An Assignment Report

On

“Time Series Anomaly Detection for IoT Sensors”

Submitted

By

Syed Zubair Yousuff

yousufzubair018@gmail.com

Submitted

To

Tsworks Technologies

AI/ML Recruitment Team

tsworks

ABSTRACT

This project presents an end-to-end anomaly detection pipeline designed for IoT sensor data. The dataset, sourced from the Numenta Anomaly Benchmark (NAB), consists of CPU utilization readings from an AWS EC2 instance. The goal was to detect abnormal system behaviours that may indicate hardware or performance issues. Two models were implemented the Isolation Forest for statistical anomaly detection and the LSTM Autoencoder for temporal sequence learning. Data preprocessing, feature engineering, and visualization were performed using Pandas, NumPy, and Matplotlib. The results showed that the Isolation Forest achieved high precision for sharp anomalies, while the LSTM Autoencoder captured broader anomaly sequences with improved recall. This project demonstrates the effectiveness of combining statistical and deep learning approaches for real-world IoT anomaly detection.

Time Series Anomaly Detection for IoT Sensors

1. Problem Understanding

IoT sensors and cloud infrastructure continuously generate large volumes of time-series data. Anomalies in these readings may indicate equipment malfunction, performance degradation, or the need for maintenance. The goal of this project is to build an end-to-end anomaly detection pipeline that can automatically detect unusual CPU utilization patterns from time-series data.

This task is unsupervised, as real-world anomaly labels are rarely available. Therefore, two complementary approaches were used:

1. **Statistical/Unsupervised Model:** Isolation Forest.
2. **Deep Learning Model:** LSTM Autoencoder.

Both methods were evaluated on the same dataset and compared in terms of accuracy, interpretability, and detection quality.

2. Data Preparation and Exploration

Dataset Description:

The dataset (ec2_cpu_utilization_5f5533.csv) was obtained from the Numenta Anomaly Benchmark (NAB). It contains a univariate time-series of CPU utilization (%) recorded from an AWS EC2 instance.

Preprocessing Steps:

- Loaded data using pandas and converted timestamps to datetime format.
- Sorted chronologically and handled missing values using forward fill.
- Checked for duplicate timestamps (none found).
- Descriptive statistics and line plots showed stable CPU activity with periodic spikes.

Outlier Inspection:

- Applied the IQR (Interquartile Range) method to identify initial outliers.
- Visualized them via boxplots and red markers on the time-series plot.

3. Feature Engineering

To capture temporal patterns and smooth out noise:

- Computed rolling mean and rolling standard deviation with a window of 10.
- Created lag features (lag_1, lag_2, lag_3) to include recent history.
- Standardized all features using StandardScaler to normalize ranges.

These features helped the models differentiate between normal fluctuations and true anomalies.

4. Approach 1 — Isolation Forest (Statistical / Unsupervised)

The Isolation Forest algorithm isolates anomalies by randomly partitioning feature space. Points requiring fewer splits to isolate are likely anomalous.

Implementation Highlights:

- Used n_estimators = 100 and contamination = 0.001.
- Input features: value, rolling_mean, and rolling_std.
- Detected anomalies were plotted on the time-series chart.

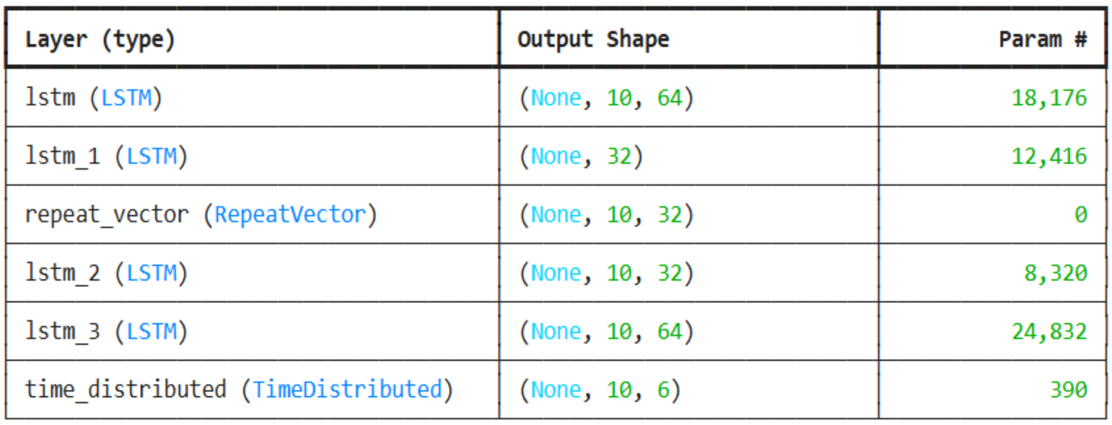
Observations:

- The model correctly identified the major spikes and deviations.
- Slight temporal shifts were observed because of rolling feature windows.
- Detection rate was sensitive to the contamination parameter — smaller values produced more accurate results.

5. Approach 2 — LSTM Autoencoder (Deep Learning)

To capture sequential dependencies, a sequence-to-sequence LSTM Autoencoder was implemented.

Model Architecture:



Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 10, 64)	18,176
lstm_1 (LSTM)	(None, 32)	12,416
repeat_vector (RepeatVector)	(None, 10, 32)	0
lstm_2 (LSTM)	(None, 10, 32)	8,320
lstm_3 (LSTM)	(None, 10, 64)	24,832
time_distributed (TimeDistributed)	(None, 10, 6)	390

Fig. 01: Model Architecture

Working Principle:

The Autoencoder learns to reconstruct "normal" time-series sequences. When an anomalous pattern appears, reconstruction error increases sharply. Anomalies were defined as sequences with error above the 99.5th percentile.

Results:

- The LSTM Autoencoder detected broader anomaly windows around actual spikes.
- It was more sensitive to temporal context but produced more false positives compared to Isolation Forest.

6. Model Evaluation and Comparison

Evaluation Setup:

Since the dataset lacked true labels, IQR-based outliers were used as the ground-truth reference.

Metrics calculated:

- Precision, Recall, F1-Score and Confusion Matrices for both models.

7. Key Insights

1. **Isolation Forest** is lightweight, interpretable, and suitable for real-time applications.
2. **LSTM Autoencoder** captures context better and is ideal when sequences show gradual deviation patterns.

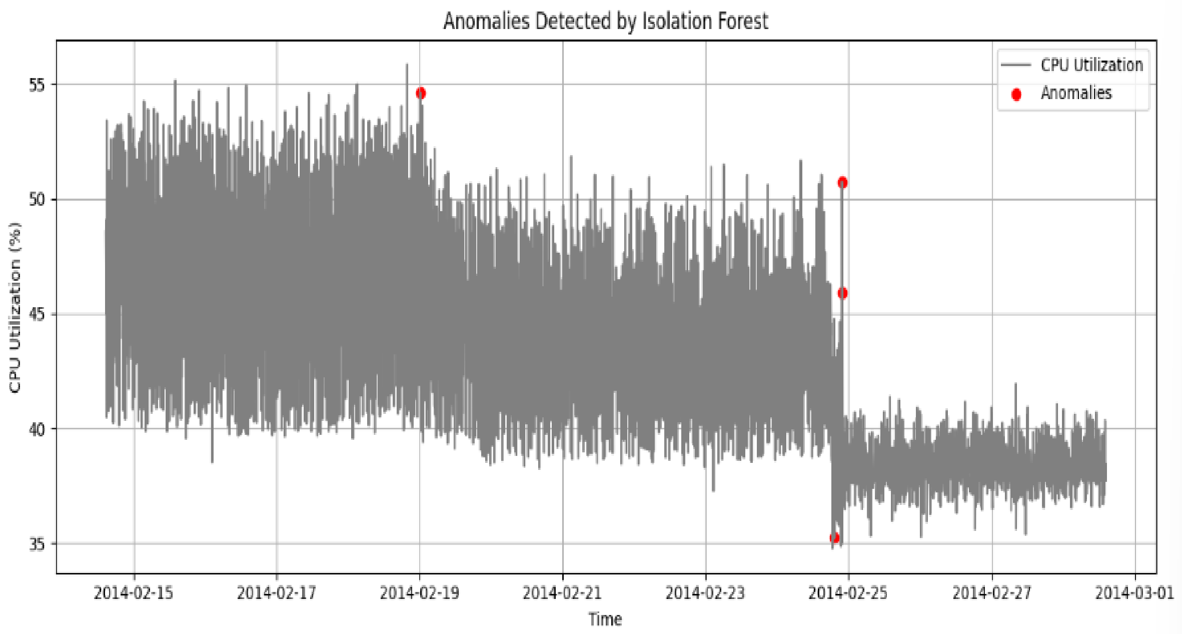


Fig. 02: Anomalies Detected by Isolation Forest.

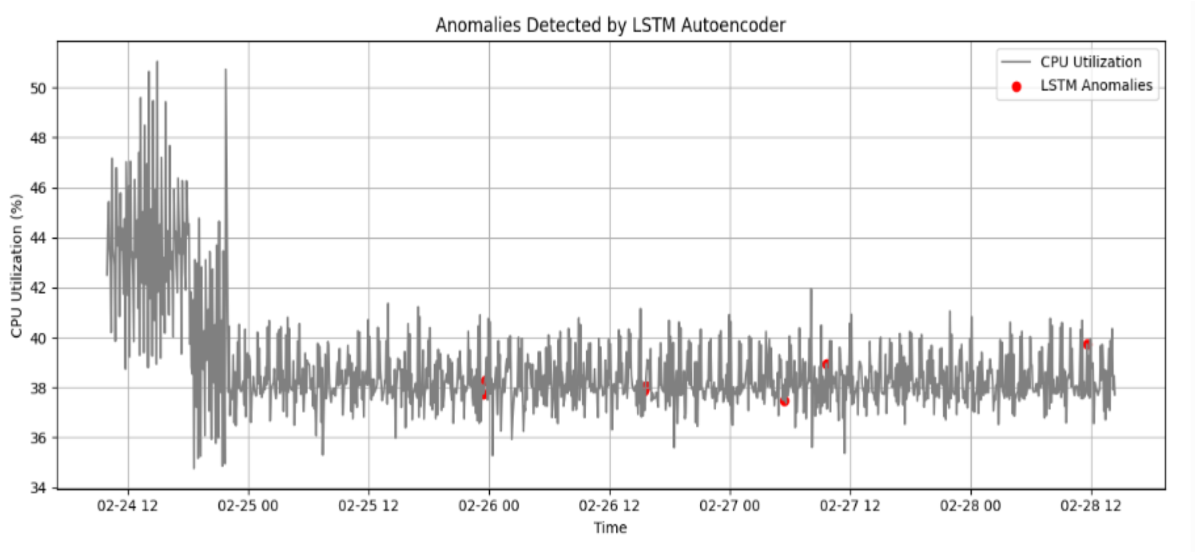


Fig. 03 Anomalies Detected by LSTM Autoencoder.

8. Conclusion

This project successfully demonstrated an end-to-end anomaly detection pipeline for time-series sensor data. Both Isolation Forest and LSTM Autoencoder effectively detected unusual CPU utilization patterns. While the Isolation Forest offered precision and simplicity, the LSTM model provided deeper insight into temporal anomalies. Together, these approaches can form the foundation of a robust IoT anomaly detection system capable of identifying early warning signals for equipment maintenance.

9. References

1. Numenta Anomaly Benchmark (NAB) Dataset – *https://github.com/numenta/NAB*
2. Liu, F. T., Ting, K. M., & Zhou, Z. (2008). *Isolation Forest.* ICDM.
3. Malhotra et al. (2016). *LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection.*