UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Enginyeria de Sistemes TIC

Anomaly Detection

Network solutions for IoT

Albert Marquillas

September 6, 2025

Contents

1 Objectives

This practical activity focuses on unsupervised learning techniques for anomaly detection. Students will work with time series data from industrial systems, exploring how to identify unusual patterns or outliers without labeled examples.

We will use methods such as Isolation Forest and DBSCAN to detect anomalies in sensor data.

2 Dataset Description

The dataset is from the Numenta Anomaly Benchmark (NAB):

Source: `https://www.kaggle.com/datasets/boltzmannbrain/nab/data`

The NAB dataset includes real-world time series with labeled anomalies across multiple domains. For this activity, you will choose a single time series (for example, `realKnownCause/ambient_temperature_system_failure.csv`) to keep the analysis tractable.

2.1 Example columns

- `timestamp`: time of the measurement.
- `value`: sensor reading.

3 Introduction to Anomaly Detection

3.1 What is Anomaly Detection?

Anomaly detection refers to identifying data points that deviate significantly from the expected behavior. In industrial settings, anomalies can indicate system failures, unusual operating conditions, or sensor faults. Detecting these early can enable predictive maintenance, reduce downtime, and improve safety.

3.2 Unsupervised Anomaly Detection

Unlike supervised classification, unsupervised methods don't require labeled anomalies in the training data. Instead, they learn patterns from the normal data distribution and flag points that appear "different" or rare.

This is especially important in industrial contexts where failures are rare and labeled data is scarce.

4 Algorithms Used in This Activity

4.1 Isolation Forest

Isolation Forest is an ensemble algorithm designed for anomaly detection. It works by randomly partitioning the data space and isolating observations. Anomalies are isolated faster (with fewer splits) because they are few and different. This approach is efficient and scales well to high-dimensional data.

4.2 DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

DBSCAN is a clustering algorithm that groups densely packed points and labels sparse regions as outliers. For time series anomaly detection, we can use DBSCAN on derived features (such as rolling averages or deltas) to find regions where the behavior is unusual. It doesn't require specifying the number of clusters in advance and can identify clusters of arbitrary shape.

5 Activity Structure

5.1 Data Exploration

- Load one of the time series CSV files.
- Plot the time series to visually inspect for potential anomalies.

Example code snippet:

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('ambient_temperature_system_failure.csv')
print(df.head())

df['timestamp'] = pd.to_datetime(df['timestamp'])
plt.figure(figsize=(12,6))
plt.plot(df['timestamp'], df['value'])
plt.title('Sensor Reading Over Time')
plt.xlabel('Time')
plt.ylabel('Value')
plt.show()
```

5.2 Feature Engineering

- Compute derived features such as rolling mean, rolling standard deviation, or lag differences.
- Add these features to the dataframe for anomaly detection.

Example code snippet:

```
df['rolling_mean'] = df['value'].rolling(window=10).mean()
df['rolling_std'] = df['value'].rolling(window=10).std()
df['diff'] = df['value'].diff()
df = df.dropna()
```

5.3 Anomaly Detection with Isolation Forest

- Select features (e.g., value, rolling mean, rolling std, diff).
- Fit an Isolation Forest model.
- Predict anomaly scores and label anomalies.
- Plot the results.

Example code snippet:

```
from sklearn.ensemble import IsolationForest

features = df[['value', 'rolling_mean', 'rolling_std', 'diff']]
model = IsolationForest(contamination=0.01)
model.fit(features)
df['anomaly'] = model.predict(features)

plt.figure(figsize=(12,6))
plt.plot(df['timestamp'], df['value'], label='Value')
plt.scatter(df['timestamp'][df['anomaly'] == -1],
            df['value'][df['anomaly'] == -1],
            color='red', label='Anomaly')
plt.legend()
plt.title('Isolation Forest Anomaly Detection')
plt.show()
```

5.4 Anomaly Detection with DBSCAN

- Scale the selected features.
- Fit DBSCAN to detect dense clusters and label noise points as anomalies.
- Visualize the detected anomalies.

Example code snippet:

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN

scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

dbscan = DBSCAN(eps=0.5, min_samples=5)
df['dbscan_label'] = dbscan.fit_predict(scaled_features)

plt.figure(figsize=(12,6))
plt.plot(df['timestamp'], df['value'], label='Value')
plt.scatter(df['timestamp'][df['dbscan_label'] == -1],
            df['value'][df['dbscan_label'] == -1],
            color='orange', label='DBSCAN␣Anomaly')
plt.legend()
plt.title('DBSCAN␣Anomaly␣Detection')
plt.show()
```

5.5 Comparison and Discussion

- Compare the anomalies detected by Isolation Forest and DBSCAN.
- Discuss the strengths and limitations of each method.
- Reflect on practical industrial applications of these techniques.

6 Optional Extensions (if time allows)

- Try other time series from the NAB dataset.
- Tune hyperparameters (contamination for Isolation Forest, eps/min_samples for DBSCAN).
- Experiment with other features (e.g., FFT components, seasonal decomposition).
- Evaluate detection with ground-truth anomaly labels (provided in NAB).

7 Conclusion and Questions

- What are the main challenges of anomaly detection in time series data?
- How does unsupervised learning help in industrial contexts with limited labels?
- How would you improve the detection in a real deployment?