise/Measurement error` and `Outliers/Extreme values`, with F2 scores of 0.872406 and 0.878212. For XGBoost OvR, the same ordering is observed. `Missingness` and `Distributional shift/Drift` again achieve the highest F2 scores, namely 0.999187 and 0.994390, while `Noise/Measurement error` and `Outliers/Extreme values` remain the weaker degradation types, with F2 scores of 0.897091 and 0.907845.

Overall, the label-wise overview shows that XGBoost OvR performs slightly better than the FFNN for all four degradation labels. Furthermore, the figure shows that `Missingness` and `Distributional shift/Drift` are the most readily detectable degradation types. In contrast to `Noise/Measurement error` and `Outliers/Extreme values`, which are more difficult to detect.

This global label-wise view already suggests that the models are not equally successful across degradation types. The next step is therefore to examine how these differences evolve across perturbation rates.

4.2 Performance across perturbation rates

After establishing the global performance advantage, this section examines whether that advantage remains stable across the perturbation spectrum. To achieve this, the analyses in this section move from macro-F2 to label-wise F2 and finally to failure regions. These analyses combined are able to uncover where and why detection becomes difficult.

4.2.1 Macro-F2 across perturbation rate

Figure 13 presents the macro-F2 performance of the FFNN and XGBoost OvR models across the perturbation rate, together with the prior baseline. The purpose of this comparison is to determine

how predictive performance changes as the degradation level increases and whether there are clear regions in which the models struggle or, conversely, begin to perform more consistently.

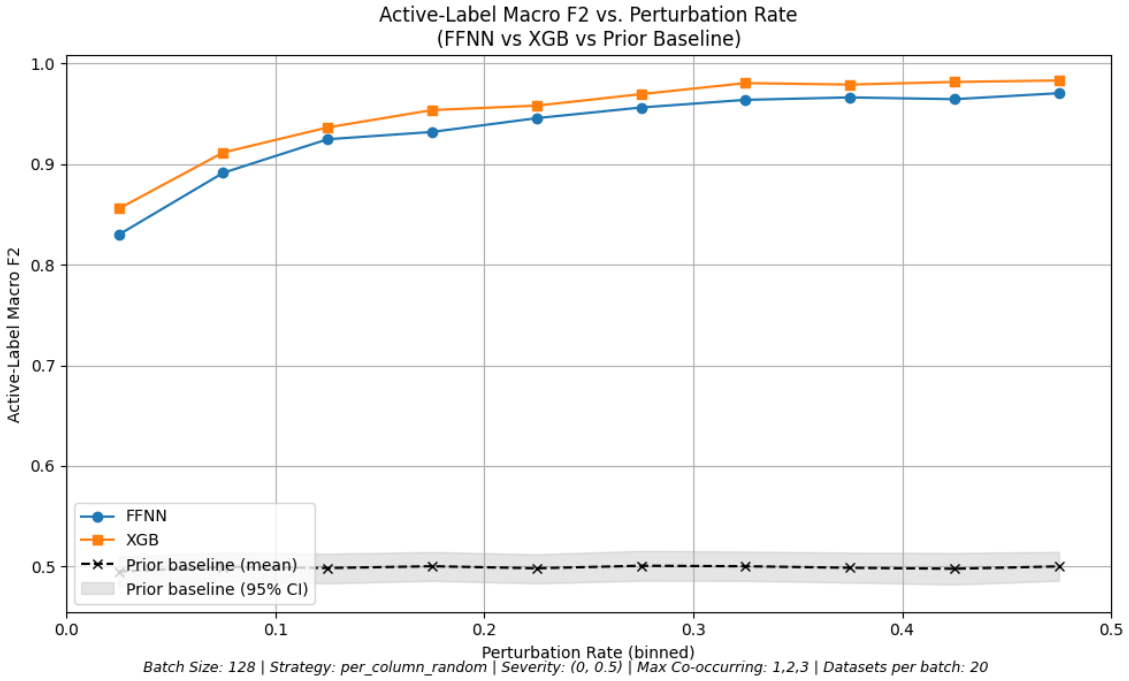


Figure 13: Macro-F2 across perturbation rates for the FFNN, XGBoost OvR, and the prior baseline. The shaded area denotes the 95% confidence interval of the prior baseline.

Across the full range, the macro-F2 results show a clear positive relationship between degradation rate and predictive performance for both models. Both models do outperform the prior baseline over the full perturbation range. In the lowest perturbation regions, the FFNN begins at a substantially lower macro-F2 than in the remainder of the spectrum, while XGBoost OvR also starts lower but already performs somewhat better than the FFNN. As degradation rate increases, both models improve steadily, after which their performance begins to stabilize at high rates of perturbation. Throughout the full spectrum, XGBoost OvR remains consistently above the FFNN.

These results indicate that predictive performance is not constant across the degradation spectrum. Instead, the figure shows that lower perturbation rates are substantially more difficult to detect than within higher regions. The aggregate pattern therefore makes clear that for lower perturbation rates, labels are substantially harder to predict than higher ones. To understand whether this difficulty is shared equally across degradation types, the analysis now turns to the label-wise results.

4.2.2 Label-wise F2 across perturbation rate

While the previous subsection considered aggregate performance, this analysis focuses on whether individual degradation labels follow similar or divergent trends across the spectrum.

Figure 14 summarizes the label-wise F2 results for both models. At a general level, the results show that the degradation rate affects the labels differently. `Missingness` and `Distributional shift/Drift` achieve strong performance already at low perturbation rates and remain close to ceiling across almost the full evaluated range. In contrast, `Noise/Measurement error` and

`Outliers/Extreme values` show substantially lower F2 scores in the lowest perturbation bins and improve more gradually as perturbation increases.

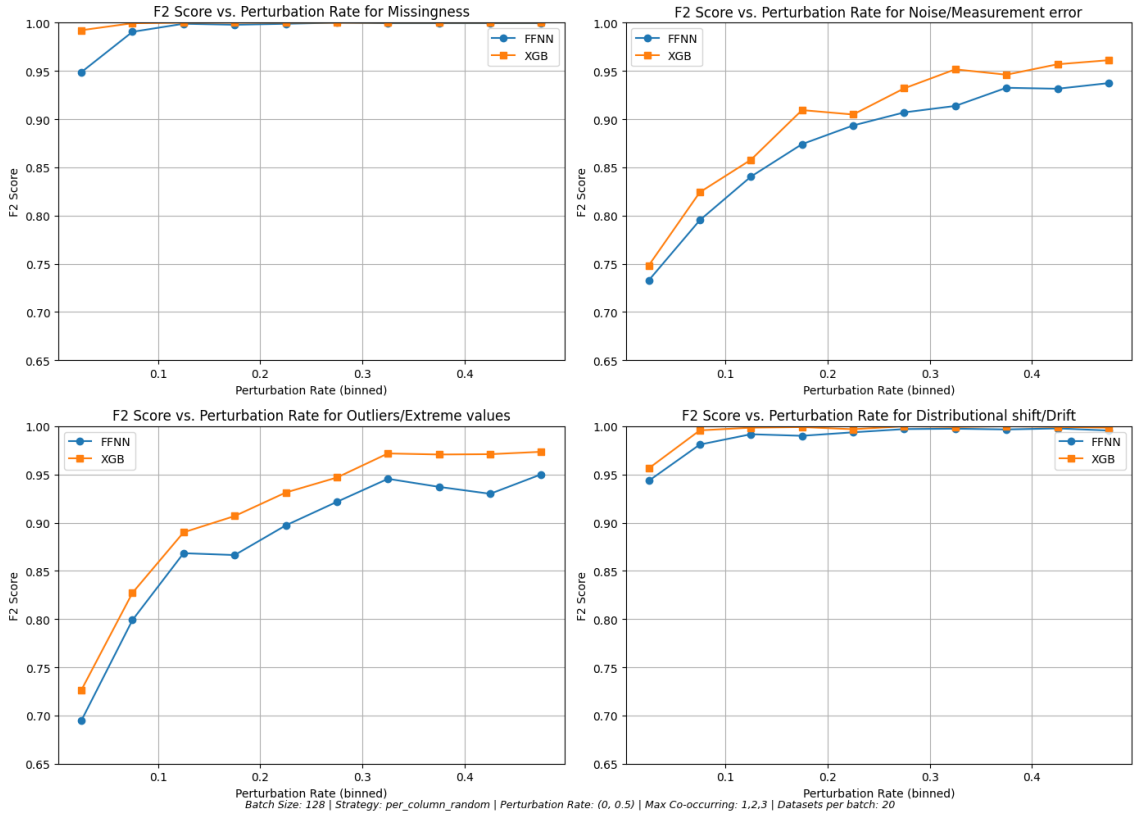


Figure 14: Label-wise F2 scores across perturbation rates for the FFNN and XGBoost one-vs-rest models.

For the FFNN, `Missingness` and `Distributional shift/Drift` are the best performing degradation types throughout almost the full spectrum. `Noise/Measurement error` and especially `Outliers/Extreme values` remain more difficult in the lower part of the spectrum, although both improve steadily once the perturbation rate increases.

For XGBoost OvR a similar ordering is observed, but with consistently stronger results for the more difficult degradation types. In particular, XGBoost OvR reaches higher F2 scores earlier in the spectrum for `Noise/Measurement error` and `Outliers/Extreme values`, while performance for `Missingness` and `Distributional shift/Drift` remains near perfect throughout.

Therefore, these results show that degradation rates affect performance differently per label. Labels such as `Missingness` and `Distributional shift/Drift` remain highly detectable even at lower perturbation rates, whereas `Noise/Measurement error` and `Outliers/Extreme values`, require more pronounced degradation before strong predictive performance is achieved.

To identify more precisely where these difficulties emerge, the next subsection examines the failure regions and the points at which model behavior begins to improve more clearly.

4.2.3 Failure regions and turning points

The previous analyses show that errors concentrate at low perturbation rates. This subsection identifies the main failure regions and the perturbation levels at which performance begins to

stabilize.

In support of the previous figure, Figures 15 and 16 examine model behavior from a false predictive perspective. The first figure shows the false-positive rate as a function of perturbation rate for each degradation type, while the second shows the corresponding recall obtained under a fixed false-positive constraint of at most 5%.

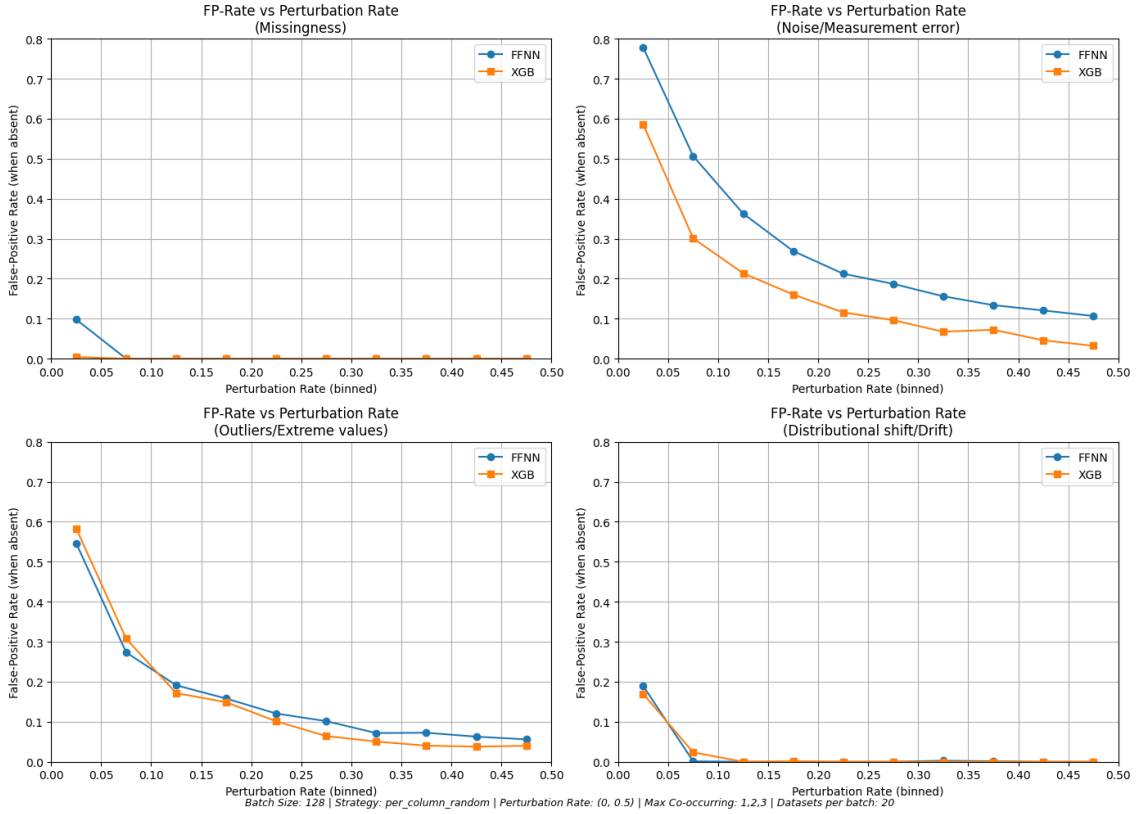


Figure 15: False-positive rate as a function of perturbation rate for the four degradation types.

A consistent pattern across both figures is that the lowest perturbation region forms the most difficult region of the classification task. In this region, false-positive rates are highest and recall under the 5% false-positive constraint is lowest, particularly for `Noise/Measurement error` and `Outliers/Extreme values`. This indicates that weak degradations of these types do not produce sufficiently distinct patterns to allow reliable classification.

For `Missingness` and `Distributional shift/Drift`, the situation is considerably different. In both models, the false-positive rates for these degradation types are already very low in the lower perturbation bins, and recall under the 5% false-positive constraint remains high across almost the full degradation spectrum. This suggests that these degradation types produce clear signatures even when the perturbation rate is limited.

The most pronounced failure region is therefore observed for `Noise/Measurement error` and `Outliers/Extreme values` at the lowest perturbation rates. For these degradation types, the FFNN shows especially high false-positive rates in the first bins, whereas XGBoost OvR reaches lower false-positive rates and higher constrained recall earlier in the spectrum. As the perturbation rate increases, both models improve substantially, indicating that stronger degradations provide a clearer signal.

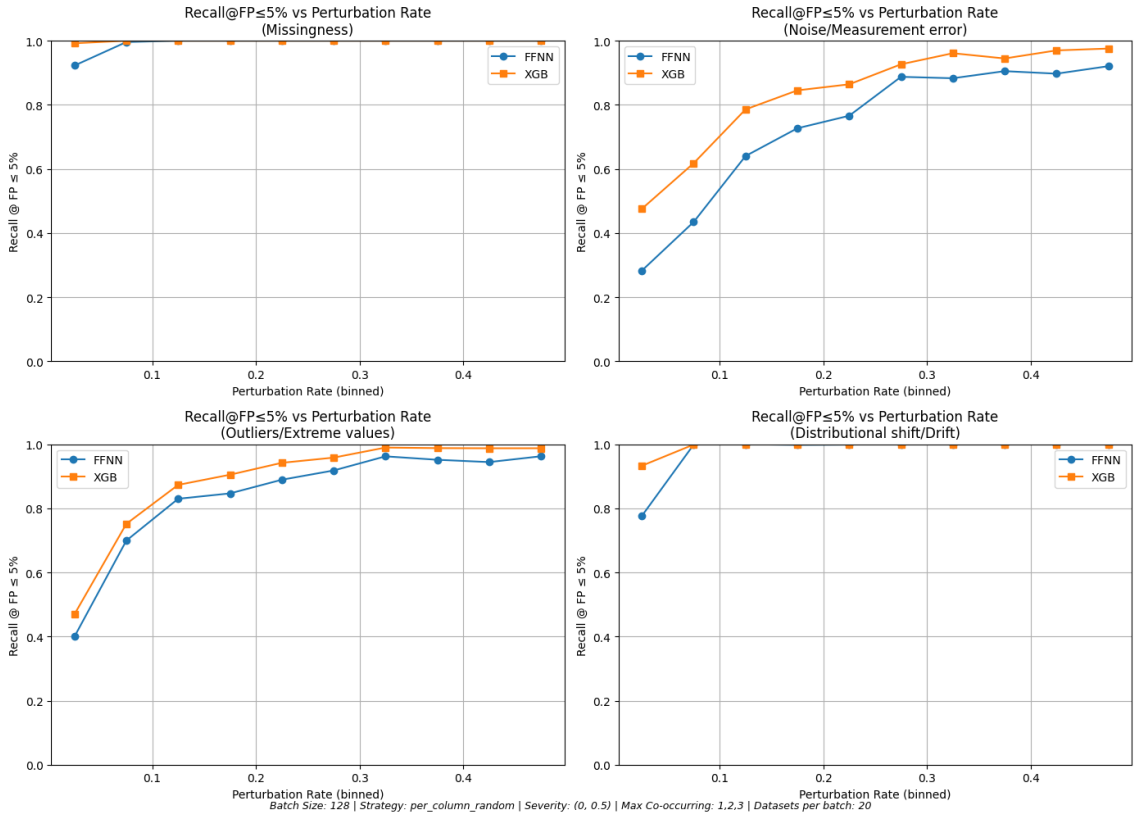


Figure 16: Recall at a false-positive rate of at most 5% as a function of perturbation rate for the four degradation types.

A first turning point appears to emerge once the perturbation rate moves beyond the lowest region, roughly from about 0.10 onward, where both false-positive rates decrease substantially and recall begins to rise. For `Noise/Measurement error` and `Outliers/Extreme values`, a more stable region appears from approximately 0.20 to 0.30 onward, where the reduction in false positives is accompanied by a substantial increase in recall. By contrast, `Missingness` and `Distributional shift/Drift` appear to operate in a stable region much earlier in the spectrum.

Taken together, these findings show that at the lowest perturbation level resides the most challenging part of the classification task and false positive rates are high. They also provide the first empirical basis for examining whether excluding the weakest region from the training design may lead to more predictive power in the higher regions.

These results identify the lowest perturbation region as the main failure zone for both models, especially for noise- and outlier-related degradations. With this aggregated perturbation pattern established, the next section considers how the predictive power is distributed with regards to co-occurrence of multiple degradation labels.

4.3 Performance across perturbation rates for different numbers of co-occurring labels

The previous section showed that perturbation rate strongly affects predictive performance. Therefore, the next step is to examine how this relationship is distributed around co-occurrence. By separating the results according to the number of co-occurring labels, this section assesses to what

extent increasing degradation complexity makes the classification task more difficult, and whether this effect is distributed uniformly across degradation types. For completeness, the distribution of label combinations in the training set is shown in Appendix H.1, whereas the validation and test sets showed similar distributions.

To study this interaction, the following figures summarize label-wise heatmaps across perturbation rates and co-occurrence levels. In these figures, degradation rates are shown on the x-axis and the number of co-occurring labels on the y-axis, while the heatmap values represent the F2 score of each individual degradation label. This makes it possible to assess whether the effect of co-occurrence is distributed uniformly across labels.

Figure 17 shows that `Missingness` is highly detectable across almost all perturbation bins and co-occurrence settings. For both models, the F2 scores remain close to perfect throughout most of the evaluated range, with only limited variation across co-occurrence levels. This suggests that missingness produces a comparatively clear and robust signal, even when additional degradations are present simultaneously.

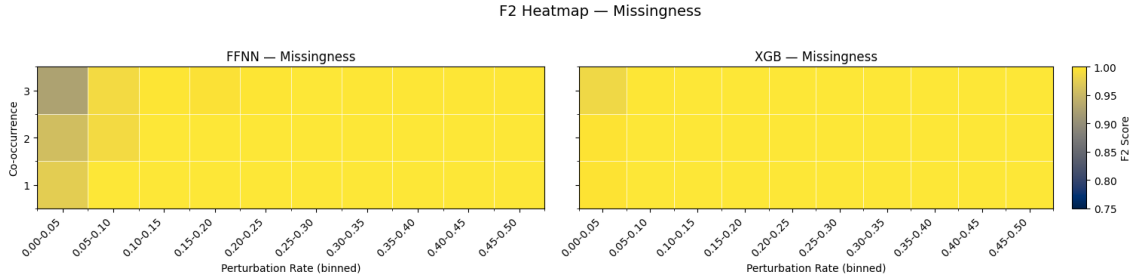


Figure 17: Label-wise F2 heatmap across perturbation rates and co-occurrence for `Missingness`.

Nevertheless, this is not the case for `Noise/Measurement error`, where a different pattern emerges (Figure 18). The lowest perturbation regions form the most difficult region, and performance increases markedly once the degradation becomes more pronounced. Compared to `Missingness`, this degradation type is more sensitive to both perturbation rate and co-occurrence. XGBoost OvR and FFNN show broadly similar co-occurrence patterns. However, for single label predictions, XGBoost OvR converges faster to high F2 scores than FFNN as the perturbation rate increases.

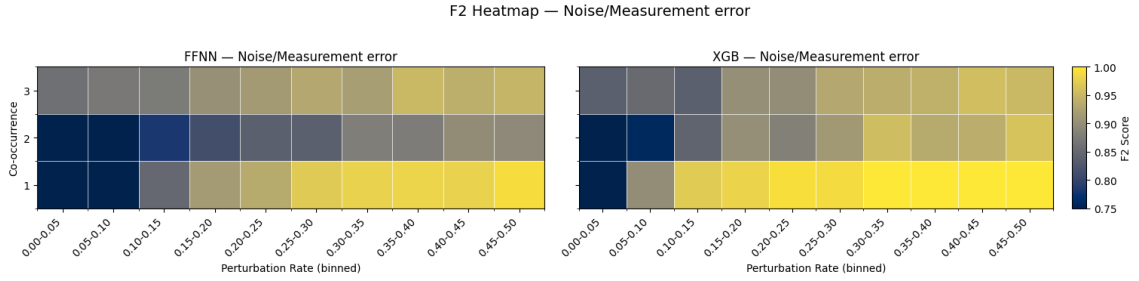


Figure 18: Label-wise F2 heatmap across perturbation rates and co-occurrence for `Noise/Measurement error`.

Figure 19 presents the results for `Outliers/Extreme values`. As with noise, performance is clearly weaker when multiple degradation labels co-occur. As perturbation increases, the F2

scores improve substantially for both models. This suggests that weaker outlier patterns are more difficult to separate from the surrounding signal, whereas stronger outlier degradations become progressively easier to detect. Across this figure, XGBoost OvR appears more stable and more robust than the FFNN, especially under higher co-occurrence.

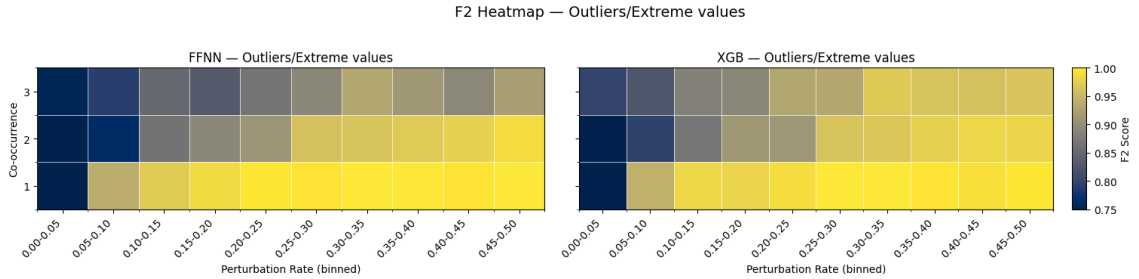


Figure 19: Label-wise F2 heatmap across perturbation rates and co-occurrence for `Outliers/Extreme values`.

Figure 20 shows that `Distributional shift/Drift` is detected well for all co-occurrences across almost the full perturbation spectrum. Even in the lower bins, the F2 score remains close to perfect across the different co-occurrence settings. This indicates that distributional drift produces relatively distinct patterns in the feature space and is therefore less sensitive to increasing co-occurrence complexity than the other degradation types.

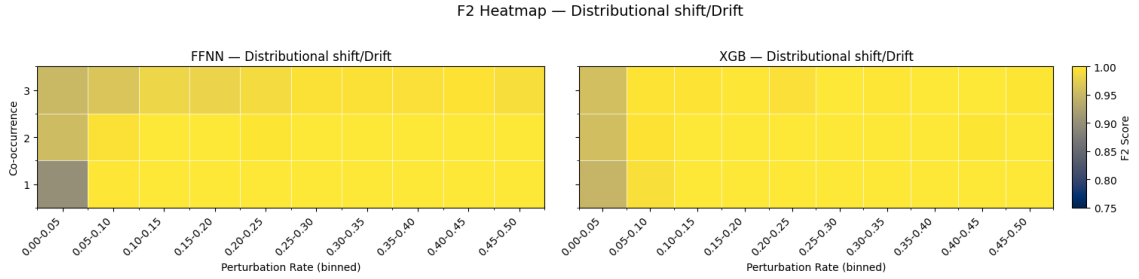


Figure 20: Label-wise F2 heatmap across perturbation rates and co-occurrence for `Distributional shift/Drift`.

Taken together, the label-wise heatmaps show that the interaction between degradation rates and co-occurrence is not uniform across degradation types. `Missingness` and `Distributional shift/Drift` remain highly detectable across almost the full spectrum and different co-occurrence settings, whereas `Noise/Measurement error` and `Outliers/Extreme values` are more visibly affected by increasing co-occurrence.

Overall, these findings show that increasing the number of co-occurring degradation labels constitutes a substantial source of difficulty for both models. For completeness, the corresponding aggregate macro-F2 comparisons across co-occurrence settings are reported in Appendix L.7. While stronger degradations generally improve detectability, this improvement is reduced when multiple degradation labels are present simultaneously. High co-occurrence therefore emerges as a major source of difficulty, besides low perturbation rates. After exploring the explainability of the XGBoost OvR model, the remaining sections examine alternative experimental designs to examine model performance under different design scenarios.

4.4 Interpretability of the XGBoost OvR model

To provide insight into the strongest-performing model, an interpretability analysis was conducted for the XGBoost one-vs-rest classifier. The purpose of this analysis is not to establish causal explanations, but to assess whether the feature reliance of the labels is consistent with their intended design. Since each degradation label is modeled by a separate binary classifier, feature contributions can be examined at the label level. Both XGBoost gain-based feature importance (FI) and SHAP were used to assess whether decision patterns are consistent with the design of the feature space.

Overall, both SHAP and feature importance show that the model relies on a combination of profiling-based, drift-related, and outlier-related metadata features. This is consistent with the construction of the Quality Feature Builder. However, it is clear that SHAP and FI come to different conclusions for which features are the most important for the model. This is plausible, since both methods capture different notions of importance. Gain-based importance reflects the usefulness of features in the tree-building process, whereas SHAP reflects their contribution to the final model predictions. Differences between both rankings may therefore arise from feature correlation and interaction effects. Nevertheless, for `Missingness` both methods agree with the intended feature design, with missingness-related features solely deciding on its presence. In contrast, the other labels contain a combination of all feature groups. The label-specific top-20 feature rankings for both methods are reported in Appendix K.

4.5 Alternative design: restricted perturbation-range

Because the previous analyses identified the lowest perturbation region as the weakest part of the spectrum, an additional design is considered in which the model is trained on a narrower perturbation interval. The aim is not to replace the main approach, but to examine whether excluding part of the weakest and most inflated regions yields a more informative and potentially more honest problem setting.

4.5.1 Motivation for the restricted range

The motivation for this additional experiment is that the lowest perturbation levels $(0, 0.1)$ may provide only weak degradation signal, thereby reducing the separability of the target labels. However, since we do not want to exclude these harder to detect cases and want the model to be able to predict cases with low degradation rates, this range is not excluded entirely. In addition, the high perturbation region $(0.3, 0.5)$ may inflate overall macro-F2 score and is generally less likely to be observed in real-world cases.

Therefore, the design is evaluated in which a part of the lower and upper regions are excluded from training. More specifically, these bounds are set such that the restricted-range model is trained on the range $(0.05, 0.3)$. The purpose of this design is to examine whether removing the weakest and inflated degradation cases improves overall learnability in the remaining perturbation range and yields a more honest assessment of the model.

4.5.2 Performance of restricted model

Across the perturbation range $(0.05, 0.3)$, the FFNN and XGBoost OvR again as expected outperform the baseline. Furthermore, within the restricted range, macro-F2 increases as perturbation rate increases, which is the same relationship observed in the main approach.

Relative to the main approach, however, the restricted-range design does not lead to a significant improvement. For the FFNN, macro-F2 is consistently lower than in the corresponding perturbation regions of the full-range model. This suggests that restricting the training range does not improve learnability for the FFNN within these smaller regions, but instead leads to weaker predictive performance across this interval. In contrast, for XGBoost OvR, the results are much more stable. Its macro-F2 remains close to that of the main approach across the retained range, with at most only limited improvement in some bins. Thus, for XGBoost OvR, the restricted-range design appears to preserve performance rather than improve it. This pattern is also reflected in the aggregate metrics and other figures.

Overall, the restricted-range experiment suggests that excluding the weakest and strongest perturbation regions does not produce a generally more learnable problem setting. The FFNN model deteriorates relative to the main approach, and XGBoost OvR shows only limited gains on secondary metrics. All plots and metrics are presented in Appendix M.

4.6 Effect of degradation application mode

A second robustness question concerns the way degradations are applied within a batch. Up to this point, the analyses have relied on the default `per_column_random` setting, in which each column receives at most one degradation. This section therefore considers whether the empirical patterns observed so far remain similar when the alternative `per_cell_random` setting is used instead.

Overall, the `per_cell_random` setting shows the same pattern as the main approach, where predictive performance remains lowest in the lower perturbation region and improves as perturbation rates increase. In addition, the overall ordering of predictive difficulty is the same. `Missingness` and `Distributional shift/Drift` remain the most readily detectable labels, whereas `Noise/Measurement error` and `Outliers/Extreme values` remain more difficult.

At the same time, the pattern differs from the main approach in several respects. Most notably, the co-occurrence effect becomes less clear. Under the default `per_column_random` design, higher co-occurrence generally made degradation detection more difficult. Under `per_cell_random`, this pattern is much weaker. In several lower-perturbation regions, higher co-occurrence does not reduce performance and may even result in slightly stronger label-wise F2. This suggests that the degradation approach affects the way co-occurrence interacts with perturbation rate.

Relative to the main approach, the effect of `per_cell_random` is again model-dependent. For the FFNN, the results are weaker than under the default `per_column_random` setting. For XGBoost OvR, the opposite pattern is observed, with performance remaining stable or improving slightly. In addition, the figures show that XGBoost OvR has a clearly higher false-positive rate for `Outliers/Extreme values` than the FFNN, whereas these rates were more similar in the

main approach. By contrast, for `Noise/Measurement error`, XGBoost OvR yields a substantially lower false-positive rate than the FFNN while also achieving clearly higher recall. This suggests that changing the degradation application regime makes the task somewhat harder for the FFNN, but does not reduce performance for XGBoost OvR.

Overall, this additional experiment suggests that the degradation application mode does influence the perturbation-performance relationship. Lower perturbation rates remain the most difficult region, but the co-occurrence pattern becomes less consistently adverse than in the main approach. In addition, whereas the FFNN decreases in performance under `per_cell_random`, XGBoost OvR remains comparatively stable. The complete results for this approach are reported in Appendix N.

4.7 Summary of findings

To conclude, across the global results, both models substantially outperform the prior baseline, indicating that the proposed feature representation provides meaningful predictive signal beyond label prevalence alone. Among the two models, XGBoost OvR consistently performs best. This final section brings these findings together and clarifies which patterns remain stable across the additional design experiments.

The main finding of the results is that perturbation rate strongly affects predictive performance. Across the main analyses, performance is weakest in the lowest perturbation region, particularly below approximately 0.10, and improves steadily as perturbation rates increase. This indicates that stronger degradations provide more distinguishable signal for the classification task. At the label level, `Missingness` and `Distributional shift/Drift` are more easily detectable degradation types, whereas `Noise/Measurement error` and `Outliers/Extreme values` remain more difficult.

In addition, the analyses across different numbers of co-occurring labels show that increasing degradation complexity generally reduces predictive performance in the main approach. This effect is most visible in the lower perturbation region and is strongest for `Noise/Measurement error` and `Outliers/Extreme values`. By contrast, `Missingness` and `Distributional shift/Drift` remain comparatively robust under co-occurrence.

The additional experiments further indicate that the observed perturbation-performance relationship is not independent of design choices. For the FFNN model, performance deteriorates relative to the main approach, whereas XGBoost OvR remains stable and shows at most limited gains on secondary metrics. Similarly, the same pattern emerged for the `per_cell_random` experiment. However, in the second experiment, the negative effect of higher co-occurrence becomes less consistently visible under `per_cell_random` than in the main approach.

Overall, the results show that the proposed method is most effective when perturbations are sufficiently pronounced and when the number of co-occurring degradation labels remains limited. However, the findings support the viability of metadata-driven root-cause diagnosis in a simulated environment, while also showing that performance depends strongly on the visibility of the degradation signal and on the model selected.

5 Discussion

This chapter reflects on the broader meaning of the findings and places them in a practical context. Therefore, it considers how the results should be interpreted, which limitations qualify them, and which directions for future research may be taken.

5.1 Interpretation of the findings

The results indicate that batch-level metadata can support diagnostic reasoning after a performance-related alert has been raised. In this framework, metadata is not only used for descriptive monitoring, but also as a supervised signal for identifying likely degradation mechanisms. However, the diagnostic value of this signal depends strongly on whether the degradation leaves a visible signature in the metadata representation. Missingness and distributional shift produce relatively stable and separable patterns, whereas noise and outlier-related degradations are less reliably detected at low perturbation rates and under higher co-occurrence. This shows that metadata-driven diagnosis is not equally effective across all degradation conditions, but depends on the extent to which degradation effects become observable in aggregated batch characteristics.

Furthermore, diagnostic difficulty is not uniform across degradation mechanisms. The results indicate that some degradation types generate more stable and distinguishable metadata patterns than others. This is important because it shows that the effectiveness of the framework does not only depend on the model family or the training procedure, but also on the degree to which a degradation mechanism is naturally reflected in profiling statistics, drift indicators, and anomaly-related signals. The feature representation is therefore informative, but not equally expressive for every type of data-quality problem. The results show that the diagnostic difficulty also increases when multiple degradation mechanisms occur simultaneously. Although the magnitude of this difficulty depends on the simulation design and on the model used, this reinforces the importance of treating the problem as a multi-label task rather than as a simplified single-cause classification problem.

Finally, model family choice has a substantial influence on performance. The empirical comparison suggests that the current feature representation is more naturally aligned with tree-based learning than with the evaluated neural architecture. This indicates that successful metadata-driven diagnosis depends not only on the features computed, but also whether the chosen model family is able to exploit the structure of the feature space and is robust to different degradation applications, such as `per_column_random` versus `per_cell_random`.

Taken together, these findings suggest that the metadata-driven approach is a viable direction for root-cause diagnosis.

5.2 Methodological limitations

The most important limitation of this thesis concerns external validity. The framework is evaluated in a controlled synthetic setting in which degradation mechanisms are known by construction and labels are generated directly from the applied corruption strategies. This is a methodological strength for proof-of-concept evaluation, because it enables precise experimentation under

known ground truth. While preventive measures have been taken to reduce overfitting, the synthetic learning problem is more structured than would be the case in operational pipelines.

In addition, a limitation of the current research concerns the empirical scope of the evaluation. The framework is tested on one tabular dataset. This limits the extent to which the findings can be generalized. Some of the observed performance patterns may depend partly or heavily on the statistical structure of the chosen dataset and feature construction, since these are the source for prediction.

Moreover, a third limitation concerns the feature representation itself. Although the Quality Feature Builder performs well overall, the current evaluation does not focus on the contribution of its individual components. This is partially achieved through the interpretability step, but does not fully establish which feature groups are essential and which may be partly redundant. The thesis only shows that the full representation is useful.

Besides, there exists a limitation concerning the model comparison. While only two model families were evaluated, this is sufficient to show that model choice matters. However, it does not justify broad claims about the relative merit of all possible approaches. The present results should therefore be interpreted as evidence about the current framework rather than as a definitive comparison between all possible model families.

Furthermore, another limitation is that the online triggering mechanism is assumed rather than evaluated. The framework is positioned as a diagnostic extension that operates after the model-centric alert has already indicated likely underperformance. This is a sensible scope choice, but it also means that the thesis does not assess the behavior of the full end-to-end monitoring process. A full monitoring evaluation could support with the hard to detect low regions, where the root-cause model struggles, since the trigger for a robust model would not generally fire at small deviations in the underlying data. In addition, with a model-centric trigger, feature importance of the original model could be parsed to provide model focused degradation.

Finally, the reference snapshot remains fixed within each experimental run. This simplifies the evaluation and makes comparisons between batches well defined, but it does not reflect the full reality of long-running production systems, where accepted data may need to be incorporated into the reference over time.

Taken together, these limitations do not undermine the central contribution of the thesis, but they do define the boundaries within which the findings should be interpreted.

5.3 Practical implications

Despite these limitations, the findings do have practical implications. In particular, the proposed framework may help narrow the search space after a performance-related alarm by predicting degradation rather than providing only an indication of deviation.

The results further suggest that the approach is most useful when degradation signals are sufficiently visible in the metadata space and when the selected model family is able to exploit the resulting feature structure reliably. This implies that the framework probably is valuable as a diagnostic-support rather than as a fully autonomous root-cause system.

At the same time, the results caution against assuming equal diagnostic reliability across all settings. In practice, a metadata-driven diagnostic model would likely be more trustworthy under some degradation conditions than under others. Through the usage of macro-F2 scores as optimization, the model focuses on diminishing false negatives, which results in a higher degree of false positives, especially in low perturbation regions. This suggests that any real deployment should be accompanied by human oversight and predictions should not be interpreted as ground-truth, but rather as a focus point of investigation.

Overall, the findings support the idea that model-centric monitoring can be made more actionable when it is linked to batch-level diagnostic reasoning. This is relevant for machine-learning operations, where the main challenge is often not merely to detect that a problem exists, but to determine where investigation should begin.

5.4 Future research

The most important direction for future research is validation on real operational incidents. The present thesis demonstrates that metadata-driven root-cause diagnosis is feasible in a controlled synthetic environment, but real-world validation is needed to assess whether the learned diagnostic patterns transfer to operational data and whether the degradation taxonomy remains appropriate in practice.

Additionally, research focused on the empirical basis of the framework across datasets and domains can support this further. Testing the approach on multiple datasets with different feature types would make it possible to distinguish more clearly between general robustness and dataset-specific performance.

To further research the overall applicability, future work could introduce additional degradation mechanisms, which introduce more complex combinations of co-occurring issues. For example, a relevant degradation mechanism for future investigation could be cross-feature consistency degradation, in which the marginal values of individual features remain plausible, but the relationships between features become internally inconsistent. Such a degradation would be valuable because it represents a realistic data-quality failure that is not detectable through univariate profiling alone. Such extensions would make the synthetic setting better aligned with the heterogeneity of real operational pipelines.

Another direction is to study the feature representation more systematically. In particular, future work could perform studies on the different feature blocks and investigate whether alternative summary statistics, richer batch representations, or learned representations improve predictability, especially in the more difficult diagnostic settings.

In addition to the predictive nature of the models, a fourth direction concerns the model families themselves. Since the current results show a strong dependence on model choice, future work should explore whether alternative neural architectures, more specialized multi-label methods, calibrated ensembles, or hybrid approaches can improve robustness.

To investigate the framework in practice, end-to-end evaluation should be considered. Since the current thesis treats the triggering mechanism as given, future work should examine how a trigger and a root-cause diagnosis model perform jointly. The focus could especially be on how an

integrated system using both models in tandem influences the number of false positives. Since human oversight on AI systems is a current heavily studied topic within AI governance and false positives increase human workload and interpretation of predictions, this is an important property of the overall applicability.

Finally, future research should investigate adaptive and online variants of the framework. In the present design, the reference snapshot remains fixed within an experimental run. While this is business context dependent, the diagnostic model itself may need recalibration over time. Subsequently, in long-running production systems, accepted batches may need to be incorporated into the reference over time or old batches must be removed to create a rolling time-window.

These directions position the thesis as an empirical starting point rather than a definitive proof of a production-ready framework.

6 Conclusion

This thesis investigated how batch-level metadata can be adopted to learn a supervised model that identifies the root causes of performance degradation in machine-learning pipelines. The work was motivated by the observation that existing monitoring approaches are often effective at detecting that incoming data or model performance has degraded, but do not determine which underlying data-quality mechanisms are responsible for the observed degradation. To address this gap, the thesis proposed a metadata-driven diagnostic framework in which batch-level quality signals are used as input for supervised multi-label prediction.

6.1 Answer to the research question

The central research question of this thesis was:

How can batch-level metadata be used to learn a supervised model that identifies and explains the root causes of machine-learning performance degradation in production pipelines, using synthetic data degradations?

Based on the empirical evaluation, the answer to the research question is that batch-level metadata can indeed be used to learn such a supervised model in a controlled synthetic environment. The results show that metadata representations derived from profiling statistics, outlier signals, and drift detectors contain sufficient information to predict degradation types at batch level. This indicates that metadata can serve as a meaningful supervised feature space for root-cause diagnosis once a model-centric performance trigger has indicated that a batch requires further investigation.

At the same time, the results also show that predictive accuracy is not uniform across degradation types, perturbation levels, or degradation co-occurrence. The approach is therefore best interpreted as a proof-of-concept rather than as a fully production-ready solution. In particular, the framework demonstrates that metadata can support root-cause diagnosis under controlled synthetic conditions, but further validation is required before such a model can be implemented into real operational pipelines.

6.2 Summary of the main findings

Several main findings emerge from the empirical evaluation. First, the proposed framework demonstrates that supervised multi-label diagnosis is feasible using batch-level metadata. Across the main evaluation, both models substantially outperform the synthetic baseline based on prior label prevalence on all reported test metrics. This indicates that the feature space captures diagnostic information beyond label frequency alone and therefore provides meaningful predictive signal.

Second, the results consistently show that the choice of model family matters. Across all settings, XGBoost OvR achieves the strongest overall results and outperforms the FFNN on the primary and complementary performance metrics. In addition, the training diagnostics suggest that XGBoost OvR converges more reliably than the FFNN, whose learning curves indicate less stable

optimization behavior and weaker generalization during training. This model difference remains visible throughout the more detailed perturbation-based analyses.

Third, predictive performance varies substantially across the perturbation spectrum. The main analyses show that the lowest perturbation region, particularly below approximately 0.10, forms the most difficult part of the task. Performance improves as perturbations become more pronounced, indicating that weak degradations often do not induce sufficiently distinct metadata signatures to support reliable diagnosis.

Fourth, the degradation labels are not equally easy to detect. `Missingness` and `Distributional shift/Drift` are the most readily detectable labels for both evaluated models, whereas `Noise/Measurement error` and `Outliers/Extreme values` are consistently more difficult to identify. This difference is particularly pronounced in the lower perturbation region, where the latter two degradation types show lower F2 scores, higher false-positive rates, and weaker recall under constrained false-positive settings. These findings suggest that some degradation mechanisms produce clearer and more stable signatures in the metadata space than others, which has direct implications for the practical applicability of the framework.

Fifth, the results show that increasing the number of co-occurring degradation labels makes the task more difficult in the main approach. Although stronger perturbations generally improve detectability, this effect is reduced when multiple degradation types are present simultaneously within the same batch. `Missingness` and `Distributional shift/Drift` remain comparatively robust under co-occurrence, whereas `Noise/Measurement error` and `Outliers/Extreme values` are more visibly affected by increasing degradation complexity. This confirms that multi-cause diagnosis is materially more challenging than single-cause identification and constitutes an important source of difficulty for the proposed framework.

Finally, the supplementary experiments provide additional context for interpreting the robustness of the main findings. The restricted perturbation-range experiment does not yield a generally more learnable problem setting. For the FFNN, performance deteriorates, whereas XGBoost OvR remains broadly stable and shows only limited gains on secondary metrics.

Similarly, under `per_cell_random`, the FFNN becomes weaker than in the main approach, whereas XGBoost OvR remains comparatively stable and in some aspects improves slightly. Together, these additional analyses indicate that the main empirical patterns are broadly robust, but not entirely independent of design choices.

6.3 Contributions of the thesis

This thesis contributes to the literature on data quality monitoring and machine-learning data quality in several ways.

First, it extends the perspective of metadata-driven monitoring from deviation detection and performance awareness toward root-cause diagnosis. Rather than limiting the use of metadata to detecting that incoming data has changed or that model performance is likely to degrade, the thesis investigates whether metadata can also be used to identify which degradation mechanisms are likely responsible once underperformance has been detected.

Additionally, the thesis formulates batch-level root-cause diagnosis as a supervised multi-label classification problem. This is an important conceptual step, since realistic degradation events are not necessarily isolated and may contain multiple co-occurring degradation mechanisms. By explicitly treating diagnosis as a multi-label problem, the framework better reflects the complexity of practical data-quality incidents.

Furthermore, the thesis introduces a framework in which synthetic degradation, metadata extraction, and supervised prediction are combined. In this design, synthetic degradation generation enables the controlled creation of labeled scenarios, while the Quality Feature Builder transforms each batch into a fixed-dimensional feature representation based on metadata profiles, outlier signals, and drift indicators. These feature vectors then form the basis for supervised root-cause prediction.

Finally, the thesis provides an empirical evaluation of this framework under varying perturbation rates, co-occurrence levels, and alternative simulation settings. As a result, the work does not only assess whether metadata-driven diagnosis is feasible in principle, but also under which conditions such diagnosis becomes more or less reliable. This contributes a more detailed understanding of where the approach is strong, where it remains limited, and which degradation types are most difficult to recover.

6.4 Concluding remarks

Overall, the findings support the viability of metadata-driven root-cause diagnosis in simulated monitoring settings. The results show that batch-level metadata can be used not only to characterize incoming data or estimate expected model behavior, but also to infer which degradation mechanisms are likely present once model underperformance has been detected. At the same time, the evaluation makes clear that diagnostic success depends strongly on the visibility and separability of the degradation signal and is therefore not equally strong across all forms of data-quality degradation.

The proposed framework should therefore be interpreted as a first step toward more informative and model-centric monitoring of machine-learning pipelines. In particular, the results suggest that metadata-driven diagnosis is most promising when degradation signals are sufficiently represented through corresponding feature engineering.

While further research is needed to validate the approach on real operational incidents, under delayed or incomplete labels, and under changing production conditions, this thesis provides evidence that metadata-driven diagnosis is a promising direction for root-cause analysis of machine-learning pipeline performance degradation.

Bibliography

[1] Z. Abedjan, X. Chu, D. Deng, R. C. Fernandez, I. F. Ilyas, M. Ouzzani, P. Papotti, M. Stonebraker, and N. Tang. Detecting Data Errors: Where are we and what needs to be done? *Proceedings of the VLDB Endowment*, 2016.

[2] M. Abughazala, M. Ibiyo, H. Muccini, and M. Sharaf. Quality by Prompt: LLM-Powered Transformation of Data Quality Requirements Into Great Expectations. *Lecture notes in computer science*, 2025.

[3] S. Ackerman, P. Dube, E. Farchi, O. Raz, and M. Zalmanovici. Machine Learning Model Drift Detection Via Weak Data Slices. *Workshop on Deep Learning for Testing and Testing for Deep Learning*, 2021.

[4] M. Aljumaili, R. Karim, and P. Tretten. Metadata-based data quality assessment. *VINE Journal of Information and Knowledge Management Systems*, 46(2):232–250, May 2016.

[5] M. Baca. *Introduction to Metadata: Third Edition*. Getty Publications, June 2016. Google-Books-ID: xgZVDQAAQBAJ.

[6] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino. Methodologies for data quality assessment and improvement. *ACM Comput. Surv.*, 41(3):16:1–16:52, July 2009.

[7] C. Batini and M. Scannapieco. *Data Quality*. Data-Centric Systems and Applications. Springer Berlin Heidelberg, 2006.

[8] C. Batini and M. Scannapieco. *Data and Information Quality: Dimensions, Principles and Techniques*. Data-Centric Systems and Applications. Springer International Publishing, Cham, 2016.

[9] V. W. Berger and Y. Zhou. Kolmogorov–Smirnov Test: Overview. In *Wiley StatsRef: Statistics Reference Online*. John Wiley & Sons, Ltd, 2014. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118445112.stat06558.

[10] L. Berti-Equille. *Quality Awareness for Managing and Mining Data*. thesis, Universté de Rennes 1, June 2007.

[11] BickelSteffen, BrücknerMichael, and SchefferTobias. Discriminative Learning Under Covariate Shift. *Journal of Machine Learning Research*, 2009.

[12] R. Blake and P. Mangiameli. The Effects and Interactions of Data Quality and Problem Complexity on Classification. *JDIQ*, 2011.

[13] T. Bleifuß, L. Bornemann, T. Johnson, D. V. Kalashnikov, F. Naumann, and D. Srivastava. Exploring Change - A New Dimension of Data Analytics. *Proceedings of the VLDB Endowment*, 2018.

[14] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, Oct. 2001.

[15] M. Breunig, P. Kröger, R. Ng, and J. Sander. LOF: Identifying Density-Based Local Outliers. volume 29, pages 93–104, June 2000.

[16] L. Budach, M. Feuerpfeil, N. Ihde, A. Nathansen, N. Noack, H. Patzlaff, H. Harmouch, and F. Naumann. The Effects of Data Quality on Machine Learning Performance. *arXiv (Cornell University)*, 2022.

[17] N. Bylois, F. Neven, and S. Vansummeren. CM-Explorer: Dissecting Data Ingestion Problems. *Proceedings of the VLDB Endowment*, 2023.

[18] Capgemini. Capgemini - about us, Apr. 2021.

[19] Capgemini. Capgemini - Analyst recognition, 2025.

[20] Capgemini. Capgemini - 2025 Annual Report, Feb. 2026.

[21] M. Carlo. Monte Carlo, 2025.

[22] E. Caveness, P. S. G. C., Z. Peng, N. Polyzotis, S. Roy, and M. Zinkevich. TensorFlow Data Validation: Data Analysis and Validation in Continuous ML Pipelines. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, pages 2793–2796, New York, NY, USA, May 2020. Association for Computing Machinery.

[23] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *CSUR*, 2009.

[24] J. Chen, F. Liu, B. Avci, X. Wu, Y. Liang, and S. Jha. Detecting Errors and Estimating Accuracy on Unlabeled Data with Self-training Ensembles. *Neural Information Processing Systems*, 2021.

[25] T. Chen and C. Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, Aug. 2016. arXiv:1603.02754 [cs].

[26] Collibra. Collibra, 2025.

[27] T. Dasu, G. T. Vesonder, and J. R. Wright. Data quality through knowledge engineering. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 705–710, New York, NY, USA, Aug. 2003. Association for Computing Machinery.

[28] L. Ehrlinger, V. Haunschmid, D. Palazzini, and C. Lettner. A DaQL to Monitor Data Quality in Machine Learning Applications. *Lecture Notes in Computer Science*, 2019.

[29] W. Elouataoui, S. El Mendili, and Y. Gahi. Active Metadata and Machine Learning based Framework for Enhancing Big Data Quality. In *Proceedings of the 7th International Conference on Networking, Intelligent Systems and Security*, NISS '24, pages 1–8, New York, NY, USA, Aug. 2024. Association for Computing Machinery.

[30] G. Expectations. Great Expectations, 2025.

[31] H. Foidl, V. Golendukhina, R. Ramler, and M. Felderer. Data Pipeline Quality: Influencing Factors, Root Causes of Data-related Issues, and Processing Problem Areas for Developers. *Journal of Systems and Software*, 2023.

[32] C. Fox, A. Levitin, and T. Redman. The notion of data and its quality dimensions. *Information Processing & Management*, 30(1):9–19, Jan. 1994.

[33] S. Greco and T. Cerquitelli. Drift Lens: Real-time unsupervised Concept Drift detection by evaluating per-label embedding distributions. *2021 International Conference on Data Mining Workshops (ICDMW)*, 2021.

[34] A. Haug and J. S. Arlbjørn. Barriers to master data quality. *J. Enterp. Inf. Manag.*, 2011.

[35] B. Heinrich, M. Klier, A. Schiller, and G. Wagner. Assessing data quality – A probability-based metric for semantic consistency. *Decision Support Systems*, 110:95–106, June 2018.

[36] J. Huang, A. J. Smola, A. Gretton, K. Borgwardt, and B. Schölkopf. Correcting Sample Selection Bias by Unlabeled Data. *Neural Information Processing Systems*, 2006.

[37] M. Hubert, M. Debruyne, and P. J. Rousseeuw. Minimum Covariance Determinant and Extensions. *WIREs Computational Statistics*, 10(3):e1421, May 2018. arXiv:1709.07045 [stat].

[38] H. D. Iii and D. Marcu. Domain Adaptation for Statistical Classifiers. *Journal of Artificial Intelligence Research*, 26:101–126, June 2006.

[39] Informatica. Informatica, 2025.

[40] Kevin N. Shah, Sandip J. Gami, and Abhishek Trehan. An Intelligent Approach to Data Quality Management AI-Powered Quality Monitoring in Analytics. *International Journal of Advanced Research in Science, Communication and Technology*, pages 109–119, Dec. 2024.

[41] M. Klier, A. Obermeier, C. Sparn, and T. Widmann. Anomaly-based Assessment of Semantic Consistency: Design and Evaluation of a Novel Probability-based Metric in Cooperation with a German Car Manufacturer. *Journal of Data and Information Quality*, 2025.

[42] S. Krishnan, M. J. Franklin, K. Goldberg, and E. Wu. BoostClean: Automated Error Detection and Repair for Machine Learning. *arXiv: Databases*, 2017.

[43] Z. Li, Y. Zhao, N. Botta, C. Ionescu, and X. Hu. COPOD: Copula-Based Outlier Detection. *Industrial Conference on Data Mining*, 2020.

[44] J. Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, Jan. 1991.

[45] Z. Lipton, Y. Wang, and A. J. Smola. Detecting and Correcting for Label Shift with Black Box Predictors. *International Conference on Machine Learning*, 2018.

[46] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, Dec. 2008.

[47] M. Lorenzini, M. Rospocher, and S. Tonelli. Automatically evaluating the quality of textual descriptions in cultural heritage records. *International Journal on Digital Libraries*, 22(2):217–231, June 2021.

[48] S. Lundberg and S.-I. Lee. A Unified Approach to Interpreting Model Predictions, Nov. 2017. arXiv:1705.07874 [cs].

[49] S. Madnick, R. Y. Wang, Y. W. Lee, and H. Zhu. Overview and Framework for Data and Information Quality Research. *JDIQ*, 2009.

[50] S. Mohammed, L. Budach, M. Feuerpfeil, N. Ihde, A. Nathansen, N. Noack, H. Patzlaff, F. Naumann, and H. Harmouch. The effects of data quality on machine learning performance on tabular data. *Information Systems*, 2025.

[51] P. Oliveira, F. Rodrigues, and P. Henriques. A Formal Definition of Data Quality Problems. 2005.

[52] V. M. Panaretos and Y. Zemel. Statistical Aspects of Wasserstein Distances. *Annual Review of Statistics and Its Application*, 6(Volume 6, 2019):405–431, Mar. 2019.

[53] K. Pearson. On the Criterion that a Given System of Deviations from the Probable in the Case of a Correlated System of Variables is Such that it Can be Reasonably Supposed to have Arisen from Random Sampling. In S. Kotz and N. L. Johnson, editors, *Breakthroughs in Statistics: Methodology and Distribution*, pages 11–28. Springer, New York, NY, 1992.

[54] L. L. Pipino, Y. W. Lee, and R. Y. Wang. Data quality assessment. *Commun. ACM*, 45(4):211–218, Apr. 2002.

[55] A. A. Pol, V. Azzolini, G. Cerminara, F. D. Guio, G. Franzoni, C. Germain, M. Pierini, and T. Krzy.zek. Deep learning for certification of the quality of the data acquired by the CMS Experiment. *Journal of Physics: Conference Series*, 1525, 2020.

[56] L. Poon, S. Farshidi, N. Li, and Z. Zhao. Unsupervised Anomaly Detection in Data Quality Control. *2021 IEEE International Conference on Big Data (Big Data)*, 2021.

[57] S. Rabanser, S. Günnemann, and Z. Lipton. Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift. *Neural Information Processing Systems*, 2018.

[58] E. Rahm and H. H. Do. Data Cleaning: Problems and Current Approaches. 2000.

[59] M. T. Ribeiro, S. Singh, and C. Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier, Feb. 2016. arXiv:1602.04938 [cs] version: 1.

[60] D. Rousidis, E. Garoufallou, P. Balatsoukas, and M. Sicilia. Metadata for Big Data: A preliminary investigation of metadata quality issues in research data repositories. *Inf. Serv. Use*, 2014.

[61] P. J. Rousseeuw and B. C. van Zomeren. Unmasking Multivariate Outliers and Leverage Points. *Journal of the American Statistical Association*, 85(411):633–639, Sept. 1990. _eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1990.10474920.

[62] K. S. Ryu, J. S. Park, and J.-H. Park. A Data Quality Management Maturity Model. *Etri Journal*, 2006.

[63] S. Ben-David and et al. Machine Learning in Non-Stationary Environments - Introduction to Covariate Shift Adaptation. *Adaptive computation and machine learning*, 2012.

[64] S. Schelter, T. Rukat, and F. Biessmann. Learning to Validate the Predictions of Black Box Classifiers on Unseen Data. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1289–1299, Portland OR USA, June 2020. ACM.

[65] S. Schelter, T. Rukat, and F. Bießmann. JENGA - A Framework to Study the Impact of Data Errors on the Predictions of Machine Learning Models. *International Conference on Extending Database Technology*, 2021.

[66] S. Schelter, P. Schmidt, T. Rukat, M. Kiessling, A. Taptunov, F. Bießmann, and D. Lange. DEEQU - Data Quality Validation for Machine Learning Pipelines. 2018.

[67] R. Shwartz-Ziv and A. Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, May 2022.

[68] R. Silvola, J. Härkönen, O. Vilppola, H. K. Vehkaperä, and H. Haapasalo. Data quality assessment and improvement. *Int. J. Bus. Inf. Syst.*, 2016.

[69] Soda. Soda, 2025.

[70] J. Song and Y. He. Auto-Validate: Unsupervised Data Validation Using Data-Domain Patterns Inferred from Data Lakes. *arXiv (Cornell University)*, 2021.

[71] K. M. Sunderland, D. Beaton, J. Fraser, D. Kwan, P. McLaughlin, M. Montero-Odasso, A. Peltsch, F. Pieruccini-Faria, D. J. Sahlas, R. H. Swartz, S. C. Strother, and M. A. Binns. The utility of multivariate outlier detection techniques for data quality evaluation in large studies: an application within the ONDRI project. *BMC Medical Research Methodology*, 2019.

[72] P. Swazinna, S. Udluft, and T. A. Runkler. Measuring Data Quality for Dataset Selection in Offline Reinforcement Learning. *IEEE Symposium Series on Computational Intelligence*, 2021.

[73] D. Tu, Y. He, W. Cui, S. Ge, H. Zhang, S. Han, D.-Y. Zhang, and S. Chaudhuri. Auto-Validate by-History: Auto-Program Data Quality Constraints to Validate Recurring Data Pipelines. *Knowledge Discovery and Data Mining*, 2023.

[74] H. Ulrich, A.-K. Kock-Schoppenhauer, N. Deppenwiese, R. Gött, J. Kern, M. Lablans, R. W. Majeed, M. R. Stöhr, J. Stausberg, J. Varghese, M. Dugas, and J. Ingenerf. Understanding the Nature of Metadata: Systematic Review. *Journal of Medical Internet Research*, 2022.

[75] G. P. Vidhya, D. Nirmala, and T. Manju. Quality challenges in Deep Learning Data Collection in perspective of Artificial Intelligence. *Journal of Information Technology and Computing*, 2023.

[76] L. Visengeriyeva and Z. Abedjan. Metadata-driven error detection. *International Conference on Statistical and Scientific Database Management*, 2018.

[77] R. Y. Wang and D. M. Strong. Beyond Accuracy: What Data Quality Means to Data Consumers. *Journal of Management Information Systems*, 12(4):5–33, Mar. 1996. _eprint: https://doi.org/10.1080/07421222.1996.11518099.

[78] E. Widad, E. Saida, and Y. Gahi. Quality Anomaly Detection Using Predictive Techniques: An Extensive Big Data Quality Framework for Reliable Data Analysis. *IEEE Access*, 11:103306–103318, 2023.

[79] J. Yang. Fast TreeSHAP: Accelerating SHAP Value Computation for Trees, July 2022. arXiv:2109.09847 [cs].

Appendix

A Statistical notation used in feature tables

The **Stats** column in Tables B.6–D.8 lists the statistics emitted for a given feature family. The following abbreviations are used:

Table A.5: Legend of statistic abbreviations used in the **Stats** column of the feature tables.

Abbreviation	**Definition**
`mean`	Arithmetic mean: $\text{mean}(x) = \frac{1}{n}\sum_{i=1}^{n} x_i$.
`median`	Median of the empirical distribution. Let $x_{(1)} \le \cdots \le x_{(n)}$ be the sorted values. Then $\text{median}(x) = \begin{cases} x_{(\frac{n+1}{2})}, & n \text{ odd}, \\ \frac{1}{2}\left(x_{(\frac{n}{2})} + x_{(\frac{n}{2}+1)}\right), & n \text{ even}. \end{cases}$
`std`	Sample standard deviation: $\text{std}(x) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \text{mean}(x))^2}$.
`min`, `max`	Minimum and maximum: $\min(x) = \min_{1\le i\le n} x_i, \quad \max(x) = \max_{1\le i\le n} x_i$.
`q50`, `q90`, `q95`, `q99`	Empirical quantiles of a score distribution. For probability level $p \in \{0.50, 0.90, 0.95, 0.99\}$, $q_p(x)$ denotes the empirical p-quantile of the values $x_1, \ldots, x_n$.
`fraction`	Fraction of rows, columns, or features satisfying a condition: $\frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[c_i]$, where c_i is the relevant condition.
`share above ref q95/q99`	For an outlier detector score $s_d(i)$, the fraction of rows in the current batch with a score above the detector-specific reference threshold: $\frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\left[s_d(i) > \kappa_{d,p}^{\text{ref}}\right]$, where $\kappa_{d,p}^{\text{ref}}$ is the empirical p-quantile of detector d on the reference batch, with $p \in \{0.95, 0.99\}$.
`tail gap`	Difference between the upper-tail quantiles of a detector score distribution: $\text{tail gap} = q_{0.99}(s) - q_{0.95}(s)$.
`tail ratio`	Ratio between the 95th percentile and the median of a detector score distribution: $\text{tail ratio} = \frac{q_{0.95}(s)}{\text{median}(s) + \varepsilon}$.
`std ratio`	Ratio between the batch and reference standard deviation of a numeric column: $\text{std ratio}_j = \frac{\sigma_j(X) + \varepsilon}{\sigma_j(R) + \varepsilon}$, where X is the current batch and R is the reference batch.
`conc`	Concentration index over nonnegative per-feature drift scores u_j: $\text{conc} = \sum_j \left(\frac{u_j}{\sum_{j'} u_{j'} + \varepsilon}\right)^2$. Higher values indicate that drift is concentrated in fewer features.

Abbreviation	Definition
agreement	Summary statistics over pairwise Pearson correlations between detector score vectors. In the outlier feature block, these summaries are reported as mean, minimum, maximum, and standard deviation of the off-diagonal detector-correlation values.

All quantiles are computed empirically. When a statistic requires division, ε denotes a small positive stabilizer to avoid division by zero.

B Metadata Feature Block

Table B.6: Metadata feature list used by the final Quality Feature Builder configuration. The features describe batch-level type composition, missingness, numeric, categorical, text, datetime, and reference-relative numeric properties.

Feature key	Computation / algorithm	Granularity	Stats	Notes
meta__types	Fraction of columns assigned to each schema type using the provided column_types mapping.	batch	numeric fraction, categorical fraction, datetime fraction	Schema-driven.
meta__missing__fraction_col	Per-column missingness fraction $m_j = \#\mathrm{NA}/n$, aggregated across columns.	batch	mean, max, p95	Completeness.
meta__missing__row_any_fraction	Fraction of rows with at least one missing value.	batch	fraction	Completeness.
meta__num__std	Per numeric column standard deviation.	batch	mean, max	Scale profiling.
meta__num__skew_abs	Absolute skewness per numeric column.	batch	mean, max	Distribution-shape profiling.
meta__num__kurtosis	Kurtosis per numeric column.	batch	mean, max	Tail-shape profiling.
meta__num__range	Per numeric column range $(\max_j - \min_j)$.	batch	mean, max	Scale profiling.

Feature key	Computation / algorithm	Granularity	Stats	Notes
meta__num__zero_fraction_col	Fraction of zero values per numeric column, aggregated across numeric columns.	batch	mean, max	Placeholder / value-pattern proxy.
meta__num__negative_fraction_col	Fraction of negative values per numeric column, aggregated across numeric columns.	batch	mean, max	Value-pattern proxy.
meta__num_col__{col}	Reference-relative numeric deltas for each numeric column: absolute changes in mean, median, min, max, and the standard-deviation ratio.	column	delta mean, delta median, delta min, delta max, std ratio	Requires fitted reference batch.
meta__cat__distinct_count_col	Per categorical column distinct count, aggregated across categorical columns.	batch	mean, max	Representation.
meta__cat__distinct_ratio_col	Per categorical column distinct ratio, defined as distinct values divided by batch size.	batch	mean, max	Representation.
meta__cat__mode_freq_col	Per categorical column modal frequency.	batch	mean, min	Dominance proxy.
meta__cat__entropy_col	Per categorical column entropy.	batch	mean, max	Distributional diversity.
meta__cat__high_dominance_fraction	Fraction of categorical columns with modal frequency at least 0.95.	batch	fraction	Threshold fixed.
meta__text__avg_length_col	Average string length per object/string column, aggregated across text-like columns.	batch	mean, max	Dtype-based text proxy.
meta__text__max_length_col	Maximum string length per object/string column, aggregated across text-like columns.	batch	mean, max	Dtype-based text proxy.
meta__dt__span_days_col	Per datetime column span in days $(\max - \min)$, aggregated across datetime columns.	batch	mean, max	Timeliness proxy.
meta__dt__max_days_since_epoch	Maximum timestamp in the batch, represented as days since the Unix epoch.	batch	max	Timeliness proxy.

C Outlier Feature Block

Table C.7: Outlier feature block used by the final Quality Feature Builder configuration. Each detector produces per-row anomaly scores, which are summarized at batch level using central tendency, dispersion, tail, reference-exceedance, and detector-agreement features.

Feature key	Computation / algorithm	Granularity	Stats	Notes
`outlier__zscore__*`	Per row: aggregate absolute per-feature z-scores fitted on the reference data.	batch	mean, std, median, q95, q99, tail gap, tail ratio, share above ref q95/q99	Univariate baseline.
`outlier__iqr_soft__*`	Per row: normalized exceedance beyond IQR fences, aggregated across numeric features.	batch	mean, std, median, q95, q99, tail gap, tail ratio, share above ref q95/q99	Robust univariate.
`outlier__mahalanobis_mcd__*`	Per row: robust Mahalanobis distance using an MCD covariance estimate fitted on the reference data.	batch	mean, std, median, q95, q99, tail gap, tail ratio, share above ref q95/q99	Robust multivariate structure.
`outlier__lof__*`	Per row: Local Outlier Factor novelty score fitted on the reference data and converted so that higher scores indicate stronger outlierness.	batch	mean, std, median, q95, q99, tail gap, tail ratio, share above ref q95/q99	Local density anomalies.
`outlier__isoforest__*`	Per row: Isolation Forest score fitted on the reference data and converted so that higher scores indicate stronger outlierness.	batch	mean, std, median, q95, q99, tail gap, tail ratio, share above ref q95/q99	Model-based multivariate detector.
`outlier__{det}__tail_gap_q99_q95`	For each detector, compute the difference between the 99th and 95th percentile of the batch row-score distribution.	batch	q99–q95	Tail-shape enrichment.

Feature key	Computation / algorithm	Granularity	Stats	Notes
outlier__{det}__tail_ratio_q95_med	For each detector, compute $q95/(\text{median} + \varepsilon)$ from the batch row-score distribution.	batch	ratio	Tail-shape enrichment.
outlier__{det}__share_above_ref_q95	For each detector, compute the fraction of batch rows with scores above the detector-specific reference 95th percentile.	batch	fraction	Fixed reference threshold.
outlier__{det}__share_above_ref_q99	For each detector, compute the fraction of batch rows with scores above the detector-specific reference 99th percentile.	batch	fraction	Fixed reference threshold.
outlier__agreement	Compute pairwise Pearson correlations between detector row-score vectors and summarize the off-diagonal correlations.	batch	mean, min, max, std	Detector agreement.

D Drift Feature Block

Table D.8: Drift Feature Block used by the final Quality Feature Builder configuration. Drift scores are computed per feature relative to the reference data and then summarized separately for numeric and categorical feature groups.

Feature key	Computation / algorithm	Granularity	Stats	Notes
drift__ks__num__*	Per numeric feature: Kolmogorov–Smirnov distance between the reference and batch empirical distributions.	batch	q50, q90, q99, conc	Numeric distribution-shape drift.
drift__wasserstein__num__*	Per numeric feature: Wasserstein-1 distance between the reference and batch distributions.	batch	q50, q90, q99, conc	Numeric location/scale distribution drift.
drift__chi2__cat__*	Per categorical feature: chi-square goodness-of-fit score comparing batch category counts against reference category proportions.	batch	q50, q90, q99, conc	Categorical frequency drift.

Feature key	Computation / algorithm	Granularity	Stats	Notes
`drift__js__cat__*`	Per categorical feature: Jensen–Shannon divergence between reference and batch category frequency vectors.	batch	q50, q90, q99, conc	Categorical distribution drift.
`drift__{det}__{type}__conc`	For each detector and applicable feature type, compute a Herfindahl-like concentration index over absolute per-feature drift scores.	batch	concentration	High values indicate drift concentrated in few features.

E Synthetic Data Generator (SDG) Configuration

Table E.9: Synthetic Data Generator (SDG) parameters and configuration options used to generate labeled degraded batches from the reference snapshot R_t.

Parameter	Type	Meaning / effect	Notes / typical values
`R_t`	DataFrame	Reference snapshot from which clean base batches are sampled and to which corruptions are applied.	Assumed accepted/clean baseline.
`n`	int	Batch size (number of rows)	Fixed per experiment (varied across runs).
`K`	int	Number of degraded batches to generate per dataset/run.	Controls training set size.
ρ	float	Target corruption rate: fraction of eligible cells to perturb. Determines $n_{\text{target}} \approx \rho \cdot n \cdot p_{\text{active}}$.	Typical: small (e.g., 0.01–0.20).
`use_feat_import`	bool	If enabled, uses feature-importance weights to create bias which columns get corrupted more often.	If false, columns are sampled uniformly.
`feature_importance` (FI_j)	dict / array	Nonnegative column weights used to define column sampling probabilities $w_j \propto \text{FI}_j$.	Optional; must align with schema columns.

Parameter	Type	Meaning / effect	Notes / typical values
`fi_topk`	int / None	Optional truncation: keep only top-k most important columns active; set others to weight 0.	Useful to focus corruption on impactful features.
`active_columns`	list / None	Explicit override list of columns eligible for corruption (after type constraints).	Optional; can enforce scope.
`strategies`	list[str]	Set of corruption strategies that SDG may apply (type-filtered per column).	E.g., `missing`, `gaussian_noise`, etc.
`strategy_mode`	str	How strategies are assigned: `fixed`, `per_column_random`, or `per_cell_random`.	Controls within-column heterogeneity.
`fixed_strategy`	str / None	Strategy used when `strategy_mode=fixed`.	Must be valid for all affected column types.
`strategy_weights`	dict / None	Optional weights over strategies when sampling in random modes.	If omitted, strategies sampled uniformly.
`missing_mode`	str	Whether missingness is injected as true NA (`missing`) or as sentinel placeholders (`placeholder_missing`).	Both map to the "missingness" label.
`missing_numeric_placeholder`	float / int / str	Sentinel used for numeric placeholder missingness.	Only used for `placeholder_missing`.
`missing_categorical_placeholder`	str	Sentinel used for categorical placeholder missingness.	E.g., `"MISSING"`.
`missing_datetime_placeholder`	str / datetime	Sentinel used for datetime placeholder missingness (parsed to datetime).	E.g., `"1970-01-01"`.
`noise_strength` (α)	float	Scale for Gaussian noise: $x' = x + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, (\alpha\sigma_j)^2)$.	Controls numeric "noise" strength.
`shift_strength` (s)	float	Shift/scale distortion: $x' = (x-\mu_j)(1+s)+\mu_j+s\sigma_j$.	Controls numeric "shift" strength.

Parameter	Type	Meaning / effect	Notes / typical values
`outlier_strength`	float	Outlier injection strength: adds large deviations (magnitude distribution scaled by σ_j).	Controls numeric "outlier" strength.
`category_swap_mode`	str / None	How alternative categories are selected for `category_swap` (e.g., uniform over observed set).	If unspecified, uniform is typical.
`type_cast_suffix`	str	Suffix/pattern appended when generating malformed strings in `type_cast_error`.	Used to create parse-breaking values.
`datetime_shift_days`	int	Large offset (days) for `datetime_shift`: $t' = t \pm \Delta$.	Controls temporal "shift" strength.
`datetime_jitter_days_std`	float	Std. dev. (days) for `datetime_jitter`: $t' = t + \delta$, $\delta \sim \mathcal{N}(0, \sigma_\delta^2)$.	Controls temporal "noise" strength.
`random_state`	int	Controls all stochasticity: row sampling, cell selection, strategy sampling, and perturbation draws.	Fixed per run; disjoint seeds for train/test.
`return_summary`	bool	Whether SDG returns a per-batch summary table of applied corruptions (column, strategy, counts/fractions).	Used for deterministic label construction.

F Synthetic Degradation Strengths

Table F.10: Synthetic degradation strength ranges used during degraded batch generation.

Configuration field	Range	Meaning
`cell_perturbation_rate_range`	$(0.05, 0.30)$	Range from which the fraction of perturbed cells is sampled.
`noise_strength_range`	$(0.10, 0.30)$	Strength range for Gaussian noise applied to numeric values.
`shift_strength_range`	$(0.25, 0.75)$	Strength range controlling the magnitude of numeric shift/scale perturbations.

Configuration field	Range	Meaning
`shift_range_width`	$(0.05, 0.10)$	Range controlling the relative width of the affected region for shift/scale degradation.
`outlier_strength_range`	$(0.30, 0.70)$	Strength range controlling the magnitude of injected numeric outliers.
`datetime_shift_days_range`	$(20, 40)$	Range for systematic datetime shifts, expressed in days.
`datetime_jitter_days_std_range`	$(1.0, 5.0)$	Range for the standard deviation of datetime jitter, expressed in days.

G SDG strategy-to-label mapping

Table G.11: Mapping from SDG implementation strategies to taxonomy labels $\mathscr{L}$ and an auxiliary operational dimension. A taxonomy label is set to 1 if at least one mapped strategy occurs in the SDG batch summary.

Degradation label ℓ	SDG strategies	Dimension(s)	Operational intent / notes
Missingness	`missing`, `placeholder_missing`	Completeness	True NA/NaT and sentinel-based missingness; label active if either strategy occurs at least once.
Noise / Measurement error	`gaussian_noise`, `datetime_jitter`	Accuracy	Attribute-level distortions that preserve type; values remain plausible but deviate from expected properties.
Outliers / Extreme values	`outlier`	Accuracy; integrity	Explicit tail-event injection in numeric columns, intended to create rare extreme deviations.
Redundancy / Duplication	*Not generated by SDG or evaluated as degradation label by Root-cause model*	Uniqueness, consistency	This label is excluded, since duplicates or near-duplicates are often caught upstream by data-centric methods.

Degradation label ℓ	SDG strategies	Dimension(s)	Operational intent / notes
Distributional shift / Drift	`shift_scale`, `permute`, `category_swap`, `datetime_shift`	Timeliness; Accuracy	Strategies that alter univariate or multivariate distributional properties relative to R_t.

H Dataset snippet & summary

The variable space excludes the primary key `transaction_id` and the derived temporal columns `hour`, `day`, `weekday`, `is_weekend`, and `is_night`, since these were not included in the analysis.

Table H.12: Illustrative sample of the enterprise fraud dataset. The table shows a small subset of rows and a selection of representative columns.

customer_id	merchant_id	timestamp	customer_segment	transaction_amount	...
9802	1487	2025-01-22 14:02:41	regular	424.75	...
9762	1412	2025-02-18 01:53:25	regular	81.54	...
9230	1690	2025-03-04 00:29:01	regular	346.67	...
6009	942	2025-04-02 21:41:53	regular	260.79	...
2818	715	2025-02-17 14:30:54	regular	143.13	...

The active feature specification excludes `transaction_id` and the derived temporal columns `hour`, `day`, `weekday`, `is_weekend`, and `is_night`, since these were commented out in the implementation and were therefore not included in the analysis.

Table H.13: Overview of active dataset columns used in the analysis. The dataset contains 200,000 rows and 38 active columns: 22 numeric, 15 categorical, and 1 datetime column.

Column	Type	Description
`customer_id`	categorical	Unique identifier for each customer.
`merchant_id`	categorical	Unique identifier for each merchant.
`timestamp`	datetime	Date and time of the transaction.
`customer_segment`	categorical	Segment or tier of the customer.
`avg_spend_profile`	numeric	Average spending profile of the customer.
`home_lat`	numeric	Latitude coordinate of the customer's home.
`home_lon`	numeric	Longitude coordinate of the customer's home.
`past_fraud_history`	numeric	Count or measure of past fraudulent activity.
`credit_score`	numeric	Credit score of the customer.
`account_age_days`	numeric	Age of the customer's account in days.
`merchant_category`	categorical	Category of the merchant, for example electronics or food.
`merchant_risk_score`	numeric	Risk score associated with the merchant.

Column	Type	Description
merchant_lat	numeric	Latitude coordinate of the merchant's location.
merchant_lon	numeric	Longitude coordinate of the merchant's location.
transaction_amount	numeric	Monetary value of the transaction.
txn_lat	numeric	Latitude coordinate of the transaction location.
txn_lon	numeric	Longitude coordinate of the transaction location.
distance_from_home_km	numeric	Distance from the transaction to the customer's home in kilometers.
is_foreign	categorical	Boolean indicator equal to 1 if the transaction is foreign and 0 otherwise.
high_risk_country	categorical	Boolean indicator equal to 1 if the transaction originates from a high-risk country and 0 otherwise.
device_type	categorical	Type of device used for the transaction.
channel	categorical	Channel through which the transaction occurred, for example POS, ATM, or online.
is_new_device	categorical	Boolean indicator equal to 1 if the device is new for the customer and 0 otherwise.
vpn_detected	categorical	Boolean indicator equal to 1 if VPN usage was detected and 0 otherwise.
tor_detected	categorical	Boolean indicator equal to 1 if TOR usage was detected and 0 otherwise.
device_id	categorical	Unique identifier for the device.
trans._velocity_1h	numeric	Transaction count or rate during the last 1 hour.
trans._velocity_24h	numeric	Transaction count or rate during the last 24 hours.
trans._velocity_7d	numeric	Transaction count or rate during the last 7 days.
sec._since_last_txn	numeric	Seconds elapsed since the previous transaction.
avg_amount_30d	numeric	Average transaction amount in the last 30 days.
amount_dev._ratio	numeric	Ratio of the current amount to the average amount.
merchant_ring_id	categorical	Identifier for a merchant group or ring.
shared_device_count	numeric	Count of other customers using the same device.
cus._merch._txn_count	numeric	Number of transactions between a customer and a merchant.
label	categorical	Target variable indicating whether a transaction is fraudulent or legitimate.
fraud_type	categorical	Type of fraud if the transaction is fraudulent.
financial_loss	numeric	Monetary loss incurred by the transaction.

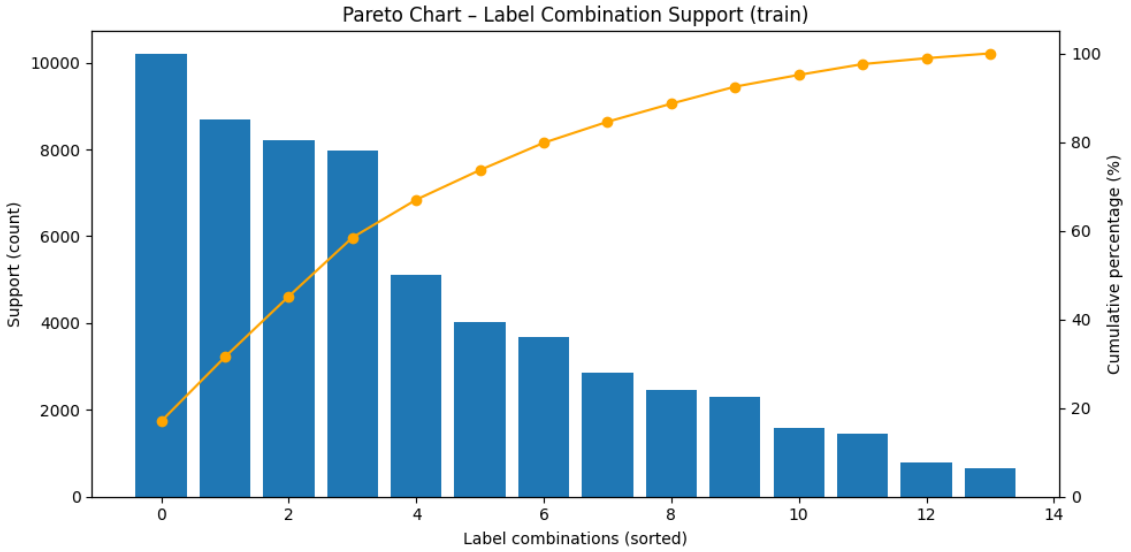


Figure H.1: Pareto distribution of label combinations in the training set.

I Reproducibility factors

Table I.14: Overview of experimental settings. All experiments use batch size 128 and 20 degradation sets per batch.

Strategy	**# co-occ. labels**	**Set sizes (Train / Val / Test)**	**perturbation range**
`per_column_random`	1	20,000 / 5,000 / 6,240	(0,0.5)
	2	20,000 / 5,000 / 6,240	(0,0.5)
	3	20,000 / 5,000 / 6,240	(0,0.5)
`Restricted range`	1	20,000 / 5,000 / 6,240	(0.05,0.3)
	2	20,000 / 5,000 / 6,240	(0.05,0.3)
	3	20,000 / 5,000 / 6,240	(0.05,0.3)
`per_cell_random`	1	20,000 / 5,000 / 6,240	(0,0.5)
	2	20,000 / 5,000 / 6,240	(0,0.5)
	3	20,000 / 5,000 / 6,240	(0,0.5)

Table I.15: Overview of stochastic components in the experimental pipeline and their corresponding seed control. All randomness is deterministically derived from `MASTER_SEED` = 20.

Component / Module	**Source of Randomness**	**Seed Control**
Dataset splitting	Shuffling and random partitioning of instances	`MASTER_SEED`
Train-validation split	Random selection of data	`MASTER_SEED`
Batch seed sets	Generation of unique per-batch random seeds	`MASTER_SEED`
Synthetic degradation (train/val/test batches)	Random strategy choice, severity, and location	*Batch seed sets*
Outlier detection	Randomized model fitting (Isolation Forest, MCD)	*Batch seed sets*
Metadata extraction	Random subsampling and initialization	*Batch seed sets*
Optuna hyperparameter sampling	sampling of hyperparameter configs `RandomSampler`	`MASTER_SEED`
Optuna trial execution	Trial order and parameter evaluation sequence	`RandomSampler`
Per-trial Model initialization and training (FFNN)	Random weight initialization, dropout and Shuffling	`trial_seed = MASTER_SEED + trial.number`
Per-trial model fitting (XGB)	Row/column subsampling and tree construction	`trial_seed`
Threshold optimization	Potential stochasticity via model outputs	Implicitly by `trial_seed`

J Models: hyperparameter & threshold optimization, computation times & learning curves

Table J.16: Selected hyperparameters for the evaluated model settings, with # of trials (N) = 100.

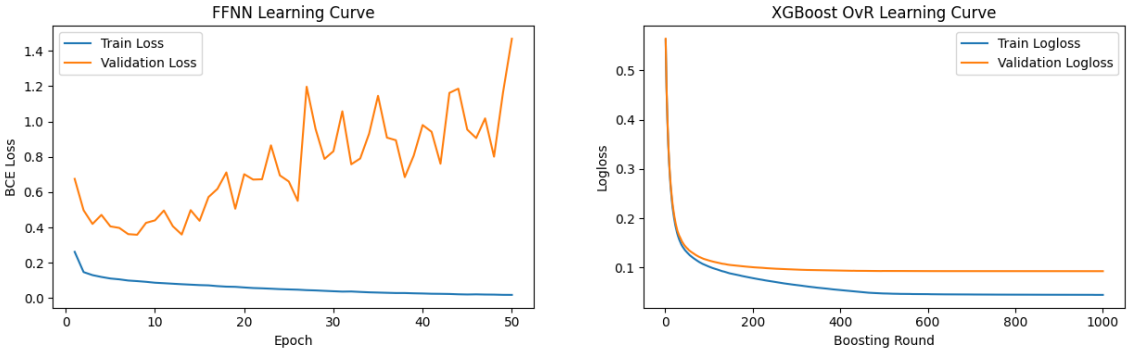
Setting	Model	Selected hyperparameters
Main approach	FFNN	$n_{\text{hidden\_layers}} = 2$; hidden_width = 64; dropout = 0.1652; learning_rate = 0.005673; weight_decay = 2.52×10^{-6}; batch_size = 256
Main approach	XGBoost OvR	max_depth = 8; eta = 0.01436; n_estimators = 2000; min_child_weight = 8.1312; gamma = 1.8370; subsample = 0.7102; colsample_bytree = 0.9407; reg_lambda = 0.07795; reg_alpha = 0.01561
Restricted range $(0.05, 0.3)$	FFNN	$n_{\text{hidden\_layers}} = 3$; hidden_width = 128; dropout = 0.2435; learning_rate = 0.0001159; weight_decay = 0.001057; batch_size = 512
Restricted range $(0.05, 0.3)$	XGBoost OvR	max_depth = 5; eta = 0.10436; n_estimators = 1000; min_child_weight = 3.4098; gamma = 2.0352; subsample = 0.9927; colsample_bytree = 0.7339; reg_lambda = 0.08327; reg_alpha = 1.21×10^{-5}
`per_cell_random` approach	FFNN	$n_{\text{hidden\_layers}} = 2$; hidden_width = 512; dropout = 0.2631; learning_rate = 0.0002052; weight_decay = 4.09×10^{-6}; batch_size = 512
`per_cell_random` approach	XGBoost OvR	max_depth = 8; eta = 0.01538; n_estimators = 1000; min_child_weight = 1.8684; gamma = 0.3088; subsample = 0.9723; colsample_bytree = 0.6092; reg_lambda = 0.003249; reg_alpha = 1.83×10^{-5}

Table J.17: Optimized label-specific thresholds for the evaluated model settings, with # of trials (N) = 100.

Setting	Model	Missingness	Noise / Meas. err.	Outliers / Ext. val.	Dist. shift / Drift
Main approach	FFNN	0.10	0.05	0.10	0.05
Main approach	XGBoost OvR	0.45	0.15	0.10	0.55
Restricted range $(0.05, 0.3)$	FFNN	0.20	0.10	0.05	0.05
Restricted range $(0.05, 0.3)$	XGBoost OvR	0.10	0.20	0.15	0.25
`per_cell_random` approach	FFNN	0.25	0.80	0.80	0.05
`per_cell_random` approach	XGBoost OvR	0.10	0.20	0.20	0.15

Table J.18: Computation times for the evaluated model settings. Degradation generation time denotes the time required to generate the degraded datasets, feature construction time denotes the time required to generate the feature representation, total training time denotes the full time required to fit the model (N=100), and inference time denotes the time required to generate predictions on the test set. These computation times are for all 3 degradation runs (max. co-occurring 1,2,3) aggregated, since these are all needed for the final model.

Setting	Model	Deg. gen.	Feat. constr.	Training	Inference
Main approach	FFNN	4:27:55	24:23:01	2:26:50	<2s
	XGB OvR			3:46:34	<2s
Restricted range	FFNN	1:30:51	24:21:49	2:24:28	<2s
	XGB OvR			3:58:38	<2s
`per_cell_random`	FFNN	4:54:27	22:21:22	2:18:24	<2s
	XGB OvR			3:05:40	<2s



(a): FFNN learning curves across epochs.

(b): XGBoost one-vs-rest training and validation log-loss across boosting rounds.

Figure J.2: Training diagnostics for the FFNN and XGBoost one-vs-rest models. The FFNN is evaluated using learning curves across epochs, whereas XGBoost one-vs-rest is evaluated using training and validation log-loss across boosting rounds.

K SHAP & Feature Importance XGBoost OvR

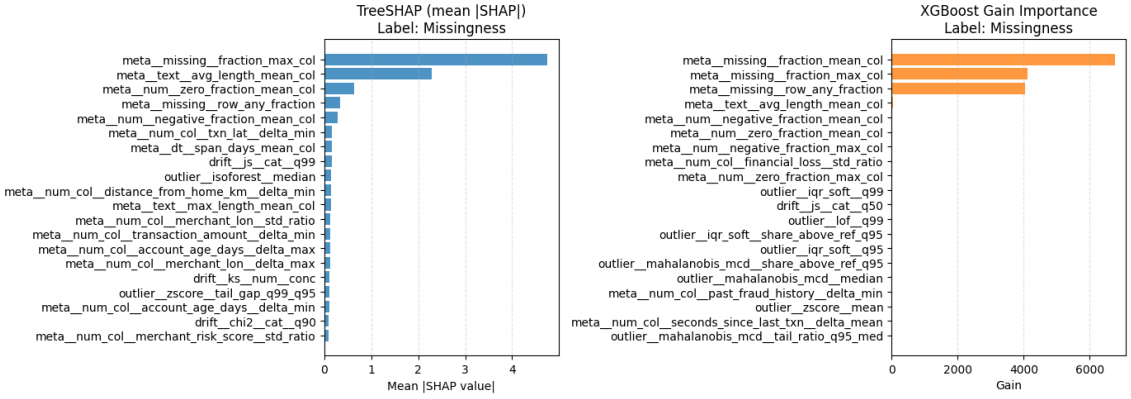


Figure K.3: Top-20 TreeSHAP and XGBoost gain importance features for the Missingness label.

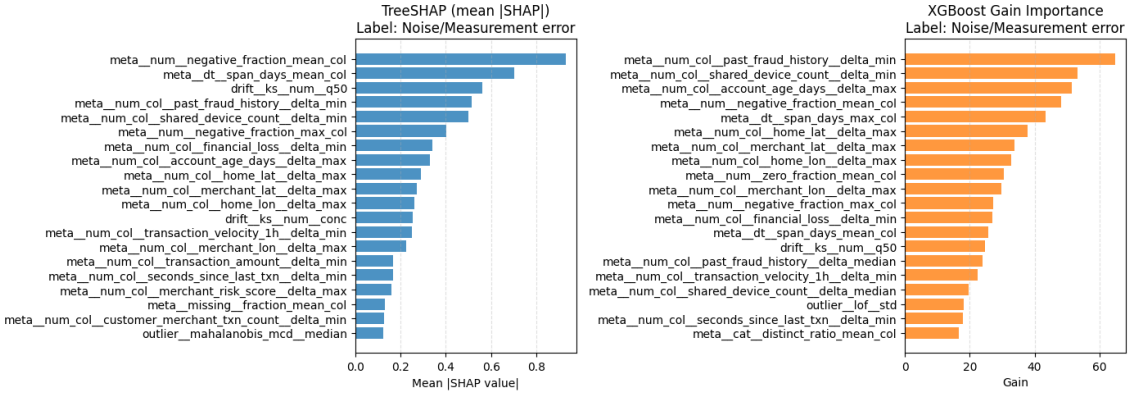


Figure K.4: Top-20 TreeSHAP and XGBoost gain importance features for the Noise/Measurement error label.

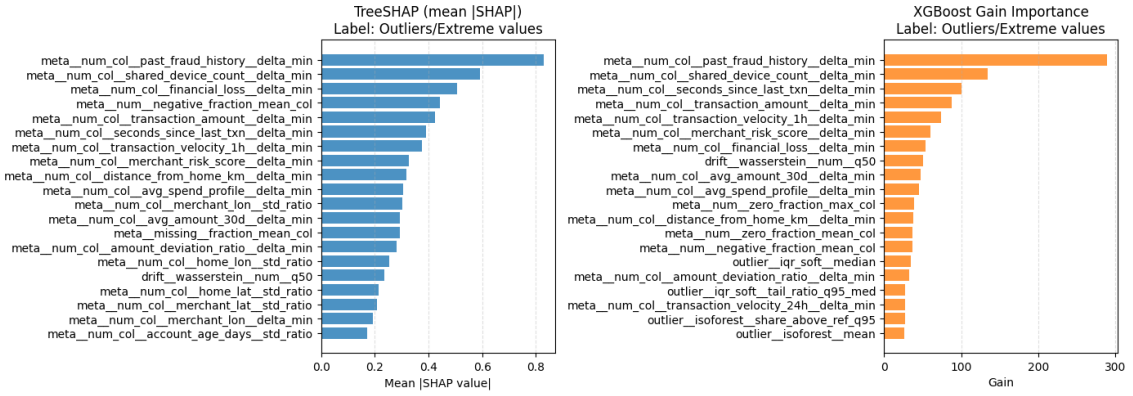


Figure K.5: Top-20 TreeSHAP and XGBoost gain importance features for the Outliers/Extreme values label.

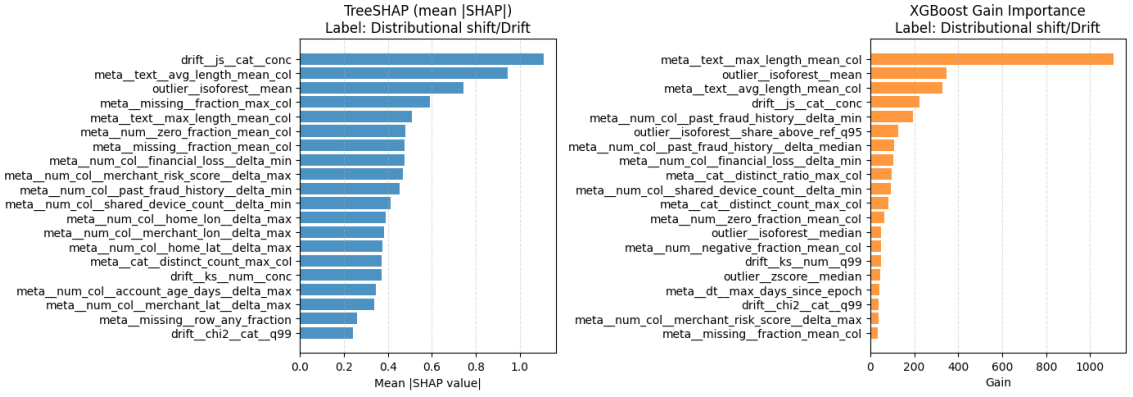
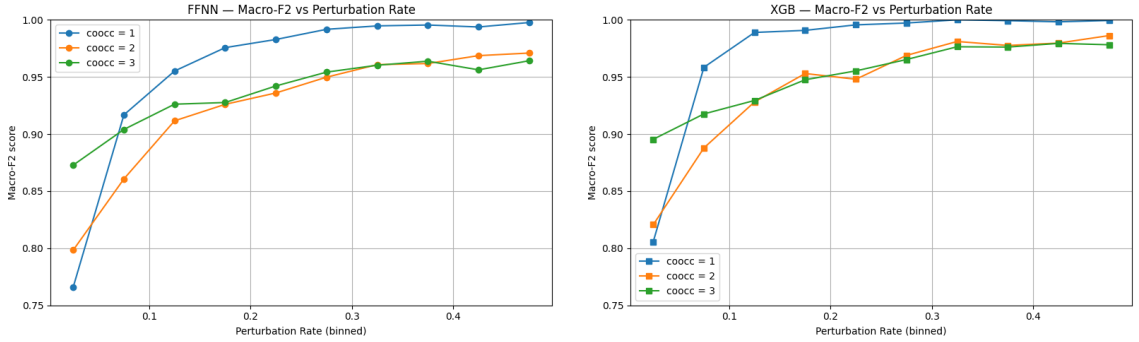


Figure K.6: Top-20 TreeSHAP and XGBoost gain importance features for the Distributional shift/Drift label.

L Results main approach

Table L.19: Additional overall predictive performance metrics of the prior baseline, FFNN, and XGBoost one-vs-rest models.

Metric		Prior baseline	FFNN	XGBoost OvR
Precision (micro)	Val	–	0.862383	0.898366
	Test	0.563370	0.862520	0.896788
Recall (micro)	Val	–	0.975905	0.983090
	Test	0.563317	0.975343	0.982786
Precision (macro)	Val	–	0.837447	0.871237
	Test	0.498797	0.836827	0.868906
Recall (macro)	Val	–	0.968276	0.975772
	Test	0.498751	0.967074	0.975296



(a): FFNN

(b): XGBoost one-vs-rest

Figure L.7: Macro-F2 plots across degradation rates and co-occurrence levels for the FFNN and XGBoost one-vs-rest models.

M Results restricted range approach

Table M.20: Overall predictive performance metrics of the prior baseline, FFNN, and XGBoost one-vs-rest on the restricted range $(0.05, 0.3)$.

Metric		Prior baseline	FFNN	XGBoost OvR
Macro-F2	Val	–	0.893675	0.951126
	Test	0.499636	0.892474	0.949479
Micro-F2	Val	–	0.917998	0.965950
	Test	0.565111	0.917938	0.965215
Subset Accuracy	Val	–	0.619200	0.811400
	Test	0.099568	0.619498	0.813141
Hamming Loss	Val	–	0.114850	0.053167
	Test	0.434590	0.115104	0.052951
Mean Jaccard Similarity	Val	–	0.853183	0.935144
	Test	0.411751	0.853508	0.935541
Precision (micro)	Val	–	0.848743	0.920526
	Test	0.565081	0.848339	0.921980
Recall (micro)	Val	–	0.937114	0.978015
	Test	0.565120	0.937159	0.976665
Precision (macro)	Val	–	0.837269	0.896691
	Test	0.499612	0.836321	0.897565
Recall (macro)	Val	–	0.919933	0.967502
	Test	0.499648	0.918807	0.965041

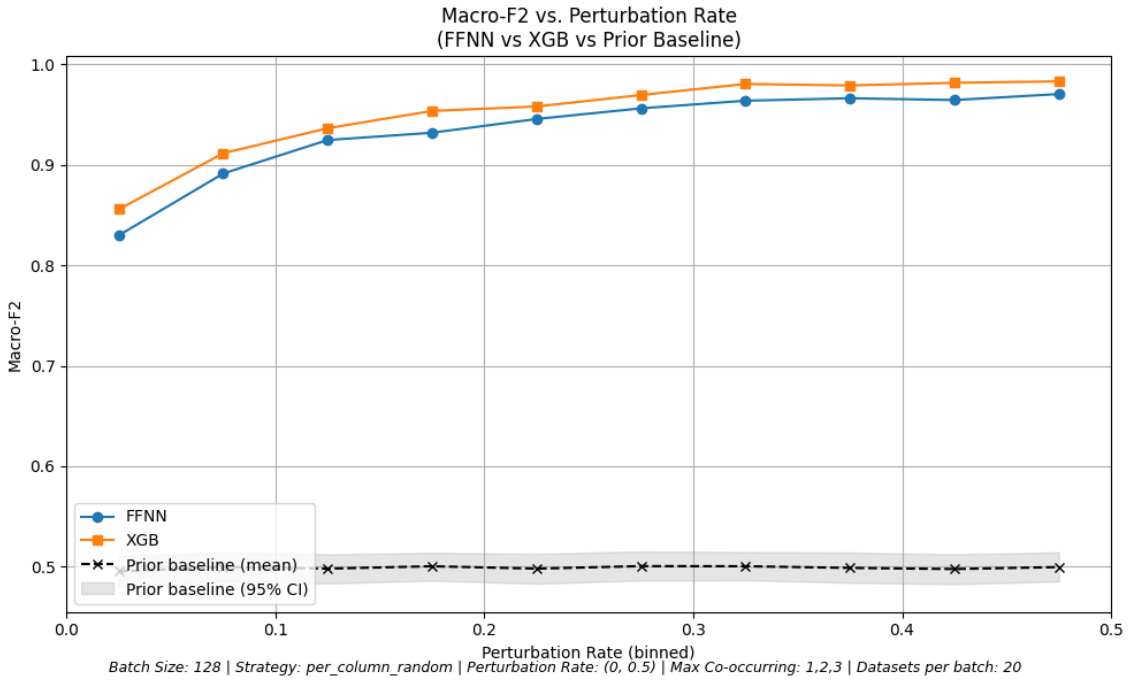


Figure M.8: Macro-F2 across perturbation rates for the FFNN, XGBoost OvR, and the prior baseline on the restricted range. The shaded area denotes the 95% confidence interval of the prior baseline.

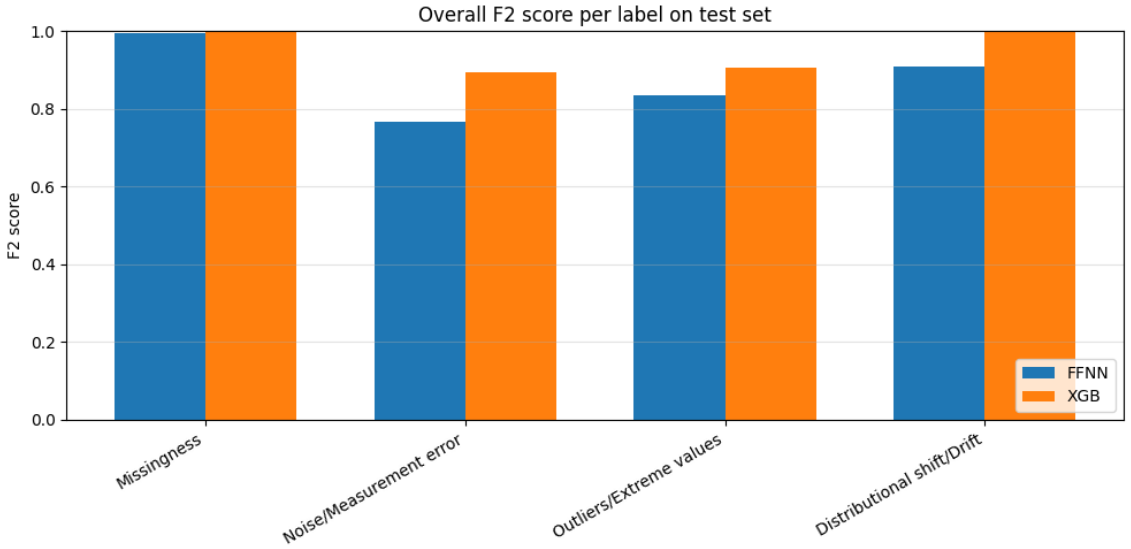


Figure M.9: Overall label-wise F2 scores on the test set for the FFNN and XGBoost one-vs-rest models on the restricted range.

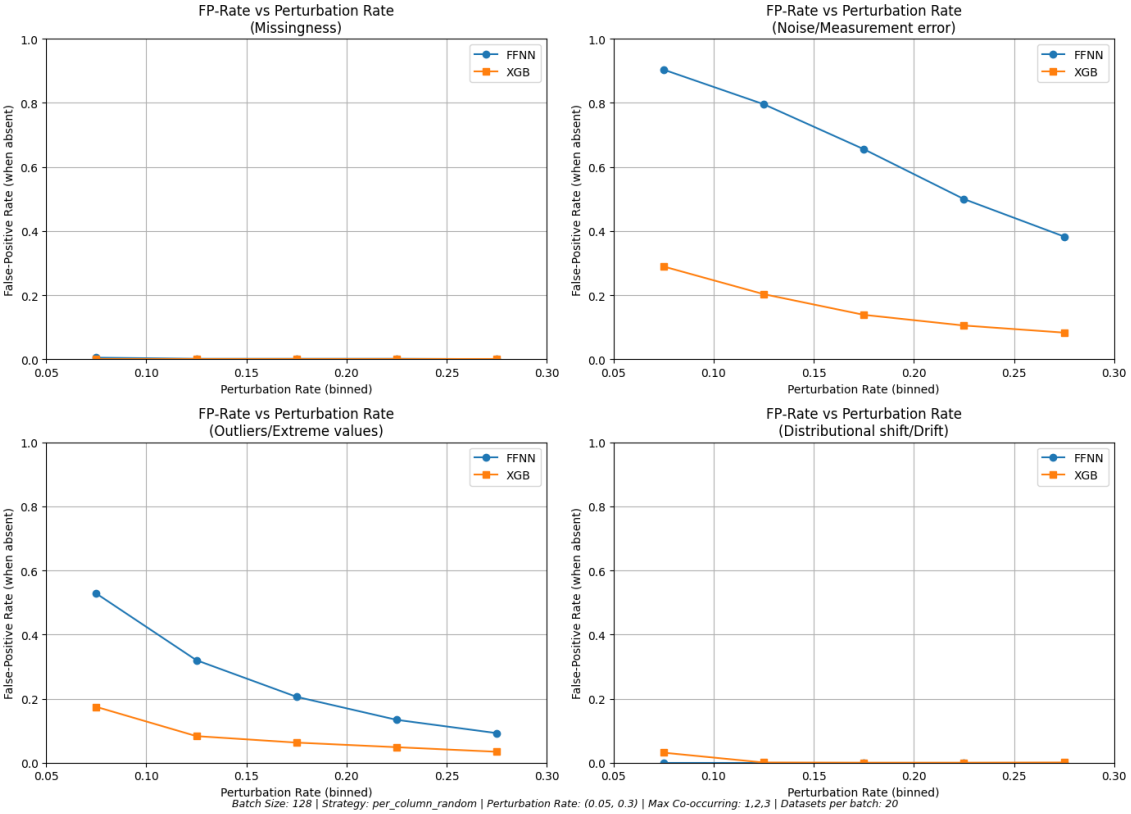


Figure M.10: False-positive rate as a function of perturbation rate for the four degradation types on the restricted range.

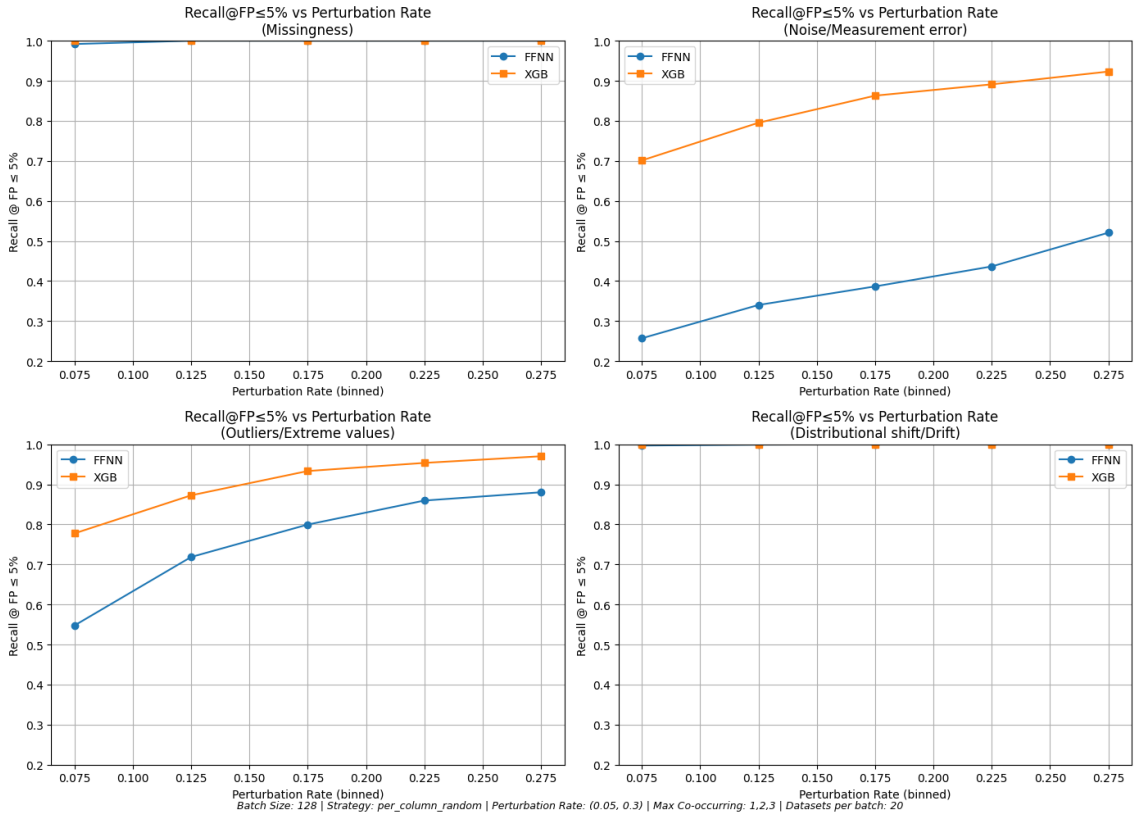


Figure M.11: Recall at a false-positive rate of at most 5% as a function of perturbation rate for the four degradation types on the restricted range.

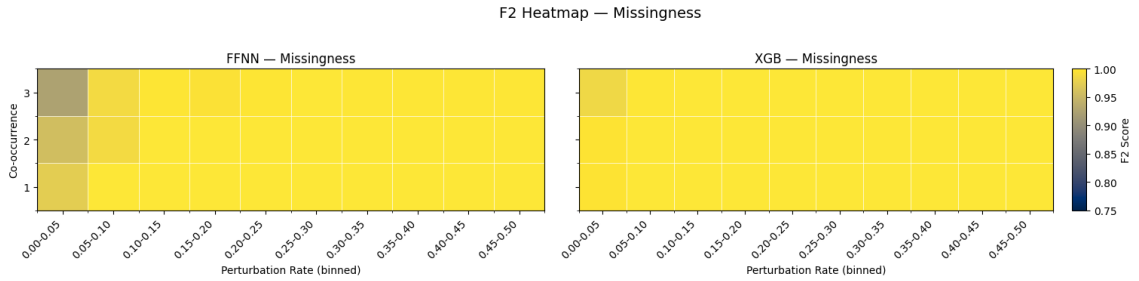


Figure M.12: Label-wise F2 heatmap across degradation rates and co-occurrence for `Missingness` on restricted range.

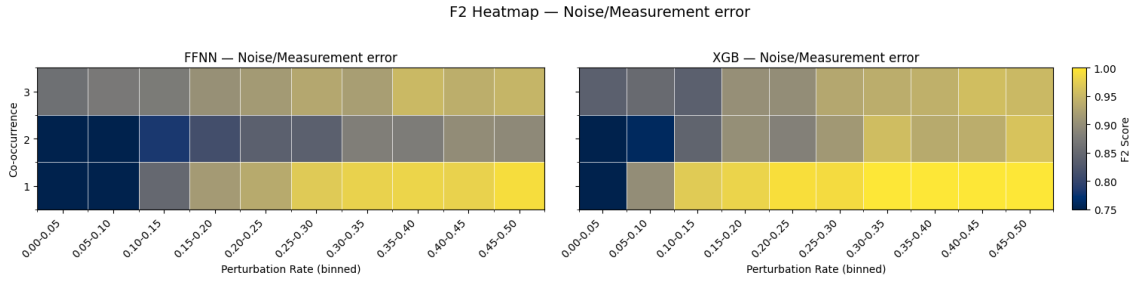


Figure M.13: Label-wise F2 heatmap across degradation rates and co-occurrence for `Noise/Measurement Error` on restricted range.

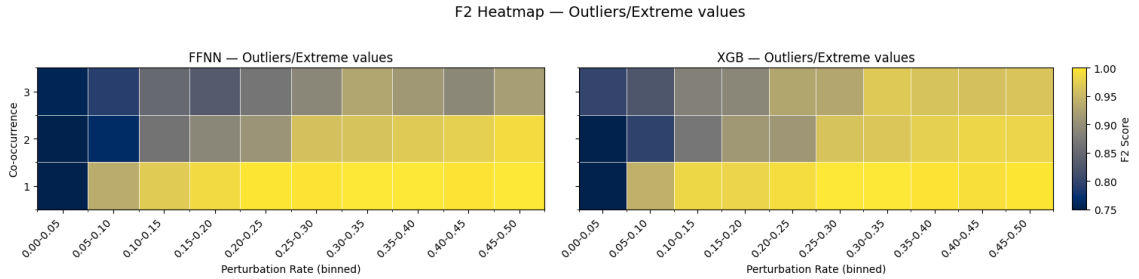


Figure M.14: Label-wise F2 heatmap across degradation rates and co-occurrence for `Outliers/Extreme values` on restricted range.

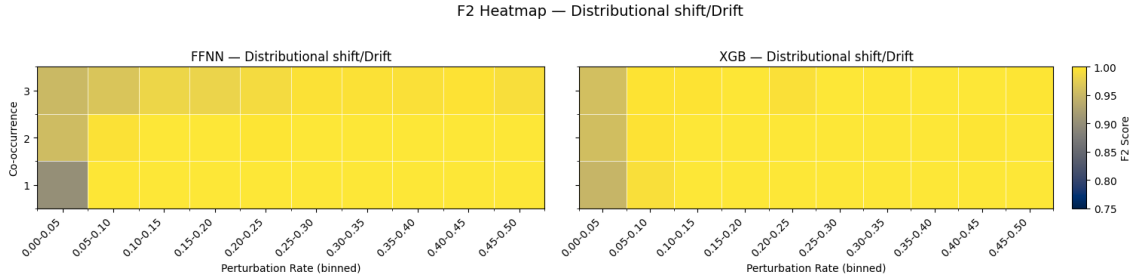


Figure M.15: Label-wise F2 heatmap across degradation rates and co-occurrence for `Distributional Shift/Drift` on restricted range.

N Results alternative degradation approach (`per_cell_random`)

Table N.21: Overall predictive performance metrics of the prior baseline, FFNN, and XGBoost one-vs-rest for the `per_cell_random` approach.

Metric		**Prior baseline**	**FFNN**	**XGBoost OvR**
Macro-F2	Val	–	0.918212	0.959506
	Test	0.499532	0.917636	0.956709
Micro-F2	Val	–	0.935037	0.971684
	Test	0.567800	0.935614	0.970308
Subset Accuracy	Val	–	0.679400	0.818400
	Test	0.104332	0.676976	0.817308
Hamming Loss	Val	–	0.103267	0.056733
	Test	0.431812	0.103552	0.057439
Mean Jaccard Similarity	Val	–	0.863333	0.931444
	Test	0.415724	0.863542	0.930311
Precision (micro)	Val	–	0.853266	0.905433
	Test	0.567782	0.852158	0.905647
Recall (micro)	Val	–	0.957989	0.989789
	Test	0.567806	0.959096	0.987943
Precision (macro)	Val	–	0.838168	0.879215
	Test	0.499519	0.833348	0.877007
Recall (macro)	Val	–	0.950898	0.984794
	Test	0.499541	0.951939	0.981854

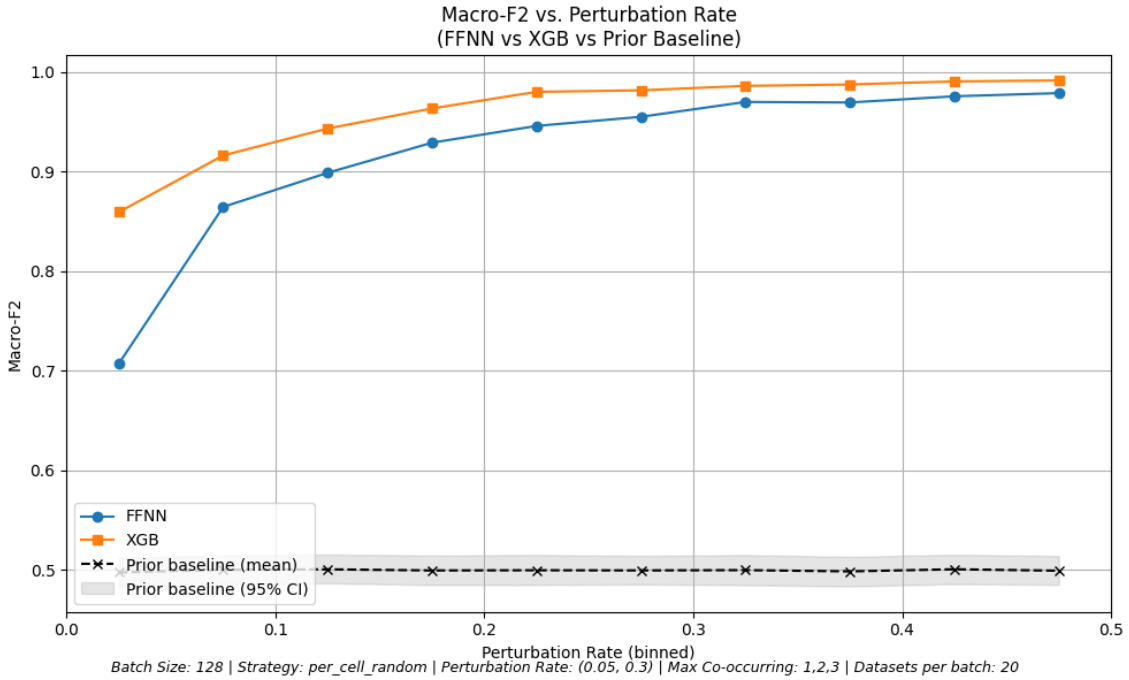


Figure N.16: Macro-F2 across perturbation rates for the FFNN, XGBoost OvR, and the prior baseline under `per_cell_random`. The shaded area denotes the 95% confidence interval of the prior baseline.

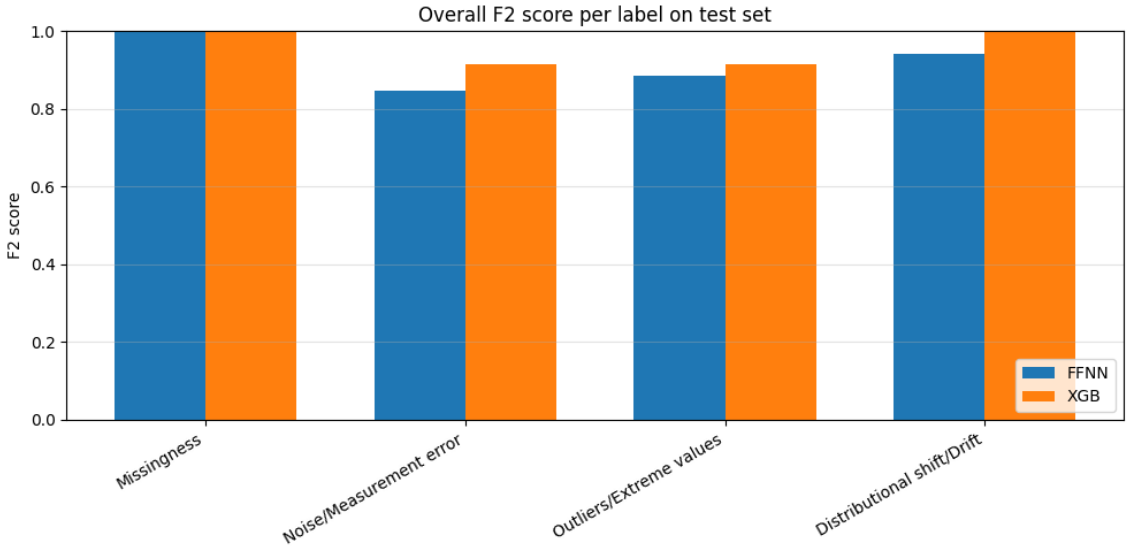


Figure N.17: Overall label-wise F2 scores on the test set for the FFNN and XGBoost one-vs-rest models under `per_cell_random`.

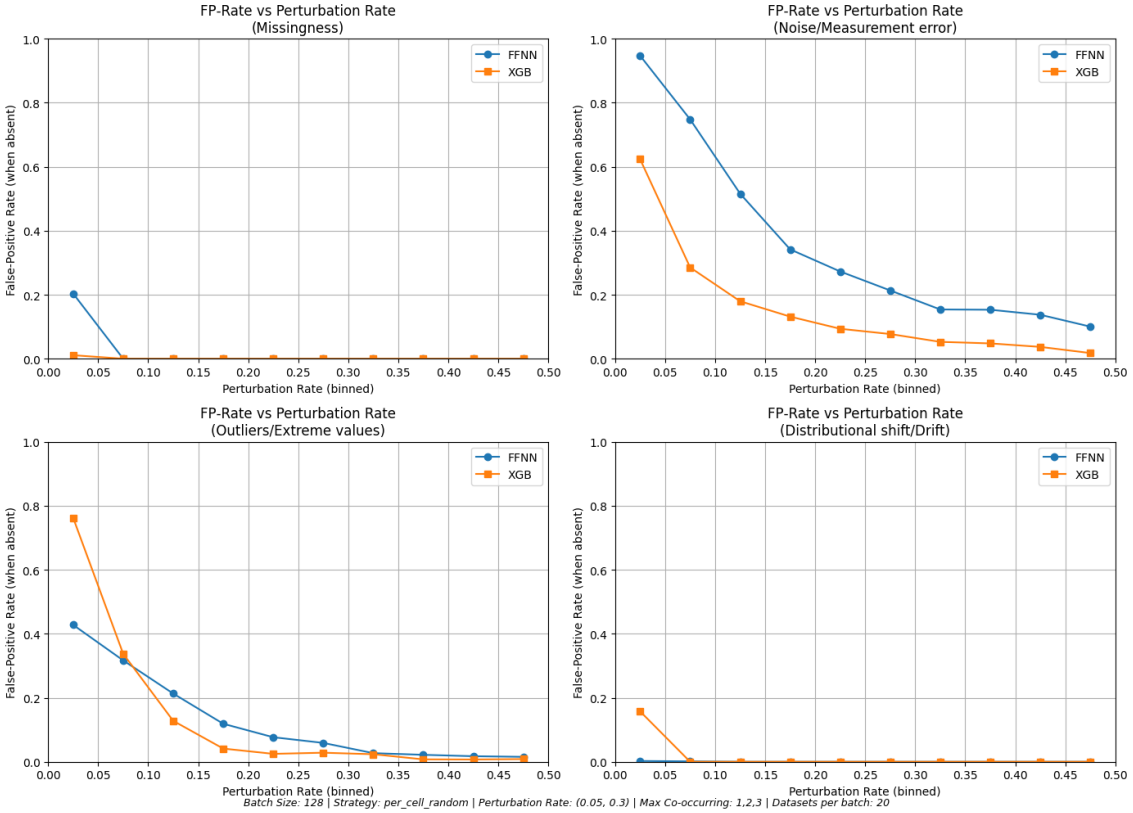


Figure N.18: False-positive rate as a function of perturbation rate for the four degradation types under `per_cell_random`.

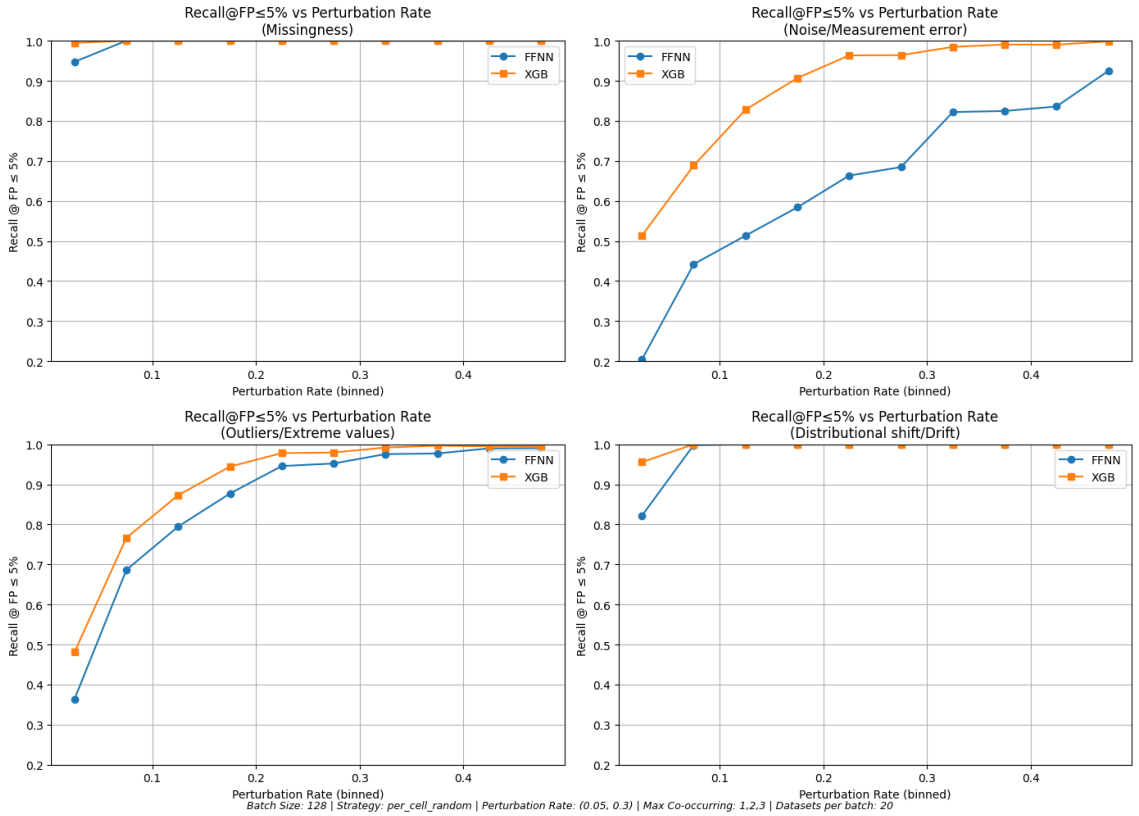


Figure N.19: Recall at a false-positive rate of at most 5% as a function of perturbation rate for the four degradation types under `per_cell_random`.

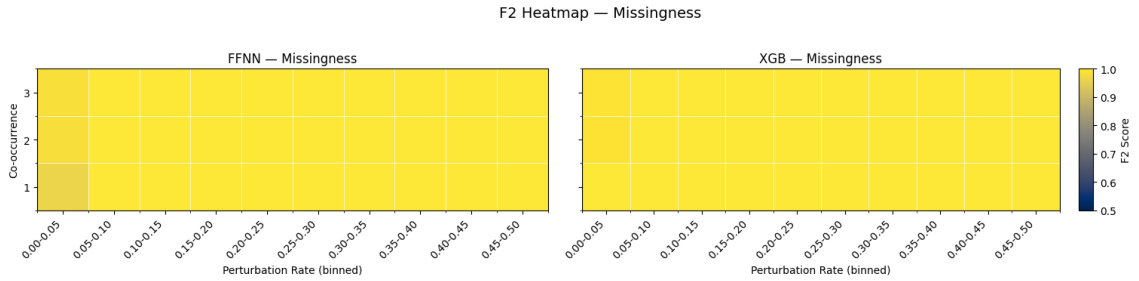


Figure N.20: Label-wise F2 heatmap across degradation rates and co-occurrence for `Missingness` under `per_cell_random`.

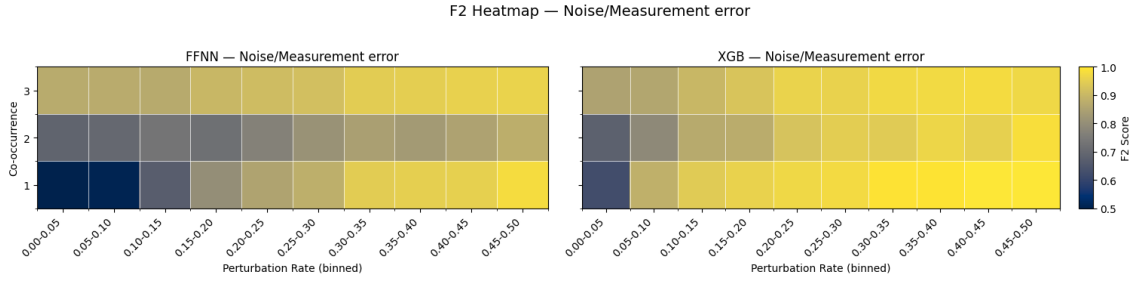


Figure N.21: Label-wise F2 heatmap across degradation rates and co-occurrence for `Noise/Measurement Error` under `per_cell_random`.

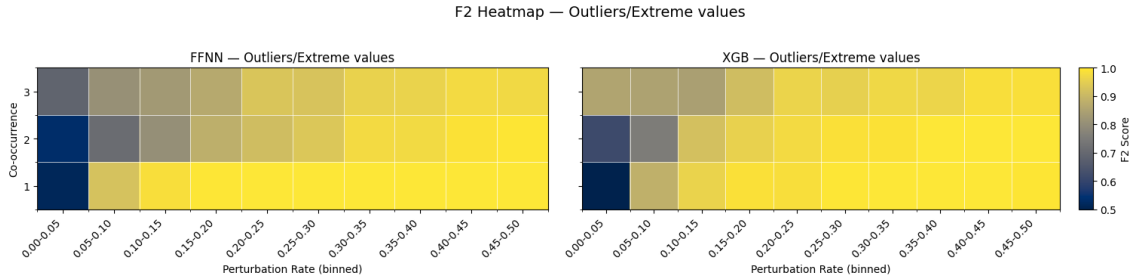


Figure N.22: Label-wise F2 heatmap across degradation rates and co-occurrence for `Outliers/Extreme values` under `per_cell_random`.

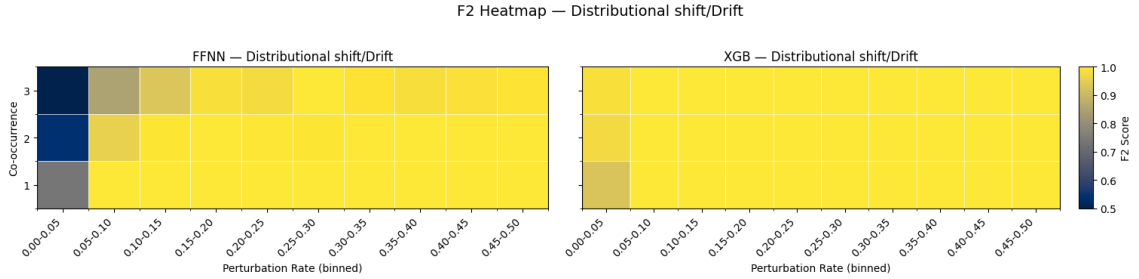


Figure N.23: Label-wise F2 heatmap across degradation rates and co-occurrence for `Distributional Shift/Drift` under `per_cell_random`.