Anomaly Detection
Lab 1 - Introduction

Andrei Hîji

October 2025

1 Environment Setup

Python will be used in our labs. It is recommended to use a virtual environment (venv).

Software packages:

- scikit-learn - install with pip
- pyod - install with pip
- keras, tensorflow - in next labs

When it comes to anomaly detection there is a subtle difference between the models from the different libraries. Be careful, models from **scikil-learn** predict -1 for outliers and 1 for inliers, while the ones from pyod use 1 for outliers and 0 for inliers.

Datasets that will be used **(no need to download them now)** - Outlier Detection DataSets - ODDS[1].

2 Outlier detection vs. novelty detection

Outlier detection[2] assumes that training data consists of both normal and anomalous events. Novelty detection is a semi-supervised approach that only permits normal samples in the training data. Some models from both **scikit-learn** and **pyod** allow to specify the wanted setup with the **novelty** parameter.

Some metrics that we will use:

- $\text{TPR} = \frac{\text{TP}}{\text{TP+FN}}$ (sensitivity, recall)
- $\text{TNR} = \frac{\text{TN}}{\text{TN+FP}}$ (specificity)
- $\text{BA} = \frac{\text{TPR+TNR}}{2}$ (balanced accuracy)

[1] `https://odds.cs.stonybrook.edu/`

[2] `https://scikit-learn.org/1.5/modules/outlier_detection.html`

- FPR = $\frac{\text{FP}}{\text{FP+TN}}$
- AUC (Area under the ROC Curve)
- ROC curve is drawn by computing TPR and FPR at every possible threshold

3 Exercises

3.1 Ex. 1

Use the **generate_data()** function from **pyod.utils.data** to generate a 2-dimensional dataset with 500 normal samples (400 training samples and 100 test samples) with a contamination rate of **0.1**.

Use **pyplot.scatter()** function to plot the training samples, choosing a different color for the outliers.

3.2 Ex. 2

Choose a model from **pyod** (ex: KNN), leave all the parameters default (except the contamination, which will match the contamination used for dataset generation) and fit it with the training data. Get the predictions of the model for both the training and the testing data.

Use the **confusion_matrix** function from **sklearn.metrics** to find the number of TN, TP, FN and FP and compute the balanced accuracy.

Use the **roc_curve()** function from sklearn.metrics to compute the ROC curve and then plot it with **plot()** function.

Change the contamination rate used by the model and see how the reported metrics change.

3.3 Ex. 3

Generate a unidimensional dataset with 10 % contamination rate, 1000 training samples and no testing samples using **generate_data()**. Try to detect the anomalies in the dataset by using the Z-scores. In order to do that you should compute the Z-score threshold that would classify the given percent (contamination rate) of data as anomalies (use **np.quantile()** function). Compute the balanced accuracy of the designed method.

3.4 Ex. 4

Same as Ex. 3 but for a multidimensional dataset. Choose your own mean vector μ and covariance matrix Σ and build your dataset by hand starting with the samples generated from a multivariate **standard** normal distribution, $x \sim \mathcal{N}(0, I)$. Using the Cholesky decomposition of the covariance matrix $\Sigma = LL^T$ draw your samples y like this: $y = Lx + \mu$ and then compute the Z-scores. All other tasks as in Ex. 3.