CSTR State Prediction and Fault Detection & Classification

By

Mokshika Kureti

(210107053)

Submission date: 25/04/2024

Final Project Submission

Application of AI and ML in Chemical Engineering

(CL 653)

Contents

1. Executive Summary

This project proposes a system state prediction based on a neural network model for a Continuous Stirred Tank Reactor (CSTR), along with fault detection using classification techniques. Given the inherent nonlinearity of the CSTR process, traditional methods may not suffice, warranting the adoption of nonlinear predictive techniques such as neural networks. The paper delineates the neural network model and its application in forecasting CSTR behavior within a specified horizon, elucidating the optimization procedure. Moreover, it incorporates fault detection capabilities employing classification algorithms to identify and address faults in the system. By harnessing Artificial Intelligence methods like Neural Networks, the approach aims to surmount modeling challenges and attain precise understanding of system state and simultaneously employing Machine Learning based fault detection & classification in the CSTR system.

2. Introduction

2.1 Problem Statement

Continuous Stirred Tank Reactors (CSTRs) play a crucial role in various chemical and industrial processes, ranging from pharmaceutical manufacturing to wastewater treatment. Their widespread application underscores the importance of ensuring efficient operation and timely fault detection to maintain process safety, product quality, and operational efficiency. Traditional methods for state estimation and fault detection in CSTRs often rely on physics-based models and rule-based approaches. While effective, these methods may struggle to capture the complex and non-linear dynamics inherent in real-world CSTR systems. Additionally, they can be sensitive to model inaccuracies and uncertainties, limiting their robustness in practical applications.
To address these challenges, there has been a growing interest in leveraging advanced computational techniques, particularly neural networks (NN) and machine learning (ML) algorithms, for predictive state estimation and fault detection in CSTR systems. Neural networks offer the capability to model complex non-linear relationships and dynamics, making them well-suited for accurate state prediction based on historical process data. Meanwhile, ML classification methods provide efficient tools for distinguishing between normal operating conditions and various fault scenarios, thereby enabling proactive fault detection and mitigation strategies.

2.2 Objectives

1. Develop a neural network for accurate prediction of CSTR system state variables.
2. Evaluate machine learning classification algorithms for effective fault detection in

CSTR systems.

2.3 Description

CONTINUOUS STIRRED TANK REACTOR

The Continuous Stirred Tank Reactor is shown in Figure 1. This CSTR model is used as a nonlinear system.

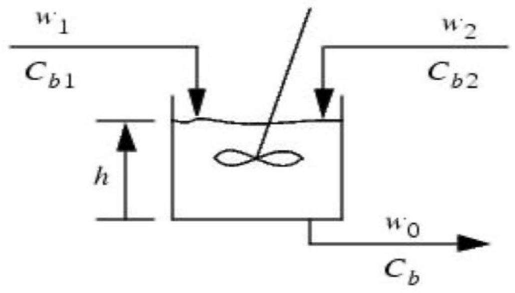


Figure 1: Continuous Stirred Tank Reactor

Continuous Stirred Tank Reactors (CSTRs) are fundamental components in chemical engineering processes, utilized for various tasks such as chemical synthesis, fermentation, and wastewater treatment. They maintain a constant operating volume and typically involve continuous inflow and outflow of reactants and products. The dynamics of CSTRs are governed by complex nonlinear equations representing mass and energy balances, making their behavior challenging to model accurately. Additionally, these systems are susceptible to various faults and disturbances, including sensor failures, actuator faults, and changes in operating conditions, which can significantly impact process performance and product quality.

The specific problem addressed in this project is the development of a comprehensive framework for predictive state estimation and fault detection in CSTR systems. The primary goal is to design an integrated system that combines the predictive capabilities of neural networks with the fault detection capabilities of machine learning classification algorithms. This system aims to accurately predict the state variables of the CSTR and detect various fault scenarios, including sensor failures, actuator faults, and process disturbances, in real-time.

Efficient operation and maintenance of Continuous Stirred Tank Reactors (CSTRs) is paramount for industrial processes, ensuring safety, product quality, and operational efficiency. The proposed framework, which incorporates predictive state estimation and fault detection, offers several key advantages. It improves process efficiency by enabling better control and optimization of CSTR conditions, thereby enhancing production efficiency. Moreover, early fault detection minimizes risks, enhancing safety for personnel and equipment. By allowing proactive maintenance and intervention, it reduces downtime and associated production losses, leading to significant cost savings. Integrating advanced computational techniques such as Neural Networks (NN) and Machine Learning (ML) not only advances process control and automation but also contributes to industry innovation. In conclusion, addressing the challenges of predictive state estimation and fault detection in CSTR systems is vital for bolstering operational reliability, safety, and efficiency in industrial processes, underscoring the project's significance.

CSTR STATE PREDICTION WITH NEURAL NETWORKS

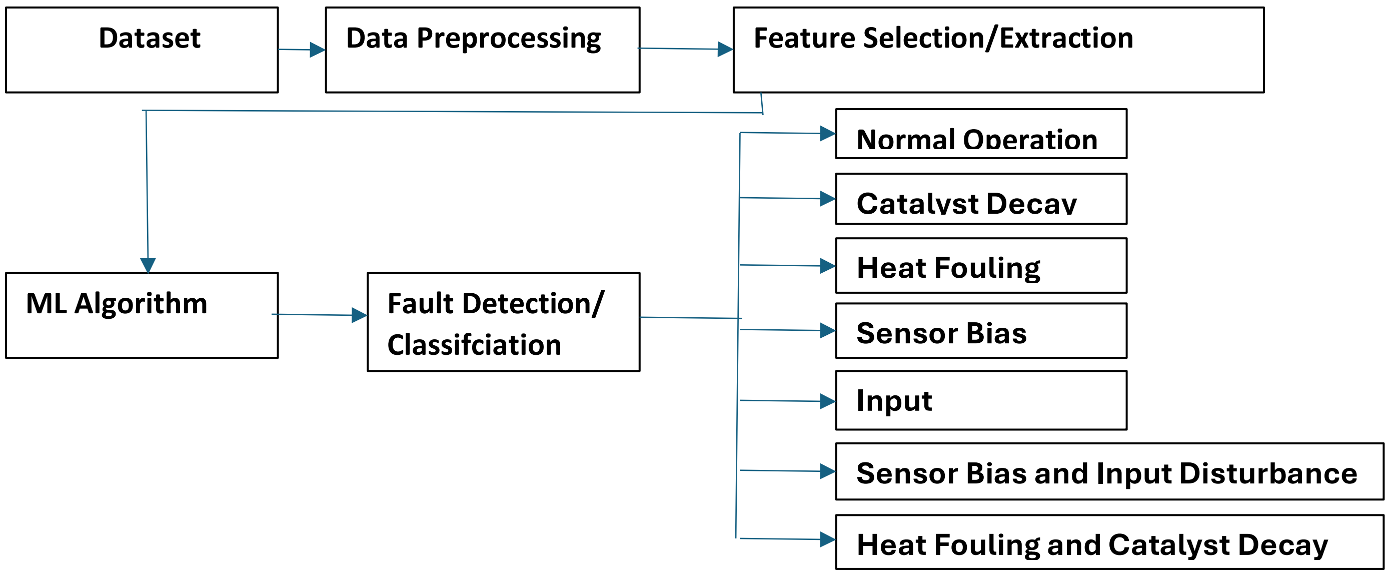
The purpose of our neural network model is to do time series prediction of the plant output. Given a series of control signals u and past data y_t, it is desired to predict the plant output series y_N. The network is trained to do one step ahead prediction, i.e., to predict the plant output y_{t+1} given the current control signal u_t and plant output y_t. The neural network will implement the function

$$Y'_{t+1} = f(u_t, y_t) \qquad (1)$$

As is discussed above, y_t has to contain sufficient information for this prediction to be possible. It is assumed that y_t is multivariable. One problem is that this method will cause a rapidly increasing divergence due to accumulation of errors. It therefore puts high demands on accuracy of the model. The better the model matches the actual plant the less significant the accumulated error. Sampling time as large as possible is an effective method to reduce the error accumulation as it effectively reduces the number of steps needed for a given time horizon. The neural network trained to do one step ahead prediction will model the plant. The acquisition of this model is also referred to as System Identification.

FAULT DETECTION AND CLASSIFICATION IN CSTR

Implementing fault detection and classification in a Continuous Stirred Tank Reactor (CSTR) using a ML model, we attempt to detect faults such as Catalyst Decay, Heat Fouling, Sensor Bias, and Input Disturbance, along with combined fault scenarios. Relevant features are extracted from this data to distinguish between normal operation and different types of faults. Fault detection algorithms are then developed to identify the presence of faults based on these features, while fault classification algorithms classify the detected faults into predefined classes such as Normal Operation, Catalyst Decay, Heat Fouling, Sensor Bias, Input Disturbance, Sensor Bias combined with Input Disturbance, and Heat Fouling combined with Catalyst Decay. The performance of the fault detection and classification system is evaluated using test data, and iterative improvements are made as necessary to refine the algorithms for accurate fault detection and classification in practical applications.



Dataset
Data Preprocessing
Feature Selection/Extraction
ML Algorithm
Fault Detection/ Classifciation
Normal Operation
Catalyst Decay
Heat Fouling
Sensor Bias
Input
Sensor Bias and Input Disturbance
Heat Fouling and Catalyst Decay

Figure 2: Block Diagram of Fault Detection and Classification in CSTR

3. Methodology

3.1 Dataset

The dataset is structured as a matrix with 2860 rows and 1404 columns. Each row represents an independent simulation of a Continuous Stirred-Tank Reactor (CSTR) process.

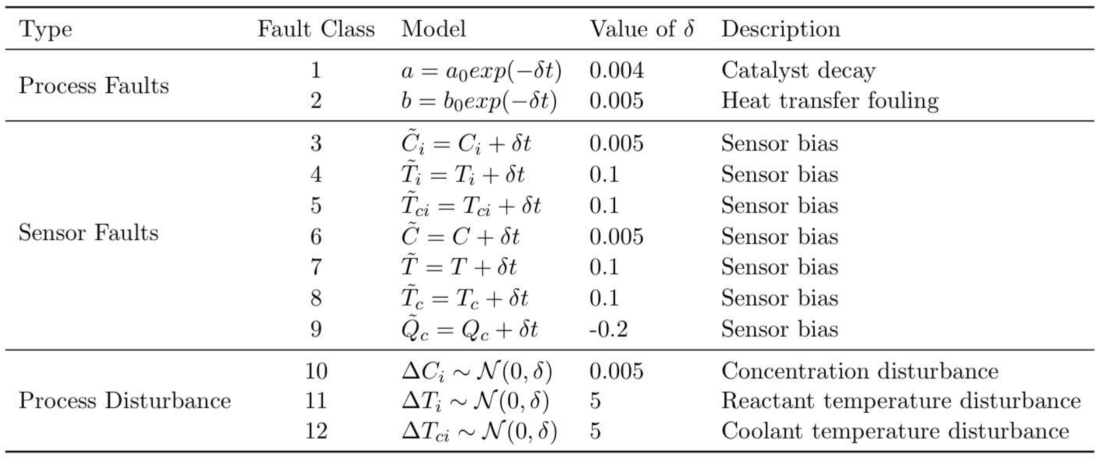
Process Variables (First 1400 Columns):

– Each row represents one simulation.
– There are 200 measurements for 7 process variables, resulting in 1400 columns.
– These 7 measured variables are:

i. Concentration of A in the inlet flow,
ii. Temperature of the inlet flow,
iii. Temperature of the inlet coolant flow,
iv. Coolant flowrate,
v. Concentration of B in the outlet flow,
vi. Temperature of the outlet flow,
vii. Temperature of the outlet coolant flow.

Class Labels (Column 1401):

- This column contains labels indicating the 12 classes.
- These class labels represent 12 faults of 7 variables of the CSTR, measured throughout 200 minutes (about 3 and a half hours), at a 1-minute rate.

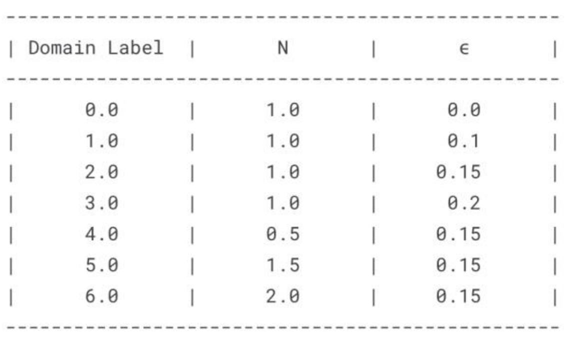


Type	Fault Class	Model	Value of δ	Description
Process Faults	1	$a = a_0 exp(-\delta t)$	0.004	Catalyst decay
	2	$b = b_0 exp(-\delta t)$	0.005	Heat transfer fouling
Sensor Faults	3	$\tilde{C}_i = C_i + \delta t$	0.005	Sensor bias
	4	$\tilde{T}_i = T_i + \delta t$	0.1	Sensor bias
	5	$\tilde{T}_{ci} = T_{ci} + \delta t$	0.1	Sensor bias
	6	$\tilde{C} = C + \delta t$	0.005	Sensor bias
	7	$\tilde{T} = T + \delta t$	0.1	Sensor bias
	8	$\tilde{T}_c = T_c + \delta t$	0.1	Sensor bias
	9	$\tilde{Q}_c = Q_c + \delta t$	-0.2	Sensor bias
Process Disturbance	10	$\Delta C_i \sim \mathcal{N}(0, \delta)$	0.005	Concentration disturbance
	11	$\Delta T_i \sim \mathcal{N}(0, \delta)$	5	Reactant temperature disturbance
	12	$\Delta T_{ci} \sim \mathcal{N}(0, \delta)$	5	Coolant temperature disturbance

- Each label corresponds to a specific simulation.

Domain Labels (Column 1402):

- There are 7 domains, each label corresponds to a combination of reaction order and parameter noise.
- There are 4 types of reaction order: 0.5, 1.0, 1.5, and 2.0. This corresponds to the velocity of the reaction A→B. Each target domain has 260 samples (20x samples for each fault + 20x samples for normal operation). There are 4 domains with N=1, i.e., the source domain (which has N=1.0 and ϵ=0.0), and N=1.0 with ϵ∈ {0.1,0.15,0.2}



```
----------------------------------------------
| Domain Label |       N       |       ϵ      |
----------------------------------------------
|      0.0     |      1.0      |      0.0     |
|      1.0     |      1.0      |      0.1     |
|      2.0     |      1.0      |     0.15     |
|      3.0     |      1.0      |      0.2     |
|      4.0     |      0.5      |     0.15     |
|      5.0     |      1.5      |     0.15     |
|      6.0     |      2.0      |     0.15     |
----------------------------------------------
```

Simulation Parameter Noise (Column 1403):

- This column contains information about the level of noise or uncertainty associated with each simulation's parameters.
- Noise could arise from measurement errors, model inaccuracies, or variability in the system.
- It provides insight into the reliability or confidence level of the simulation results.

Reaction Order (Column 1404):

- This column contains information about the reaction order of the CSTR process.

There are four different reaction orders: 0.5, 1.0, 1.5, and 2.0, which determine the rate at which reaction A transforms into B.

- The reaction order indicates how the rate of reaction depends on the concentration of reactants.
- Different reaction orders can significantly impact the behavior of the CSTR.

3.2 Data preprocessing

- All the data used related to CSTR system state are continuous in nature, however the variables involved are of different scales. Variables like concentration, flow rate and temperature are of different numerical scales.
- Hence, we need to normalize the data and bring all the different variables to the same scale. This can be done by transforming all the features to the same mean and variance.
- Another step required is to check for any outliers in the data and then either drop those data points or perform clipping

3.3 Model Architecture

1. CSTR STATE PREDICTION

This project aims to develop a regression model to predict CSTR state variables using simulated data. The project utilizes machine learning techniques, specifically focusing on the Multi-Layer Perceptron (MLP) regression model.

The MLP regression model was chosen due to its capability to learn complex patterns in data, making it suitable for the non-linear behavior of CSTR. The model architecture includes two hidden layers with 100 and 50 neurons, respectively, employing the ReLU activation function and the Adam optimizer. These choices were made to balance model complexity and performance.

1. FAULT DETECTION & CLASSIFICATION

The objective of this project is to develop machine learning models capable of accurately classifying faults based on input data. The dataset consists of features extracted from various fault scenarios, with the goal of predicting the type of fault present. For the fault detection and classification task, ML models can be used. Given the input of the 7 system variables over 200-time stamps, our model needs to classify the CSTR operation into 13 classes (normal operation + 12 faults).

Model Selection

To address the problem of fault classification, we considered following machine learning models:

- Logistic Regression: A simple linear model suitable for binary and multi-class classification tasks.
- K-Nearest Neighbors (KNN): A non-parametric algorithm that classifies instances based on the majority vote of its neighbors. (Used 5 nearest neighbors)
- Support Vector Machine (SVM) Classifier: Capable of capturing complex decision boundaries in high-dimensional spaces.
- Shallow Neural Network (Multi-Layer Perceptron, MLP): A neural network architecture capable of learning intricate patterns in the data. (Used 2 hidden layers)

4. Implementation Plan

4.1 Model Training

1. CSTR STATE PREDICTION

The dataset was preprocessed to extract features and targets specific to CSTR state variables. Features were reshaped to fit the input requirements of the MLP model (using a 2 layer neural network). The dataset was split into training and testing sets using an 80-20 split ratio to evaluate model performance on unseen data. The model was trained using the training set of 1040 samples till convergence.

The neural network was trained on Mean Squared Error loss for regression task, the loss reached a value of 21.43 when the model was fit to training data.

2. FAULT DETECTION & CLASSIFICATION

We adopted a standardized training approach for all four ML models chosen:

- Data preprocessing: Standardization of features to zero mean and unit variance.
- Dimensionality reduction: Reducing dimensions from 1400 to 64 features using Principal Component Analysis (PCA). An analysis of the number of PCA components required has also been done.

- Model training: Each model was trained using a pipeline that included data preprocessing, dimensionality reduction steps and model fitting.
- Evaluation: Models were evaluated based on accuracy, confusion matrix, precision, recall, and F1-score. Also, the models have been evaluated on domains outside the training domain to check model robustness and generalizability.

4.2 Model Evaluation

1. CSTR STATE PREDICTION

Evaluation and Validation: Two evaluation metrics were employed to assess model performance:

- **Mean Squared Error (MSE):** This metric provides insight into the average squared differences between predicted and true values, emphasizing larger errors. This is also the loss function used.
- **Mean Absolute Deviation Percentage (MAD %):** MAD % calculates the mean absolute deviation of predicted values from true values relative to the true mean, offering a perspective on error magnitude.

The model's performance was evaluated using these metrics on the testing set. Additionally, validation techniques such as cross-validation and external validation on unseen datasets could be considered to further validate the model's generalizability and robustness.

2. FAULT DETECTION & CLASSIFICATION

Evaluation Metrics: Accuracy was used as the primary evaluation metric to measure the overall correctness of the model's predictions. Confusion matrix and derived metrics provided insights into the model's performance across different fault classes. Models are tested on both same domain and cross-domain. Accuracies of the models:

Validation Strategy: Train test split and external validation (cross domain) was employed to validate the model's performance robustness. External validation on unseen datasets ensured generalizability beyond the training data.

An analysis on varying the number of PCA components of the data used in pipeline was also done to find optimal performance.

5. Testing and Deployment

5.1 Testing Strategy

1. CSTR STATE PREDICTION

The model is trained and tested for the case where it is provided with the 20 previous readings of the 7 variables of the CSTR system, and the model's task is to predict the 7 state variables at the current time. The model is further tested for its ability to predict more than one step ahead. This is done by first feeding the model with 20 readings to get the 21st reading, then the 2nd to 20th reading along with the predicted 21st reading is used as the 20 inputs to predict the 22nd reading. In this manner the model is evaluated for prediction up to 10 steps ahead using the current 20 reading. In such an autoregressive method, the errors accumulate as we try to predict further ahead, and the error will grow fast.

2. FAULT DETECTION & CLASSIFICATION

Our dataset has 7 domains, the first domain contains 1300 samples and the other 6 contain 260 each. We evaluate our model performance within the same domain, where the model is trained & tested on the same domain and model performance is also evaluated on cross domain testing. In cross domain testing the model is trained on domain 1 (with 1300 samples) but tested on one of the 6 other domains (with 260 samples). Such analysis allows to evaluate model robustness.

6. Results & Analysis

6.1 CSTR STATE PREDICTION

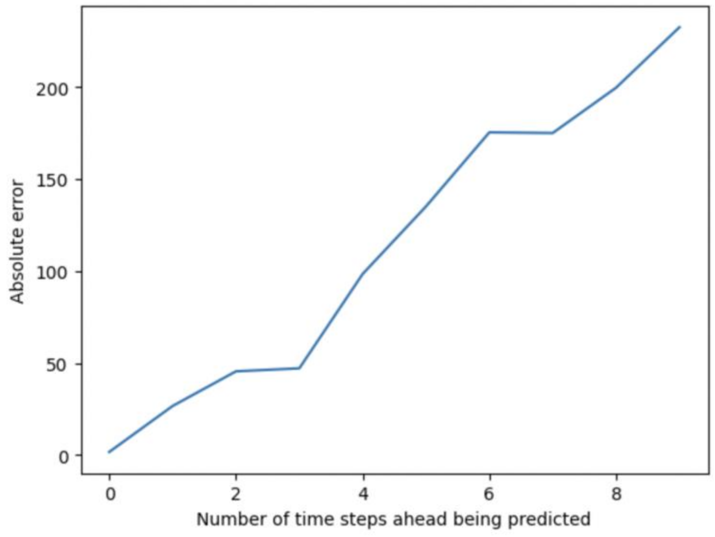
The performance of each model varied across different fault scenarios. Overall, the MLP model demonstrated the highest accuracy and performed well in capturing complex patterns in the data. Also, accuracies of models in same domain are higher than cross domain. However, further analysis is required to identify the strengths and weaknesses of each model.

After training and evaluation, the model achieved a Mean Squared Error of [21.7118] and a Mean Absolute Deviation Percentage of [0.963056901245642]. These results indicate that the predicted value of each variable deviates ~0.963% from the actual value. If we look at the Mean Absolute Deviation Percentage for each variable:

Prediction error in Concentration of A in the inlet flow - 0.955%
Prediction error in Temperature of the inlet flow- 0.940%
Prediction error in Temperature of the inlet coolant flow - 1.174%
Prediction error in Coolant flowrate - 0.887%
Prediction error in Concentration of B in the outlet flow - 0.981%
Prediction error in Temperature of the outlet flow - 0.947%
Prediction error in Temperature of the outlet coolant flow - 0.761%

We observe that the error is approximately the same for all variables, the model performs best for predicting outlet coolant flow temperature and has the highest error for inlet coolant flow temperature.

Below plot shows the error in prediction at each number of future time steps ahead. It is very evident from the plot that such an autoregressive method will accumulate errors rapidly.

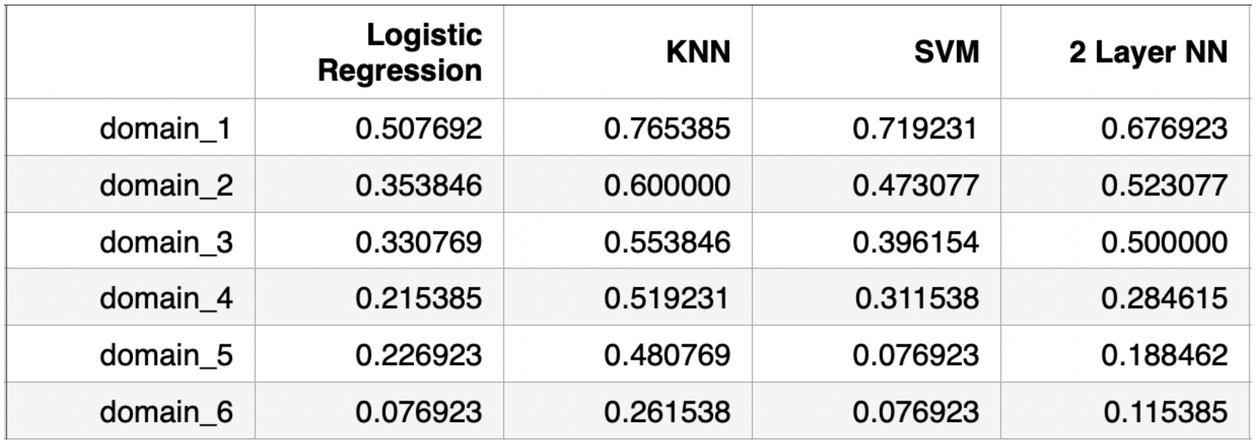


6.1 FAULT DETECTION & CLASSIFICATION

Below are the results for accuracy of the four applied ML models in same domain testing (training & testing on domain 0) and cross domain (training on domain 0 & testing on domain 1 here)

1. Logistic Regression:
   Same domain - 0.7784615384615384
   Cross domain - 0.4653846153846154
2. K-Nearest Neighbors:
   Same domain - 0.9661538461538461
   Cross domain - 0.7615384615384615
3. Support Vector Machine (SVM) Classifier:
   Same domain - 0.9292307692307692
   Cross domain - 0.7423076923076923
4. Shallow Neural Network (Multi-Layer Perceptron, MLP):
   Same domain - 0.96
   Cross domain - 0.6538461538461539

A further evaluation on cross domain performance is done to check for model robustness. All models are applied to cross domain testing where all models are trained on domain 0 and are tested separately on domains 1 to 6.



	Logistic Regression	KNN	SVM	2 Layer NN
domain_1	0.507692	0.765385	0.719231	0.676923
domain_2	0.353846	0.600000	0.473077	0.523077
domain_3	0.330769	0.553846	0.396154	0.500000
domain_4	0.215385	0.519231	0.311538	0.284615
domain_5	0.226923	0.480769	0.076923	0.188462
domain_6	0.076923	0.261538	0.076923	0.115385

From this table we can infer that KNN is the most robust model here across all domains, showing best performance in all cross-domain settings against other models and has least performance variance between domains (especially domain 1, 2, 3).

Another analysis done was on the optimal number of PCA components for a pipeline. For a fixed setting, using SVM classifier and same domain evaluation, varying the number of PCA components used by the model yields the below results:

PCA Components	**Accuracy**
2	0.803
4	0.833

8	0.873
16	0.929
64	0.929
128	0.929
512	0.926

After a certain point, the number of components captures all the required information and increasing the number of dimensions provides no extra information to the model to improve performance. 16 dimensions are sufficient as per this analysis.

7. Conclusion and Future Works

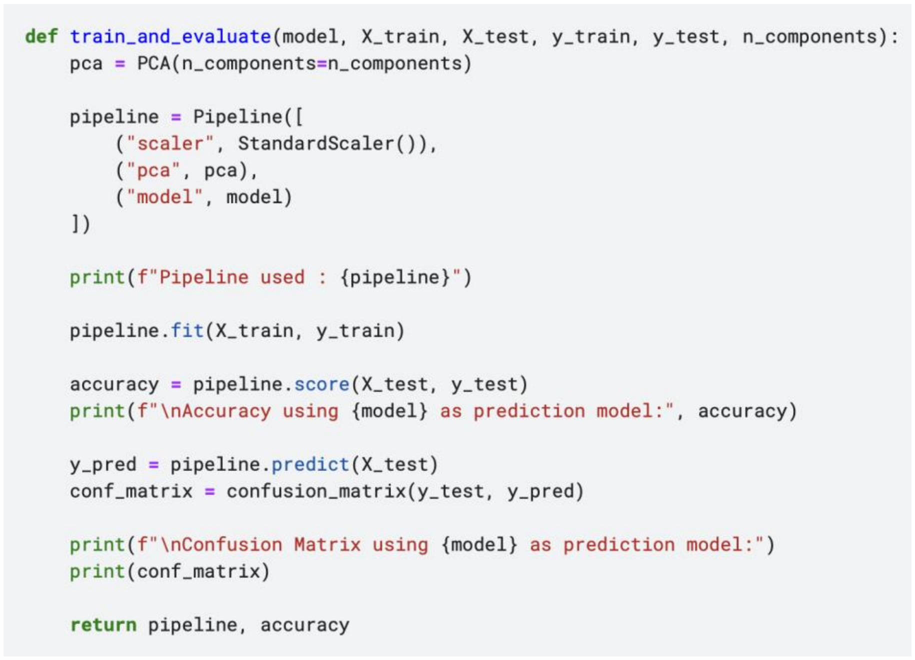
This project proposes a comprehensive framework for state prediction and fault detection in Continuous Stirred Tank Reactors (CSTRs), leveraging advanced AI/ML techniques to address inherent challenges in these systems. By developing predictive state estimation models based on neural networks, the project enables accurate forecasting of CSTR behavior, facilitating better control and optimization of process parameters. Moreover, the integration of fault detection and classification algorithms enhances operational reliability by enabling proactive identification and mitigation of faults and abnormalities. The significance of this work lies in its potential to revolutionize CSTR operations, leading to enhanced efficiency, safety, and cost savings for industries across various sectors. Through innovation in computational techniques and real-world applicability, this project represents a significant step forward in the realm of process automation and control. Overall, the project underscores the transformative power of AI/ML in addressing complex industrial challenges and driving continuous improvement in operational performance.

8. References

- O. Levenspiel, Chemical Reaction Engineering, 3rd Edition, Wiley Eastern, 2006.
- H. S. Fogler, Elements of Chemical Reaction Engineering, 4th Edition, Prentice Hall, 2013.
- Piyush Shrivastava, August 2012, Modeling and Control of CSTR using Model based Neural Network Predictive Control, Cornell University.
- Montezuma, E., 2021. Cross-Domain Fault Diagnosis through Optimal Transport. Bachelor dissertation. Universiade Federal do Ceará.

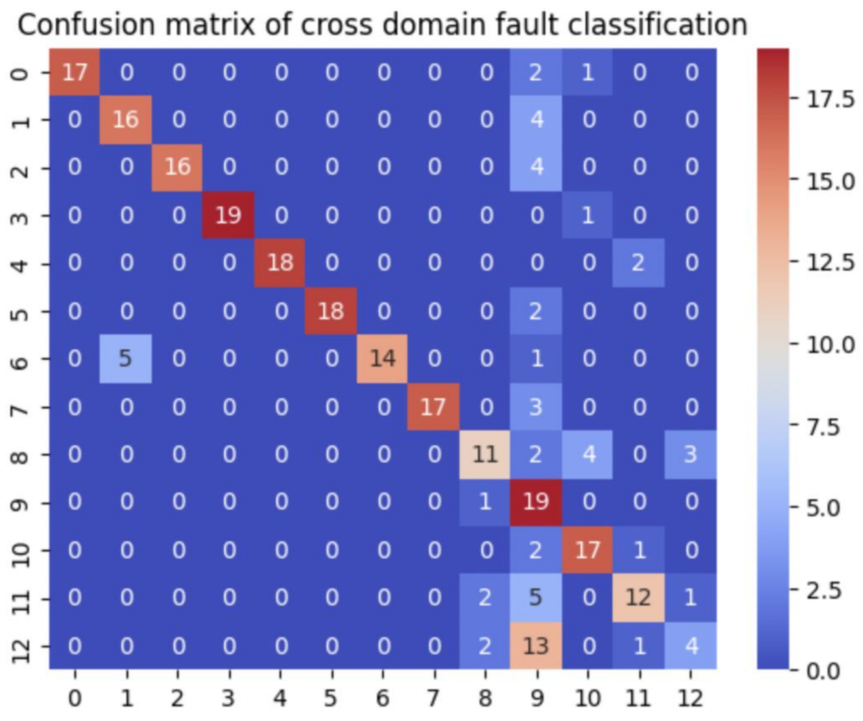
9. Appendices

The below code snippet shows the training & testing function used for fault detection and classification. A sklearn pipeline is used to carry out the steps of scaling the data and then applying PCA to extract features for the model



```python
def train_and_evaluate(model, X_train, X_test, y_train, y_test, n_components):
    pca = PCA(n_components=n_components)

    pipeline = Pipeline([
        ("scaler", StandardScaler()),
        ("pca", pca),
        ("model", model)
    ])

    print(f"Pipeline used : {pipeline}")

    pipeline.fit(X_train, y_train)

    accuracy = pipeline.score(X_test, y_test)
    print(f"\nAccuracy using {model} as prediction model:", accuracy)

    y_pred = pipeline.predict(X_test)
    conf_matrix = confusion_matrix(y_test, y_pred)

    print(f"\nConfusion Matrix using {model} as prediction model:")
    print(conf_matrix)

    return pipeline, accuracy
```

Below is a confusion matrix that shows the performance of our model for fault detection and classification explaining which classes are more prone to error and what pairs of classes are hard for the model to differentiate.



	0	1	2	3	4	5	6	7	8	9	10	11	12
0	17	0	0	0	0	0	0	0	0	2	1	0	0
1	0	16	0	0	0	0	0	0	0	4	0	0	0
2	0	0	16	0	0	0	0	0	0	4	0	0	0
3	0	0	0	19	0	0	0	0	0	0	1	0	0
4	0	0	0	0	18	0	0	0	0	0	0	2	0
5	0	0	0	0	0	18	0	0	0	2	0	0	0
6	0	5	0	0	0	0	14	0	0	1	0	0	0
7	0	0	0	0	0	0	0	17	0	3	0	0	0
8	0	0	0	0	0	0	0	0	11	2	4	0	3
9	0	0	0	0	0	0	0	0	1	19	0	0	0
10	0	0	0	0	0	0	0	0	0	2	17	1	0
11	0	0	0	0	0	0	0	0	2	5	0	12	1
12	0	0	0	0	0	0	0	0	2	13	0	1	4

10. Auxiliaries

Data Source: https://www.kaggle.com/datasets/eddardd/continuous-stirred-tank-reactor-domain-adaptation/data

Python file:
https://colab.research.google.com/drive/1F7xsbuQhdBI1Pr1RlwzMVLAZUUNNvJnI#scrollTo=ozgagpQpNHV1