Detecting Retail Sales Anomalies — Mini Paper

Author: Sri Charan Konidina
Date: October 14, 2025
Project: retail-anomaly-poc (GitHub) — https://github.com/konidinasricharan/retailanomaly-poc

Abstract

This mini-paper summarizes a proof-of-concept pipeline to detect anomalous retail transactions using Isolation Forest on synthetic and real retail datasets (Kaggle Supermarket Sales). The work includes interactive and static visualizations, a Jupyter notebook, a streaming simulation, and a short public dissemination effort (LinkedIn + Medium). The artifact documents: data sources, model choices, results, and a suggested roadmap for productionization.

1. Motivation

Retailers suffer revenue loss and operational pain from anomalous transactions (fraud, mispricing, refunds, mis-scans). Rapid detection reduces losses and improves auditability. The aim was to build a compact, reproducible pipeline demonstrating feasibility using accessible tools.

2. Data

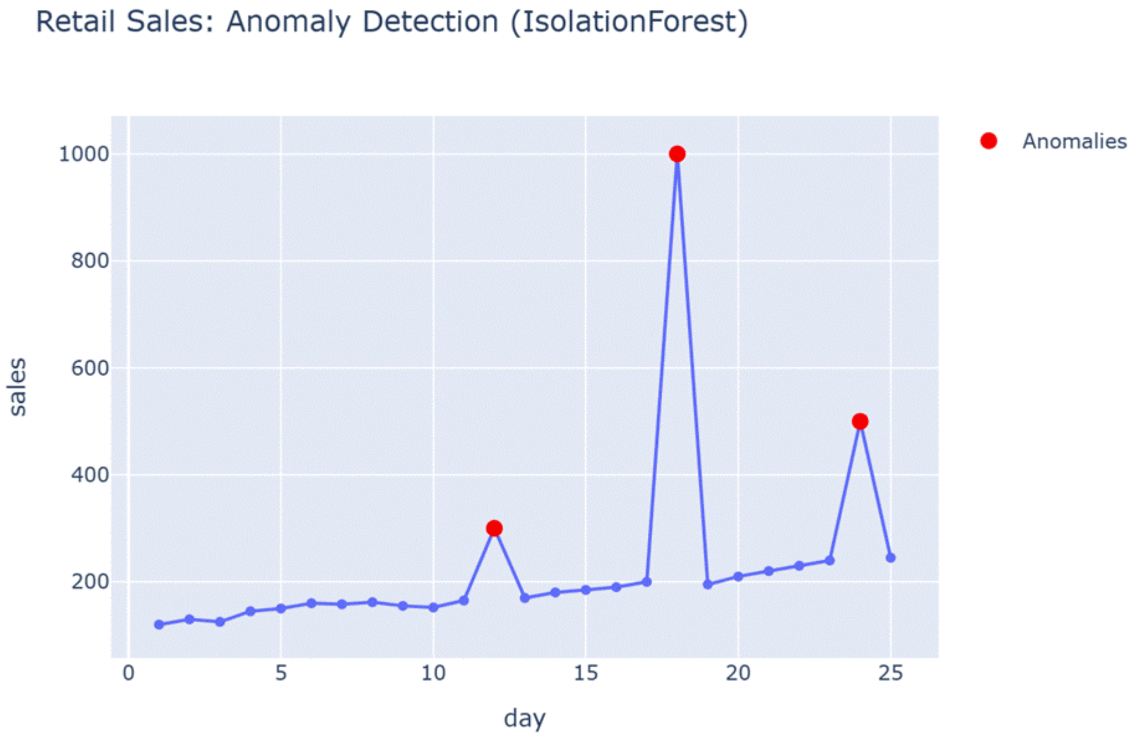
- Synthetic sample for initial POC (Days 0–2).
- Kaggle "Supermarket Sales" dataset (Day 4) — saved as `sales.csv` .
- Simulated streaming sample for real-time demonstration (Day 6).

3. Methods

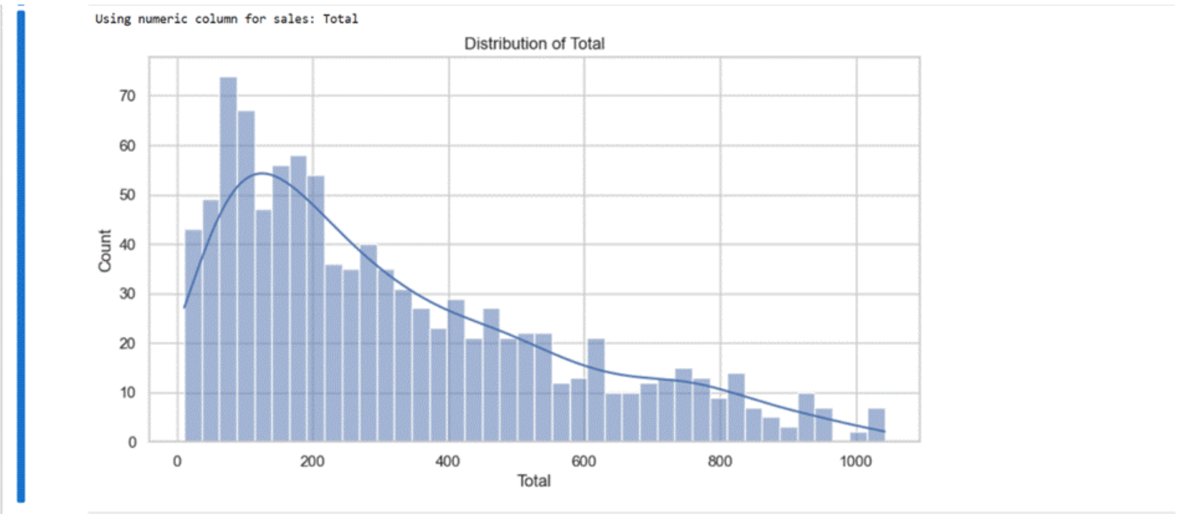
- Preprocessing: basic cleaning, parse `Date` / `Time` where available.
- Unsupervised detection: `IsolationForest` (scikit-learn) with small contamination rates (0.5–1% for large datasets; 8–10% for toy streams to highlight anomalies). Visualizations:
- Matplotlib (static), Plotly (interactive), and live-updating plots for streaming demo.

4. Experiments & Results

- **Baseline (Day 0–2):** small PoC using IsolationForest; demo script `demo.py` detected synthetic outliers and shows reproducible output examples.



- **Real data (Day 4):** EDA on Kaggle supermarket dataset (`retail_sales_analysis.ipynb`); histogram of sales, total-by-product-line bar chart, weekday boxplots; IsolationForest found ~1% anomalous transactions (high-value spikes).



```
Using numeric column for sales: Total
```

- **Streaming demo (Day 6):** `retail_stream_anomaly.ipynb` simulates incremental data and visualizes anomalies appearing in real time.

Key numeric summary: - Dataset rows (example): see `retail_sales_analysis.ipynb` head and info. - Anomalies detected (example counts): reported inline in the notebooks and PDF exports.

5. Reproducibility & Files

All code, notebooks, and reports are in the repo: - `demo.py` — baseline PoC - `visual_demo.py` — interactive Plotly demo - `retail_anomaly_demo.ipynb` , `retail_anomaly_demo_report.pdf` — Day 3 notebook + report `retail_sales_analysis.ipynb` , `retail_sales_analysis_report.pdf` — Day 4 real-data analysis - `retail_stream_anomaly.ipynb` , `retail_stream_anomaly_report.html` — Day 6 streaming demo - `mini_paper.md` , `mini_paper_report.pdf` — this artifact

6. Discussion

- IsolationForest is quick and interpretable for simple anomaly detection; for temporal patterns, a sequence model (LSTM-autoencoder) or statistical control charts may be superior.
- Visualizations (interactive + static) accelerate human triage.
- For production: ingest streaming data (Kafka), maintain a model registry, backfill labels where available, and set up alerting dashboards.

7. Next Steps (roadmap)

1. Build LSTM autoencoder baseline for temporal anomalies (Q4 2025).
2. Develop a Streamlit dashboard for triage and human-in-the-loop verification.
3. Prepare a workshop submission or short conference demo (submit abstract + notebook).
4. Gather expert recommendation letters and invite review from a retail analytics practitioner.

8. How to run (quick)

-```bash

Example: run baseline PoC

python demo.py

Run interactive Plotly demo

python visual_demo.py

Start Jupyter for notebooks

python -m notebook

9. Acknowledgements & Contacts

Author: Sri Charan Konidina — Applied Security & Analytics GitHub: https://github.com/konidinasricharan/retail-anomaly-poc LinkedIn: https://www.linkedin.com/in/sricharankonidina/