Coursework on Anomaly Detection

In this coursework, you will work with data from the Tennessee Eastman process. The Tennessee Eastman process is a virtual plant that Downs and Vogel designed based on an actual plant of the Eastman Chemical Company[1]. In this virtual plant, a series of reactants, A, C, D, and E are processed to obtain two products, G and H, as pure as possible, with an unavoidable inert component mixed in the feed, B, and a by-product, F.

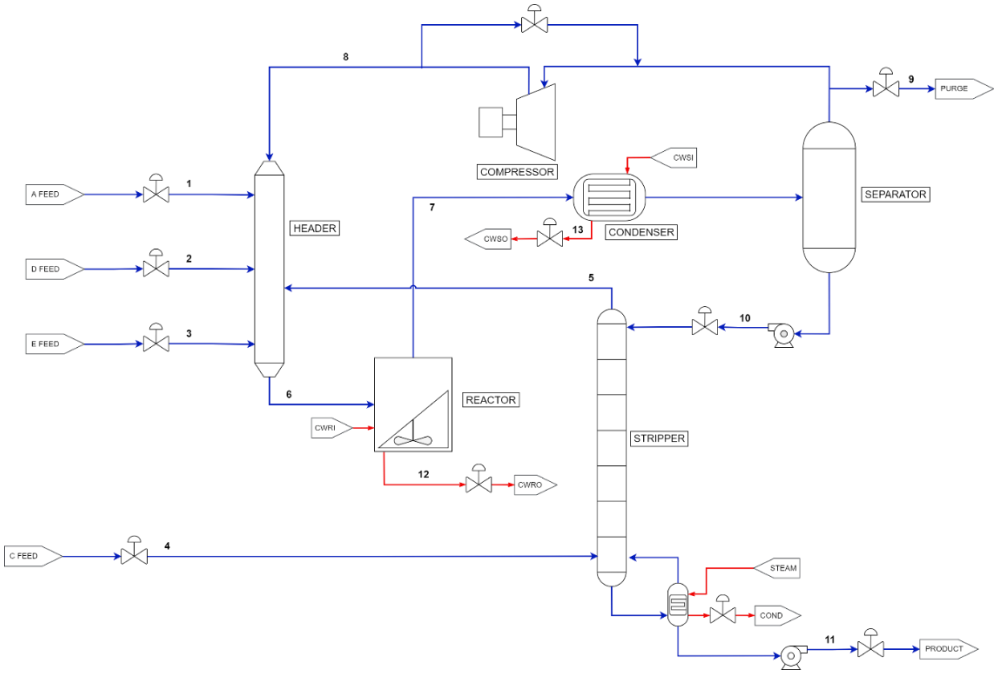

Figure 1. Process diagram of the Tennessee Eastman process.

To obtain the desired products, the system is made up of five unit operations: a reactor, a condenser, a liquid-vapor phase separator, a compressor, and a stripper column.

In terms of the user interface, the TE process shows 41 measurable variables and 11 first-order dynamics actuators that allow for external interaction with the system. In the data you will use, you will find these variables named as "`XMEAS(1)`", "`XMEAS(2)`", …, "`XMEAS(41)`" for the sensors, and "`XMV(1)`" through "`XMV(11)`" for actuators. The sampling time is 3 min. All variables have noise, and the composition variables `XMEAS(23)` to `XMEAS(36)` have measurement delay and dead time of 6 min, and `XMEAS(37)` to `XMEAS(41)` show a delay and dead time of 15 min. A summary of the variables is provided at the end of this documents. The process allows the activation of a series of faults. These disturbances represent step changes, drifts, random noise, and component malfunctions in the supply conditions, cooling streams, and actuators of the plant.

For this task, we have identified a series of disturbances that are well known and compiled historical data of them into the file `data_train.csv`. Your task is to create an anomaly detection model to flag unknown disturbances. We have also managed to collect some example data from unknown faults and located them in the file `data_test.csv`, which you can use to test your implementations.

---

[1] Downs, J. J., & Vogel, E. F. (1993). A plant-wide industrial process control problem. *Computers & Chemical Engineering*, *17*(3), 245–255. https://doi.org/10.1016/0098-1354(93)80018-I

You can find and load the data from this repository, where you can also find a submission example that you must check before starting your code.

You have complete freedom in what algorithm or sequence of algorithms you want to implement, as long as you justify its usage on your report and restrict yourself to the packages and code limitations stated below. Since the takeaway is on working with chemical engineering systems with a limited amount of data, we operate under the following assumptions:

- You will be partly tested on data similar to the train and test sets provided, but also with different data. Remember, an important goal of anomaly detection models is to be robust against unknown anomalies.
- You must use semi-supervised anomaly detection techniques, meaning by this that you must **fit your models using the training set only**. The test set is there only for you to gauge how different approaches compare.
- Your code is limited by a fixed time budget constraint. We will not consider algorithms that take longer than 2 minutes, included loading, preprocessing and model fitting.

Grading (per group):

- 2-page Report on your algorithm (20%).
    - The report should have the following sections:
        - Big picture explanation and intuition behind the algorithm or pipeline of algorithms selected
        - Methodology
        - Pseudocode
        - You are allowed a figure for your algorithm which is not considered for the length of the report.
        - References are not considered for the length of the report.
    - The report should include a pseudocode following the format specified here.
    - The report should also explain the rationale behind the algorithm and in paragraph form the main steps in the algorithm.
    - Please do not use a letter smaller than size 11 (with a decent font: e.g., Arial, Times New Roman, Calibri, Latex font), and margins no smaller than 2cm Top, Bottom, Left, Right.
    - You will be graded based on clarity of communication, creativity of your algorithm and scientific explanation of your methods. You are encouraged to comment on the trials and errors you have done, and stress which approaches worked better and why.
- Your implementation will be graded based on your performance on a larger dataset (80%).
    - You will upload a single python file ".py" such that the file is "team_name.py". This file MUST contain:
        - A function called `fit_preprocess` that returns a single object representing the parameters used to preprocess the data
        - A function called `load_and_preprocess` that returns the preprocessed data and labels,
        - A function called `fit_model` that returns a trained model that will be used by `predict`,
        - A function called `predict` that returns the anomaly detection prediction for the input data.

      Check the example script for a deeper description. Besides these functions, you can use other pieces of code that may help you. In fact, modularizing your

code in several blocks of code (functions) is encouraged and in line with the "good programming practices" mentioned at the end of this document.

- The data over which to fit and evaluate your models is provided on this repository. Data are contained in comma-separated values (CSV) files, on which the last column represents the label and the rest of columns are the proper features of the data. An instance is normal if label = 0, and anomalous otherwise.
- Take into account that files with disturbances (anomalies) might contain a mix of normal instances and anomalous instances, so you need to always check the label of the instance you are dealing with.
- You must load the data you need for training and testing during runtime from our GitHub repository, directory **`data`**. For instance, to download the normal operation data of you can use the GitHub raw link (check example files for usage with the pandas package):

  https://raw.githubusercontent.com/iraola/ML4CE-AD/main/coursework/data/data_train.csv

- The criteria to assess the anomaly detection model will be checking its performance over different datasets. We will be looking for a balance between the accuracy predicting normal instances and the accuracy predicting anomalies.

IMPORTANT:

- Good coding practice: Make sure your function takes the same inputs and outputs as the example functions provided, and make sure that your implementation is flexible enough to handle different dimensionalities. This is **very important**, as otherwise we have no way to test your algorithms, and you will not get marks for those problems.
- **Do not use global variables**, always use local variables declared inside functions.
- Take a look at the following link. It contains the most used standard for coding practices in Python.
- Data exploration. We encourage spending some time visualizing what data looks like before blindingly applying algorithms to it.
- Avoid hard-coding thresholds or other tuned values into the code. Your code needs to show how it reaches any of these values in some systematic way.
- Stochasticity: Do not set random seeds to your algorithm.
- Packages: please restrict to use numpy, scipy, random, time and scikit-learn. **Make absolutely sure** that your code runs in an environment where only numpy, scipy, random, time and scikit-learn packages are available.
- Before submitting, make sure your code can be called from an external script located in the same directory as your code, like this.

Code example:

- Example codes include:
  - solution_example.py

Table 1. Summary of measured variables.

XMEAS	Description	Units
1	Feed flow component A (stream 1)	kscmh
2	Feed flow component D (stream 2)	kg/h
3	Feed flow component E (stream 3)	kg/h
4	Feed flow components A & C (stream 4)	kscmh
5	Recycle flow to reactor from separator (stream 8)	kscmh
6	Reactor feed (stream 6)	kscmh
7	Reactor pressure	kPa gauge
8	Reactor level	%
9	Reactor temperature	°C
10	Purge flow (stream 9)	kscmh
11	Separator temperature	°C
12	Separator level	%
13	Separator pressure	kPa gauge
14	Separator underflow (liquid phase)	m^3/h
15	Stripper level	%
16	Stripper pressure	kPa gauge
17	Stripper underflow (stream 11)	m^3/h
18	Stripper temperature	°C
19	Stripper steam flow	kg/h
20	Compressor work	kW
21	Reactor cooling water outlet temperature	°C
22	Condenser cooling water outlet temperature	°C
23	Concentration of A in Reactor feed (stream 6)	mol %
24	Concentration of B in Reactor feed (stream 6)	mol %
25	Concentration of C in Reactor feed (stream 6)	mol %
26	Concentration of D in Reactor feed (stream 6)	mol %
27	Concentration of E in Reactor feed (stream 6)	mol %
28	Concentration of F in Reactor feed (stream 6)	mol %
29	Concentration of A in Purge (stream 9)	mol %
30	Concentration of B in Purge (stream 9)	mol %
31	Concentration of C in Purge (stream 9)	mol %
32	Concentration of D in Purge (stream 9)	mol %
33	Concentration of E in Purge (stream 9)	mol %
34	Concentration of F in Purge (stream 9)	mol %
35	Concentration of G in Purge (stream 9)	mol %
36	Concentration of H in Purge (stream 9)	mol %
37	Concentration of D in stripper underflow (stream 11)	mol %
38	Concentration of E in stripper underflow (stream 11)	mol %
39	Concentration of F in stripper underflow (stream 11)	mol %
40	Concentration of G in stripper underflow (stream 11)	mol %
41	Concentration of H in stripper underflow (stream 11)	mol %

Table 2. Summary of actuator variables.

XMV	Description	Units
1	D feed flow (stream 2)	%
2	E feed flow (stream 3)	%
3	A feed flow (stream 1)	%
4	A and C feed flow (stream 4)	%
5	Compressor recycle valve	%
6	Purge valve (stream 9)	%
7	Separator liquid flow (stream 10)	%
8	Stripper liquid product flow (stream 11)	%
9	Stripper steam valve	%
10	Reactor cooling water flow	%
11	Condenser cooling water flow	%