STRUCTURED EVENT LOGGING FOR TRACKING MODEL BEHAVIOR UNDER DISTRIBUTIONAL DRIFT

Amrutha Muralidhar[1] **Yathindra Lakkanna**[2]
[1]Department of Computer Science and Engineering, B.M.S. College of Engineering, India
[2]National Institute of Fashion Technology, India
[1]amrutham.cs21@bmsce.ac.in, [2]yathindra.lakkanna@nift.ac.in

ABSTRACT

Distributional drift poses fundamental challenges for deployed machine learning systems. While drift detection methods exist, current monitoring approaches lack systematic mechanisms for recording relationships between drift events, intervention decisions, and performance outcomes. We propose a structured event logging methodology that organizes the drift management lifecycle through an event taxonomy spanning detection, intervention, and monitoring phases. We evaluate this approach using 5-fold temporal cross-validation across three domains (prediction markets, electricity pricing, weather forecasting) with varying drift characteristics. Our results show that structured event logs can support retrospective analysis of drift patterns and their relationship to model performance across different drift regimes.

1 INTRODUCTION

Distributional drift poses fundamental challenges for deployed machine learning systems (Gama et al., 2014; Gemaque et al., 2020; Paleyes et al., 2022). While drift detection methods are well-established (Dos Reis et al., 2016; Rabanser et al., 2019), existing approaches focus on identifying *when* drift occurs rather than systematically recording relationships between drift events, interventions, and performance outcomes. Current monitoring infrastructure (Hummer et al., 2019; Schelter et al., 2018) emphasizes real-time metrics but often lacks structured event provenance, limiting retrospective analysis of model behavior.

Production environments present particular challenges where multiple models operate concurrently, each experiencing different drift patterns. Without structured event histories, practitioners face difficulties answering questions such as: Which features exhibit temporal instability? Do specific drift patterns correlate with performance degradation? How effective are different intervention strategies? These questions require linking drift detection events to subsequent interventions and performance outcomes, a capability that can be limited in current monitoring infrastructure (Sculley et al., 2015; Breck et al., 2017).

We propose a structured event logging methodology that organizes the drift management lifecycle through an event taxonomy spanning detection, intervention, and monitoring phases. This approach supports systematic analysis of relationships between distributional shifts and model performance. The event schema enables temporal queries and feature-level attribution analysis, addressing gaps identified in production ML systems research (Polyzotis et al., 2017; Paleyes et al., 2022).

Contributions: (1) An event taxonomy organizing drift management into detection, intervention, and monitoring phases; (2) Empirical evaluation using 5-fold temporal cross-validation showing that structured event logs support retrospective analysis of drift patterns; (3) Validation across three domains with varying drift characteristics using regularized models to isolate drift effects from overfitting.

Table 1: Event taxonomy with trigger conditions and severity levels

Event Type	Severity	Trigger
DRIFT_DETECTED	WARNING	KS $p < 0.05$
MODEL_RETRAINED	INFO	Retraining executed
PREDICTION_ANOMALY	WARNING	Out of range
FEATURE_SHIFT	INFO	Distribution change
PERFORMANCE_DEGRADATION	CRITICAL	RMSE $> 1.5\times$ baseline
INTERVENTION_REQUIRED	CRITICAL	Multiple warnings
MODEL_DEPLOYED	INFO	Version change
AUDIT_CHECKPOINT	INFO	Periodic snapshot

2 METHODOLOGY

2.1 EVENT TAXONOMY

We organize the drift management lifecycle through a structured event taxonomy comprising three phases: **detection** (identifying distributional shifts), **intervention** (adaptive responses), and **monitoring** (outcome tracking). This yields eight event types:

Detection events capture distributional shifts through statistical testing and feature-level analysis. DRIFT_DETECTED records the outcome of two-sample statistical tests (e.g., Kolmogorov-Smirnov, Mann-Whitney) with test statistics, p-values, and significance thresholds. FEATURE_SHIFT provides granular feature-level drift quantification, identifying which specific features exhibit temporal instability. These events provide the empirical foundation for subsequent intervention decisions.

Intervention events record adaptive actions taken in response to detected drift. MODEL_RETRAINED captures retraining execution with hyperparameters, training duration, and convergence metrics. MODEL_DEPLOYED tracks deployment transitions including model versions, deployment timestamps, and rollback capabilities. INTERVENTION_REQUIRED flags cases requiring manual review when automated responses are insufficient or when drift patterns are ambiguous. These events document the decision-making process linking detection to action.

Monitoring events track ongoing system state and performance outcomes. PERFORMANCE_DEGRADATION records metric deterioration with baseline comparisons and degradation ratios. PREDICTION_ANOMALY captures outlier detection in model outputs, identifying predictions that fall outside expected ranges. AUDIT_CHECKPOINT provides periodic snapshots of system state for compliance verification and debugging. These events support assessment of intervention effectiveness and long-term system health.

2.2 EVENT SCHEMA

Each event $e \in \mathcal{E}$ is defined by a tuple (id, t, τ, s, u, m) where id is a unique identifier, t is the timestamp, $\tau \in \mathcal{T}$ is the event type, $s \in \{\text{INFO, WARNING, CRITICAL}\}$ is severity, u is user attribution, and m is event-specific metadata. The metadata m varies by type: for DRIFT_DETECTED events, m includes drift statistics (KS statistic D, p-value) and affected features; for PERFORMANCE_DEGRADATION events, m contains performance metrics and degradation ratios; for MODEL_RETRAINED events, m includes hyperparameters and training metrics.

Events are persisted in append-only logs $\mathcal{L} = \langle e_1, e_2, \ldots, e_n \rangle$ ordered by timestamp, ensuring immutability and auditability. This structure supports temporal queries such as finding all interventions following drift detection in a specific feature or computing time-to-intervention distributions. The schema enables both automated event generation during pipeline execution and manual event injection for human-in-the-loop scenarios. Event context fields maintain references to related events, supporting reconstruction of sequences linking drift detection to interventions to performance outcomes.

Table 2: Drift detection and performance across three domains (5-fold CV, mean ± std)

Dataset	Drift %	$\bar{D}$	Model	Degradation
Markets	90.2 ± 3.5	0.195 ± 0.009	Linear	$0.98 \pm 0.23\times$
			Random Forest	$0.84 \pm 0.29\times$
			Gradient Boost	$0.82 \pm 0.29\times$
Electricity	45.7 ± 5.7	0.043 ± 0.008	Logistic	$1.01 \pm 0.02\times$
			Random Forest	$1.08 \pm 0.02\times$
			Gradient Boost	$1.08 \pm 0.03\times$
Weather	8.0 ± 9.8	0.019 ± 0.003	Logistic	$1.01 \pm 0.03\times$
			Random Forest	$1.07 \pm 0.04\times$
			Gradient Boost	$1.09 \pm 0.06\times$

3 Empirical Evaluation

3.1 Experimental Setup

Datasets. We evaluate across three domains with varying drift characteristics: (1) *Prediction Markets*: real-world data from Manifold Markets ($n = 3{,}156$, 60 markets, $d = 52$ features) predicting next-day price changes; (2) *Electricity*: Australian electricity market data ($n = 45{,}312$, $d = 7$ features) predicting price movements; (3) *Weather*: meteorological observations ($n = 18{,}159$, $d = 5$ features) predicting rainfall. These datasets exhibit different temporal patterns: prediction markets show strong drift due to evolving information, electricity shows moderate seasonal drift, and weather shows minimal drift due to stable physical processes.

Protocol. We employ 5-fold temporal cross-validation to provide robust estimates with confidence intervals. We apply two-sample Kolmogorov-Smirnov tests (Dos Reis et al., 2016) to quantify distributional shifts with significance level $\alpha = 0.05$. For regression tasks (prediction markets), we evaluate linear regression, random forest (max_depth=5, min_samples_leaf=20), and gradient boosting (max_depth=3, early stopping). For classification tasks (electricity, weather), we evaluate logistic regression, random forest, and gradient boosting with the same regularization. All pipelines are instrumented with event logging at detection, training, and evaluation stages.

3.2 Results

Distributional Shift Across Domains. Table 2 summarizes drift detection and model performance across all three datasets with 5-fold cross-validation. Prediction markets exhibit severe drift (90.2% ± 3.5% features, $\bar{D} = 0.195 \pm 0.009$), electricity shows moderate drift (45.7% ± 5.7% features, $\bar{D} = 0.043 \pm 0.008$), and weather shows minimal drift (8.0% ± 9.8% features, $\bar{D} = 0.019 \pm 0.003$). This variation provides evaluation across different drift regimes.

Performance Patterns. With regularized models, degradation patterns are consistent across model types within each dataset. Prediction markets show stable or improved performance (0.82 to 0.98×), likely due to increasing predictability as markets approach resolution. Electricity shows modest degradation (1.01 to 1.08×) consistent with moderate drift. Weather shows minimal degradation (1.01 to 1.09×) consistent with low drift. These patterns show that structured event logs can capture performance trends across different drift regimes.

Event Log Analysis. Retrospective analysis of logged events reveals domain-specific drift patterns. For prediction markets, temporal and liquidity features exhibit largest shifts, corresponding to markets approaching resolution. For electricity, price features show seasonal drift patterns. The structured event history supports this root cause analysis across diverse domains, which would be more difficult to reconstruct from aggregate metrics alone.

4 Related Work

Drift Detection and Adaptation. Extensive work addresses drift detection through statistical methods (Gama et al., 2014; Gemaque et al., 2020; Dos Reis et al., 2016; Rabanser et al., 2019) and

adaptive learning strategies (Losing et al., 2018; Bhardwaj et al., 2022; Lu et al., 2018). Rabanser et al. (2019) provide empirical comparison of drift detection methods, while Lu et al. (2018) survey concept drift adaptation techniques. These approaches focus on algorithmic aspects of detection and adaptation rather than systematic event provenance linking drift to interventions.

ML Governance and Documentation. Model cards (Mitchell et al., 2019), datasheets (Gebru et al., 2021), and fairness audits (Raji et al., 2020) provide static documentation of model properties and design decisions. Our work addresses complementary runtime event tracking that captures temporal evolution and relationships between drift events and system responses.

MLOps and Monitoring. Recent work on ML lifecycle management (Hummer et al., 2019; Paleyes et al., 2022) and data quality verification (Schelter et al., 2018) emphasizes monitoring infrastructure. Paleyes et al. (2022) identify deployment challenges across case studies, highlighting the need for systematic monitoring. These systems primarily track aggregate metrics rather than structured event histories linking drift to interventions.

Production ML Systems. Research on production ML systems (Sculley et al., 2015; Breck et al., 2017; Polyzotis et al., 2017) identifies technical debt and data management challenges. Sculley et al. (2015) characterize hidden technical debt in ML systems, while Breck et al. (2017) propose production readiness metrics. Polyzotis et al. (2017) discuss data management challenges including data validation and monitoring. Our event logging methodology addresses the specific challenge of maintaining structured provenance for drift-related events in production systems.

5 Discussion

Our empirical evaluation across three domains shows that structured event logging supports retrospective analysis of drift-performance relationships under varying drift regimes. Using regularized models with temporal cross-validation to isolate drift effects from overfitting, we observe distinct patterns across domains. Prediction markets exhibit severe drift yet show improved performance on test data, revealing an information convergence phenomenon where markets become more predictable as they approach resolution. Electricity shows moderate drift with modest, consistent degradation across model types, reflecting temporal shifts in pricing patterns. Weather shows minimal drift with correspondingly minimal degradation, reflecting the stability of physical processes. This variation demonstrates that our methodology can be applied across different temporal patterns and drift severities, from rapidly evolving information markets to stable physical processes.

The structured event histories support several forms of analysis that are more difficult with aggregate metrics alone. First, event sequences link specific drift events to intervention decisions and performance outcomes, supporting assessment of intervention effectiveness. Second, temporal pattern analysis can identify recurring drift patterns and seasonal effects. Third, feature-level attribution reveals which features contribute most to distributional shifts, informing feature engineering and data collection strategies. These capabilities support both offline analysis for model improvement and online monitoring for production systems.

Limitations. The retrospective analysis does not address real-time monitoring challenges or automated intervention decision-making. The event taxonomy may require domain-specific adaptation for specialized applications. Future work could explore integration with active learning and online adaptation strategies, as well as automated intervention policies based on event history patterns.

6 Conclusion

We propose a structured event logging methodology for tracking model behavior under distributional drift. Our approach organizes the drift management lifecycle through an event taxonomy spanning detection, intervention, and monitoring phases. The event schema enables temporal queries and sequence reconstruction linking drift patterns to performance outcomes. Empirical evaluation using regularized models and temporal cross-validation across three domains with varying drift characteristics reveals that drift does not always lead to degradation, and that structured event histories enable nuanced analysis of drift-performance relationships. The methodology supports root cause analysis by linking distributional shifts to performance outcomes while controlling for overfitting, providing insights that are more difficult to obtain with conventional metric-based monitoring. This work

contributes to the growing body of research on ML lifecycle management and model governance, providing a foundation for systematic drift management in production systems.

REFERENCES

Romil Bhardwaj, Zhengxu Xiao, Ganesh Patel, Ramesh Ramakrishnan, Ganesh Ananthanarayanan, Ravi Netravali, and Junchen Shen. Ekya: Continuous learning of video analytics models on edge compute servers. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pp. 119–135, 2022.

Eric Breck, Shanqing Cai, Eric Nielsen, Michael Salib, and D Sculley. The ml test score: A rubric for ml production readiness and technical debt reduction. In *IEEE International Conference on Big Data*, pp. 1123–1132. IEEE, 2017.

Denis M Dos Reis, Peter Flach, Stan Matwin, and Gustavo Batista. Fast unsupervised online drift detection using incremental kolmogorov-smirnov test. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1545–1554, 2016.

João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64 (12):86–92, 2021.

Raquel N Gemaque, Allan FB Costa, Rafael Giusti, and Eulanda M dos Santos. An overview of unsupervised drift detection methods. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(6):e1381, 2020.

Waldemar Hummer, Vinod Muthusamy, Thomas Rausch, Parijat Dube, Kaoutar El Maghraoui, Anupama Murthi, and Punleuk Oum. Modelops: Cloud-based lifecycle management for reliable and trusted ai. In *2019 IEEE International Conference on Cloud Engineering (IC2E)*, pp. 113–120. IEEE, 2019.

Viktor Losing, Barbara Hammer, and Heiko Wersing. Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing*, 275:1261–1274, 2018.

Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2018.

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pp. 220–229, 2019.

Andrei Paleyes, Raoul-Gabriel Urma, and Neil D Lawrence. Challenges in deploying machine learning: A survey of case studies. *ACM Computing Surveys*, 55(6):1–29, 2022.

Neoklis Polyzotis, Sudip Roy, Steven Euijong Whang, and Martin Zinkevich. Data management challenges in production machine learning. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pp. 1723–1726, 2017.

Stephan Rabanser, Stephan Günnemann, and Zachary Lipton. Failing loudly: An empirical study of methods for detecting dataset shift. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Inioluwa Deborah Raji, Andrew Smart, Rebecca N White, Margaret Mitchell, Timnit Gebru, Ben Hutchinson, Jamila Smith-Loud, Daniel Theron, and Parker Barnes. Closing the ai accountability gap: Defining an end-to-end framework for internal algorithmic auditing. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pp. 33–44, 2020.

Sebastian Schelter, Dustin Lange, Philipp Schmidt, Meltem Celikel, Felix Biessmann, and Andreas Grafberger. Automating large-scale data quality verification. *Proceedings of the VLDB Endowment*, 11(12):1781–1794, 2018.

David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

A System Architecture and Implementation

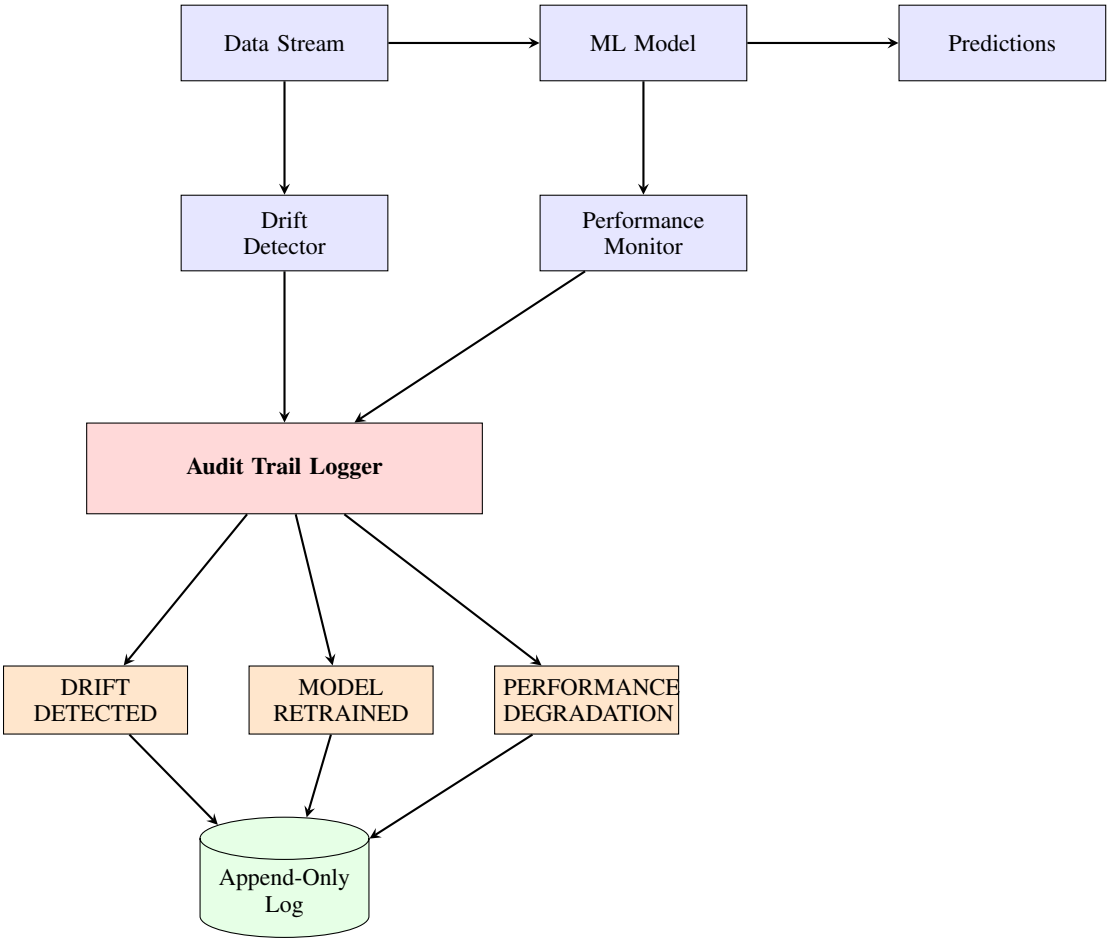

Figure 1: System architecture showing event logging integration with ML pipeline. The logger captures events from drift detection and performance monitoring components.

A.1 Implementation Details

The system comprises two components: an event representation class and a persistence manager. Events are serialized to append-only JSONL format for tamper resistance. Integration requires instrumentation at three pipeline stages: drift detection, model training, and performance evaluation.

B Additional Experimental Results

B.1 Feature Drift Statistics

Table 3 shows complete drift statistics for top 10 features. Temporal features (days_until_resolution, market_age_days) show strongest drift, followed by liquidity features.

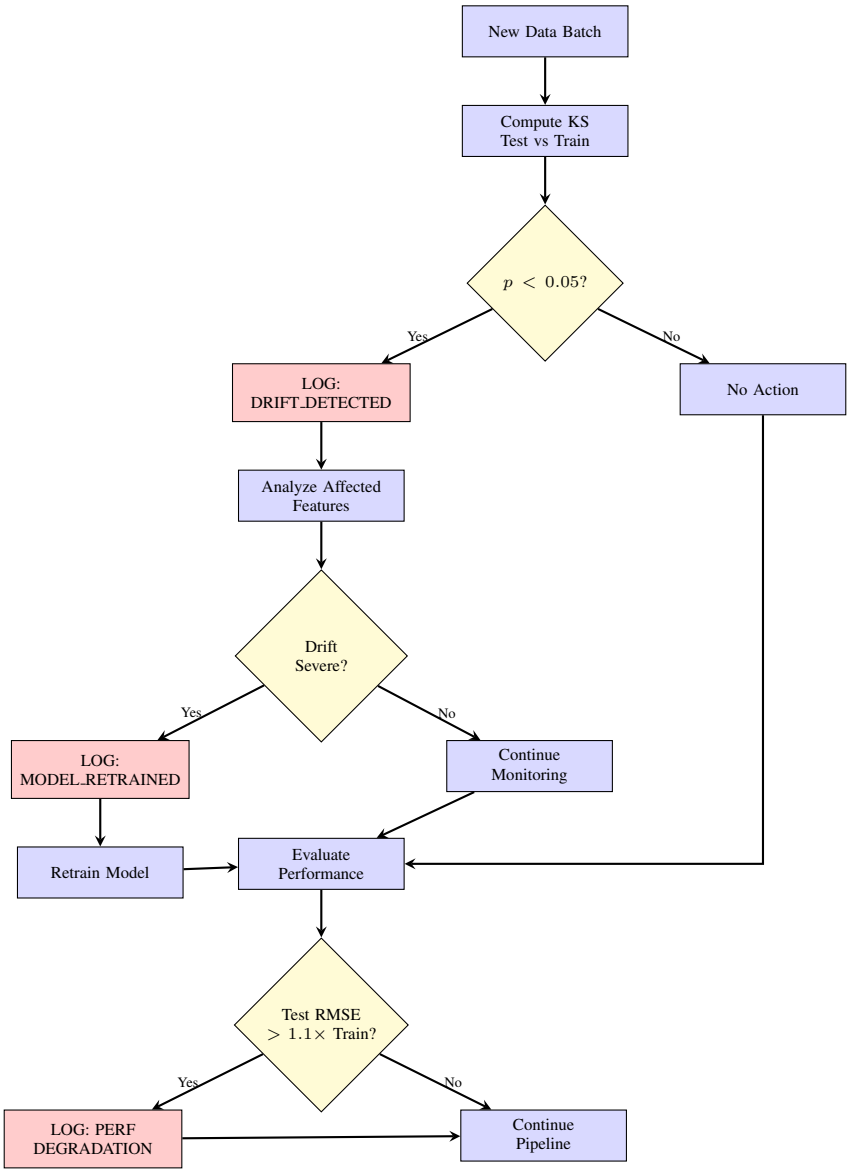

Figure 2: Drift detection and intervention workflow with audit logging. Each decision point triggers structured event logging.

B.2 Audit Log Sample

Below is a sample DRIFT_DETECTED event from the experiment:


```
{
  "event_id": "a3f2c1...",
  "timestamp": "2025-01-30T10:23:45Z",
  "event_type": "DRIFT_DETECTED",
  "severity": "WARNING",
  "user": "system",
  "model_name": "prediction_market_model",
  "metadata": {
    "drift_metric": 0.198,
    "threshold": 0.05,
```

Table 3: Top 10 features by KS statistic showing distribution shift between train and test sets

Feature	KS Statistic	p-value
days_until_resolution	0.407	<0.001
liquidity	0.313	<0.001
volume_24h	0.246	<0.001
num_traders	0.215	<0.001
bid_ask_spread	0.167	<0.001
market_age_days	0.156	<0.001
price_volatility	0.142	<0.001
volume_7d	0.138	<0.001
unique_traders_24h	0.129	<0.001
price_momentum	0.121	<0.001


```
    "num_features_drifted": 39,
    "total_features": 44,
    "top_drifted_features": [
      "days_until_resolution",
      "liquidity",
      "volume_24h"
    ]
  }
}
```