Aprendizagem Aplicada à Segurança

Unsupervised Anomaly Detection

Mário Antunes

October 17, 2025

Universidade de Aveiro

Table of Contents i

What is Anomaly Detection?

Anomaly Detection (or Outlier Detection) is the task of identifying data points or events that are rare and deviate significantly from the “normal” majority of the data.

These “anomalies” or “outliers” can represent:

- Fraudulent transactions
- A failing sensor on a piece of equipment
- A network intrusion
- A new, emerging trend
- Errors in data entry

Types of Anomalies

1. **Point Anomalies:** A single data point that is far from the rest of the data (e.g., a credit card transaction for \$10,000 when all others are < \$100).
2. **Contextual Anomalies:** A data point that is normal in a global sense but abnormal in its specific context (e.g., buying a winter coat in July).
3. **Collective Anomalies:** A *group* of data points that are not anomalous individually, but their *collection* as a whole is (e.g., a “heart flutter” in an EKG, which is a *sequence* of unusual-but-not-impossible heartbeats).

The Unsupervised Challenge

In many real-world problems, we **do not have labels** for what is an anomaly. We often have a large dataset of what we *assume* is “normal.”

The Unsupervised Strategy: The core idea is to **build a model of “normalcy.”**

1. Train a model on the entire dataset, assuming most of it is normal.
2. The model learns the underlying patterns, structures, and densities of the normal data.
3. Anomalies are points that **do not fit this learned model** of “normalcy.”

We will explore several methods to build this model.

The Problem: Categorical Data i

The models we will discuss (K-Means, GMM, PCA, Autoencoders) are mathematical. They operate on **numerical vectors.**

They **cannot** directly process categorical text features (e.g., `protocol: "TCP"`, `user: "admin"`).

Why?

- **Distance Models (K-Means):** How do you calculate the distance between “TCP” and “UDP”?
- **Probabilistic Models (GMM):** How do you fit a Gaussian (bell curve) to text labels?

The Problem: Categorical Data ii

- **Gradient Models (Autoencoders):** You cannot multiply an input like “admin” by a weight during training (backpropagation).

We **must** convert these categories into numbers first. This is **Feature Encoding**.

Method 1: Label Encoding i

This method assigns a unique integer to each unique category.

Example: A feature `protocol` with values `[TCP, UDP, HTTP, TCP]`

...would be encoded as: `[0, 1, 2, 0]`

- `TCP = 0`
- `UDP = 1`
- `HTTP = 2`

Method 1: Label Encoding ii

The Problem: This creates a **false ordinal relationship**. The model now thinks `HTTP (2)` is “larger” than `UDP (1)` and that the “distance” between `TCP (0)` and `HTTP (2)` is twice that of `TCP (0)` and `UDP (1)`. This is usually wrong and will confuse distance-based models.

Method 2: One-Hot Encoding (OHE) i

This method creates a new **binary (0/1)** feature for *each* unique category.

Example: `[TCP, UDP, HTTP]`

...becomes three new columns:

	`is_TCP`	`is_UDP`	`is_HTTP`
TCP	1	0	0
UDP	0	1	0
HTTP	0	0	1

Method 2: One-Hot Encoding (OHE) ii

The Benefit: No false ordinal relationship. All categories are now geometrically equidistant. This is the **preferred method** for most anomaly detection models.

The Drawback: If a feature has 10,000 unique categories (e.g., `user_id`), OHE will create 10,000 new columns, which can be computationally expensive (Curse of Dimensionality).

Encoding Comparison i

Method	How it Works	Best For	Cons
Label Encoding	Maps N categories to integers `[0, 1 … N-1]`.	**Ordinal data** (e.g., `low`, `medium`, `high`). Tree-based models (like Isolation Forest).	Creates a false ordinal relationship. **Bad for distance/gradient models.**

Encoding Comparison ii

One-Hot Encoding	Maps N categories to N binary (0/1) features.	**Nominal data** (no order). Distance, density, and gradient models (K-Means, GMM, PCA, AE).	**Curse of Dimensionality**: Can create thousands of features.

Method 1: K-Means Clustering

The Algorithm:

- K-Means is an algorithm that partitions data into K distinct, non-overlapping clusters.
- It is an iterative algorithm that minimizes **inertia**, which is the sum of squared distances from each point to its assigned cluster center (centroid).

How it Works (EM-like): 1. **Initialization:** Randomly place K centroids. 2. **Expectation (Assign Step):** Assign each data point to its *nearest* centroid. 3. **Maximization (Update Step):** Recalculate each centroid as the **mean** of all points assigned to it. 4. **Repeat** steps 2 and 3 until the centroids no longer move significantly.

K-Means for Anomaly Detection

The Core Idea: Normal points will be close to other, similar points and thus will form dense clusters. Anomalies will be “lone wolves,” far away from any cluster center.

The Anomaly Score:

1. Train K-Means on the data to find the K cluster centroids.
2. For any new data point, its **anomaly score** is its **Euclidean distance (L_2 norm) to its closest centroid.**
    - $Score(x) = \min_{j \in \{1 \dots K\}} \|x - \mu_j\|^2$
3. **Interpretation:**
    - **Low Score:** The point is close to a known cluster. It is **normal**.
    - **High Score:** The point is far from *all* known clusters. It is an **anomaly**.
4. A **threshold** is set on this score to make a classification.

Method 2: Gaussian Mixture Models (GMM)

The Algorithm:

- GMM is a **probabilistic** clustering method. It's more flexible than K-Means.
- It assumes the data is generated from a "mixture" of several **Gaussian distributions** (bell curves).
- Unlike K-Means (hard assignment), GMM provides a **"soft" assignment**, giving the *probability* that a point belongs to each cluster.

How it Works:

- It uses an algorithm called **Expectation-Maximization (EM)** to find the parameters (mean μ, covariance Σ) of each Gaussian.
- **Expectation:** Calculates the probability (responsibility) that each cluster k has for generating each point i.
- **Maximization:** Updates the μ_k and Σ_k parameters for each cluster based on the weighted responsibilities from the E-step.

GMM for Anomaly Detection

The Core Idea: The GMM learns a “density” function for the data. Normal points will have a high probability of being generated by this model. Anomalies will have a very low probability.

The Anomaly Score:

1. Train the GMM on the data.
2. For a new data point, its **anomaly score** is the **negative log-likelihood** of the point under the model.
    - $Score(x) = -\log(P(x))$

3. **Interpretation:**
   - **Low Score:** The model says $P(x)$ is high (e.g., 0.9). The point is very likely and **normal**.
   - **High Score:** The model says $P(x)$ is very low (e.g., $1e^{-50}$). The log-likelihood is a large negative number, so the *negative* log-likelihood is a large positive number. The point is highly unlikely and an **anomaly**.

Method 3: Principal Component Analysis (PCA)

The Algorithm:

- PCA is a **linear dimensionality reduction** technique.
- It finds a new set of orthogonal (perpendicular) axes, called **Principal Components**, that align with the directions of maximum **variance** in the data.
- The first component (PC1) captures the most variance, PC2 captures the next most, and so on.
- It's a "Blind Signal Separation" method: it separates the "signal" (the main components) from the "noise."

How it Works:

- It finds the **eigenvectors** and **eigenvalues** of the data’s covariance matrix.
- The eigenvectors are the Principal Components.
- The eigenvalues tell you how much variance each component explains.
- You can “compress” data by projecting it onto the first k components.

PCA for Anomaly Detection

The Core Idea: Anomalies, by definition, do not follow the same “normal” patterns as the rest of the data. The principal components (which model “normal” variance) will not be able to represent anomalies well.

This is a **Reconstruction Error** method.

The Anomaly Score:

1. Train PCA on *normal* data, keeping the top k components that explain (e.g.) 95% of the variance.
2. For a new data point x:
   a. **Project:** Transform x into the low-dimensional k-space (x_{proj}).
   b. **Reconstruct:** Transform x_{proj} *back* into the original high-dimensional space (x_{recon}).
   c. The **anomaly score** is the **reconstruction error**: $\|x - x_{recon}\|^2$
3. **Interpretation:**
   - **Low Score:** x was reconstructed well. It fits the normal patterns. **Normal**.
   - **High Score:** x was reconstructed poorly. It deviates from the normal patterns. **Anomaly**.

Method 4: Autoencoders (AE)

The Algorithm:

- An Autoencoder is an unsupervised neural network. It's conceptually a **non-linear** version of PCA.
- It's trained to learn an **identity function**: the output should be as close to the input as possible ($X' \approx X$).
- It has two parts:
    1. **Encoder:** A network that compresses the high-dimensional input X into a low-dimensional **latent space** or "bottleneck" z.
    2. **Decoder:** A network that tries to reconstruct the original X' from the compressed representation z.

How it Works:

- The network is forced to learn a compressed representation (a “code”) of the data.
- To do this successfully, it *must* learn the most important, underlying patterns and correlations in the data.

Autoencoders for Anomaly Detection

The Core Idea: This is also a **Reconstruction Error** method, just like PCA.

1. Train the Autoencoder **only on normal data**.
2. The model becomes an “expert” at compressing and decompressing *normal* data points. It learns the “rules” of normalcy.
3. When the model is given an **anomaly**, it will fail to reconstruct it, because the anomaly doesn’t follow the “rules” the model learned.

The Anomaly Score:

1. For a new data point x, feed it through the trained model to get the reconstruction x'.
2. The **anomaly score** is the **reconstruction error**, typically Mean Squared Error (MSE):
   - $Score(x) = \|x - x'\|^2$
3. **Interpretation:**
   - **Low Score:** The model reconstructed x perfectly. **Normal**.
   - **High Score:** The model's reconstruction x' is very different from x. **Anomaly**.

Method 5: Isolation Forest

The Core Idea: This method works on a completely different principle:

- Anomalies are **“few and different.”**
- Because they are different, they are **easier to isolate** from the rest of the data.

The Algorithm:

1. Build an “ensemble” (a forest) of many random trees.
2. To build each tree, “isolate” points by randomly selecting a feature and a random split point.
3. Repeat this process until every point is in its own leaf node.

The Anomaly Score:

- **Normal points** are in dense regions. It takes *many* random splits to isolate them, so they have a **long path length** in the tree.
- **Anomalies** are “out on their own.” It takes *very few* random splits to isolate them, so they have a **short path length**.
- The **anomaly score** is based on the *average path length* for a point across all trees in the forest.

Summary of Unsupervised Methods

Method	Core Idea	Anomaly Score
K-Means	Clustering	Distance to nearest centroid.
GMM	Probabilistic Density	Negative log-likelihood ($-\log(P(x))$).
PCA	Linear Reconstruction	Reconstruction error $\|x - x_{recon}\|^2$.
Autoencoder	Non-Linear Reconstruction	Reconstruction error $\|x - x'\|^2$.
Isolation Forest	Ease of Isolation	Average path length in a random tree.