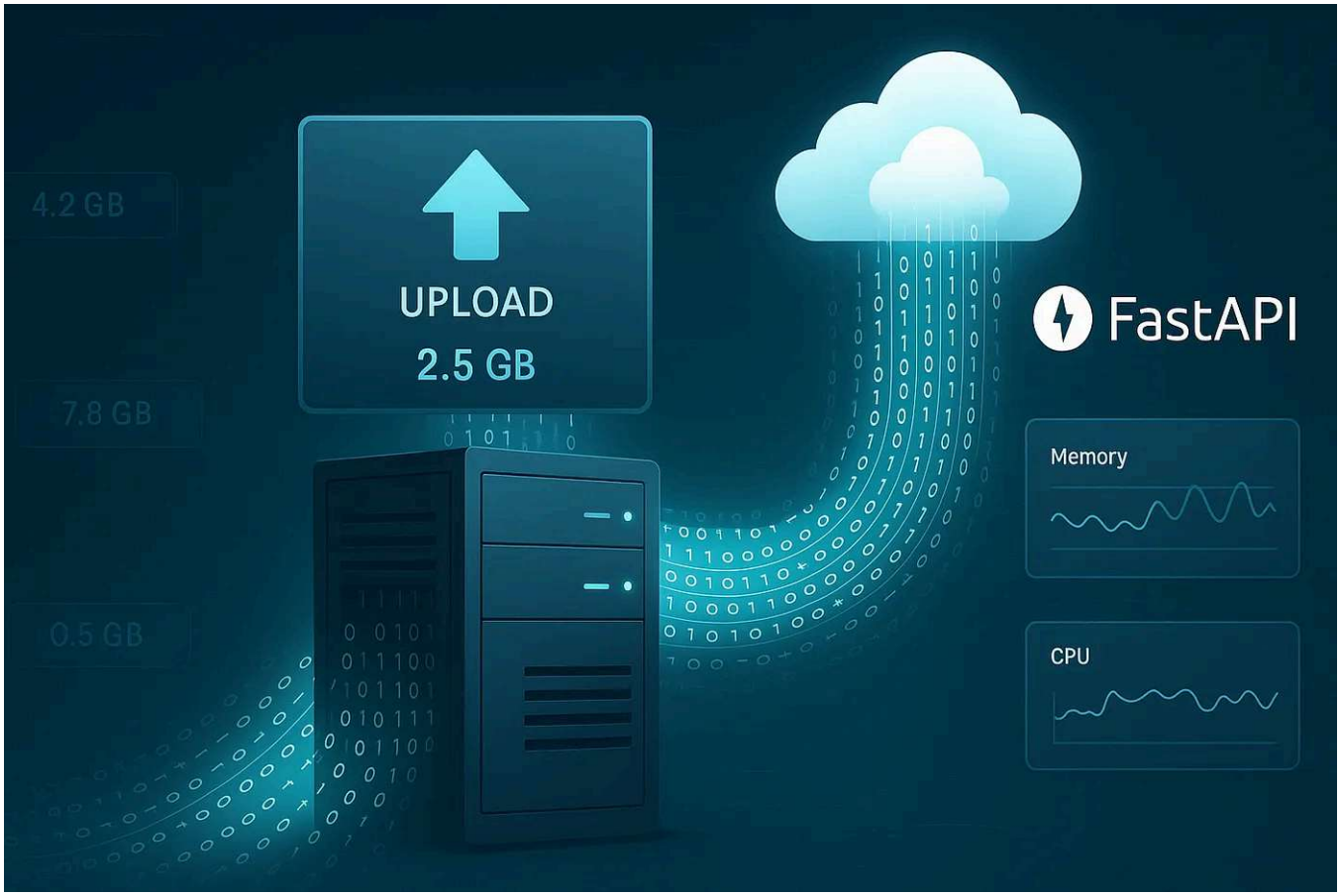
How I Used Pandas to Detect Anomalies in Time-Series Data Without ML

Windowed averages, IQR, z-score, and rolling variance techniques

3 min read · Jul 15, 2025

Bhagya Rana



Detecting anomalies in time-series data doesn't always require complex machine learning models. In many real-world scenarios, traditional statistical methods

combined with the power of Pandas are more than enough. In this article, I'll walk through how I used Pandas to uncover spikes, dips, and inconsistencies in time-stamped data using four core techniques — windowed averages, IQR filtering, z-scores, and rolling variance.

📈 The Dataset: Synthetic But Realistic

To illustrate the techniques, I used a simulated dataset resembling hourly sensor readings from an IoT device. These values follow a sinusoidal trend with occasional random anomalies injected.

```
import pandas as pd
import numpy as np

# Simulated time-series data
np.random.seed(42)
timestamps = pd.date_range(start='2023-01-01', periods=1000, freq='H')
signal = np.sin(np.linspace(0, 20, 1000)) + np.random.normal(0, 0.2, 1000)

# Inject anomalies
signal[300] += 5
signal[600] -= 4
signal[850] += 3

df = pd.DataFrame({'timestamp': timestamps, 'value': signal})
df.set_index('timestamp', inplace=True)
```

🧠 Technique 1: Windowed (Rolling) Averages

Rolling averages help smooth the data and reveal points that deviate significantly from the local trend.

```
df['rolling_mean'] = df['value'].rolling(window=24).mean()
df['rolling_std'] = df['value'].rolling(window=24).std()
```

```
# Flag values > 3 std deviations from rolling mean
df['anomaly_rolling'] = (abs(df['value'] - df['rolling_mean']) > 3 * df['rollir
```

This method is great for local trend comparison — a sudden jump in value relative to its recent history gets flagged.

📊 Technique 2: IQR-Based Outlier Detection

The Interquartile Range (IQR) is another robust method for spotting outliers without assuming normal distribution.

```
Q1 = df['value'].quantile(0.25)
Q3 = df['value'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

df['anomaly_iqr'] = (df['value'] < lower_bound) | (df['value'] > upper_bound)
```

Use this when you want a global view of anomalies — great for detecting point anomalies in relatively stable data.

🧮 Technique 3: Z-Score Standardization

Z-score helps find anomalies assuming the values are normally distributed.

```
mean_val = df['value'].mean()
std_val = df['value'].std()
```

```
df['z_score'] = (df['value'] - mean_val) / std_val
df['anomaly_zscore'] = abs(df['z_score']) > 3
```

Although this assumes normality, it's surprisingly effective in many datasets, especially when you're trying to find global outliers.

🔁 Technique 4: Rolling Variance Spikes

Sometimes, it's not the value but the **variance** that signals something odd is happening — such as noisy equipment or unstable metrics.

```
df['rolling_var'] = df['value'].rolling(window=12).var()
threshold = df['rolling_var'].mean() + 3 * df['rolling_var'].std()

df['anomaly_var'] = df['rolling_var'] > threshold
```

You can use this to detect sudden volatility that might indicate systemic issues or unstable behavior.

✅ Summary of Techniques

Here's a quick comparison of the methods:

```
Method             | Local/Global | Suitable For              | Notes
------------------ | ------------ | ------------------------- | --------------
Rolling Mean/Std   | Local        | Smooth anomalies          | Sensitive to w
IQR                | Global       | Spiky outliers            | No distributic
Z-Score            | Global       | Normal-like distributions | Simple, fast
Rolling Variance   | Local        | Volatility detection      | Useful for uns
```

🔍 Final Thoughts

You don't always need TensorFlow or PyTorch to find anomalies. Often, your best tool is a well-thought-out statistical method implemented with a few lines of Pandas.

These approaches are especially useful in edge devices, streaming pipelines, or lightweight systems where machine learning isn't feasible. They also offer more explainability than opaque models — something that's crucial in regulated or safety-critical environments.

If you're dealing with time-series data, try these first. They just might be all you need.

Pandas | Time Series Analysis | Anomaly Detection | Data Science | Python

Follow

Written by Bhagya Rana

1.4K followers · 366 following

Professional & Motivational Helping you turn chaos into clarity through systems, focus & smart productivity. Progress over perfection. ⏳ ✨

More from Bhagya Rana