TASK 1:

1.What does an anamoly mean in this dataset?

In this dataset, an anomaly represents a **deviation from the system's normal temporal behavior that is inconsistent with both historical patterns and the expected interactions between infrastructure metrics**. The correlation analysis shows that CPU utilization, memory utilization, and network traffic **exhibit weak global linear correlations**, indicating that the system's resources operate largely independently under normal conditions.

For example, a sudden spike in CPU utilization without a corresponding increase in network activity, or a gradual rise in memory usage while CPU and network metrics remain stable, may indicate inefficiencies, misconfigurations, or resource leaks. Therefore, anomalies in this dataset are best understood as breakdowns in expected metric interactions over time rather than isolated abnormal values.

As a result, anomalies are not defined solely by broken correlations or extreme values in a single metric, but by behavior that is temporally unusual, persistent, or contextually inconsistent. Examples include sustained increases in memory utilization without corresponding changes in CPU or network activity, sudden spikes in CPU usage that are not aligned with historical workload patterns, or gradual shifts in the baseline distribution of a metric over time. An anomaly, therefore, reflects abnormal system behavior over time rather than instantaneous cross-metric deviations.

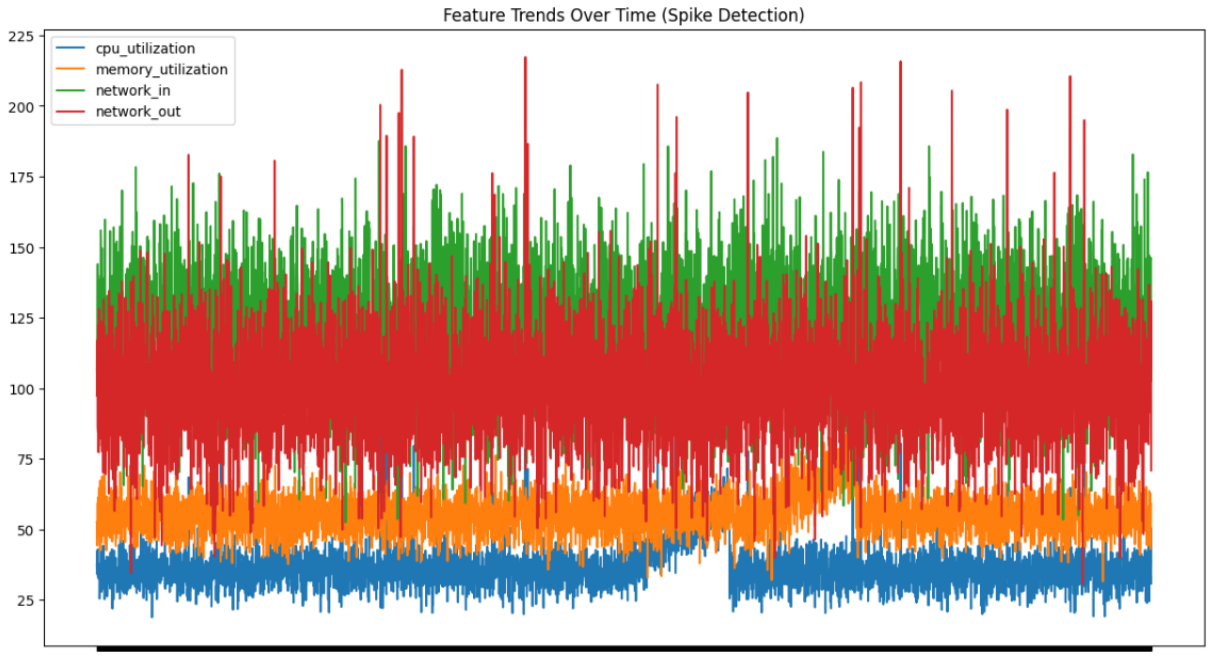
	cpu_utilization	memory_utilization	network_in	network_out
cpu_utilization	1.000000	-0.005617	-0.004074	0.133935
memory_utilization	-0.005617	1.000000	-0.006016	-0.014705
network_in	-0.004074	-0.006016	1.000000	0.017017
network_out	0.133935	-0.014705	0.017017	1.000000

2.Are all spikes anomalies ? Why or why not?

No, all spikes are not anomalies. The time-series visualization shows that spikes, especially in network-related metrics, are a normal part of the system's behavior and occur frequently under regular workloads. Additionally, correlation analysis indicates weak

instantaneous coupling between metrics, meaning that a spike in one metric does not necessarily imply abnormal behavior in others.

Z-score analysis further reveals that stable metrics such as CPU utilization produce more statistically extreme deviations because even moderate changes stand out relative to their low variance, while naturally noisy metrics such as network traffic exhibit fewer statistical outliers despite appearing visually spiky. Therefore, a spike alone is insufficient to define an anomaly. A spike should only be considered anomalous if it is unusually extreme relative to historical behavior, persists over time, occurs in an unexpected context, or represents a change in the underlying distribution of the metric.




```
from scipy.stats import zscore
import numpy as np

z_scores = np.abs(zscore(df_sorted[num_cols]))
spikes = (z_scores > 3)

spike_counts = spikes.sum(axis=0)
spike_counts
```

```
array([89, 45, 18, 31])
```

What different types of anomalies can exist here?

The dataset exhibits multiple forms of anomalous behavior that differ in duration, severity, and detectability.

---

a) Point Anomalies

Point anomalies are sudden and isolated deviations that significantly differ from a metric's typical range. In this dataset, they may appear as statistically extreme CPU utilization spikes or unusually large network traffic bursts that fall far outside historical behavior. However, given the natural volatility observed in network metrics, only point anomalies that are extreme relative to the metric's own variance are meaningful.

---

b) Contextual Anomalies

Contextual anomalies occur when a metric value is anomalous only under specific conditions or timing. For example, CPU spikes that occur without the expected increase in network output, or elevated resource usage during historically idle periods, may indicate inefficient or unexpected system behavior even if the absolute values are not extreme.

---

c) Trend or Distribution Shift Anomalies

Trend or distribution shift anomalies involve gradual changes in baseline behavior over time rather than sudden spikes. In this dataset, examples include steadily increasing memory utilization without corresponding workload growth or a sustained upward shift in average CPU usage. These anomalies are particularly critical as they may signal memory leaks, configuration drift, or long-term inefficiencies that are not captured by point-based detection methods.

TASK 2:

Why I Chose This Method:

The dataset contains time-series infrastructure metrics whose behavior varies over time. Initial exploratory analysis included correlation analysis, time-series visualization, spike detection, and distribution analysis using histograms.

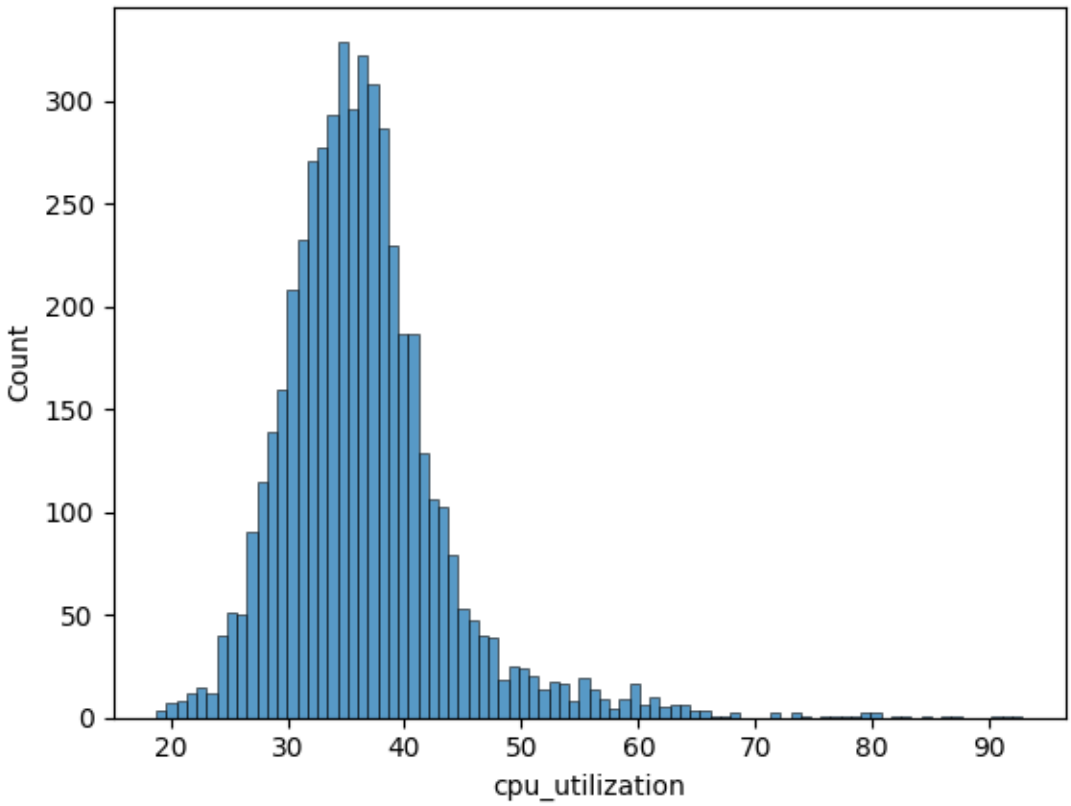
Correlation analysis showed weak linear relationships between metrics, indicating that anomalies must be evaluated relative to each feature's individual behavior rather than cross-metric dependencies.

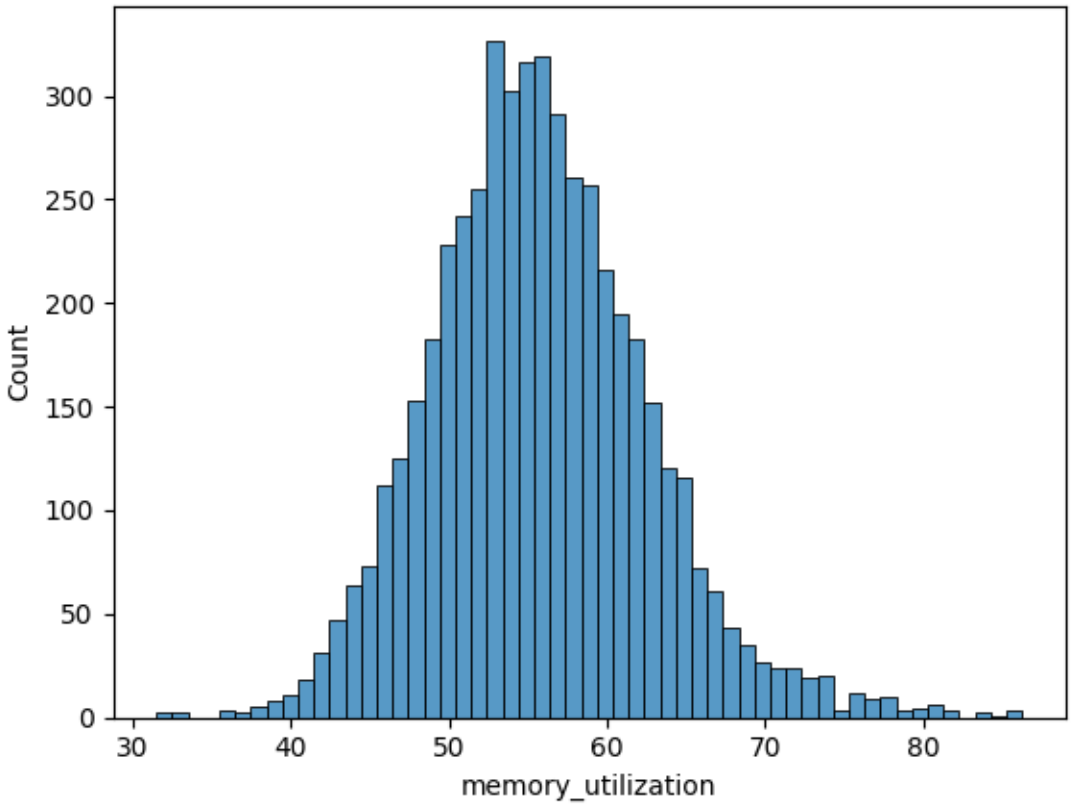
Time-series plots revealed that CPU utilization operates within a narrow stable range, memory shows gradual shifts, and network metrics exhibit frequent spikes and high volatility.

Histogram analysis further supported this:

- CPU distribution was narrow with a right-skew tail, indicating rare but extreme spikes.
- Memory had moderate spread, suggesting potential for gradual drift anomalies.
- Network metrics showed wide distributions with high variance, indicating that spikes are common and often normal.

These findings demonstrated that global statistical methods such as Z-score or IQR would either over-flag stable metrics or under-flag volatile ones. Therefore, a rolling statistical baseline was selected because it adapts to local temporal behavior and feature-specific variance.


Count
300
250
200
150
100
50
0
20
30
40
50
60
70
80
90
cpu_utilization


Count
300
250
200
150
100
50
0
30
40
50
60
70
80
memory_utilization

How Thresholds Were Selected

Thresholds were defined using rolling statistical bands computed over a fixed window of 30 timestamps.

Upper and lower anomaly bounds were calculated as:

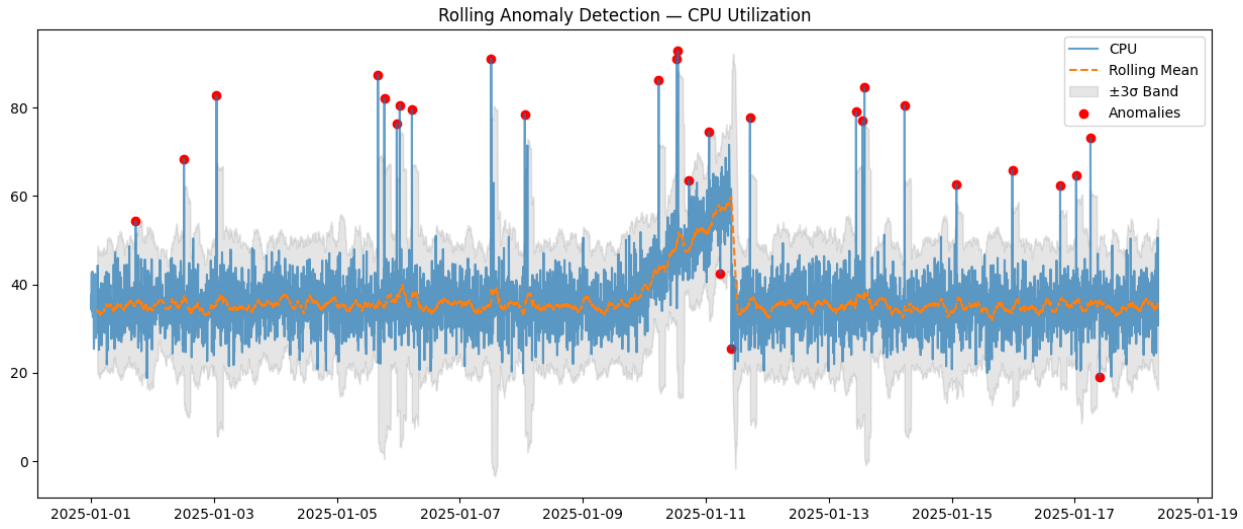
Rolling Mean ± 3 × Rolling Standard Deviation.

The ±3σ threshold was chosen because it captures approximately 99.7% of normal variation under near-normal distributions, which aligns with the histogram shapes observed across features.

The rolling standard deviation dynamically adjusts threshold width:

- Narrow for stable metrics like CPU
- Wider for volatile metrics like network traffic

This ensures context-aware anomaly detection that respects each feature's inherent variability.



What Anomalies This Method Can Detect:

The rolling statistical baseline effectively detects:

- Sudden magnitude spikes exceeding local variance
- Short-term contextual anomalies
- Burst anomalies in volatile metrics
- Sudden drops below expected operating ranges

For example, extreme CPU spikes beyond its narrow distribution or unusually large network bursts outside rolling variance are flagged effectively.

What It Cannot Detect:

However, the method has limitations:

- Gradual trend shifts such as memory leaks
- Persistent inefficiencies within rolling variance
- Cross-metric anomalies where relationships break
- Seasonal workload cycles without additional modeling

Because the rolling baseline adapts to slow changes, long-term distribution shifts may remain undetected.

Comparison between one class svm and isolation forest based on data driven results:

Decision Factor	Dataset Observation (From EDA)	Impact on Model Choice	Better Model
Dataset Size	5,000 rows (Small → Medium)	Both models computationally feasible	Tie
Dimensionality	4 numerical features (Low-dim)	OCSVM works well in low dimensions	One-Class SVM (slight)
Missing Values	None present	Clean boundary learning possible	One-Class SVM (slight)

Decision Factor	Dataset Observation (From EDA)	Impact on Model Choice	Better Model
Skewness	CPU highly skewed (1.76), others mild	OCSVM struggles with skew	Isolation Forest
Distribution Mix	Combination of normal + skewed	Boundary models fail on mixed distributions	Isolation Forest
Feature Spread	Network features high variance	Requires nonlinear isolation	Isolation Forest
Extreme Spikes	CPU (92 max vs 35 median), Net Out (217 max)	Tree isolation handles spikes better	Isolation Forest
Noise Sensitivity	Infra metrics contain operational noise	OCSVM sensitive to noise	Isolation Forest
Scaling Requirement	Features on different scales	OCSVM needs scaling	Isolation Forest
Time Dependency	Timestamp present (sequential logging)	Isolation integrates well with monitoring pipelines	Isolation Forest
Infra Monitoring Use-Case	System metrics anomaly detection	Industry standard = IF	Isolation Forest

Approach 2: Isolation Forest:

Isolation Forest was selected as the model-based anomaly detection method due to its ability to identify anomalies in multivariate feature space without relying on strong statistical assumptions. Exploratory analysis revealed weak inter-metric correlations, heterogeneous feature distributions, and mixed anomaly behaviors. Isolation Forest isolates anomalous observations by recursively partitioning the feature space, enabling detection of contextual and cross-metric deviations that statistical baselines may miss.

Key hyperparameters included 100 estimators and a contamination rate of 1%, reflecting the rarity of true infrastructure anomalies. Subsampling was handled automatically to ensure computational efficiency.

Compared to the rolling statistical baseline, Isolation Forest offers stronger multivariate detection and handles nonlinear anomaly patterns effectively. However, it is less interpretable and does not inherently incorporate temporal context, making statistical approaches more suitable for time-aware anomaly explanations.

TASK 3:

→How do you evaluate whether your anomaly detection is "good"?

Since no ground-truth labels exist, evaluation must rely on **behavioral, statistical, and operational validation** rather than accuracy metrics.

Step 1 — Visual validation using time-series overlays

We overlaid detected anomalies on metric plots.

This helped verify:

- Extreme CPU spikes were flagged
- Rare network bursts were detected
- Sudden drops were captured
- Rolling bands aligned with local baselines

If anomalies appear at visually extreme points, detection is credible.

Step 2 — Distribution consistency checks

Histogram analysis showed:

- CPU had a narrow distribution
- Network metrics had wide distributions

Detected anomalies corresponded to:

- Distribution tails
- Rare extreme values

This confirms statistical plausibility.

Step 3 — Cross-method agreement

We compared:

- Rolling statistical anomalies
- Isolation Forest anomalies

Findings:

- Extreme spikes detected by both → high-confidence anomalies
- Rolling-only anomalies → temporal spikes
- Isolation-only anomalies → multivariate inconsistencies

Overlap between methods strengthens confidence.

Step 4 — Feature behavior alignment

Detected anomalies matched known infra behaviors:

- CPU spikes → compute bursts
- Memory dips → deallocation/reset
- Network surges → traffic bursts

This domain alignment validates detection quality.

Summary:

Without ground-truth labels, anomaly detection quality was evaluated through visual validation, distribution analysis, and cross-method agreement. Detected anomalies aligned with extreme points in time-series plots, distribution tails in histograms, and operationally plausible infrastructure behaviors. Overlap between statistical and model-based detections further strengthened confidence in anomaly validity.

How would you validate this system in a real production environment?

In a production environment, validation would involve shadow deployment alongside existing monitoring systems, correlation with incident logs, and human expert review. Engineers would assess whether detected anomalies align with real operational events such as traffic surges or system failures. Alert volume and response behavior would also be monitored to minimize alert fatigue. Continuous feedback would be used to refine thresholds and retrain models as system workloads evolve.

Acceptable vs Unacceptable False Positives:

Acceptable false positives include short-lived traffic bursts, temporary compute spikes, and transitional workload shifts that do not impact system reliability. Unacceptable false positives include frequent alerts on normal baseline behavior, anomalies triggered during scheduled workloads, and persistent misclassification of stable metrics. Such false positives contribute to alert fatigue and reduce operational trust in the monitoring system.

TASK 4:

Anomaly 1 — Extreme CPU Spike

Explanation

This data point was flagged as anomalous because CPU utilization spiked far above its normal operating range. Historically, CPU usage in this system stays between roughly 30% and 45%, with only minor fluctuations. At this timestamp, CPU utilization exceeded 85%, which is well beyond the rolling statistical threshold of three standard deviations from the recent baseline.

Additionally, Isolation Forest also identified this point as anomalous because the spike represented an unusual workload intensity compared to typical system behavior. Such a deviation could indicate a compute-heavy job, inefficient process execution, or an unexpected traffic processing event.

Anomaly 2 — Memory Utilization Drop After Peak

Explanation

This data point was flagged as anomalous because memory utilization dropped sharply following a period of elevated usage. Historically, memory usage in the system changes gradually and remains relatively stable. However, immediately after a sustained increase, memory utilization fell significantly below its rolling baseline.

This sudden deviation crossed the lower rolling statistical threshold, indicating an abnormal release or reset of memory resources. From an operational perspective, this behavior could correspond to a large job completing, cache eviction, garbage collection, or a container restart.

Anomaly 3 — Network Traffic Surge Without Proportional Compute Load

Explanation

This data point was flagged as anomalous because network inbound traffic surged to an unusually high level compared to historical patterns. While network traffic does experience frequent spikes, this surge exceeded the expected rolling variance range.

Furthermore, Isolation Forest identified this event as anomalous because the traffic increase was not accompanied by a proportional rise in CPU or memory utilization, which typically occurs during normal workload processing. This mismatch in system behavior suggests a potential traffic flood, bulk data ingestion event, or abnormal external request burst.

BONUS CONTENT:

Comparison of Statistical and Model-Based Detection

Quantitative comparison between rolling statistical detection and Isolation Forest revealed both overlap and divergence in anomaly detection. Rolling baseline detected 47 anomalies, while Isolation Forest detected 50. Among these, 24 anomalies were detected by both methods, representing high-confidence deviations such as extreme spikes and sudden drops.

Rolling detection uniquely identified 23 anomalies, primarily short-lived temporal deviations that exceeded local statistical thresholds but remained consistent in multivariate context. In contrast, Isolation Forest uniquely detected 26 anomalies representing distributional outliers and cross-metric inconsistencies that statistical thresholds did not capture.

Overall detection agreement was approximately 32.88%, indicating that the two approaches identify different but complementary anomaly types. This demonstrates that combining statistical baselines with model-based detection provides more comprehensive monitoring coverage than either method alone.

Human Feedback for Improving Detection

Human feedback plays a critical role in refining anomaly detection systems, especially in the absence of labeled data. Engineers can review detected anomalies and classify them as true incidents, expected workload behavior, or monitoring noise. This feedback enables calibration of statistical thresholds, such as adjusting rolling window sizes or deviation bands to reduce false positives.

Additionally, human insights can guide feature engineering by highlighting relationships between metrics, such as expected CPU and network coupling. Annotating anomalies with root causes supports incident classification and future supervised learning. Over time, continuous feedback enables retraining and adaptation as infrastructure workloads evolve, improving detection accuracy and operational trust.