CHE 4230 Advanced Process Control Systems – Spring 2023 Semester Project

Introduction:

The Tennessee Eastman (TE) process[1] created by the Eastman Chemical Co. has been widely used as a benchmark process for evaluating process supervision methods. The process uses four gaseous reactants (A, C, D, and E) to produce two liquid products (G and H) in competition with an unwanted byproduct (F) and in the presence of an inert (B). There are four competing reactions related to temperature through Arrhenius laws (two of which produce desired products, while the other two produce the byproduct):

$$\begin{gathered} \mathrm{A(g)} + \mathrm{C(g)} + \mathrm{D(g)} \rightarrow \mathrm{G(liq)} \\ \mathrm{A(g)} + \mathrm{C(g)} + \mathrm{E(g)} \rightarrow \mathrm{H(liq)} \\ \mathrm{A(g)} + \mathrm{E(g)} \rightarrow \mathrm{F(liq)} \\ 3\mathrm{D(g)} \rightarrow 2\mathrm{F(liq)} \end{gathered} \tag{1}$$

The entire simulation includes five unit operations: a reactor, condenser, separator, compressor, and a stripper and has 41 measured variables along with 12 manipulated variables for a total of 53 process variables that are given in **Tables 1** and **2**.

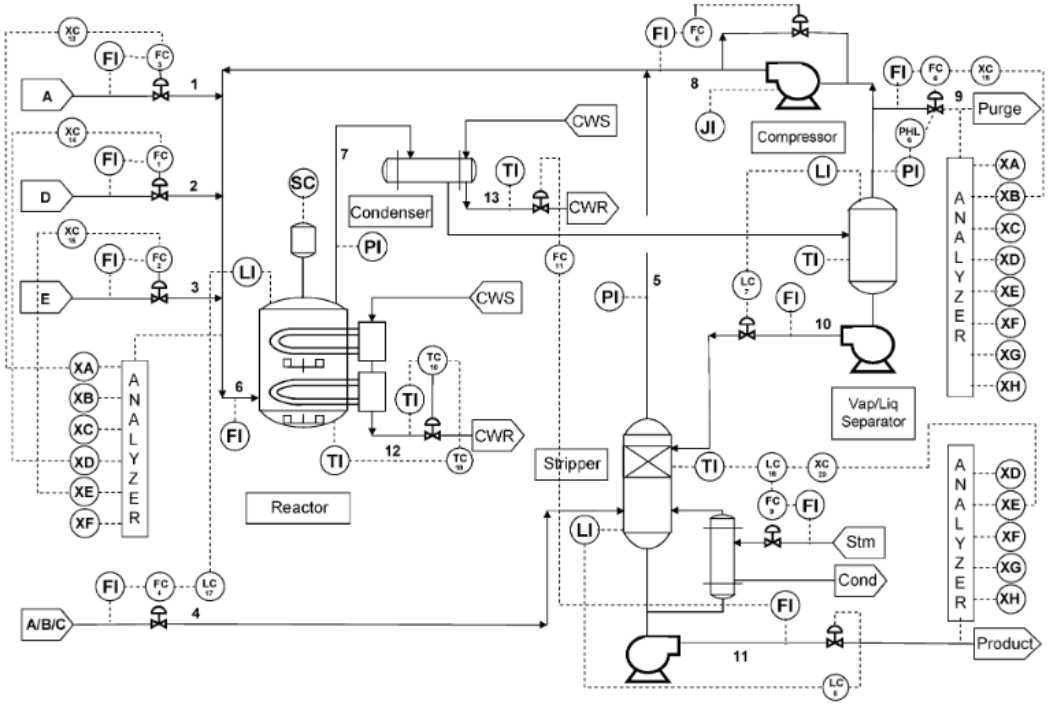


Figure 1. Process flow diagram of the TE process.

Table 1. TEP manipulated variables.

Variable	Description
XMV (1)	D feed flow (stream 2)
XMV (2)	E feed flow (stream 3)
XMV (3)	A feed flow (stream 1)
XMV (4)	Total feed flow (Stream 4)
XMV (5)	Compressor recycle valve
XMV (6)	Purge valve (stream 9)
XMV (7)	Separator pot liquid flow (stream 10)
XMV (8)	Stripper liquid product flow (stream 11)
XMV (9)	Stripper stream valve
XMV (10)	Reactor cooling water flow
XMV (11)	Condenser Cooling Water flow
XMV (12)	Agitator speed

Table 2. TEP process measurements (3 minute sampling interval).

Variable	Description	Units
XMEAS (1)	A feed (stream 1)	kscmh
XMEAS (2)	D feed (stream 2)	kg/h
XMEAS (3)	E feed (stream 3)	kg/h
XMEAS (4)	Total feed (stream 4)	kscmh
XMEAS (5)	Recycle flow (stream 8)	kscmh
XMEAS (6)	Reactor feed rate (stream 6)	kscmh
XMEAS (7)	Reactor pressure	kPa gauge
XMEAS (8)	Reactor level	%
XMEAS (9)	Reactor temperature	°C
XMEAS (10)	Purge rate (stream 9)	Kscmh
XMEAS (11)	Product sep. temperature	°C
XMEAS (12)	Product sep. level	%
XMEAS (13)	Product sep. pressure	kPa gauge
XMEAS (14)	Prod. sep. underflow (stream 10)	m^3/h
XMEAS (15)	Stripper level	%
XMEAS (16)	Stripper pressure	kPa gauge
XMEAS (17)	Stripper underflow (stream 11)	m^3/h
XMEAS (18)	Stripper temperature	°C
XMEAS (19)	Stripper steam flow	kg/h
XMEAS (20)	Compressor work	kW
XMEAS (21)	Reactor cooling water outlet temp	°C
XMEAS (22)	Separator cooling water outlet temp	°C

Project description:

Part 1: Software development environment and project planning

Establishing an environment for code management, version control and project planning and tracking is essential for software development, especially in teams. For this part of the project you will setup a GitHub repository with a directory structure organized according to the project parts, tasks and deliverables (see Appendix 1 for an example of a directory structure). This is where the team will share and keep track of all source code and deliverables. All TAs must be granted added to the repository with full view privileges.

Additionally, prepare a 2-page document outlining how you plant to organize the work for this project, included tentative tools/methods to be used and provisional internal deadlines.

Deliverables:

1. GitHub repository with appropriate directory structure. (Delivered by granting access to TAs).
2. 2-page project plan.

Part 2: Data analysis and pattern recognition

For this part of the project, the objective is to find important patterns from the unlabeled TEP data. Dimensionality reduction (DR) and clustering are commonly used methods in the domain of unsupervised learning. DR methods can find important features from the high dimensional spaces and project these features into low dimensional spaces. Then, clustering methods can be applied on the low dimensional spaces to easily group up data samples with similar information. After that, further analysis can be done based on the DR and clustering results.

In this part, apply DR and clustering methods to analyze the given datasets in two stages. In the first stage, investigate the performance of different DR methods (≥5) and clustering methods (≥3). In the second stage, the corresponding label of the given dataset will be given. Compare the ground truth label with the data mining results from stage 1, and discuss the comparison results.

Deliverables:

1. Source code for data preprocessing, dimensionality reduction, clustering, and plotting implementations.
2. Annotated Jupyter Notebook with the tests, experiments and analyses performed using the code in Deliverable 1.
3. 1-page summary of the main findings.

Part 3: Fault detection and classification

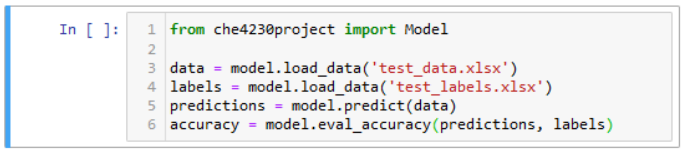
Artificial neural network (ANN) and support-vector machine (SVM) are two of the most commonly used supervised learning models in data science areas. For the fault identification and classification purposes, they can be used to build a real-time fault classification model. For this part, build upon the results from Part 1 and:

a. Think about the preprocessing procedures using different subsets of faulty and normal datasets.

b. Build a single layer ANN for the fault classification. Test the ANN performance with different configurations of hidden-layer neurons inside and other necessary hyper-parameters.
c. Build an SVM fault classifier.
d. Compare their performance and make a discussion.

Deliverables:

1. Source code for data preprocessing, classification, and plotting implementations.
2. Annotated Jupyter notebook with the tests, experiments and analyses performed using the code in Deliverable 1.
3. A pip-installable package with the necessary methods to load a dataset, make predictions, and report accuracy. See example of how the package will be evaluated.



```
In [ ]:
from che4230project import Model

data = model.load_data('test_data.xlsx')
labels = model.load_data('test_labels.xlsx')
predictions = model.predict(data)
accuracy = model.eval_accuracy(predictions, labels)
```

4. A 1-page summary of the main findings.

Preliminary and final reports:

For the preliminary report, include Part 1 in Project Description. For the final report include Part 2 and Part 3.

Grading:

Preliminary report:

- Repository: 50 %
- Project plan: 50 %

Final report:

- Source code, annotated Jupyter Notebooks, and summaries for parts 2 and 3: 80 %
- Accuracy of model in part 3: 20 % (evaluated as $20 \times Accuracy$).

Due dates:

Preliminary report: February 17

Final report: March 24

References:

1. Downs, J. J.; Vogel, E. F., A plant-wide industrial process control problem. *Computers & Chemical Engineering* **1993,** *17* (3), 245-255.

Appendix 1. Sample of directory structure

```
├── LICENSE
├── Makefile           <- Makefile with commands like `make data` or `make train`
├── README.md          <- The top-level README for developers using this project.
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been transformed.
│   ├── processed      <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
│
├── docs               <- A default Sphinx project; see sphinx-doc.org for details
│
├── models             <- Trained and serialized models, model predictions, or model summaries
│
├── notebooks          <- Jupyter notebooks. Naming convention is a number (for ordering),
│                         the creator's initials, and a short `-` delimited description, e.g.
│                         `1.0-jqp-initial-data-exploration`.
│
├── references         <- Data dictionaries, manuals, and all other explanatory materials.
│
├── reports            <- Generated analysis as HTML, PDF, LaTeX, etc.
│   └── figures        <- Generated graphics and figures to be used in reporting
│
├── requirements.txt   <- The requirements file for reproducing the analysis environment, e.g.
│                         generated with `pip freeze > requirements.txt`
│
├── setup.py           <- Make this project pip installable with `pip install -e`
├── src                <- Source code for use in this project.
│   ├── __init__.py    <- Makes src a Python module
│   │
│   ├── data           <- Scripts to download or generate data
│   │   └── make_dataset.py
│   │
│   ├── features       <- Scripts to turn raw data into features for modeling
│   │   └── build_features.py
│   │
│   ├── models         <- Scripts to train models and then use trained models to make
│   │   │                 predictions
│   │   ├── predict_model.py
│   │   └── train_model.py
│   │
│   └── visualization  <- Scripts to create exploratory and results oriented visualizations
│       └── visualize.py
│
└── tox.ini            <- tox file with settings for running tox; see tox.readthedocs.io
```