Uncertainty Quantification and Data Provenance for Data Pipeline Security Analysis

Alberta Dadeboe*
albertaod@vt.edu
Virginia Tech
Virginia, USA

Farzaneh Mansourifard*
fmansourifard@thinksense.com
Thinksense, Inc.
Virginia, USA

Shridatt Sugrim
ssugrim@a2labs.com
A2Labs, LLC
Virginia, USA

Abstract

Ensuring data integrity and reliability is essential for real-world applications, especially in automated decision-making and anomaly detection systems. In this study, we introduce a data pipeline augmentation tool that combines Uncertainty Quantification (UQ) techniques with Data Provenance Tracking to detect anomalies and shifts. By leveraging a task runner for pipeline orchestration, our approach ensures scalable, fault-tolerant execution while maintaining full traceability and monitoring at each processing stage.

To validate our framework, we conduct two experiments using the Lawrence Berkeley National Laboratory (LBNL) Fault Detection and Diagnostics (FDD) datasets, focusing on Fan Coil Unit (FCU) operations in HVAC systems. Our experiments assess the pipeline's ability to detect anomalies under different corruption scenarios: (1) Detecting corruption in a single pipeline stage, (2) Capturing inline data corruption.

We integrate statistical tests, such as the Kolmogorov-Smirnov (KS) test, to identify distributional shifts between sequential data batches. Additionally, we apply UQ techniques to quantify uncertainty, enhancing confidence in detected anomalies. The results demonstrate that our work effectively identifies computational corruption, providing a robust and scalable solution for anomaly detection in real-world data pipelines.

Keywords

Uncertainty Quantification, Provenance, Pipeline, Diagnostic, Augmentation

ACM Reference Format:
Alberta Dadeboe, Farzaneh Mansourifard, and Shridatt Sugrim. 2025. Uncertainty Quantification and Data Provenance for Data Pipeline Security Analysis. In *The 7th Workshop on Design Automation for CPS and IoT (DESTION '25), May 6–9, 2025, Irvine, CA, USA*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3722573.3727831

*Both authors contributed equally to this research.

DESTION '25, Irvine, CA, USA

ACM ISBN 979-8-4007-1604-1/2025/05
https://doi.org/10.1145/3722573.3727831

1 Introduction

Lack of sanity checks at each stage of existing data pipelines results in unreliable systems that cannot be fully trusted. Traditional pipelines typically perform error analysis only after the prediction model. As a result, they miss malicious attacks or data corruption that occur earlier in the process. This limitation leaves many critical applications—such as HVAC fault detection, industrial monitoring, and financial analytics—vulnerable to undetected data manipulation, processing errors, or unexpected shifts in distributions including adversarial shifts introduced by malicious actors.

To address these challenges, we introduce data pipeline uncertainty quantification (UQ) and data provenance, a framework that integrates UQ methods and Data Provenance Tracking to systematically detect shifts, anomalies, and potential corruptions in data processing workflows. Unlike conventional anomaly detection approaches, this work applies sanity checks at each stage of the pipeline, enabling early detection of malicious alterations, faulty computations, and distributional shifts. The UQ can be used to alert pipeline users of any unexpected departures from historical norms. Additionally, we integrate Prefect [19], the task runner of our choice for workflow orchestration, ensuring a scalable, fault-tolerant, and traceable data processing system. To evaluate the effectiveness of our framework, we use the Lawrence Berkeley National Laboratory (LBNL) Fault Detection and Diagnostics (FDD) Data Sets [6, 7], focusing on Fan Coil Unit (FCU) operations in HVAC systems. We introduce controlled anomalies to assess the pipeline's ability to detect computational corruption.

We conduct experiments to test different data corruption scenarios: Case 1: Detection of Corrupted Computation in a Single Pipeline Stage, and Case 2: Inline Data Corruption Detection.

We employ Kolmogorov-Smirnov (KS) statistical tests to measure distributional shifts between consecutive batches and leverage UQ techniques to quantify confidence levels in detected anomalies. The results demonstrate that our approach effectively detects computational corruption, enhances trust in automated data pipelines.

2 PROBLEM SETUP

Managing a pipeline and monitoring data to provide assurance on the pipeline and functions performed by the individual stages requires a scalable approach to measuring uncertainty and a task runner that enables seamless data management and workflow orchestration. In this section, we review various UQ approaches and task runners to determine the best approach for our implementation.

2.1 Uncertainty Quantification

Potential Sources of uncertainty include errors in sensor observation, approximation errors during model computations, disturbances in the setup environment[20], and malfunctioning of physical components. This implies that for some specific Quantities of Interest (QOI) obtained from the computational model, we might be uncertain about the corresponding true values [3].This necessitates the need for quantifying this uncertainty between the model predictions and the true values at each stage of a data pipeline. Thus, in quantifying uncertainty, we need to account for all sources of uncertainty[3]. There are many methods of UQ a few of which we consider and discussed with relevant attributes in Table 1. These methods vary in computational cost, sensitivity, ability to reflect specific relationships, and ability to model the specific phenomenon of interest.

It is important to note that our framework can employ any UQ methodology. The chosen method would have to match with the data type and the computational resources available to complete the analysis in bounded time. The use case explored in this implementation of our framework is intended to be suited for connected sequential stages and focuses on shifts in statistical moments, IQR, and entropic uncertainty between pipeline stages. Thus, we aim for non-repetitive sampling, allowing for fewer computations and expedited UQ analysis for diagnostics.

2.2 Workflow Orchestration

Workflow orchestration plays a crucial role in automating and managing data pipelines, ensuring efficiency, scalability, and fault tolerance [14, 15]. Prefect [19] has emerged as a powerful alternative to traditional orchestration tools like Apache Airflow [1, 5, 8], Snowpark [10, 18], AWS Data Pipelines [4, 16, 17], such as MWAA (Managed Workflows for Apache Airflow) and AWS Step Functions. Based on our comparative analysis in table 2., Prefect offers several advantages that make it suitable for modern data workflows. It provides a modern, flexible, and easy-to-use workflow orchestration solution. Its built-in monitoring, scalability, and event-driven capabilities make it a strong alternative to other tools.

3 DATA PIPELINE UNCERTAINTY QUANTIFICATION ARCHITECTURE

This section details the components of our proposed architecture. We describe the propagation of uncertainty through pipeline stages, the compilation of UQ statistics to form a data provenance chain, and the importance of these components.

3.1 Uncertainty Quantification (UQ)

The general framework consists of a combination of a forward process and a backward process, both contributing to estimating uncertainty and making modifications to reduce uncertainty respectively.

The forward process consists of uncertainty propagation, where uncertainty from the input data is characterized in relation to the output. The backward process compares model computations with experiments to efficiently update the statistical model to reduce the output uncertainty [20]. Uncertainty propagation is a method of uncertainty analysis in which the input distribution is propagated through the pipeline stage to produce a measure of uncertainty at the output [3]. It measures the impact of disturbances in the input variables on the system output [12]. Some techniques treat the computational model (pipeline stage in this case) as a black box in a non-intrusive manner while in intrusive techniques, modifications are made to the model during propagation [3]. The backward process is characteristic of an intrusive technique which requires knowledge of the pipeline stage to make updates.

In our architecture, UQ statistics for QOIs are propagated from the input to the output of each pipeline stage. In propagation, an input to a function or data processing performed in a pipeline stage influences the output of that stage. If an unpredicted variation is present in any input or a combination or inputs, the effect is seen on the output as a consequential shift in the output distribution. Each pipeline stage in our framework takes samples representative of the input characteristics and produce outputs for these samples. Over a given sampling window, we derive a probability distribution for the outputs, which reflect uncertainties from the input due to linear or nonlinear dependence on input values. The stages are treated as black boxes, a non-intrusive approach agnostic to the computations within a pipeline stage. This non-intrusive technique allows our implementation to be scalable to a variety of applications and encourage data privacy.

3.2 Data Provenance Chain

Creating a documented trail of the movement of data in the pipeline is essential for security and diagnostics. In our method, data is packaged into a keyed container representation where UQ information for data at each stage is stored in a dictionary 1. Every stage will have its own entry under a different key in the dictionary. If there is an unexpected model output, we can look into its provenance purely by examining the contents of the container that the model output was in. The history of each output is easily identified, This is in contrast to existing pipelines where identifying the source of a particular output is challenging. By capturing the metadata associated with data shifts, we can identify potential change points. These change points might contain a pattern which indicates if the shift is natural or malicious.

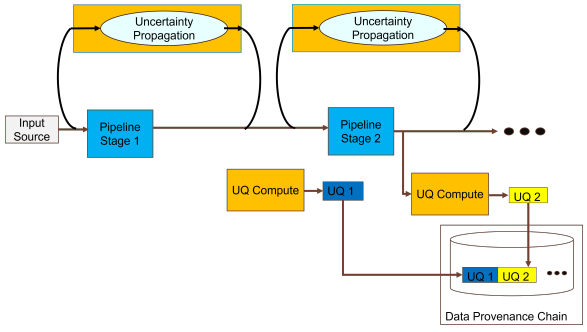

Figure 1: Framework for Computing UQ and Data Provenance Chain:Deep orange color (UQ components).

Table 1: Comparison of UQ Methods with Corresponding Properties

Method	Sampling Approach	Number of Function Evaluations	Function Approximation	Computational Cost
Monte Carlo	Employs multiple input sample combinations	Multiple function computations to approach accuracy	No approximations	Expensive for large number of function evaluations
Polynomial Chaos	Limited number of samples	Fewer function evaluations	Uses approximations based on the mathematical structure of input probability measures	Inexpensive
Emulator Propagation	Limited number of samples	Fewer function evaluations	Uses approximated emulator	Inexpensive

Monte Carlo method takes multiple samples of the input distribution and evaluates the function outputs for each input combination [3]. The sampling is repeated a large number of times, and the resultant point values are then used to evaluate the system's uncertainty [12]. This method can be beneficial for a forward model which is fast, complicated and with many inputs since the sampling does not depend on the dimension of the input space or model complexity [3]. The caveat is that it requires multiple computations to arrive at a sufficient accuracy. This can be reduced by using the quasi-Monte Carlo approach[3]. Another disadvantage is that sampling does not reflect the functional dependence of outputs on inputs and as such cannot take advantage of mathematical of structural optimizations to a function to expedite computations [3]. This is a problem solved with the Polynomial Chaos approach. Polynomial Chaos makes use of function approximations based on the mathematical structure of input probability measures. It first describes stochastic functions, variables, or processes with respect to a basis in a suitable vector space. The choice of basis can be adapted to the distribution of the input parameters . It then computes the coordinates in this representation using functional analysis machinery, such as orthogonal projections and error minimization[3].

Table 2: Comparison of Workflow Orchestration Tools Across Various Dimensions

Category	Prefect	Apache Airflow	Snowpark	AWS Data Pipeline	MWAA	AWS Step Functions
Ease of Setup [1]	Easy	Moderate	Easy	Easy	Complex	Easy
Scalability[2]	High	Manual	High	High	High	High
Flexibility [3]	Flexible, integrates with data sources	Flexible for batch	Flexible for batch and analytics	Moderate flexibility	Flexible, supports third-party integration	Less flexible, AWS-focused
Monitoring and Logging [4]	Built-in Prefect Cloud, real-time	Custom, less real-time	Built-in, requires tools for real-time	Built-in CloudWatch	Built-in CloudWatch, requires configuration	Built-in CloudWatch
Community and Support [5]	Small, growing	Large	Small, growing	AWS-only	Large	AWS-only
Version Control [6]	Built-in	Not built-in	Not built-in, external handling	Not built-in, external handling	Not built-in, external handling	Not built-in, external handling
Code Complexity [7]	Low	High	Moderate	Moderate	High	Moderate
UI/UX [8]	Built-in Prefect Cloud	Built-in web interface	Built-in for SQL and job orchestration	Built-in user interface	Not built-in, requires Airflow UI	Built-in visual development interface
Handling Streaming Data [9]	Yes, event-driven, supports real-time	No, batch-oriented	No, batch-oriented	No, batch-oriented	Yes, batch and streaming	Yes, task-based workflows
Ease of Cloud Deployment [10]	Easy	Hard	Easy	Moderate	Moderate	Easy

[1] How easy or difficult it is to set up and configure the workflow orchestration tool.[2] The ability of the tool to handle increasing workloads and efficiently manage distributed execution.[3] The extent to which the tool supports various data sources, third-party integrations, and different workload types. [4] Availability of built-in monitoring, real-time tracking, and logging capabilities. [5] The level of community adoption, availability of documentation, and official support. [6] Whether the tool has built-in version control. [7] The level of technical expertise required to implement and maintain workflows. [8] The quality and accessibility of the tool's user interface for designing and managing workflows. [9] The tool's capability to handle both batch processing and real-time (streaming) data workflows.[10]How well the tool supports cloud-based deployment and integrates with cloud-native services.

3.3 Pipeline Demonstration

To effectively detect anomalies and track data shifts, we build a pipeline using Prefect[19]. This allows us to smoothly orchestrate each stage of our data pipeline, even when dealing with complex, real-world data challenges.

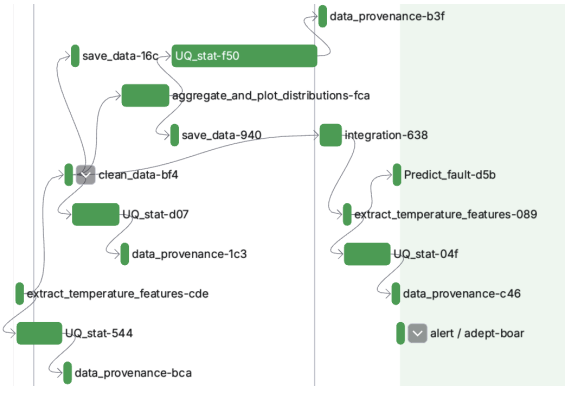

Figure 2: Prefect Workflow for Fault Detection and Uncertainty Quantification (UQ).

Figure 2 shows the Prefect workflow used for fault detection and uncertainty quantification (UQ). The pipeline consists of multiple interconnected stages (tasks). To explore the application of Uncertainty Quantification (UQ) in a Prefect-based pipeline, we design an environmental data processing workflow. This pipeline ingests temperature data and processes it through multiple stages to identify potential faults along the way.

Then, we apply uncertainty quantification (UQ-stat) methods to measure variability and assess confidence at each pipeline stage. To keep track of every transformation, we populate the data container with provenance details, making it easier to trace changes throughout the pipeline. The cleaning stage removes inconsistencies to improve data quality before it reaches the model. The next stage aggregates and transforms the records from multiple sources. Finally, an alerting mechanism monitors the UQ and notifies the user if significant anomalies are detected, helping catch potential faults early.

4 EXPERIMENTAL SETUP

In this study, we conduct two experiments to evaluate the effectiveness of statistical methods and UQ in detecting anomalies, focusing on different types of data corruption to evaluate their impact on distributional shifts and system faults. To ensure a realistic and controlled testing environment, we base our analysis on a dataset that reflects actual HVAC system behavior. To illustrate the practicality of our framework, the experiments were repeated for data retrieved from UAVs Under Normal Operations and Cyberattacks [9]. Results can be found in this Github reopsitory.

We use the FDD Datasets [6, 7] which comes from a fully instrumented testbed that replicates real-world HVAC system operations under both normal and faulty conditions. It contains a variety of sensor readings that track temperature, humidity, pressure, airflow rates, and equipment control states.

We focus our analysis on temperature-related features across six sequential batches, each representing 30 minutes of recorded data, resulting in a 3-hour observation window. To detect potential anomalies, we investigate shifts in temperature distributions between

batches. Additionally, we incorporate Uncertainty Quantification (UQ) methods, as presented in Section 3.1, to further assess data variability and confidence in detected shifts.

To detect these distributional shifts, we apply Kolmogorov-Smirnov (KS) test, a non-parametric statistical test to compare the distributions of two independent samples by measuring the maximum deviation between their cumulative distribution functions (CDFs) [2, 11, 13]. The test is particularly useful for detecting differences in location, shape, or spread between two datasets, making it a powerful tool for identifying shifts in data distributions. A low p-value (< 0.05) suggests that the two samples are drawn from significantly different distributions, indicating a distributional shift or anomaly in the data.

Instead of comparing all possible batch pairs, we specifically focus on consecutive batch comparisons (e.g., Batch 1 vs. 2, Batch 2 vs. 3, etc.). This method is better suited for detecting gradual process drifts rather than abrupt isolated changes. If the goal is to detect abrupt shifts rather than gradual drifts, one approach is to compare all batch pairs instead of limiting the analysis to consecutive batches. This broader comparison allows for the identification of sudden deviations in distribution that may not be evident in sequential comparisons alone.

In our experiments, we observe a high variability in p-values across different batch comparisons, we apply a logarithmic scale to the p-values for better visualization.

4.1 Case 1: Single-Stage Computational Corruption Detection

This experiment considers scenarios where all pipeline stages operate under normal conditions, except for a single stage where an anomaly introduces unintended shifts in the data distribution. Such anomalies can emerge due to system faults, adversarial manipulations, or parameter misconfigurations.

Threat Model. In a pipeline designed for monitoring data from a physical system, various computations occur at different stages to ensure data integrity and reliable decision-making. A potential adversary, could instead introduce shifts at specific stages, modifying how the data is preprocessed. This manipulation can result in unnoticed but systematic errors, triggering false alarms or disrupting fault detection mechanisms, leading to incorrect responses in automated decision systems.

In this experiment, we aim to detect computational anomalies occurring in a specific stage of the pipeline while ensuring that only fault-free data are utilized. Anomalies in a pipeline stage can arise due to various factors, including adversarial manipulation, system faults, or unintended parameter changes. To evaluate our detection approach, we introduce a controlled shift in the data cleaning stage of pipeline by modifying the normalization method at a specific batch. To simulate an attacker-induced shift, we intentionally alter the normalization technique used for Batch 3 in the cleaning stage while keeping the rest of the batches unchanged. The processing scheme is as follows: Batches 1, 2, 4, 5, and 6: Standard Normalization (mean subtraction and variance scaling), while Batch 3: Min-Max Scaling (rescaling values to a fixed range). By analyzing variations between consecutive batches at the data cleaning stage, this experiment highlights how a localized computational shift affects downstream processing. The objective is to determine whether our pipeline can accurately identify deviations introduced within a single pipeline stage. In addition, we assess the impact of this anomaly on feature distributions. By quantifying statistics and distributional shift, we establish a basis for automated anomaly detection in real-world data pipelines.

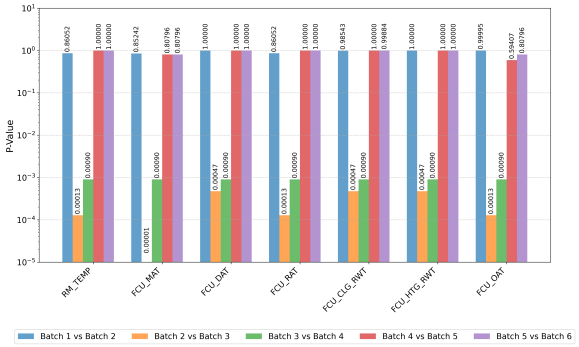

Figure 3: KS Test Results for Batch Comparisons in case 1. The y-axis shows log-scaled p-values, indicating the statistical similarity between batches(x-axis). Values on top of the bars indicate the exact p-values (p-values < 0.05 suggest significant distributional shifts, particularly in Batch 3 comparisons.

Figure 3. shows the p-values of KS tests in consecutive batch comparisons. The logarithmic scale highlights distributional differences, where smaller p-values indicate statistically significant shifts. As Batch 3 was intentionally altered, we observe significantly lower p-values when comparing Batch 2 vs. 3 and Batch 3 vs. 4. This confirms that the modification introduced in Batch 3 was detected by the pipeline with distributional changes.

Figure 4 illustrates the batch-wise statistical analysis for two features, highlighting the distributional changes occurring across different batches. In this analysis, we focus on two features, RM-TEMP and FCU-OAT (For other features analysis, refer to this Github repository). The change observed in Batch 3 confirms the presence of a significant shift in data distribution, which was intentionally introduced. We notice a drop in standard deviation (STD), variance, and IQR at Batch 3, meaning the data became less spread out, suggesting a loss of variability due to the introduced corruption. Together, these changes confirm that something unusual happened in Batch 3. After Batch 3, all return to their usual patterns, reinforcing their reliability in detecting shifts in the data. Additionally, entropy increases from Batch 2 to Batch 3. This suggests that the distribution of values in these batches is different.

By leveraging statistical measures, entropy-based evaluations and comparison tests, this framework can detect process drifts and unexpected interventions without requiring predefined anomaly thresholds. The ability to detect such distributional shifts makes this approach valuable for early fault detection, proactive maintenance, and ensuring data integrity in automated systems. The distributional shift is observed across all features, highlighting the pervasiveness of the anomaly rather than being isolated to a single variable. If a fault detection model were designed to monitor only

temperature related features, it might overlook the broader distribution shift. However, the alerting framework is feature agnostic, meaning it identifies anomalies in any feature rather than relying on a predefined set of monitored variables.

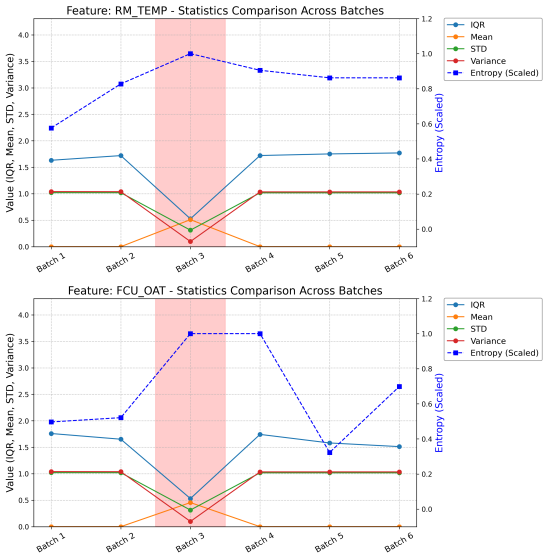

Figure 4: Case 1. Statistical Feature Comparison across Batches. The figure presents IQR, Mean, and STD, Variance, and scaled Entropy across different batches. A significant deviation is observed in Batch 3, where all statistical measures drop, followed by a recovery in Batch 4 and beyond.

4.2 Case 2: Inline Data Corruption Detection

This experiment accounts for scenarios where all pipeline stages are in ideal operating conditions but the ingestion of data is corrupted as a result of an attacker introducing false readings manually to mimic a non-existent fault in the physical system being monitored.

Threat Model. Given a pipeline setup to monitor the data derived from a physical system, a sequence of computations are performed at various stages in the pipeline. A potential attacker with no access to the pipeline modification environment and aims to launch an attack without tampering with the pipeline computations can target the ingestion point or the data supply. The goal would be to manipulate the data at the point of ingestion to trigger multiple false alerts at the decision-making point. The result is corrupted data interpretations made by the model.

We aim to identify shifts occurring in the UQ statistics through different batch processes to determine the point at which corrupted data enters the pipeline. Fault-free data is passed through the pipeline for a given time, followed up with faulty data, and then shifts in UQ statistics between batches are observed. The statistics are averaged across all stages to create a unified impression of data changes across the pipeline as a single unit. The values of mean in the graph had a much higher range than the other statistics so We scaled the mean to make shifts for all statistics visible.

Depending on the applied attack, some features will show more distributional shifts than others, allowing the user to determine the specific subsystem within the whole physical system that is being targeted. In the attack type used for this experiment, a sensor bias in zone air temperature measurements [7], the trends indicate some specific features are affected the most. In the physical system, it is likely that components associated with these features are being targeted for an attack. Other features might not be as much affected and thus are not indicated as targets. For this data set, we focus on 2 features for our illustrations, room temperature and fan coil unit outdoor air temperature. Based on observations from this experiment, room temperature (RM_TEMP) is affected the most. We observe that in its plots, when corrupted data is inserted, all statistics will have a shift, with at least one statistic having a sharp shift, at the batch where corrupt data is inserted. This pattern is not reflected in the plots for fan coil unit outdoor air temperature (FCU_OAT). For trends in other features, refer to this Github repository.

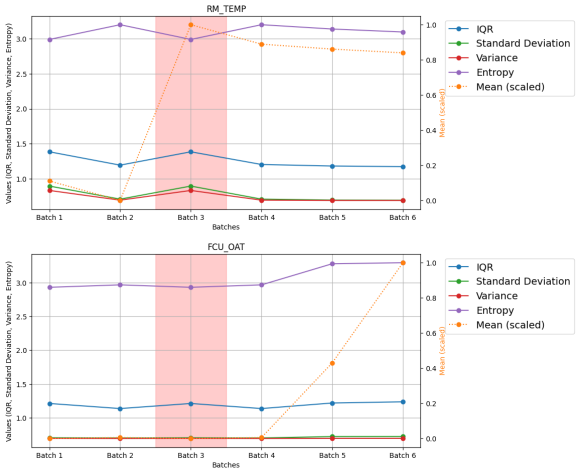

Figure 5: Case 2. Statistical feature comparison across batches for 1 hour of fault-free data processing followed by 2 hours of faulty data processing. The applied fault, a sensor bias for zone temperature, affects RM_TEMP and shows shifts in all UQ statistics and a sharp shift in mean at Batch 3, the point of data corruption. FCU_OAT is not affected as much and thus there are no significant shifts in UQ statistic at Batch 3.

Figure 5 shows the batch-wise statistical trend for pipeline processing of a timed sequence of fault-free and faulty data. Fault-free data is sent into the pipeline for 1 hour, followed by 2 hours of faulty data to mimic an attacker-induced corrupt data ingestion. The pattern across all affected plots show that when some new data is introduced (faulty or fault-free), there is a slight rise in entropy and a slight drop in all other UQ statistics. After this, all statistics stabilize at an almost fixed value. This means there is an expected shift in statistics with new data entry. The shift should be slight across all statistics. If any statistic shows a sharp change before the statistics become stabilized, it could indicate corrupt data.

From observations of the RM_TEMP plot, the transition from Batch 1 to Batch 2 shows all statistics dropping slightly with entropy

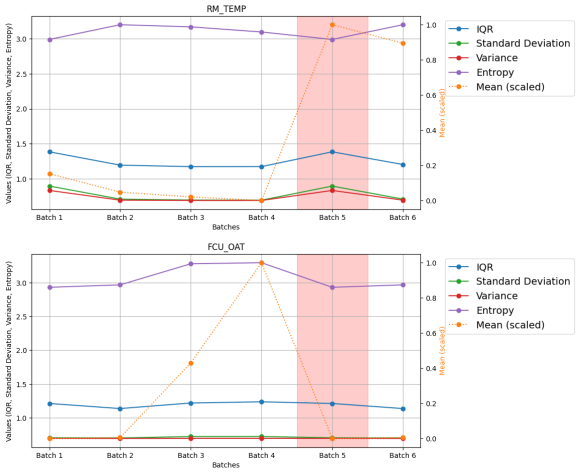

Figure 6: Case 2. Statistical feature comparison across batches for 2 hours of fault-free data processing followed by 1 hour of faulty data processing. Fault affects RM_TEMP indicated at Batch 5. FCU_OAT is not affected as much.

rising slightly as expected for a new data entry. From Batch 2 to Batch 3, the statistics do not stabilize and we see a sharp rise in the mean value. This indicates that at Batch 3, corrupted data has been introduced. We can conclude that new corrupt data has entered the pipeline and as such an expected slight rise and drop in entropy and all other UQ statistics is shown from Batch 3 to Batch 4. After the corrupt data continues to be processed within the pipeline, the statistics stabilize for the remaining Batches. This trend is not identified in the plot for FCU_OAT. Variance and Standard deviation do not make any significant shifts across all batches, indicating that FCU_OAT is not primarily affected by corrupt data entry.

In figure 6, fault-free data is sent into the pipeline for 2 hours followed by 1 hour of faulty data. From observations of the RM_TEMP plot, the transition from Batch 1 to Batch 2 shows all statistics dropping slightly with entropy rising slightly as expected for a new data entry. From Batch 2 to Batch 4, the statistics stabilize and we do not see a considerable sharp change in any UQ statistics. From Batch 4 to Batch 5, there is a sharp rise in the mean value and all other statistics also see a shift. This indicates that at Batch 5, corrupted data has been introduced. We can conclude that new data has entered the pipeline and as such an expected slight rise in entropy and a slight drop in all other UQ statistics is shown from Batch 5 to Batch 6. This trend is not identified for FCU_OAT.

5 CONCLUSION AND FUTURE WORK

In this paper, we apply a combination of uncertainty quantification and data provenance to implement a data pipeline augmentation tool that dynamically detects shifts in statistics at each processing stage to provide diagnostic information for further security analysis. We do this by (1) computing the statistical moments, IQR and entropy of the output distribution of every pipeline stage, (2) Capturing these UQ statistics in a keyed dictionary, and (3) Analyzing the shifts in UQ statistics to detect and identify distributional shift during data processing through the stages. While these systems offer advantages, their implementation does present certain challenges. Addressing the issue of resource overhead for these tools in a resource constrained environment is pertinent. In subsequent work, we plan to implement emulator models of resource intensive computations and compute UQ statistics based on the output distributions of these models.

Acknowledgments

This report is based on work supported by the National Aeronautics and Space Administration (NASA) under STTR Phase I Contract No. 80NSSC24PB234. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NASA. The authors extend their gratitude to Dr. Paul Ampadu from Virginia Tech and the entire teams at A2 Labs, LLC and Thinksense, Inc. for their assistance and contributions.

References

[1] Censius AI. 2024. Prefect vs Airflow: Which One to Choose for Your Workflow Orchestration? https://censius.ai/blogs/prefect-vs-airflow Accessed: 2024-02-28.
[2] Vance W Berger and YanYan Zhou. 2014. Kolmogorov–smirnov test: Overview. *Wiley statsref: Statistics reference online* (2014).
[3] National Research Council. 2012. *Assessing the Reliability of Complex Models: Mathematical and Statistical Foundations of Verification, Validation, and Uncertainty Quantification.* The National Academies Press, Washington, DC. https://doi.org/10.17226/13395
[4] Gareth Eagar. 2021. *Data Engineering with AWS: Learn how to design and build cloud-based data transformation pipelines using AWS.* Packt Publishing Ltd.
[5] Apache Software Foundation. 2025. Apache Airflow Documentation. https://airflow.apache.org/ Accessed: 2025-02-28.
[6] Jessica Granderson, Guanjing Lin, Yimin Chen, Armando Casillas, Piljae Im, Sungkyun Jung, Kyle Benne, Jiazhen Ling, Ravi Gorthala, Jin Wen, et al. 2022. *LBNL fault detection and diagnostics datasets.* Technical Report. DOE Open Energy Data Initiative (OEDI); Lawrence Berkeley National ....
[7] Jessica Granderson, Guanjing Lin, Yimin Chen, Armando Casillas, Jin Wen, Zhelun Chen, Piljae Im, Sen Huang, and Jiazhen Ling. 2023. A labeled dataset for building HVAC systems operating in faulted and fault-free states. *Scientific data* 10, 1 (2023), 342.
[8] Bas P Harenslak and Julian De Ruiter. 2021. *Data pipelines with apache airflow.* Simon and Schuster.
[9] Samuel Chase Hassler, Umair Ahmad Mughal, and Muhammad Ismail. 2024. Cyber-Physical Intrusion Detection System for Unmanned Aerial Vehicles. *IEEE Transactions on Intelligent Transportation Systems* 25, 6 (2024), 6106–6117. https://doi.org/10.1109/TITS.2023.3339728
[10] Snowflake Inc. 2025. Snowflake Snowpark Documentation. https://www.snowflake.com/en/data-cloud/snowpark/ Accessed: 2025-02-28.
[11] K Kendall and Maurice George. 2008. Kolmogorov–Smirnov Test. *The Concise Encyclopedia of Statistics* (2008), 283–287.
[12] Hiromitsu Kumamoto and Ernest J. Henley. 1996. *Uncertainty Quantification.* IEEE, 535–572. https://doi.org/10.1109/9780470546277.ch11
[13] Frank J Massey Jr. 1951. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American statistical Association* 46, 253 (1951), 68–78.
[14] Mihhail Matskin, Shirin Tahmasebi, Amirhossein Layegh, Amir Hossein Payberah, Aleena Thomas, Nikolay Nikolov, and Dumitru Roman. 2021. A Survey of Big Data Pipeline Orchestration Tools from the Perspective of the DataCloud Project.. In *DAMDID/RCDL (Supplementary Proceedings).* 63–78.
[15] Anthony Mbata, Yaji Sripada, and Mingjun Zhong. 2024. A survey of pipeline tools for data engineering. *arXiv preprint arXiv:2406.08335* (2024).
[16] Amazon Web Services. 2024. *What is AWS Data Pipeline?* https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/what-is-datapipeline.html Accessed: 2024-02-28.
[17] Amazon Web Services. 2025. AWS Data Pipeline Documentation. https://aws.amazon.com/datapipeline/ Accessed: 2025-02-28.
[18] RST Software. 2024. What is Snowflake Snowpark? https://www.rst.software/blog/what-is-snowflake-snowpark Accessed: 2024-02-28.
[19] Prefect Technologies. 2023. Prefect: The Modern Data Workflow Orchestration Tool. https://docs.prefect.io.
[20] Juan Zhang, Junping Yin, and Ruili Wang. 2020. Basic Framework and Main Methods of Uncertainty Quantification. *Mathematical Problems in Engineering* 2020 (Aug. 2020), 1–18. https://doi.org/10.1155/2020/6068203