
DETECTING SILENT FAILURES IN MULTI-AGENTIC AI TRAJECTORIES

Divya Pathak
IBM Research
Divya.Pathak2@ibm.com

Harshit Kumar
IBM Research
harshitk@in.ibm.com

Anuska Roy
IIIT Bangalore
anuska.roy@iiitb.ac.in

Felix George
IBM Research
Felix.George@ibm.com

Mudit Verma
IBM Research
mudiverm@in.ibm.com

Pratibha Moogi
IBM Research
pratibha.moogi@ibm.com

ABSTRACT

Multi-Agentic AI systems, powered by large language models (LLMs), are inherently non-deterministic and prone to silent failures such as drift, cycles, and missing details in outputs, which are difficult to detect. We introduce the task of anomaly detection in agentic trajectories to identify these failures and present a dataset curation pipeline that captures user behavior, agent non-determinism, and LLM variation. Using this pipeline, we curate and label two benchmark datasets comprising **4,275 and 894** trajectories from Multi-Agentic AI systems. Benchmarking anomaly detection methods on these datasets, we show that supervised (XGBoost) and semi-supervised (SVDD) approaches perform comparably, achieving accuracies up to **98%** and **96%**, respectively. This work provides the first systematic study of anomaly detection in Multi-Agentic AI systems, offering datasets, benchmarks, and insights to guide future research.

Keywords Multi-Agentic AI · Anomaly Detection · Silent Failures · Agentic Trajectories · Non-deterministic Systems

1 Introduction

The adoption of AI agent systems has been rapidly increasing across various industries, such as insurance, customer service chatbots, etc Bhattacharya et al. [2025], McKinsey & Company [2025], DamcoGroup [2025], AiMultiple Research [2025]. An AI agent consists of a set of tools, large language models (LLMs), and system prompts. These LLMs drive agent decision-making, including selecting tools, invoking other agents, reasoning, and planning, based on system prompts, input queries, context from other agents/tools, and available tools. Due to the reliance on LLMs for decision-making, AI Agents are inherently **non-deterministic** Bronsdon [2025], Cemri et al. [2025], Hammond et al. [2025] i.e. the sequence of execution order of steps (trajectory) in the agent workflow is decided dynamically. Moreover, changing the system prompt or the LLM for the same input may result in a different trajectory. Furthermore, even for the same input query, using the same model and system prompt, the agent’s execution may vary, thereby leading to different trajectories, sometimes erroneous too.

Due to this non-determinism and dynamic nature, Agentic AI systems are more prone to failures compared to traditional microservice based applications Bronsdon [2025], Cemri et al. [2025]. Unlike microservice applications, where errors are typically explicit (with error codes) and predictable, failures in agentic systems can often be **silent**, occurring without generating clear error signals while still deviating from the intended behavior. Table 1 summarizes the *different failures scenarios* that can arise in such systems. For instance, a drift is when an agent chooses next agent A and tool B instead of agent D and tool E. Such behaviors are observed through Agentic AI traces (Figure 1 (B)), which capture the complete execution workflow of an input request across agents, tools, and LLMs. We use the terms trajectories and traces interchangeably throughout this paper.

Effective detection mechanisms are essential, as such *silent failures* can rapidly escalate operational costs, including **computational resources, token usage, and time**. We define **anomaly detection** as the task of detecting the existence

Table 1: Silent Failure Scenarios in Multi-Agentic AI Systems

Failure	Description
Drift	The agent diverges from the intended path, selecting tools or subsequent agents that are irrelevant for an input query.
Cycles	The agent repeatedly invokes itself or other agents/tools by re-planning resulting in redundant loops, wasted computation.
Missing Details in Final Output	The agent returns a response without errors, but misses crucial information requested in the input query.
Tool Failures	External tools(APIs) may fail silently, return unexpected results, hit rate limits—issues that the agent may not detect or handle gracefully.
Context Propagation Failures	Failure in propagating the correct context to dependent agents/tools.

of such silent failures in Agentic AI trajectories. Detecting anomalies and evaluating these detection techniques in Agentic AI systems requires datasets that capture diverse types of failures. To the best of our knowledge *no publicly available datasets* currently exist that capture the diverse behaviors and failure scenarios of Multi-Agentic AI systems. Although anomaly detection is extensively explored in areas like microservices and networks Zhang et al. [2024], Chen et al. [2023], Rodriguez et al. [2024], established techniques for AI Agents remain unexplored. Recent work like He et al. [2025] applies graph-based methods but offers limited evaluation. To bridge this gap, this work makes the following three key contributions:

- **Dataset Curation Pipeline (Section 2):** We design a comprehensive pipeline for curating datasets from agentic traces for the anomaly detection task. This includes collecting traces that capture *user behavior, agent non-determinism, and LLM model variation*, extracting relevant features, labeling traces into two binary classes (normal/anomaly). Using this pipeline, we curate labeled datasets from two Multi-Agentic AI systems, comprising **4,275** and **894** datapoints. We plan to open-source these datasets to foster community-driven benchmarking and further research on anomaly detection in agentic systems.
- **Benchmarking Anomaly Detection Methods (Section 3):** We leverage the two datasets to benchmark anomaly detection methods, demonstrating their utility for both research and practical applications. Among the evaluated methods, XGBoost Chen and Guestrin [2016], a supervised binary classification approach, achieved accuracies of **98%** and **94%** on the two datasets. While, SVDD Tax and Duin [2004], a one-class semi-supervised method, obtained **96%** and **89%**, respectively.
- **Detailed Error Analysis and Insights (Section 3):** We conduct a thorough analysis of detection results, identifying causes of misclassifications, areas for improvement, and directions for future research in Agentic AI systems.

These contributions offer researchers and practitioners practical guidance for building robust and reliable AI-Agentic systems, and provides insights to current limitations and future directions for developing approaches to detect silent failures.

2 Dataset Curation Pipeline for Multi-Agentic AI Systems

In this section we present a *dataset curation pipeline* used to curate datasets for anomaly detection task, which can also be extended to other tasks such as drift detection and cycle detection. Figure 1 illustrates the curation pipeline consisting of three components: (1) collect Agentic AI traces, (2) extract key features from these traces and (3) define labeling criteria to label these traces into two binary classes - normal and anomaly. Finally, we apply this pipeline to construct datasets from two representative Multi-Agentic AI systems: (1) Stock Market Assistant and (2) Research Writing Assistant. This pipeline can be readily extended to other Agentic AI systems.

2.1 Agentic Traces: The Building Blocks

The first step in our dataset curation process is the collection of agentic traces. Just as traces in microservice based application represent the end-to-end flow of a request across distributed services, Agentic AI traces capture the complete execution workflow of an input request across agents, tools, and LLMs. For instance shown in Figure 1 (B), a request executed by Agent 1 is recorded as a span, which triggers Tool 1 and an LLM call, forming child spans linked to the parent Agent 1 span. To collect agentic traces, we instrument Agentic AI applications using OpenTelemetry distributed tracing OpenTelemetry Project [2021], a standard for capturing traces in distributed systems.

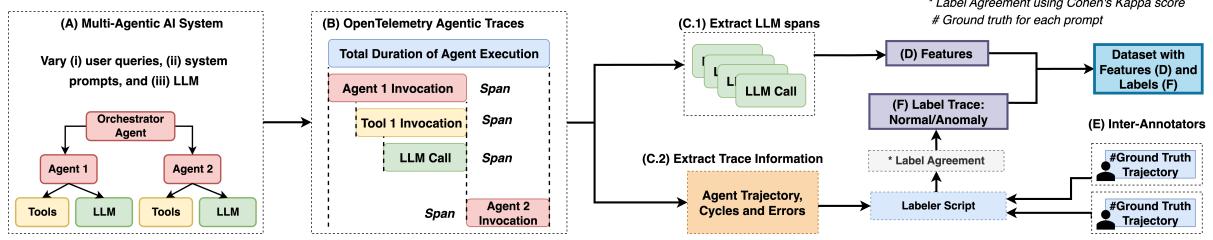


Figure 1: Dataset Curation Pipeline for Multi-Agentic AI Systems

Next, we carefully generate these traces by simulating three key factors that influence agentic behavior: **Non-determinism of AI agents**: Executing the same or similar query multiple times may result in different trajectories due to variations in reasoning and planning. **LLM model**: Acts as the decision-making engine of the AI agent, shaping its reasoning and planning. **User behavior**: Each user or developer has a distinct style of writing system prompts, which influences agent behavior.

To systematically capture these factors in the traces, we vary three key variables: the **input query prompt**, the **LLM model employed**, and the **system prompt**. We categorize system prompts into three types: a *poor* prompt is highly ambiguous and provides minimal structure; a *good* prompt is well-structured and follows the ReAct-prompts pattern Yao et al. [2022]; and a *strict* prompt extends a good prompt with additional rules and illustrative examples.

2.2 Feature Extraction from Agentic Traces

Once traces are collected, the next step is to extract **relevant features** capturing key characteristics of Agentic AI executions. Features are extracted primarily from *LLM call* span attributes, as they drive decision-making, token usage, latency, and reasoning behavior, whereas agent and tool spans contribute minimally to these attributes. We extract **16** features organized into five key categories:

- **Token Features**: Captures the number of tokens consumed by the LLMs for its input, and output as well as aggregated (sum, mean, std) token usage across all the LLM calls in the trace. These feature provides insights into computational cost.
- **Latency Features**: Including per-span latency, sum and mean latency, and end-to-end total time. These features highlight performance bottlenecks and help benchmark execution efficiency.
- **Path Features**: Representing the sequence of agent, tool, and LLM calls, length of hierarchical delegation, unique agent/tool calls. These features capture the trajectory of the application and extracted using trace structure.
- **Prompt and Context Features**: System and user prompts, prompt lengths, and intermediate inputs/outputs. These features provide semantic context, explaining variations in execution behavior and reasoning patterns.
- **Model Features**: Information about the specific LLM or tool versions used, allowing fair comparison between different models.

2.3 Labeling traces: Normal or Anomalous

Next is to define a labeling scheme to flag traces as anomalous (experiencing silent failures) or normal (executed successfully). Providing these labels serve as a ground truth and is essential as it enables systematic evaluation, benchmarking, and comparison of anomaly detection task, and help in performance optimization of agentic workflow execution. We label a trace as an anomaly if it exhibits any of the three failure scenarios defined in Table 1: **drift**, **cycles**, or **errors**. Drift is detected by domain experts who define the ground truth trajectory for each input prompt, allowing deviations from the expected path to be identified. For cycles, we define a strict criterion: no agent or tool should be invoked more than once within a single execution. Errors are identified directly from the trace spans by checking the error status recorded during execution.

Next, using a script, we extract trace information such as observed trajectory, cycles, and errors for each trace corresponding to a given input prompt. The inter-annotator defines its own ground truth (expected trajectory) for each input prompt. The inter-annotator uses automatic labeling script (Figure 1) which takes the ground truth and the extracted information for each trace to automatically assign a label of normal or anomalous. This labeler script is highly extensible, as future annotators only need to provide the ground truth for new prompts, and the script can automatically generate the corresponding labels.

We apply the pipeline¹ (Figure 1) to **two** Multi-Agent AI systems:

- **Stock Market Analysis Assistant:** This system has 3 agents and 9 tools – an orchestrator overseeing *stock_market* and *search_agent*, calling tools such as *stock_quote*, *stock_recommendation*, *company_transactions*, *search_symbol*, *timeseries_daily_info*, *search_internet*. Traces were generated using 525 user prompts, 3 system prompt types (poor, good, strict), and 3 LLMs (gpt-4o, ibm-granite-3-1-8B, meta-llama-3-3-70B), producing 4,275 datapoints. Two annotators labeled traces (Cohen’s kappa Cohen [1960] 97.6%). Among the datapoints, 1,804 are normal and 2,921 anomalous, including cycles (1,484), errors (1,245), and drift (1,952), note that these categories are non-exclusive.
- **Research and Writing Assistant:** This system has 9 agents and 6 tools – an orchestrator overseeing research and writing orchestrator agents, which call multiple helper agents (*doc_writer*, *note_taker*, *chart_maker*, *rag_agent*, *web_scraping*, *search_internet* each with one tool). Traces were generated from 112 user prompts, 2 system prompt types (good, poor), and 2 LLM models (ibm-granite-3-1-8B, meta-llama-3-3-70B/405B), producing 894 datapoints. Two annotators labeled the traces (Cohen’s kappa 80.6%). Of these, 321 are normal and 573 are anomalous, including cycles (320), errors (285), and drift (410).

3 Anomaly Detection in Multi-Agentic AI Systems

In this section, we leverage the curated datasets to perform anomaly detection task. We start by understanding the datasets and visualizing feature distributions, then describe the experimental setup in both supervised, semi-supervised, and unsupervised settings. Finally, we present the results, highlighting feature importance, error analysis, and key insights.

3.1 Dataset Feature Space Visualization

To understand the feature space, we project the 16-dimensional features using t-SNE van der Maaten and Hinton [2008]. Figure 2 shows the 2D distribution of normal and anomalous traces. In the stock market (Figure 2a), some anomalous points overlap with normal, indicating higher detection complexity, whereas the research writing (Figure 2b) shows clearer separation. This motivates the need for sophisticated models as feature distributions vary across applications.

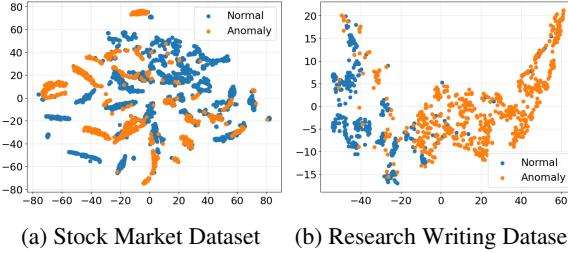


Figure 2: t-SNE of normal and anomalous agentic traces

3.2 Experimental Setup

To perform anomaly detection task, the datasets for the two applications — The Stock Market dataset (4,275 traces; 1,804 normal, 2,921 anomalous) and Research Writing dataset (894 traces; 321 normal, 573 anomalous) was split into 70-15-15% for train-validation-test. We consider anomaly as **positive class**, normal as **negative class**. We employ three settings enabling a comprehensive evaluation of accuracy, robustness, and generalizability: **(A) Supervised:** Includes classical ML models like Tree-based models (XGBoost, Random Forest), linear models (Logistic Regression, Gaussian Naïve Bayes), and SVMs. These serve as upper-bound baselines assuming both normal and anomalous labels are available. **(B) Semi-Supervised:** Given the relative ease of collecting normal traces, we include one-class classifiers (OCC) such as SVDD and Isolation Forest which model the normal behavior and flag deviations as anomalies. **(C) Unsupervised:** Since labeling datasets is expensive and not always feasible, we use K-Means clustering to flag anomalies as points in small or low-density clusters. Hyperparameters were tuned via grid search on the validation set.

¹The dataset and curation pipeline will be released after paper acceptance in accordance with organizational policies.

Table 2: Comparison of Model Performance Across Two Datasets. *For Anomaly (Positive) Class

Method Type	Model	Stock Market Dataset				Research Writing Dataset			
		Accuracy	Macro-F1	Precision*	Recall*	Accuracy	Macro-F1	Precision*	Recall*
Supervised	XGBoost	0.9803	0.9793	0.9977	0.9703	0.9481	0.9426	0.9341	0.9884
	Random Forest	0.9704	0.9690	0.9952	0.9566	0.9407	0.9347	0.9333	0.9767
	Logistic Regression	0.9464	0.9438	0.9717	0.9406	0.9185	0.9115	0.9310	0.9419
	SVM	0.9732	0.9717	0.9816	0.9749	0.9111	0.9011	0.9022	0.9651
	Naïve Bayes	0.8815	0.8789	0.9707	0.8333	0.8444	0.8402	0.9577	0.7907
Semi-Supervised (OCC)	SVDD	0.9647	0.9628	0.9791	0.9635	0.8963	0.8820	0.8750	0.9767
	Isolation Forest	0.8914	0.8803	0.8706	0.9680	0.8889	0.8793	0.9080	0.9186
Unsupervised	KMeans	0.8533	0.8359	0.8327	0.9543	0.8296	0.8220	0.9091	0.8140

3.3 Results and Discussion

Table 2 summarizes the performance of supervised, semi-supervised, and unsupervised models on the Stock Market and Research Writing Assistant datasets. On the Stock Market, **XGBoost** achieves the best results overall, with accuracy of **98.03%**, macro-F1 of **97.93%**, and precision of **99.77%**. Recall (**97.03%**) is slightly lower (**-0.46%**) than SVM (**97.49%**). On the Research Writing, **XGBoost** again outperforms other models with accuracy of **94.81%**, macro-F1 of **94.26%**, and recall of **98.84%**. Naïve Bayes achieves the highest precision (**95.77%**, **+2.36%** over XGBoost). Semi-supervised **SVDD** also perform competitively (up to **96.47%** accuracy on Stock Market and **89.63%** on Research Writing), showing their ability to detect anomalies even when trained only on normal traces. On the Research Writing, Isolation Forest achieves higher precision than SVDD (**+3.30%**). Compared to supervised methods, unsupervised **K-Means** yields moderate performance across both datasets, highlighting the challenges of anomaly detection in the absence of labeled data.

Overall, **XGBoost** emerges as the best-performing supervised model, while in the semi-supervised setting, the one-class approach **SVDD** performs the best. However, there remains significant room for improvement in the unsupervised case. Notably, XGBoost and SVDD outperform inter-annotator agreement—97.6% for Stock Market and 80.6% for Research Writing—indicating misclassification are likely due to ambiguous traces where even humans disagree. This highlights that while supervised and semi-supervised models perform comparable, the cost and difficulty of annotation make semi-supervised methods more practical for real-world applications. Next, we analyze **feature importance** for models (XBoost and SVDD) — using SHAP Lundberg and Lee [2017] to identify which features most strongly influence anomaly detection. There are several common features that appear in the **top-10** for both models across the two datasets. **Path-level features** such as tool_count, total_steps, unique_steps, and agent_count—are consistently **ranked highest**, highlighting their critical role in anomaly detection. While some **latency and prompt features** such as total_duration, std and avg_prompt_similarity features also appear, they tend to have **lower rankings** in terms of feature importance.

To understand model misclassifications, we conducted a detailed **error analysis**. For each false negative (FN, anomaly predicted as normal), we compared the mean feature values to those of the corresponding ground-truth class (TPs) to understand which features most strongly contributed to misclassification. For the Stock Market dataset, XGBoost has 13 FNs and SVDD 16 out of 438 anomalies, with all 13 XGBoost FNs overlapping with SVDD. For Research Writing, XGBoost has 1 FN and SVDD 2 out of 86 anomalies, with 1 XGBoost FN overlapping with SVDD. While FPs also occur, FNs are key to understand model weaknesses in our use case. Upon analyzing the FNs, several interesting insights emerge: **(Insight 1)** Most anomalies (TP) involve cycles and errors, which also caused drift and increase path-level feature values making the models easier to predict them as anomalies. **(Insight 2)** Some anomalies follow shorter, drifted paths without explicit cycles or errors, having very less value for path-level, prompt & latency features

(negative difference with TP mean) closely resembles normal traces and thus harder to flag as anomaly leading to misclassifications. **(Insight 3)** t-SNE (Figure 2) support these findings: anomalies with cycles/errors form well-separated clusters, while anomalies with subtle drift overlap with normal traces. To conclude, these FNs though few in number are strong examples of silent failures where the system drifts subtly while feature values remain within normal ranges flagging them as normal by both XGBoost and SVDD, a key area for improvement in future works.

4 Conclusions and Future Plans

In this work, we introduced anomaly detection of agentic traces in Multi-Agentic AI systems, addressing challenges like non-deterministic behavior and silent failures. We curated datasets from traces of two such systems and evaluated on supervised, semi, unsupervised methods. Our results show that while XGBoost and SVDD perform comparable in both supervised and semi-supervised settings, preparing labeled data is easier in the semi-supervised setting. Error analysis indicates that a small number of false negatives persist, where both models miss anomalies due to their close resemblance to normal behavior. As part of **future work**, we plan to enhance anomaly detection methods by thoroughly analyzing feature statistics, particularly focusing on anomalies that mimics normal behavior. This includes investigating additional feature engineering strategies beyond the current set to improve model sensitivity. Given the high cost and scarcity of labeled data, developing robust unsupervised or semi-supervised approaches suitable for both offline and online settings will also be a key focus. Further analysis is needed to explore and address the other failure modes identified in our study (see Table 1), aiming to expand the coverage and robustness of detection methods across varied silent failure scenarios.

References

- Sukriti Bhattacharya, Giovanni Castignani, Lorenzo Masello, and Barry Sheehan. Ai revolution in insurance: bridging research and reality. *Frontiers in Artificial Intelligence*, 8:1568266, 2025. doi:10.3389/frai.2025.1568266. URL <https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2025.1568266/full>.
- McKinsey & Company. The future of ai for the insurance industry. Technical report, McKinsey & Company, July 2025. URL <https://www.mckinsey.com/industries/financial-services/our-insights/the-future-of-ai-in-the-insurance-industry>.
- DamcoGroup. Ai agents in insurance: The intelligent leap beyond traditional chatbots. <https://www.damcogroup.com/blogs/ai-agents-in-insurance-intelligent-leap-beyond-traditional-chatbots>, June 2025. Accessed: November 7, 2025.
- AiMultiple Research. Top 5 insurance chatbots with real-life use cases in 2025. <https://research.aimultiple.com/insurance-chatbot/>, August 2025. Accessed: November 7, 2025.
- Conor Bronson. Ai agent reliability strategies that stop ai failures before they start. <https://galileo.ai/blog/ai-agent-reliability-strategies>, July 2025. Directly compares AI agent reliability challenges to traditional software systems, showing how non-deterministic behavior creates entirely new failure categories.
- Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, et al. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657*, 2025.
- Lewis Hammond, Alan Chan, Jesse Clifton, Jason Hoelscher-Obermaier, Akbir Khan, Euan McLean, Chandler Smith, Wolfram Barfuss, Jakob Foerster, Tomáš Gavenčíak, The Anh Han, Edward Hughes, et al. Multi-agent risks from advanced ai. *arXiv preprint arXiv:2502.14143*, 2025. URL <https://arxiv.org/abs/2502.14143>. Comprehensive analysis of risks in multi-agent AI systems.
- Yuhan Zhang, Xiaoyun Chen, Jiawei Liu, Haochen Wang, and Yuxin Su. Unsupervised microservice system anomaly detection via contrastive multi-modal representation clustering. *Information Processing & Management*, 62(1):103728, 2024. URL <https://www.sciencedirect.com/science/article/abs/pii/S0306457324003728>. Proposes MAD-CMC framework for microservice anomaly detection using multimodal clustering.
- Xueke Chen, Chunfeng Yan, Xiaoli Zhang, and Yingying Li. Deep attentive anomaly detection for microservice systems with multimodal time-series data. *IEEE Transactions on Services Computing*, 2023. URL <https://www.semanticscholar.org/paper/Deep-Attentive-Anomaly-Detection-for-Microservice-Chen-Yan/d20ac00f8d7139a087ee67d705ee5ff0ca5d5b19>. Introduces DAM approach with multimodal fusion and attentive LSTM for microservice anomaly detection.

- Maria Rodriguez, John Smith, and David Johnson. Machine learning-based network anomaly detection: Design, implementation, and evaluation. *Journal of Cybersecurity and Privacy*, 5(4):143, 2024. URL <https://www.mdpi.com/2673-2688/5/4/143>. Comprehensive evaluation of ML techniques for network anomaly detection including unsupervised and supervised approaches.
- Xu He, Di Wu, Yan Zhai, and Kun Sun. Sentinelagent: Graph-based anomaly detection in multi-agent systems. *arXiv preprint arXiv:2505.24201*, 2025.
- Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, pages 785–794. ACM, 2016. doi:10.1145/2939672.2939785.
- David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54(1):45–66, 2004.
- OpenTelemetry Project. OpenTelemetry: An Observability Framework for Cloud-native Software. 2021. <https://opentelemetry.io>.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.
- Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1): 37–46, 1960. doi:10.1177/001316446002000104.
- Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. In *Journal of Machine Learning Research*, volume 9, pages 2579–2605, 2008.
- Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.