RECURRING UNKNOWN ANOMALY DETECTION

USING MACHINE LEARNING

PROJECT REPORT

By

ARSHANA N

DATA ANALYST

UNDER THE GUIDANCE AND SUPERVISION OF

Miss THENNEL K A

Palakkad, Kerala, 678001

CONTENTS

INTRODUCTION

In today's data-driven environment, complex systems generate large volumes of time-series data through sensors and monitoring mechanisms. These data streams capture system behavior over time and are essential for ensuring reliability and performance. However, systems may occasionally exhibit abnormal behavior that deviates from expected patterns, known as **anomalies.**

Anomaly detection focuses on identifying such deviations, which may indicate system faults, performance degradation, or unexpected operational conditions. Traditional anomaly detection methods rely on predefined rules or fixed threshold values and are often ineffective in complex environments where abnormal patterns are unknown or continuously evolving.

In many real-world applications, anomalies are **unlabeled,** making supervised learning approaches unsuitable. **Unsupervised machine learning** techniques address this challenge by learning normal system behavior directly from data and detecting deviations without prior labeling, making them well suited for time-series telemetry data.

This project aims to detect recurring unknown anomalies **using machine learning** as a major project, while a minor project uses **Power BI** and **Streamlit dashboards** to visually present anomaly trends and recurring patterns, supporting effective analysis and decision-making.

ABSTRACT

Anomaly detection plays a crucial role in monitoring complex systems that generate large volumes of time-series data. In many real-world applications, anomalies are unknown and unlabeled, making traditional rule-based and supervised learning approaches ineffective. This project presents an unsupervised machine learning approach for detecting **recurring unknown anomalies** by learning normal system behavior and identifying deviations without prior labeling.

The proposed approach focuses on identifying anomaly patterns that repeatedly occur over time, as recurring anomalies often indicate underlying systemic issues. Machine learning techniques are applied to analyze high-dimensional data and generate anomaly scores that reflect abnormal behavior. To enhance interpretability and usability, the detection results are integrated into a Streamlit-based application and visualized through an interactive Power BI dashboard.

The dashboard provides insights into anomaly trends, frequency, and channel-wise behavior, enabling effective monitoring and data-driven decision-making. This combined analytical and visualization-based approach offers a scalable and interpretable solution for detecting and understanding recurring unknown anomalies in complex systems.

OBJECTIVE

The main objective of this project is to develop a machine learning-based solution for detecting **recurring unknown anomalies** in complex time-series data. The project aims to provide an automated, scalable, and interpretable system that identifies abnormal patterns without relying on prior labeling or predefined thresholds.

Primary Objectives:

1. Detect unknown anomalies in high-dimensional time-series telemetry data using **unsupervised machine learning techniques**.

2. Identify recurring anomaly patterns that repeat over time, indicating underlying systemic issues.

3. Implement the anomaly detection workflow in a **Streamlit application** for interactive visualization and analysis.

4. Visualize anomaly trends, frequency, and channel-wise behavior using **Power BI dashboards** for enhanced interpretability.

Secondary Objectives:

1. Improve the interpretability of unsupervised anomaly detection results for effective decision-making.

2. Reduce false positives by leveraging recurring pattern analysis.

3. Enable proactive monitoring and early detection of potential system failures.

PROBLEM STATEMENT

Modern complex systems generate vast amounts of time-series data through sensors and monitoring tools. While these data streams are crucial for system reliability and performance monitoring, they can also exhibit unexpected or abnormal behavior, known as **anomalies**. Detecting these anomalies is essential to prevent system failures, performance degradation, or unexpected operational events.

Traditional anomaly detection approaches, such as rule-based methods or supervised learning models, are limited in real-world scenarios because many anomalies are **unknown, unlabeled, or recurring over time**. Isolated or one-off anomaly detection may fail to identify patterns that repeat and indicate underlying systemic issues.

The primary challenge addressed in this project is to develop an **automated machine learning framework** capable of detecting **recurring unknown anomalies** in high-dimensional time-series data. The framework must identify abnormal patterns without prior labeling, quantify recurring anomalies, and provide **visual insights** to facilitate effective monitoring and decision-making.

By integrating **unsupervised machine learning techniques** with a **Streamlit application** and **Power BI dashboards**, the project provides a comprehensive solution that combines automated anomaly detection with interactive visualization, enabling both technical analysis and human interpretability.

DATASET DESCRIPTION

This project uses the **NASA Anomaly Detection Dataset (SMAP & MSL)**, a publicly available telemetry dataset hosted on Kaggle at

https://www.kaggle.com/datasets/patrickfleith/nasa-anomaly-detection-dataset-smap-msl.

The dataset contains **real spacecraft telemetry data collected from NASA missions**— specifically the Soil Moisture Active Passive (SMAP) satellite and the Mars Science Laboratory (MSL) rover — making it suitable for anomaly detection research.

The dataset includes time-series measurements from multiple telemetry channels with expert-provided anomaly labels listed in a separate file (`labeled_anomalies.csv`). These labels contain the start and end indices of known anomalous segments for each channel, allowing comparison with machine learning-detected anomalies.

Key Features of the Dataset

1. **Multi-channel time-series telemetry** from two NASA missions, capturing real operational behavior.

2. **NASA-labeled anomaly sequences** that serve as reference points for evaluating recurring anomaly detection. :

3. **High variability and noise**, reflecting real-world conditions in spacecraft telemetry.

4. A total of **82 telemetry channels** (55 from SMAP, 27 from MSL), with labeled anomaly sequences covering hundreds of anomalous segments.

5. The telemetry values for each channel are provided as `.npy` files, containing **time-series numerical data**. Each `.npy` file corresponds to a single telemetry channel. These files are loaded in the Streamlit application to compute features, detect anomalies, and visualize recurring anomaly patterns.

This dataset is used to analyze the performance of the unsupervised anomaly detection model implemented in this project, facilitating both quantitative comparison and visual interpretation of recurring anomalous behavior using Streamlit and Power BI dashboards.

EXPLORATORY DATA ANALYSIS (EDA)

Exploratory Data Analysis (EDA) is a crucial step in understanding the underlying patterns, trends, and characteristics of the telemetry dataset before applying machine learning models. In this project, EDA was performed on the NASA SMAP and MSL telemetry data to analyze sensor behavior and identify preliminary anomaly patterns.

Steps and Observations:

1. **Channel-wise Distribution**
   Each telemetry channel was analyzed individually to observe the distribution of sensor values over time. Visualizations such as line plots and histograms were used to understand value ranges, detect outliers, and assess the level of noise present in the data.

2. **Time-Series Behavior**
   Raw sensor signals were plotted to examine overall trends, fluctuations, and sudden spikes or drops in values. This analysis helped identify channels with unstable behavior and regions that were potentially prone to anomalous activity.

3. **Comparison with NASA-Labeled Anomalies**
   NASA-labeled anomaly segments were overlaid on the raw time-series plots to observe how known anomalies manifest in the telemetry data. This comparison provided an initial reference for later evaluating machine learning–detected anomalies.

4. **Feature Exploration**
   Derived features such as rolling mean, rolling standard deviation, z-score, and lag values were computed to understand how

statistical properties of the signal change over time. This step supported feature selection for the anomaly detection model.

5. **Preliminary Outlier Analysis**
   Initial outlier detection was performed using simple statistical measures such as z-scores. Channels and time segments showing significant deviations were noted for further processing during the machine learning phase.

Visualizations Included:

- Line plots of raw sensor values for selected telemetry channels
- Histograms showing the distribution of sensor readings
- Time-series plots with highlighted NASA-labeled anomaly segments

DATA ANALYSIS AND INTERPRETATION

The data analysis in this project provides critical insights into the behavior of telemetry signals, feature trends, and recurring anomaly patterns. These interpretations guided the design of the machine learning model and the visualization approach in both Streamlit and Power BI dashboards.

1. **Sensor Data Behavior**

- Raw telemetry signals exhibited varying patterns across channels, including steady signals, periodic fluctuations, and sudden spikes.
- Channels with high variability or frequent spikes were identified as more prone to anomalies.
- This observation validated the use of feature engineering (rolling mean, rolling standard deviation, z-score, and lag features) to highlight deviations in sensor behavior.

2. **Feature Trends**

- Derived features effectively captured abnormal deviations in sensor signals.
- High z-score values corresponded to sudden deviations, while rolling statistics smoothed minor fluctuations and highlighted sustained anomalies.
- These features formed the input to the Isolation Forest model for anomaly detection.

3. **Recurring Anomaly Patterns**

- Applying a rolling window over ML-detected anomalies enabled identification of recurring anomaly segments.

- Many recurring anomalies aligned with NASA-labeled segments, confirming the reliability of the ML model.
- Some ML-detected anomalies did not match NASA labels, indicating potential unknown or previously unrecorded recurring anomalies.

4. **Insights from Comparison with NASA Labels**

- Overlaying NASA-labeled anomalies on sensor signals provided a benchmark for evaluating model performance.
- The comparison showed the model successfully detected both isolated and recurring anomalies, demonstrating its effectiveness in unsupervised anomaly detection.

5. **Actionable Interpretation**

- Channels with high recurring anomaly rates should be closely monitored in real systems.
- Combining ML-detected anomalies with NASA-labeled segments provides a robust framework for anomaly tracking and visualization.
- These insights guided the development of Streamlit dashboards to display raw signals, recurring anomaly segments, and comparative views with NASA labels.

6. **Conclusion**

The data analysis and interpretation confirm that the feature engineering and ML approach effectively capture recurring unknown anomalies. These insights directly informed the modeling and visualization steps,

providing a practical and interpretable solution for real-world telemetry monitoring.

PREPROCESSING

Preprocessing is a crucial step in preparing the NASA telemetry data for machine learning–based anomaly detection. Raw telemetry data contains noise, fluctuations, and varying scales, which require feature engineering and cleaning before applying the Isolation Forest model.

1. **Data Cleaning**

- Missing or invalid values were removed from each telemetry channel.
- Shifted values at the start of the rolling window (NaNs from rolling calculations) were dropped.
- Each channel was ensured to contain a continuous, clean time-series suitable for ML processing.

2. **Feature Engineering**
   To enhance anomaly detection, additional features were derived from the raw sensor values:

- **Rolling Mean (rolling_mean)**
  Captures local trends in sensor behavior and smooths short-term fluctuations to highlight sustained anomalies.
- **Rolling Standard Deviation (rolling_std)**
  Measures local variability and helps identify regions with unusually high variation.
- **Z-Score (zscore)**
  Standardizes sensor values relative to local mean and standard

deviation. High absolute z-score indicates significant deviations from expected behavior.

- **Lag Feature (lag_1)**
  Stores the previous time-step value and helps the model understand temporal changes and sudden jumps.

3. **Feature Selection**

- The features value, rolling_mean, rolling_std, zscore, and lag_1 were selected as input to the Isolation Forest model.
- These features effectively capture both short-term deviations and long-term trends, essential for detecting recurring anomalies.

4. **Data Structuring**

- Each telemetry channel was converted into a DataFrame with the derived features.
- This structure ensured compatibility with the Streamlit application, enabling easy visualization of raw values, computed features, and detected anomalies.

5. **Outcome of Preprocessing**

- Cleaned and feature-enhanced datasets for each channel were ready for model training.
- Preprocessing ensured that the Isolation Forest could accurately detect anomalous patterns without interference from missing values, noise, or scale differences.

MODELING

In this project, the **Isolation Forest (IForest)** algorithm was used to detect anomalies in telemetry data. Isolation Forest is an unsupervised learning method suitable for time-series anomaly detection because it identifies unusual observations without requiring labeled data for training.

1. **Model Selection**

- Isolation Forest isolates anomalies based on the principle that anomalies are few and different.
- It is particularly effective for high-dimensional datasets, such as telemetry channels, where anomalous patterns may appear across multiple features simultaneously.

2. **Input Features**
   The model was trained on the following features derived during preprocessing:

- value (raw sensor reading)
- rolling_mean (captures local trends)
- rolling_std (captures local variability)
- zscore (standardized deviation)
- lag_1 (previous time-step value to capture temporal changes)

3. **Model Pipeline**

- A scikit-learn Pipeline was created with two steps:

1. **StandardScaler:** Scales all features to have zero mean and unit variance.
2. **IsolationForest:** Detects anomalies using the scaled features.

- **GridSearchCV** was applied for hyperparameter tuning, including:
  - Number of estimators (n_estimators)
  - Maximum samples (max_samples)
  - Contamination level (contamination)

4. **Recurring Anomaly Detection Logic**

- After detecting anomalies with Isolation Forest, a rolling window approach was applied to count consecutive anomalies.
- Recurring anomaly segments were defined where the number of anomalies exceeded a threshold within the rolling window.
- This allowed detection of sustained, recurring anomalous behavior rather than isolated spikes.

5. **Comparison with NASA Labels**

- Recurring anomaly segments were compared with NASA-labeled anomalies to validate the model.
- ML-detected anomalies that matched NASA segments confirmed the model's accuracy, while unmatched segments indicated potential unknown anomalies.

6. **Outcome**

- The Isolation Forest model, combined with feature engineering and recurring anomaly logic, successfully identified both isolated and recurring anomalies.
- Model outputs were visualized using Streamlit and Power BI dashboards, providing clear, interpretable insights for each telemetry channel.

HYPERPARAMETER TUNING

Hyperparameter tuning is a critical step in optimizing the performance of the Isolation Forest model for anomaly detection. In this project, tuning was performed using **GridSearchCV** to select the best combination of parameters for detecting anomalies in NASA telemetry data.

1. **Parameters Tuned**
   The following Isolation Forest parameters were considered:

- **n_estimators**: Number of trees in the forest
  - Values tested: 100, 200
  - More trees increase stability but also computation time
- **max_samples**: Number of samples drawn to train each tree
  - Values tested: 0.6, 0.8
  - Controls the size of the subset used for training each tree
- **contamination**: Expected proportion of anomalies in the data
  - Values tested: 0.01, 0.02, 0.05
  - Helps the model distinguish normal behavior from anomalies
- **max_features**: Number of features used for each tree
  - Value used: 1.0 (all features considered)

2. **Grid Search with Cross-Validation**

- GridSearchCV was applied with 3-fold cross-validation using the ROC AUC score as the evaluation metric.
- The pipeline included StandardScaler followed by Isolation Forest to ensure proper scaling of all features.

- The combination of parameters achieving the best ROC AUC score was selected as the final model configuration.

3. **Outcome**

- Hyperparameter tuning improved the model's ability to detect anomalies accurately while reducing false positives.
- The final tuned model was then used to identify recurring anomalies across telemetry channels and compare them with NASA-labeled segments.

MODEL VALIDATION AND RESULT ANALYSIS

The performance of the Isolation Forest model was validated by comparing ML-detected anomalies with NASA-labeled anomalies and analyzing recurring anomaly detection across telemetry channels.

1. **Comparison with NASA Labels**

- ML-detected anomalies were overlaid on raw sensor signals along with NASA-labeled segments.
- This comparison allowed a qualitative assessment of the model's ability to detect true anomalies.

Observations:

- Many ML anomalies matched NASA-labeled segments, confirming model reliability.
- Some ML anomalies did not overlap with NASA labels, indicating potential unknown anomalies.

2. **Recurring Anomaly Detection**

- A rolling-window approach was applied to ML anomalies to identify recurring anomalous segments.

Metrics:

- Total ML anomalies: Count of anomalies detected per channel.
- Recurring anomaly segments: Number of sustained anomaly sequences.
- Recurrence rate: Percentage of recurring anomalies relative to total observations.

- Recurring segments aligned closely with NASA-labeled segments, demonstrating that the model captures sustained anomalous behavior effectively.

3. **Observations**

- Channels with high variability showed more recurring anomalies, validating the importance of feature engineering (rolling statistics and z-score).
- The model successfully detected anomalies in both isolated spikes and long recurring segments, providing actionable insights for telemetry monitoring.

4. **Visualization of Results**

Streamlit Visualizations:

- Raw Sensor Signals: Line charts of sensor readings.
- ML-Detected Anomalies: Red points indicate isolated anomalies.
- Recurring Anomalies: Orange highlights indicate recurring segments.
- NASA-Labeled Anomalies: Cyan highlights for labeled anomalies.

Tables:

- Start and end indices of recurring anomalies.
- Duration of each anomaly segment.
- ML-only anomalies not labeled by NASA.

5. **Power BI Dashboard Methodology**

After generating ML-detected and recurring anomalies, the data was exported to Power BI for interactive dashboard visualization.

Design Approach:

- One-color theme (teal) with shades and opacity variations for all charts to maintain a professional and consistent look.
- Dark gradient background to reduce visual strain and make charts and KPIs stand out.

Charts and KPIs:

- Trend Charts: Line charts displaying raw sensor values, ML anomalies, and recurring anomaly segments.
- Comparison Charts: Column charts comparing ML-detected anomalies with NASA-labeled anomalies.
- KPI Cards: Total anomalies, recurrence rate, total NASA-labeled anomalies, and novelty counts.

Interactivity:

- Filters for selecting channels, adjusting rolling windows, and setting anomaly thresholds dynamically.
- Hover-over tooltips to display detailed values for anomalies and recurring segments.

Outcome:

- Enabled clear, interpretable insights for each telemetry channel.
- Complemented Streamlit visualizations with polished, interactive dashboards suitable for reporting and monitoring.

6. **Conclusion**

- Model validation confirms that Isolation Forest, combined with rolling-window logic, effectively identifies both isolated and recurring anomalies in NASA telemetry data.
- ML results are consistent with NASA labels while also highlighting previously undetected anomalies, demonstrating the model's practical utility.
- Streamlit and Power BI dashboards provided complementary visualizations for quick analysis and reporting.

FEATURE IMPORTANCE

In unsupervised anomaly detection using Isolation Forest, traditional feature importance metrics are not directly available. However, understanding the contribution of each feature to the detection of anomalies helps interpret the model results and guide future analysis.

1. **Features Used**
   The following features were used as input to the Isolation Forest model:

- value – Raw sensor readings
- rolling_mean – Captures local trends
- rolling_std – Measures local variability
- zscore – Standardized deviation from the local mean
- lag_1 – Previous time-step value to capture temporal changes

2. **Contribution of Features**

- **value**: Provides the baseline sensor measurement; essential for identifying deviations.
- **rolling_mean** and **rolling_std**: Highlight sustained anomalies rather than isolated spikes, making them critical for recurring anomaly detection.
- **zscore**: Standardizes deviations, allowing the model to detect extreme variations consistently across channels.
- **lag_1**: Helps capture abrupt changes and sudden jumps in sensor readings.

3. **Interpretation**

- Analysis of anomaly detection results showed that rolling statistics (rolling_mean and rolling_std) and z-score were the most effective features in highlighting recurring anomalies.
- The combination of these features enabled the model to detect both isolated spikes and sustained anomalous segments, providing interpretable outputs for visualization in Streamlit and Power BI dashboards.

4. **Outcome**

- Although Isolation Forest does not output feature importance scores, examining the behavior of the features in detected anomalies provided a qualitative understanding of their contribution.
- This guided the design of dashboards and helped focus attention on features most indicative of recurring anomalies.

CONCLUSION

This project successfully demonstrates the detection of recurring unknown anomalies in NASA telemetry data using machine learning. By applying the **Isolation Forest** algorithm along with feature engineering and rolling-window logic, the model was able to:

- Detect both isolated and recurring anomalies across multiple telemetry channels.
- Identify anomalies that align with NASA-labeled segments, validating the model’s accuracy.
- Highlight potential unknown recurring anomalies not previously labeled, providing actionable insights for monitoring spacecraft systems.
- Present results clearly through Streamlit and Power BI dashboards, enabling visual interpretation of recurring anomalies and anomaly trends.

The combination of ML modeling, preprocessing, and visualization offers a robust and interpretable solution for real-world telemetry anomaly detection.

SUGGESTIONS

For future work and improvements:

1. **Extend to Other Datasets:** Apply the framework to additional spacecraft or industrial time-series telemetry to test generalizability.
2. **Enhance Feature Engineering:** Include additional temporal or domain-specific features to improve detection of subtle anomalies.
3. **Real-Time Detection:** Integrate the model into a streaming system for live anomaly detection.
4. **User-Friendly Dashboards:** Add interactive controls in Streamlit or Power BI to allow selection of channels, rolling windows, and thresholds dynamically.
5. **Evaluation Metrics:** Implement more quantitative metrics (precision, recall, F1-score) to compare ML-detected anomalies with labeled data in future iterations.

APPENDIX - I

	chan_id	spacecraft	anomaly_sequences	class	num_values
0	P-1	SMAP	[[2149, 2349], [4536, 4844], [3539, 3779]]	[contextual, contextual, contextual]	8505
1	S-1	SMAP	[[5300, 5747]]	[point]	7331
2	E-1	SMAP	[[5000, 5030], [5610, 6086]]	[contextual, contextual]	8516
3	E-2	SMAP	[[5598, 6995]]	[point]	8532
4	E-3	SMAP	[[5094, 8306]]	[point]	8307
5	E-4	SMAP	[[5450, 8261]]	[point]	8354
6	E-5	SMAP	[[5600, 5920]]	[point]	8294
7	E-6	SMAP	[[5610, 5675]]	[point]	8300
8	E-7	SMAP	[[5394, 5674]]	[point]	8310
9	E-8	SMAP	[[5400, 6022]]	[point]	8532

REFERENCES

NASA Anomaly Detection Dataset (SMAP & MSL), Kaggle.

https://www.kaggle.com/datasets/patrickfleith/nasa-anomaly-detection-dataset-smap-msl

APPENDIX -II

Importing Required Python Libraries

```
import os
import ast
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
```

Loading Labeled Anomaly Metadata

```
DATASET_PATH = r"C:\Users\arsha\Downloads\archive (2)"
LABELED_FILE = os.path.join(DATASET_PATH, "labeled_anomalies.csv")
labeled_df = pd.read_csv(LABELED_FILE)
labeled_df["anomaly_sequences"] = labeled_df["anomaly_sequences"].apply(
   lambda x: ast.literal_eval(x) if isinstance(x, str) else x
)
labeled_df.head()
```

Loading Telemetry Sensor Data

```
CHANNEL_ID = "P-1"    # Select the channel
channel_file = os.path.join(
   DATASET_PATH,
```

```
    "data", "data", "test",
    f"{CHANNEL_ID}.npy"
)
values = np.load(channel_file)
values_1d = values[:, 0]
channel_data = pd.DataFrame({
    "value": values_1d
})
channel_data.head()
```

Raw Sensor Signal Visualization

```
plt.figure(figsize=(15,4))
plt.plot(channel_data["value"])
plt.title(f"Raw Sensor Signal – Channel {CHANNEL_ID}")
plt.xlabel("Time Index")
plt.ylabel("Sensor Value")
plt.show()
```

Feature Engineering

```
ROLLING_WINDOW = 50
channel_data["rolling_mean"] =
channel_data["value"].rolling(ROLLING_WINDOW).mean()
channel_data["rolling_std"]  = channel_data["value"].rolling(ROLLING_WINDOW).std()
channel_data["zscore"] = (
    channel_data["value"] - channel_data["rolling_mean"]
) / channel_data["rolling_std"]
channel_data["lag_1"] = channel_data["value"].shift(1)
```

```
channel_data.dropna(inplace=True)
channel_data.head()
```

Rolling Context Feature Visualization

```
plt.figure(figsize=(15,4))
plt.plot(channel_data["value"], alpha=0.6, label="Value")
plt.plot(channel_data["rolling_mean"], label="Rolling Mean")
plt.fill_between(
    channel_data.index,
    channel_data["rolling_mean"] - channel_data["rolling_std"],
    channel_data["rolling_mean"] + channel_data["rolling_std"],
    alpha=0.2,
    label="± Std"
)
plt.title("Rolling Context Features")
plt.xlabel("Time Index")
plt.ylabel("Value")
plt.legend()
plt.show()
```

Feature Matrix Construction

```
FEATURES = ["value", "rolling_mean", "rolling_std", "zscore", "lag_1"]
X = channel_data[FEATURES]
```

Isolation Forest Model Pipeline

```
pipeline = Pipeline([
```

```
    ("scaler", StandardScaler()),
    ("iforest", IsolationForest(random_state=42))
])
```

Hyperparameter Tuning using GridSearchCV

```
param_grid = {
    "iforest__n_estimators": [100, 200],
    "iforest__max_samples": [0.6, 0.8],
    "iforest__contamination": [0.01, 0.02, 0.05],
    "iforest__max_features": [1.0]
}
grid = GridSearchCV(
    pipeline,
    param_grid,
    cv=3,
    scoring="roc_auc",
    n_jobs=-1
)
grid.fit(X)
print("Best Parameters:", grid.best_params_)
```

Training Final Model and Detecting Anomalies

```
best_model = grid.best_estimator_
channel_data["anomaly_ml"] = best_model.predict(X)
channel_data["anomaly_ml"] = channel_data["anomaly_ml"].map({1: 0, -1: 1})
channel_data["anomaly_ml"].value_counts()
```

Visualization of ML-Detected Point Anomalies

```
plt.figure(figsize=(15,4))
plt.plot(channel_data["value"], alpha=0.6)
plt.scatter(
  channel_data.index[channel_data["anomaly_ml"] == 1],
  channel_data["value"][channel_data["anomaly_ml"] == 1],
  color="red",
  s=10
)
plt.title("Isolation Forest – Detected Point Anomalies")
plt.xlabel("Time Index")
plt.ylabel("Value")
plt.show()
```

Recurring Anomaly Detection Logic

```
RECURRING_WINDOW = 200
RECURRING_THRESHOLD = 10
channel_data["recurring_count"] = (
  channel_data["anomaly_ml"]
  .rolling(RECURRING_WINDOW)
  .sum()
)
channel_data["recurring_flag"] = (
  channel_data["recurring_count"] > RECURRING_THRESHOLD
)
```

Extraction of Recurring Anomaly Segments

```
segments = []
start = None
for i in range(len(channel_data)):
    if channel_data["recurring_flag"].iloc[i] and start is None:
        start = i
    elif not channel_data["recurring_flag"].iloc[i] and start is not None:
        segments.append((start, i - 1))
        start = None
if start is not None:
    segments.append((start, len(channel_data) - 1))
segments
```

Visualization of Recurring Anomalies

```
plt.figure(figsize=(15,5))
plt.plot(channel_data["value"], label="Sensor Value", alpha=0.7)
mask = channel_data["anomaly_ml"].astype(bool)
plt.scatter(
    channel_data.index[mask],
    channel_data["value"][mask],
    color="red",
    s=10,
    label="ML Anomalies"
)
for i, (start, end) in enumerate(segments):
    if i == 0:
        plt.axvspan(start, end, color="orange", alpha=0.3, label="Recurring Anomaly")
    else:
```

```
        plt.axvspan(start, end, color="orange", alpha=0.3)
plt.title(f"Recurring Anomaly Detection – Channel {CHANNEL_ID}")
plt.xlabel("Time Index")
plt.ylabel("Value")
plt.legend()
plt.show()
```

Validation with NASA-Labeled Anomalies

```
channel_info = labeled_df[labeled_df["chan_id"] == CHANNEL_ID].iloc[0]
labeled_sequences = channel_info["anomaly_sequences"]
plt.figure(figsize=(15,5))
plt.plot(channel_data["value"], label="Sensor Value", alpha=0.7)
plt.scatter(
    channel_data.index[mask],
    channel_data["value"][mask],
    color="red",
    s=10,
    label="ML Detected Anomalies"
)
for i, (start, end) in enumerate(segments):
    if i == 0:
        plt.axvspan(start, end, color="orange", alpha=0.3, label="Detected Recurring")
    else:
        plt.axvspan(start, end, color="orange", alpha=0.3)
for i, (start, end) in enumerate(labeled_sequences):
    if i == 0:
        plt.axvspan(start, end, color="blue", alpha=0.25, label="NASA Labeled")
```

```
    else:
        plt.axvspan(start, end, color="blue", alpha=0.25)
plt.title(f"Labeled vs Detected Anomalies – Channel {CHANNEL_ID}")
plt.xlabel("Time Index")
plt.ylabel("Sensor Value")
plt.legend()
plt.show()
```

FIGURE 1: Streamlit

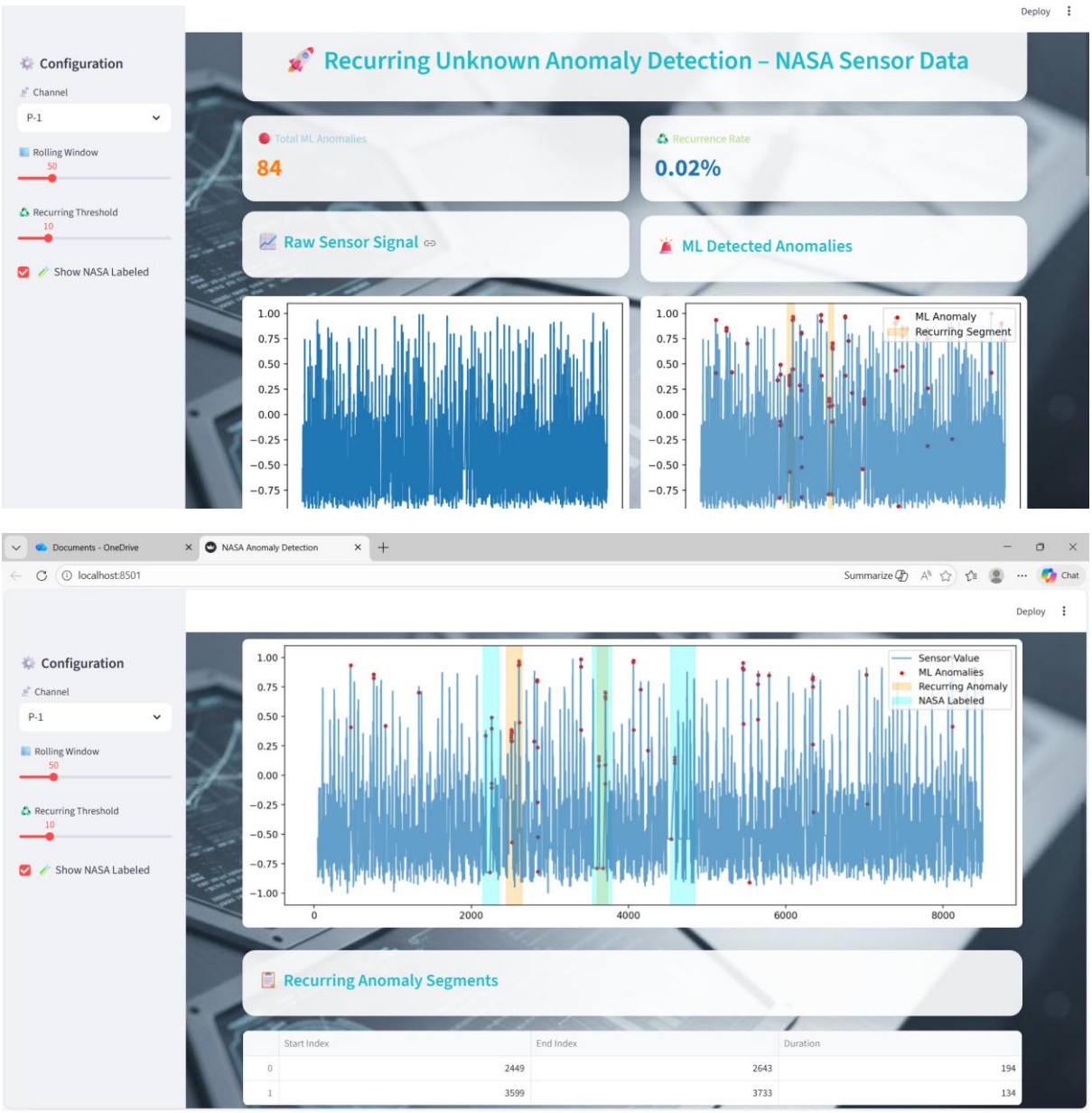

FIGURE 2: Power Bi Dashboard