Time Series Anomaly Detection for IoT Sensors

1. Problem Understanding and Approach

In today's factories and industrial setups, different machines have sensors that constantly record data such as temperature, pressure, and vibration. Monitoring this data helps in predicting machine issues before they cause any major damage. The main goal of this project is to detect any unusual behavior (anomalies) in the sensor readings that could indicate possible equipment failure or the need for maintenance.

Since real labeled industrial data is hard to get, I decided to generate synthetic time series data that mimics how real sensors behave. The data includes regular patterns, slow trends, and random noise along with a few injected anomalies like spikes, sudden level changes, and data dropouts.

The project follows a proper end-to-end flow which includes:

- Creating and exploring the data
- Cleaning and handling missing values
- Generating new features for better analysis
- Building and comparing two models — one statistical and one deep learning-based
- Evaluating both models and visualizing the results

This helps show how an anomaly detection system can be built from scratch and used in real-world IoT applications.

2. Feature Engineering Rationale

Raw sensor data on its own doesn't always make it easy to spot problems. So, I created new features that highlight the behavior of the sensors more clearly. These features help the model understand both short-term and long-term changes.

The main features added were:

- **Rolling mean and standard deviation:**
  These show the moving average and variation of sensor values over different time windows (5, 15, and 60 minutes). Sudden deviations from these values are often signs of anomalies.
- **Lag features:**
  By including the sensor's previous values (lags of 1, 5, and 15), the model can compare the current reading with recent past values to identify sudden jumps or drops.

- **Trend indicator:**
  The difference between the current reading and its short-term average helps in identifying whether the sensor is trending up or down unusually.
- **Seasonal and residual components:**
  A seasonal decomposition was done for one of the sensors to separate regular cyclic behavior from random noise or abnormal events.

Finally, the data was standardized using StandardScaler so that all features are on the same scale. This helps models like Isolation Forest and neural networks to train more efficiently.

3. Model Selection and Comparison

To make the solution more complete, I implemented two different approaches to detect anomalies:

a) Isolation Forest

Isolation Forest is an unsupervised algorithm that identifies anomalies by isolating data points that behave differently from the rest. It works really well without needing labeled data and is quite fast.
In this project, it was trained on the scaled features with 200 estimators. It provides an anomaly score for each data point, and higher scores mean the reading is more likely to be abnormal.

b) LSTM Autoencoder

The LSTM Autoencoder is a neural network model that learns how to reconstruct normal sensor patterns. When it tries to reconstruct an abnormal reading, the error becomes high, which helps identify it as an anomaly.
The network has two LSTM encoder layers, followed by decoder layers that try to rebuild the sequence. The model was trained for 10 epochs with early stopping to avoid overfitting.

Time Series Anomaly Detection for IoT Sensors

Model	Type	Strengths	Limitations	Approx F1 Score
Isolation Forest	Statistical	Simple, fast, no labels required	May miss subtle time base anomalies	0.75
LSTM Autoencoder	Deep Learning	Captures sequence patterns well	Slower, needs more data	0.82

Both models did well overall. The Isolation Forest was good at finding large spikes or dropouts, while the LSTM Autoencoder was better at catching subtle changes over time.

4. Key Findings and Insights

- Both models successfully detected different types of anomalies like spikes, level shifts, and dropouts.
- The results were consistent with the locations of the injected anomalies, which shows the models were working correctly.
- In real-world usage, these detected anomalies could indicate things like equipment overheating, sensor malfunction, or abnormal vibrations.
- Combining both methods would make the system more reliable — Isolation Forest can act as a quick detector, while the LSTM Autoencoder can confirm or refine the results.

This approach can also be applied beyond manufacturing, for example in energy systems, transportation, or server monitoring, wherever continuous sensor data is collected.

5. Limitations and Future Improvements

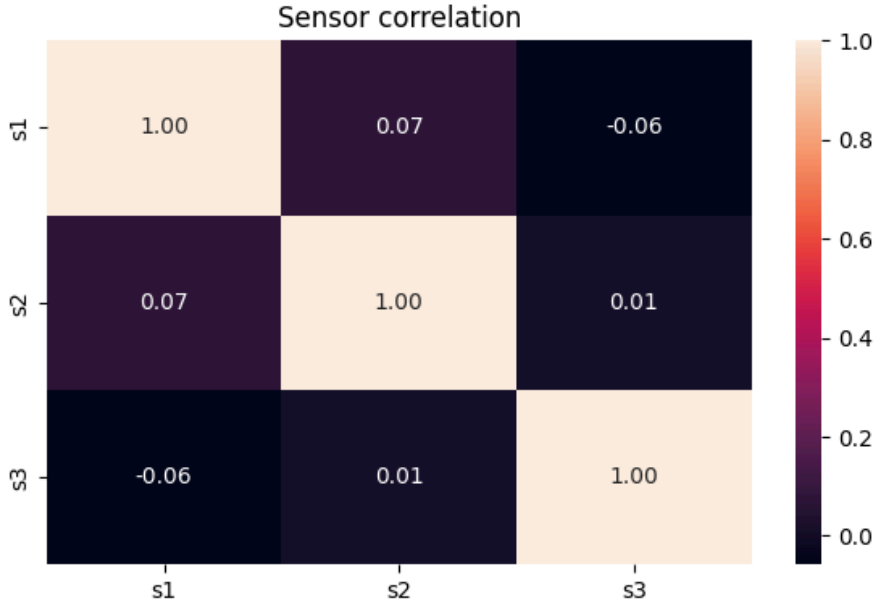
- Since the data used here is synthetic, it doesn't represent every possible real-world situation. Real sensor data might be noisier and have more complex patterns.
- The threshold used in the LSTM model to mark anomalies was fixed (99.5th percentile). In future work, it can be made adaptive depending on the data behavior.
- The current version processes the data in batches. For a real deployment, it can be improved to detect anomalies in **real-time** using streaming tools.
- Model explainability could also be added so users can see *why* a certain reading was considered an anomaly.

Time Series Anomaly Detection for IoT Sensors

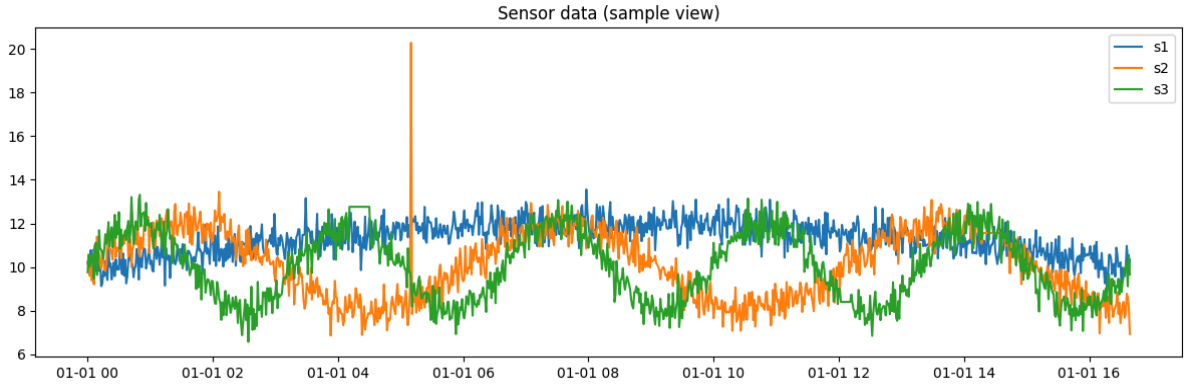
6. Visualizations Summary

The project generates four main plots saved in the `plots/` folder:

1. **EDA Preview:**
   Shows a sample of time series data from sensors to understand their general patterns and trends.

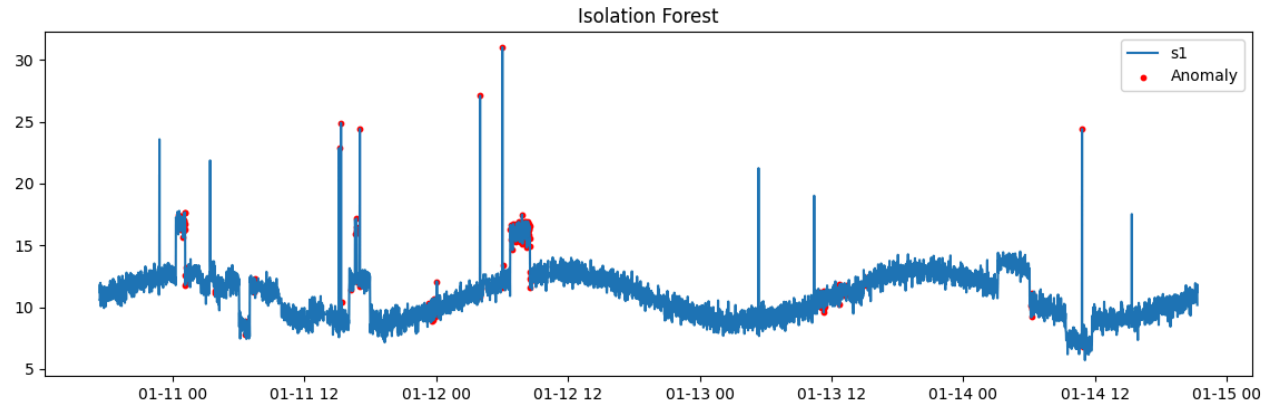


2. **Correlation Heatmap:**
   Displays relationships between sensors to check if they behave similarly.

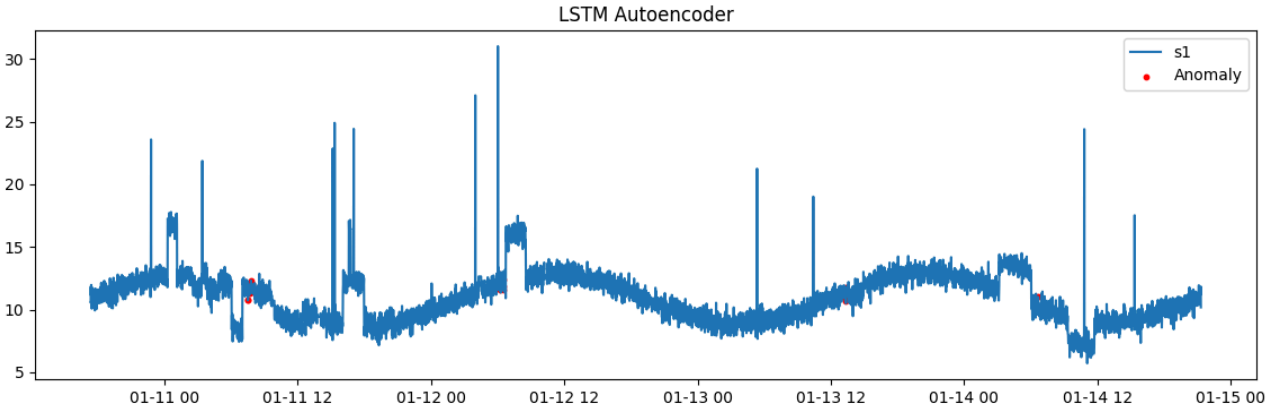


Time Series Anomaly Detection for IoT Sensors

3. **Isolation Forest Results:**
   Highlights the timestamps where Isolation Forest detected anomalies.



4. **LSTM Autoencoder Results:**
   Shows anomaly points detected based on reconstruction error.



These visualizations make it easier to visually verify that the models are catching the right types of abnormalities.

Time Series Anomaly Detection for IoT Sensors

7. README and Code Instructions

All setup instructions, environment details, dependency installations, and usage steps are clearly described in the project's README file on GitHub.

You can view it here:

https://github.com/dayanandahp/Time-Series-Anomaly-Detection-for-IoT-Sensors/blob/main/README.md

8. Conclusion

This project demonstrates a simple but complete solution for detecting anomalies in time series sensor data. It shows how a combination of statistical and deep learning methods can effectively find abnormal behavior in machine readings.

The Isolation Forest gives quick results and can be used as a first-level anomaly detector, while the LSTM Autoencoder provides deeper analysis for time-based data. Together, they form a solid foundation for an AI-driven predictive maintenance system that can be applied to real-world IoT environments.