

Evidence-Ready AI Systems: Logging, Telemetry, and Experiment Infrastructure for Auditable AI Workflows

Akshaya Jayasankar
Email: akshayajayasankar01@gmail.com

Abstract—As AI systems move from prototypes to high-stakes production settings, organizations increasingly need *evidence-ready* infrastructures: logging, telemetry, and experimentation pipelines that make it possible to study, audit, and improve AI-mediated decisions over time. Many AI deployments still treat logging as an afterthought, yielding sparse, siloed traces that are difficult to connect to user outcomes, fairness concerns, or organizational processes. This paper proposes a design for evidence-ready AI systems that integrates (i) principled logging schemas, (ii) telemetry pipelines that link AI behaviour to user, task, and organizational context, and (iii) experiment infrastructure for causal evaluation and ongoing audits. I synthesize prior work on ML systems, observability, and responsible AI; develop design principles and an architecture for “analysis-ready” logs; and outline how synthetic data and replay experiments can be used to test pipelines before real-world deployment. The goal is to provide a blueprint that both enables rigorous empirical research and supports organizations in governing AI systems as they evolve.

Index Terms—Artificial intelligence, logging, telemetry, experimentation, observability, responsible AI, audits, field experiments.

I. INTRODUCTION

AI tools are increasingly embedded in everyday workflows: drafting emails, summarizing documents, ranking support tickets, and recommending actions to frontline staff. In many organisations, these systems already influence who receives which information, which cases are prioritised, and how workers allocate their attention. Yet the underlying instrumentation is often minimal. Logs capture raw prompts and responses, but omit contextual signals such as task type, user role, or downstream decisions. As a result, it is difficult to answer basic questions about performance, equity, or safety once an AI system is in production.

This paper starts from a simple requirement: AI systems used in organisations should be *evidence-ready*. That is, they should be designed from the outset so that researchers, practitioners, and auditors can reconstruct what happened, link AI behaviour to human decisions and outcomes, and run credible evaluations—including experiments—as the system evolves.

I use the term *evidence-ready AI system* to describe an AI-enabled workflow in which:

- logging schemas are explicit and stable enough to support quantitative analysis,
- telemetry pipelines reliably link events across services and time,
- experimentation hooks (e.g., flags, treatment assignments) are available without extensive re-engineering, and
- privacy, access control, and governance rules are built into the data structure rather than retrofitted.

The contributions of this paper are threefold. First, I synthesise prior work on ML systems, observability, and responsible AI into a set of design principles for evidence-ready infrastructures. Second, I propose an architecture and logging schema for AI-mediated workflows that separate raw events, derived features, and analysis-ready tables while preserving reproducibility. Third, I outline an evaluation plan that uses synthetic data, replay experiments, and staged rollouts to exercise the logging and experiment stack before applying it to real users.

Throughout, I focus on AI systems that support human decision-makers rather than fully autonomous systems. Typical examples include internal copilots for operations teams, triage assistants in customer support, or recommendation systems embedded in organisational platforms. These settings combine technical and organisational complexity, making the need for traceable, analysable logs especially salient.

The remainder of the paper is structured as follows. Section II reviews related work and motivates the need for evidence-ready infrastructures. Section IV presents design principles. Section V describes the architecture and logging schema. Section VII discusses telemetry, experimentation, and synthetic data. Section VIII outlines an evaluation plan, and Section IX discusses governance implications.

II. BACKGROUND AND RELATED WORK

A. *ML Systems, Technical Debt, and Observability*

Work on large-scale ML systems has highlighted that most real-world effort goes into data management, monitoring, and infrastructure rather than the model itself. Sculley *et al.* describe the “hidden technical debt” that arises when ML systems lack modularity, testing, and clear data contracts [1]. Subsequent work on ML observability

and testing proposes checklists, validation tools, and continuous monitoring for data and models [2], [3].

These contributions mainly focus on reliability and performance. They say less about how logs should be structured to support social-science style questions about fairness, accountability, and organisational impact. Evidence-ready systems build on this line of work but emphasise analysis and audit as first-class use cases.

B. Documentation, Transparency, and Audits

Parallel streams in responsible AI emphasise documentation and transparency for models and datasets. Model Cards [4] and Datasheets for Datasets [5] propose structured artefacts that record intended use, limitations, and evaluation results. Audit studies and governance frameworks argue that organisations should maintain records of how AI systems are deployed and how they affect different groups over time.

Evidence-ready logging complements these artefacts. While documentation describes the system at a point in time, logs and telemetry provide the raw evidence needed to test whether the system behaves as described, and whether its impact changes as data and contexts shift.

C. Experimentation and Organisational Learning

Online platforms and firms increasingly rely on experiment infrastructure to evaluate product changes. Controlled experiments and staggered rollouts support causal inference about the effects of algorithmic changes on user behaviour and outcomes. In AI-mediated work, similar infrastructures can be used to test interfaces, explanations, and decision policies.

However, many operational AI deployments lack the instrumentation required for credible experiments. Treatment assignment may not be logged; outcome measures may live in separate systems; and key covariates are missing or captured only in free text. Evidence-ready systems treat experiment hooks and outcome logging as part of the core design, not as an afterthought.

D. Literature Matrix

Table I summarises how selected strands of prior work address (or omit) three aspects that evidence-ready systems must integrate: logging schemas, telemetry/traceability, and experiment readiness.

III. MOTIVATING SCENARIO AND REQUIREMENTS TRACEABILITY

This section grounds the design in a concrete organisational scenario and traces backwards from the questions we want to answer to the fields that must be present in the logs.

TABLE I
HIGH-LEVEL LITERATURE MATRIX FOR EVIDENCE-READY AI SYSTEMS.

Stream / exemplar	Logging schema	Telemetry / traceability	Experiment readiness
ML technical debt [1]	○	○	● (testing emphasis)
ML observability / validation [2], [3]	●	●	○
Documentation artefacts (Model Cards, Datasheets) [4], [5]	○	○	○
Experimentation platforms and A/B testing	○	●	●
Responsible AI, audits, governance	● (often ad hoc)	○	○
This paper: evidence-ready AI	●	●	●

Notes: ● = core focus; ○ = partial / implicit.

A. Scenario: AI-Assisted Internal Support Triage

Consider an internal support team that handles tickets from employees about access issues, tooling problems, and data requests. Agents work in a web console where they can:

- view incoming tickets with metadata (requester, category, severity),
- call an AI assistant to propose a draft reply, diagnosis, or routing decision, and
- escalate complex cases to specialists or other teams.

A typical interaction unfolds as follows. An agent opens a ticket; the system records a `TASK_CREATED` event. The agent clicks “Ask AI,” which generates an `AI_REQUEST` with the ticket text, relevant context, and a task type such as “triage” or “diagnosis.” The AI service returns suggested actions and a draft response, logged as `AI_RESPONSE`. The agent either accepts the suggestion, edits it, ignores it, or escalates the ticket; this becomes a `USER_ACTION` event. When the ticket is eventually resolved, a `TASK_OUTCOME` event records resolution time, status, and (when available) requester satisfaction.

Over time, product managers, operations leaders, and researchers may want to ask questions such as:

- How does AI assistance affect resolution time and quality for different ticket types?
- Do novice and experienced agents rely on AI in different ways?
- Are there systematic differences in escalation patterns across teams?
- Do interface or policy changes (e.g., explanations) change these behaviours?

Answering such questions requires that the logging schema and telemetry pipelines be structured deliberately, not assembled from ad hoc print statements or siloed logs.

TABLE II
EXAMPLE REQUIREMENTS TRACEABILITY FOR THE TRIAGE SCENARIO.

Question	What must be logged	Where
How often do agents fully accept vs. edit AI suggestions?	Action type (<code>accept/edit/ignore</code>); optional edit distance between AI draft and final reply.	<code>USER_ACTION</code> table
Do novice vs. experienced agents rely on AI differently?	Agent role/tenure; AI usage events linked to <code>user_id</code> .	User dimension; interaction table
Does AI reduce resolution time for high-severity tickets?	Ticket severity; timestamps for creation, first AI use, and resolution; treatment flags if policies vary.	Task panel; experiment table
Do explanation variants change escalation behaviour?	Experiment arm for explanation variant; whether explanation was shown; escalation outcome.	<code>EXP_ASSIGN</code> ; <code>USER_ACTION</code> ; task panel

B. Requirements Traceability

Table II presents a simple traceability matrix. Each row starts from a question that stakeholders might ask and maps it to required log fields and the analysis tables where those fields should appear. In practice, such a matrix can be maintained as a living design document and updated when schemas evolve.

This traceability perspective reinforces the “analysis comes first” principle. Instead of treating logging as a purely technical concern, designers work backwards from the evaluative and governance questions that evidence-ready systems must support. Subsequent sections show how these requirements are reflected in the logging schema, data model, and reproducibility artefacts.

IV. DESIGN PRINCIPLES FOR EVIDENCE-READY AI SYSTEMS

I propose five design principles for evidence-ready AI systems. They apply both to new deployments and to retrofits of existing tools.

A. P1: Analysis Comes First

Instead of asking “What can we log?”, start from the questions that researchers, product teams, or regulators may want to answer. For example:

- How often do users follow, edit, or override AI recommendations?
- Do outcomes differ by user role, segment, or geography?
- How do changes in prompts, models, or interfaces affect behaviour?

From these questions we derive required entities (users, sessions, tasks, items), keys, and outcome variables, and ensure that logs contain the joins needed to reconstitute relevant panels.

TABLE III
DESIGN PRINCIPLES AND THEIR IMPLICATIONS FOR LOGGING.

Principle	Implication for logs
P1: Analysis first	Explicit join keys, outcome variables, and timestamps supporting panels.
P2: Versioned schemas	Event versions, migration scripts, and compatibility layers.
P3: Traceability	Request/session IDs linking model IO, user actions, and outcomes.
P4: Experiment hooks	Flags for treatments, cohorts, and feature exposures.
P5: Privacy & minimisation	Sensitive fields isolated, access-controlled, or synthesised.

B. P2: Stable, Versioned Schemas

Evidence-ready systems separate *schema evolution* from ad hoc logging. Changes to event structures (e.g., adding a field for “explanation shown”) are versioned and documented. Downstream pipelines translate from raw event versions to stable analysis tables, preserving backwards compatibility where possible.

C. P3: End-to-End Traceability

Every AI-mediated decision should be traceable from input to output to downstream actions. This requires identifiers that propagate across services (e.g., request IDs, session IDs, task IDs) and link:

- model inputs (prompts, features),
- model outputs (scores, rankings, generations),
- user actions (accept, edit, ignore, escalate), and
- outcomes (resolution, time, quality, survey responses).

Traceability enables both audits and causal analyses.

D. P4: Experiment Hooks by Design

Flags and assignment variables for experiments (e.g., which explanation variant a user saw) should be embedded in logs and carried through to analysis tables. Even when no experiment is running, systems should support randomisation, counterfactual logging, or shadow evaluation without redesign.

E. P5: Privacy, Access, and Minimisation

Finally, evidence-ready does not mean logging everything. Schemas should respect data minimisation and encapsulate sensitive attributes (e.g., protected characteristics, free-text content) in controlled tables with differential access. Aggregated views and synthetic data can be used for many tests before touching identifiable logs.

Table III summarises how these principles translate into concrete requirements for event design.

V. ARCHITECTURE AND LOGGING SCHEMA

This section describes an architecture that implements the above principles. The focus is on data structures and flows rather than on specific vendor technologies.

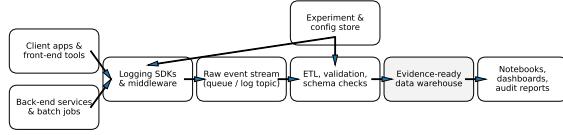


Fig. 1. High-level architecture for an evidence-ready AI system. Client applications and services emit structured events via logging SDKs; events flow through a raw stream into validated ETL, an evidence-ready warehouse, and downstream analysis and audit surfaces.

TABLE IV
CORE EVENT TYPES IN THE EVIDENCE-READY LOGGING SCHEMA.

Event type	Description
AI_REQUEST	User or system sends input to an AI model; includes task type, context size, and model version.
AI_RESPONSE	Model output, including scores, alternatives, and explanation metadata.
USER_ACTION	User behaviour in response to the AI output (accept, edit, ignore, escalate, ask clarification).
TASK_OUTCOME	Resolution status and quality at task level (e.g., solved, reopened, satisfaction rating).
EXP_ASSIGN	Treatment flags and randomisation metadata for experiments or feature rollouts.
SCHEMA_CHANGE	Records of changes to logging schema and feature definitions.

A. High-Level Architecture

Figure 1 illustrates the proposed architecture. User-facing applications send requests to an AI service layer (e.g., a retrieval-augmented generation API). An observability layer captures structured events from both the AI service and the client applications. A central event store retains immutable raw logs, while downstream pipelines materialise analysis-ready tables.

B. Event Types and Entities

Table IV summarises the main event types. Each event shares a small set of global keys (e.g., `user_id`, `session_id`, `task_id`, `request_id`) and timestamps.

C. Raw Events and Analysis-Ready Tables

Raw events are written in an append-only log (e.g., object storage or a message bus). A separate transformation layer produces:

- **Interaction tables**, with one row per AI interaction, including input, output, user action, and experiment flags.
- **Task-level panels**, with one row per task per period, aggregating interactions and outcomes.
- **User-period panels**, capturing user-level behaviour (e.g., how frequently they rely on AI).

Each analysis table is defined by a versioned specification (e.g., a SQL or notebook file in version control). This allows researchers to reproduce historical analyses even if underlying event schemas evolve.

D. Synthetic Schema Example

For illustration, the synthetic evaluation in Section VIII will use a simplified interaction table with the following fields:

- `user_id`, `role`, `team_id`
- `session_id`, `task_id`, `task_type`
- `request_id`, `model_version`, `prompt_template_id`
- `treatment_arm` (e.g., control vs. explanation vs. counterfactual)
- `response_tokens`, `latency_ms`
- `action_type` (accept / edit / ignore / escalate)
- `outcome_quality_score` (synthetic scalar)

Synthetic data for this schema can be generated programmatically to stress-test the logging and analysis code before any real logs are used.

VI. DATA MODEL AND REPRODUCIBILITY ARTEFACTS

The logging schema described in Section V feeds into a data model that is convenient for analysis and reproducibility. This section sketches a simple star-schema design and the accompanying artefacts that make the system evidence-ready in practice.

A. Star-Schema Data Model

At a high level, the proposed data model consists of a small number of fact tables and several dimensions. Fact tables capture events or outcomes; dimensions capture relatively slowly changing attributes such as user role or team.

A minimal design includes:

- **fact_interactions**: one row per AI interaction (request-response pair), including input metadata, model version, treatment arm, user action, and immediate outcomes (e.g., acceptance).
- **fact_tasks**: one row per task per period (e.g., ticket per day), aggregating interactions, escalations, and final resolution status and quality.
- **dim_users**: user-level attributes such as role, tenure band, team membership, and region.
- **dim_teams**: team-level attributes such as function, queue type, and management structure.
- **dim_experiments**: definitions of experiments, treatment arms, and activation windows.

Table V summarises the purpose of each table and examples of key fields.

This star schema supports a variety of analyses via straightforward joins, while keeping entity boundaries clear. It also makes it easier to expose synthetic datasets, since each table can be generated independently from a parameterised simulator.

TABLE V
SUMMARY OF CORE FACT AND DIMENSION TABLES.

Table	Role and example fields
<code>fact_interactions</code>	Grain: interaction (AI request-response). Keys: <code>request_id</code> , <code>user_id</code> , <code>task_id</code> . Fields: task type, model version, treatment arm, response length, latency, user action.
<code>fact_tasks</code>	Grain: task per period. Keys: <code>task_id</code> , period. Fields: counts of AI calls, escalations, resolution time, outcome quality score.
<code>dim_users</code>	Grain: user. Keys: <code>user_id</code> . Fields: role, tenure band, team, region; optional anonymised identifiers.
<code>dim_teams</code>	Grain: team. Keys: <code>team_id</code> . Fields: function, queue type, leadership structure.
<code>dim_experiments</code>	Grain: experiment arm. Keys: experiment and arm IDs. Fields: description, start/end dates, targeting rules, primary metrics.

B. Reproducibility and Artefact Bundle

To make the system genuinely evidence-ready, the data model must be accompanied by artefacts that allow others to understand and reuse it. A practical artefact bundle for this paper would include:

- **Schema definitions:** SQL or YAML files that define each table, including data types, keys, and foreign-key relationships.
- **Synthetic data generators:** scripts that produce synthetic `fact_interactions`, `fact_tasks`, and dimension tables, following the schemas and mimicking realistic distributions.
- **Example analysis notebooks:** notebooks that load the synthetic tables and reproduce the descriptive, experimental, and slice analyses sketched in Section VIII.
- **Configuration files:** human-readable files declaring experiment arms, metrics, and key slices (e.g., novice vs. experienced users) that audits should consider.

Table VI summarises how these artefacts map onto the goals of evaluation, audit, and collaboration.

By keeping these artefacts under version control alongside the code that generates analysis tables, organisations can support internal reproducibility, external collaboration, and future doctoral research that builds on the same infrastructure.

VII. TELEMETRY, EXPERIMENTS, AND SYNTHETIC DATA

A. Telemetry and Health Signals

Telemetry extends basic logging with derived signals and health checks. For evidence-ready systems, useful telemetry includes:

TABLE VI
ARTEFACT COVERAGE FOR AN EVIDENCE-READY AI SYSTEM.

Artefact	Purpose
Schema definitions	Make table structure explicit; support validation and migration; allow others to recreate the schema in their own environments.
Synthetic datasets	Enable testing of queries, visualisations, and models without exposing sensitive operational data.
Analysis notebooks	Demonstrate how to compute key metrics, treatment effects, and audit slices from the tables.
Configuration files	Document experiments, metrics, and slices in a form that product teams, researchers, and auditors can inspect and version.

- distribution shifts in inputs (e.g., length, language, task mix),
- drift in model outputs (e.g., sentiment, ranking position, safety scores),
- user reliance patterns (e.g., acceptance rate, edit distance to AI output),
- latency and error rates across segments, and
- coverage of evaluation slices used in audits.

These signals can be computed in batch and near-real time. Dashboards built on top of telemetry help both operations teams and researchers identify periods or cohorts where closer analysis is warranted.

B. Experiment Infrastructure

The same identifiers that support traceability also support experiments. For example, a system might randomise:

- whether an explanation is shown,
- which explanation style is used (rationale vs. bullet points),
- ordering of AI and human recommendations, or
- thresholds for escalation or human review.

Experiment assignments are logged in `EXP_ASSIGN` events and carried into interaction and task tables. This enables standard analyses using difference-in-means, regression with covariates, or more advanced estimators (e.g., difference-in-differences for staggered rollouts).

C. Synthetic Data and Replay Experiments

Before running experiments with real users, organisations can perform synthetic tests:

- **Schema tests:** generate synthetic events that exercise edge cases (missing fields, rare combinations) and validate transformations into analysis tables.
- **Replay experiments:** feed logged historical requests through new model variants or interface policies, logging counterfactual responses while leaving production behaviour unchanged.

- **Load and privacy checks:** use synthetic users and tasks to ensure that logs respect minimisation rules and that storage and pipelines can handle expected scale.

These synthetic tests are particularly valuable when collaborating with external researchers, since detailed schemas and example datasets can be shared without exposing sensitive user data.

VIII. EVALUATION PLAN AND EXAMPLE ANALYSES

This section sketches an evaluation plan using synthetic data to demonstrate how an evidence-ready system supports analysis. The goal is not to present empirical results, but to show that the logging and experiment infrastructure is sufficient for future studies.

A. Synthetic Data Generator

I assume a generator that produces interaction-level data with:

- N_{users} users nested in teams,
- T time periods (e.g., weeks),
- a mix of task types (e.g., triage, drafting, summarisation),
- randomised treatment arms (e.g., control vs. explanation vs. counterfactual).

Outcomes depend on user skill, task type, and treatment, plus noise. Parameters can be chosen so that treatment effects are modest but detectable, mimicking realistic organisational settings.

B. Example Analyses

Using this synthetic panel, we can demonstrate three classes of analysis.

1) *Behavioural Descriptives*: First, we characterise reliance on AI: acceptance rates, edit distances (if text similarity is available), and escalation frequencies by role and task. Simple plots and summary tables here validate that logs capture expected distributions.

2) *Experiment Effects*: Second, we estimate the effect of interface or policy variants on outcomes. For example, we might regress an outcome Y_{it} (e.g., resolution quality or time) on treatment indicators and covariates with team and period fixed effects:

$$Y_{it} = \alpha + \beta_1 \text{TreatA}_{it} + \beta_2 \text{TreatB}_{it} + \gamma_i + \delta_t + \varepsilon_{it}, \quad (1)$$

where γ_i and δ_t capture unobserved heterogeneity by user and period. With synthetic data, we can verify that the estimator recovers known treatment effects.

3) *Audit and Slice Analyses*: Third, we perform slice analyses across user and task subgroups (e.g., novice vs. experienced users; different regions). The logging schema ensures that required fields are present and that sample sizes per slice are visible, helping auditors avoid over-confident conclusions for tiny subgroups.

C. Reproducibility Artefacts

Finally, evidence-ready systems should ship with artefacts that allow others to reproduce analyses:

- SQL or notebook files that define each analysis table,
- scripts that generate synthetic datasets from schemas,
- configuration files that declare experiment arms and outcome metrics.

These artefacts can be versioned in a public repository, with real data replaced by synthetic examples.

IX. DISCUSSION: GOVERNANCE, RISKS, AND OPPORTUNITIES

Evidence-ready infrastructures create opportunities and risks. On the one hand, they support more rigorous evaluation and accountability. On the other hand, they can easily become surveillance infrastructures if not governed carefully.

A. Benefits for Governance and Research

For organisations, evidence-ready AI systems:

- make it easier to detect regressions when models or prompts change,
- support internal reviews and external audits with clear artefacts,
- allow targeted experiments on explanations, guardrails, or escalation policies, and
- enable collaboration with academic researchers without exposing raw operational systems.

For researchers, access to well-structured logs opens up questions about learning, expertise, and inequality in AI-mediated work that are currently difficult to study.

B. Risks and Mitigations

At the same time, detailed logs can capture sensitive information about both users and workers. Potential risks include:

- **Over-monitoring of workers**, where every interaction is used for fine-grained performance evaluation rather than for learning and safety.
- **Re-identification or misuse**, if logs combine identifiers in ways that make individuals easy to trace.
- **Function creep**, where data collected for evaluation is repurposed for unrelated goals.

Mitigations include:

- explicit governance policies and review boards for AI instrumentation,
- role-based access control and differential logging for sensitive attributes,
- aggregation and de-identification in analysis tables, with raw logs restricted.

C. Future Extensions

Future work could extend the architecture to:

- integrate causal discovery and adaptive experimentation,
- support cross-organisational federated audits, and

- combine structured logs with qualitative fieldwork and worker narratives.

These directions would further connect system design with organisational research on authority, learning, and fairness.

X. CONCLUSION

This paper has argued that AI systems deployed in organisations should be designed to be *evidence-ready*: their logs, telemetry, and experiment hooks should make it straightforward to study how they behave, how people use them, and how they affect outcomes over time. I synthesised prior work on ML systems, documentation, and experimentation into design principles; proposed an architecture and logging schema; and outlined an evaluation plan that uses synthetic data to exercise the infrastructure before working with real users.

The core message is simple: instrumentation is not a convenience feature to be added after deployment. It is part of the sociotechnical design of an AI system. By investing in evidence-ready infrastructures, organisations can make it easier to detect problems, run careful experiments, and collaborate with researchers and regulators. For doctoral research, such infrastructures also create a fertile empirical foundation for studying how AI reshapes work, authority, and opportunity.

REFERENCES

- [1] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, “Hidden technical debt in machine learning systems,” in *Proc. 28th Int. Conf. Neural Information Processing Systems (NIPS)*, 2015, pp. 2503–2511.
- [2] E. Breck, N. Polyzotis, S. Roy, S. Whang, and M. Zinkevich, “The ML test score: A rubric for ml production readiness and technical debt reduction,” in *Proc. IEEE BigData*, 2017.
- [3] N. Polyzotis, S. Roy, S. Whang, and M. Zinkevich, “Data validation for machine learning,” in *Proc. 2nd SysML Conf.*, 2019.
- [4] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Wasserman, B. Hutchinson, E. Spitzer, I. D. Raji, and T. Gebru, “Model cards for model reporting,” in *Proc. Conf. Fairness, Accountability, and Transparency (FAT*)*, 2019, pp. 220–229.
- [5] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. D. III, and K. Crawford, “Datasheets for datasets,” *Communications of the ACM*, vol. 64, no. 12, pp. 86–92, 2021.