Int. J. Nonlinear Anal. Appl. In Press, 1–10
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2024.33191.4941

The application of machine learning algorithms with a fuzzy approach to investigating faults in the control system of consumable water resources

Majid Rahi[a], Ali Ebrahimnejad[b,*], Homayun Motameni[c]

[a]*Department of Computer Engineering, Babol Branch, Islamic Azad University, Babol, Iran*
[b]*Department of Mathematics, Qaemshahr Branch, Islamic Azad University, Qaemshahr, Iran*
[c]*Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran*

(Communicated by Seyed Hossein Siadati)

Abstract

Nowadays, proper management of water resources is essential due to the scarcity of freshwater resources and the high cost of wastewater treatment. To control and optimize while avoiding human mistakes, it is necessary to design and implement an intelligent automated system to minimize human intervention. On the other hand, inevitable deficiencies in system equipment require fault detection and localization methods, all of which involve time and cost. Nevertheless, these costs can be reduced by examining faults in the design phase before entering the implementation phase. This article uses a simulated sample system to demonstrate the method's effectiveness. In this way, system faults in the design phase are predicted through machine learning methods. The system's tolerance for dealing with them is evaluated using fuzzy approaches. The proposed approach consists of a process-oriented framework comprising offline and online phases.

Keywords: fault Tolerance, Decision Tree, Fuzzy Sets, Water Resources Management System
2020 MSC: 68T05, 68T01

1 Introduction

In today's world, given the limited nature of water resources, intelligent management methods are paramount. In this regard, automated water consumption control has been highlighted for efficiency and targeted utilization. As a fundamental basis for systematic consumption and storage, there is a need for physical equipment such as liquid-level sensors, pumps, controllable valves, reservoirs, and control software systems to receive input data from sensors, process them, and consequently issue commands to open or close valves. Therefore, faults, or defects in these equipments, are inevitable. If these faults can be predicted and evaluated during the system design phase, the cost is much lower than when the system is physically implemented. Predicting faults in the design phase provides a clear picture.

*Corresponding author
Email addresses: majid_rh_2006@yahoo.com (Majid Rahi), a.ebrahimnejad@qaemiau.ac.ir (Ali Ebrahimnejad), motameni@iausari.ac.ir (Homayun Motameni)

Received: February 2024 *Accepted:* April 2024

However, machines' expected stable performance diminishes due to inappropriate functionality and challenging environments. Therefore, there is a continuous effort to develop Fault Detection and Diagnosis (FDD) techniques to enable continuous and reliable system health monitoring during operations and daily maintenance [11].

Since faults are inherent in systems, fault detection and localization methods have been employed to prevent system failures and breakdowns. This aims to enhance system reliability. Predicting potential faults before constructing a system is challenging. This may lead to a system with a high failure rate and low reliability. Therefore, presenting a solution in the design phase to predict and estimate the faults of the intended system is crucial to prevent the construction of a system with high risks.

With the development of Artificial Intelligence (AI), extensive research has been conducted on rapid and efficient fault detection based on intelligent methods [17]. Generally, the system's condition can be categorized as normal or faulty based on input data. Therefore, one of the tasks of the proposed solution is to detect the system's current status (normal or faulty) so that it can either proceed with its routine under typical conditions or halt in the presence of a fault. However, if the system is halted by detecting any fault and becomes inaccessible, its reliability is compromised, reducing its efficiency. Fuzzy methods and predefined threshold criteria evaluate system faults.

Based on such motivation, this study presents a comprehensive, process-oriented, and integrated approach that includes prediction, fault classification focusing on a decision tree, and fault tolerance assessment based on ANFIS during the system design phase. By doing so, a system with high failure rates is prevented, and its reliability is enhanced.

The continuation of the article is organized as follows: Section 2 outlines previous works. Section 3 defines fundamental concepts (related to Petri net and machine learning algorithms, including decision tree C5.0, CHAID, QUEST, SVM, and adaptive neuro-fuzzy inference systems). Section 4 discusses the proposed method. Section 5 concludes and suggests future research directions.

2 Preliminaries

In [18], an approach was proposed for evaluating three-reservoir system performance using hybrid Petri net. The model was examined based on structural features and constraint information for liquid-level processes. In [5] the proposed method, the flexible job-shop scheduling problem systems was modeled by color Petri net and CPN tool and then a scheduled job was programmed by GSA algorithm. In [1, 2, 3] Colored petri net (CPN) was used for modeling to predict different modes of traffic management system and probable occurrences. In [4] the methodology of firewalled LAN models construction in the form of Colored Petri Net was introduced and model the ACL (Access Control List) in firewalls. In [16], an architecture was introduced and described for showing the mobile agent request process in mobile environments. Hence, a model has been suggested using Stochastic Colored Petri Nets that encompasses two sections. In [15], a machine-learning framework for predicting software faults was suggested. Seven classifiers were compared: k-Nearest Neighbors, Naive Bayes, Linear Discriminant Analysis, Linear Regression, Decision Tree, Support Vector Machine, and Random Forest. In [20], a High Impedance Fault (HIF) detection method based on Decision Trees (DTs) is presented. HIF features, serving as inputs to the DTs, include known indicators such as root mean square (RMS), current, second, third, and fifth harmonic magnitudes, and third harmonic phase.

In [10], a comprehensive review of various machine learning methods is presented, including a Naive Bayes classifier, Decision Tree, Random Forest, k-Nearest Neighbors, Support Vector Machine, as well as artificial neural networks such as Feedforward Neural Network, Convolutional Neural Network, and Adaptive Neuro-Fuzzy Inference System. These methods are used to identify, classify, and localize faults on transmission lines. In [19], a survey of recent machine learning-based techniques for fault detection, classification, and location estimation on transmission lines is provided. This article offers a comprehensive overview of various machine learning methods, including Naive Bayes classifier, Decision Tree, Random Forest, k-Nearest Neighbors, Support Vector Machine, as well as artificial neural networks such as Feedforward Neural Network, Convolutional Neural Network, and Adaptive Neuro-Fuzzy Inference System, used for the identification, classification, and localization of faults in transmission lines. In [23], the problem of Fault Tolerant Control (FTC) is studied, specifically, an observer-based fully distributed event-triggered adaptive fuzzy FTC for nonlinear Multi-Agent Systems (MASs) exposed to unknown actuator faults.

An innovative mechanism for compensating with a reconfigurable operator is adopted to cope with unknown actuator faults. Then, based on the backstepping technique and the design of a dual-channel event-triggered mechanism, a fault-tolerant approach that relies on fully distributed event-triggered development is devised. A fault-tolerant mechanism is introduced based on a Multi-Class Support Vector Machine (SVM) and an Adaptive Neuro-Fuzzy Inference System (ANFIS). Multi-class SVM identifies and locates system fault states, and ANFIS estimates real sensor fault data. After estimation, the system compares the real sensor fault data with the estimated ANFIS data [22].

3 Definitions and Basic Concepts

This section explains the fundamental concepts of the topics discussed in the following sections. Basic definitions of Petri net, decision trees, and ANFIS are provided.

3.1 Petri net

Below are some concepts and properties of colored Petri net. Further details can be found in [8] and [9].

Definition 3.1. A marked discrete Petri net denoted by the quintuple $R = (P, T, Pre, Post, m_0)$ is defined as follows:

$P = \{P_1, P_2, ...P_n\}$ is a finite and non-empty set of places (represented by circles).

$T = \{T1, T2, ..., Tm\}$ is a finite and non-empty set of transitions (represented by rectangular boxes).

$P \cap T = \emptyset$, meaning that P and T are disjoint.

$Pre : P \times T \to \{0, 1\}$ is the input incidence matrix

$Post : P \times T \to \{0, 1\}$ is the output incidence matrix.

$m_0 : Pi \to Z^{\alpha}$ is the initial marking of all places, where Z^+ corresponds to the set of non-negative integers, and $\to$ indicates the assignment of P_i to Z^+.

3.2 Decision Trees

Decision trees are among the best-studied and widely used tools in machine learning and data science. Decision tree capabilities are one of the most practical methods extensively applied in data mining, particularly anomaly detection [6, 7]. Data object classification is a data mining and knowledge management technique for grouping similar data objects. While various classification algorithms exist in the literature, decision trees are the most common due to their ease of implementation and straightforward understanding compared to other classification algorithms [21].

Decision trees (DT) are commonly used for identifying and classifying faults. DT models trained on datasets have demonstrated high fault detection and fault classification accuracy in experimental sets [24].

3.3 Adaptive Neuro-Fuzzy Inference System (ANFIS)

The Adaptive Neuro-Fuzzy Inference System (ANFIS) is an artificial neural network built upon the Takagi-Sugeno fuzzy inference system. This technique was developed in the early 1990s [12]. Since ANFIS combines both neural network capabilities and fuzzy logic principles, it can simultaneously utilize both. Its inference system employs if-then fuzzy rules to learn nonlinear functions' approximation. Therefore, ANFIS is considered a universal approximator [13]. In the ANFIS structure, two distinct parameter groups exist: antecedent and consequent. Training ANFIS involves determining these parameters using optimization algorithms. Jang's initial ANFIS model proposed a hybrid learning approach. ANFIS utilizes input-output pairs during training [14].

4 Proposed Methodology

In this section, the proposed approach is detailed. The approach consists of two phases: offline and online. In the offline phase, a model of the studied system is created using stochastic hybrid Petri nets (SHPN). The dataset is extracted during the system monitoring using a simulator. These datasets are labeled for training machine learning algorithms in three stages (as per Section 4.1.1). In the next stage, the labeled datasets train machine learning algorithms for fault detection, decision tree-based fault classification, and fault tolerance assessment based on ANFIS.

In the online phase, the trained models from the offline phase are utilized to process the streaming of real-world data. Initially, the data (as the system states) is fed into the fault detection model based on the decision tree.

If this state is predicted as a fault, it enters the fault classification model based on the decision tree to identify the location of the fault. Finally, the classified fault is input into the fault tolerance assessment model based on ANFIS. This model predicts the fuzzy membership degree of fault tolerance within zero to one. A value of one represents high fault tolerance, while a value of zero indicates low fault tolerance. If the system state is normal, the system resumes normal operation. To illustrate the effectiveness of the proposed solution, a case study related to the water resource management domain has been utilized. The details of the proposed solution are discussed in the following sections.

Offline Phase

4.1 Simulation, Data Extraction, and Preprocessing

This section illustrates the studied system in Figure (1) and its simulation as a stochastic hybrid Petri net in Figure (2).

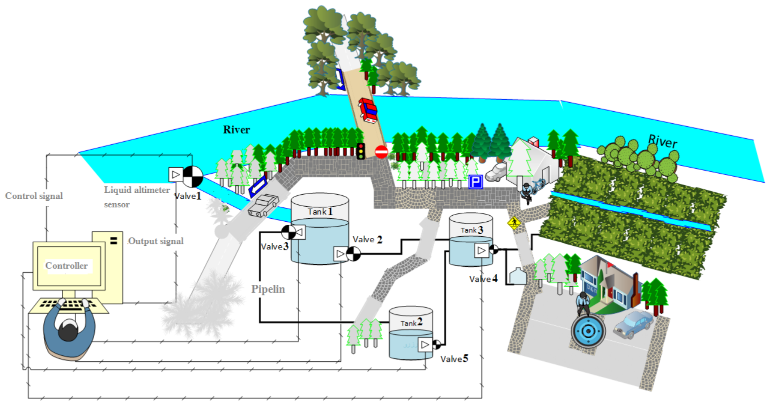


Figure 1: The studied system.

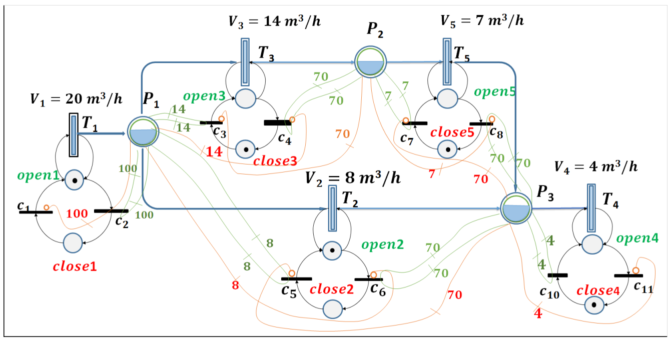
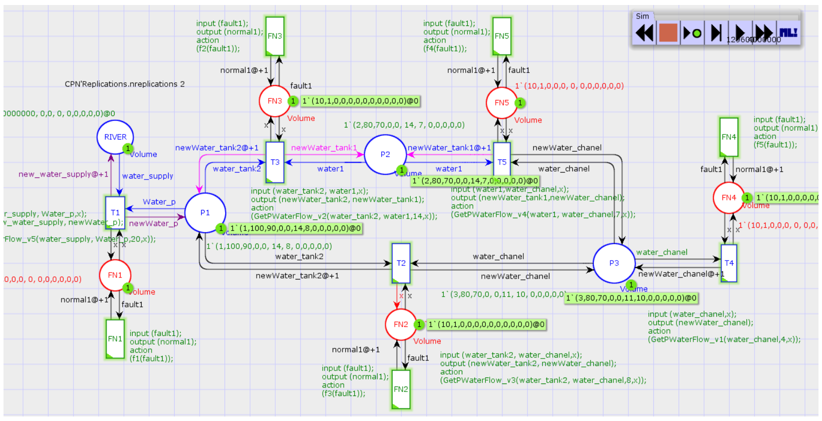


Figure 2: Simulated system equivalent to Figure (1) using Petri net.



```
input (fault1);
output (normal1);
action
(f2(fault1));

input (fault1);
output (normal1);
action
(f4(fault1));

input (fault1);
output (normal1)
action
(f5(fault1));

input (fault1);
output (normal1);
action
(f1(fault1));

input (fault1);
output (normal1);
action
(f3(fault1));

r_supply, Water_p,x);
v_water_supply, newWater_p);
Flow_v5(water_supply, Water_p,20,x));

input (water_tank2, water1,x);
output (newWater_tank2, newWater_tank1);
action
(GetPWaterFlow_v2(water_tank2, water1,14,x));

input (water1,water_chanel,x);
output (newWater_tank1,newWater_chanel);
action
(GetPWaterFlow_v4(water1, water_chanel,7,x));

input (water_chanel,x);
output (newWater_chanel);
action
(GetPWaterFlow_v1(water_chanel,4,x));

input (water_tank2, water_chanel,x);
output (newWater_tank2, newWater_chanel);
action
(GetPWaterFlow_v3(water_tank2, water_chanel,8,x));

0000000, 0,0, 0, 0,0,0,0,0)@0
0,0,0, 0, 0,0,0,0,0,0)
1`(10,1,0,0,0,0,0,0,0,0,0,0)@0
1`(2,80,70,0,0, 14, 7, 0,0,0,0,0)
1`(10,1,0,0,0, 0, 0,0,0,0,0,0)
1`(2,80,70,0,0,14,7,0,0,0,0,0)@0
1`(10,1,0,0,0,0,
1`(10,1,0,0,0, 0, 0,0,
1`(1,100,90,0,0,14,8,0,0,0,0,0)@0
1`(1,100,90,0,0, 14, 8, 0,0,0,0,0)
1`(3,80,70,0, 0,11, 10, 0,0,0,0,0)
1`(3,80,70,0,0,11,10,0,0,0,0,0)@0
1`(10,1,0,0,0,0,0,0,0,0,0,0)@0
```

Figure 3: The simulated studied system in CPNTools.

Details related to the implementation of the studied system are presented in Tables (1) to (4).

Table 1: Continuous Transitions and Places.

#	Equivalent in petri net	Equivalent to Figure 1	(volume/fluid output speed)
1	P_1	Tank1	100 m^3
2	P_2	Tank2	80 m^3
3	P_3	Tank3	80 m^3
4	T_1	Valve1	20 m^3/h
5	T_1	Valve2	8 m^3/h
6	T_1	Valve3	14 m^3/h
7	T_1	Valve4	4 m^3/h
8	T_1	Valve5	7 m^3/h

Table 2: Conditions of Valve Open or Closed (Continuous Transitions).

#	**Valve**	**Closed conditions**	**Open conditions**
1	Valve1	The liquid volume of Tank 1 should be less than $100m^3$	The liquid volume of Tank1 should be more than 100 m^3
2	Valve2	The liquid volume of Tank3 and Tank1 should be less and more than $70m^3$ and $8m^3$ respectively	The liquid volume of Tank3 and Tank1 should be more and less than $70m^3$ and $8m^3$ respectively
3	Valve3	The liquid volume of Tank2 and Tank1 should be less and more than $70m^3$ and $14m^3$ respectively	The Liquid jam Tank2 and Tank1 should be more and less than $70m^3$ and $14m^3$ respectively
4	Valve4	The liquid volume of Tank 3 should be less than $4m^3$	The liquid volume of Tank 3 should be more than $4m^3$
5	Valve5	The liquid volume of Tank3 and Tank2 should be less and more than $70m^3$ and $7m^3$ respectively	The liquid volume of Tank3 and Tank2 should be more and less than $70m^3$ and $7m^3$ respectively

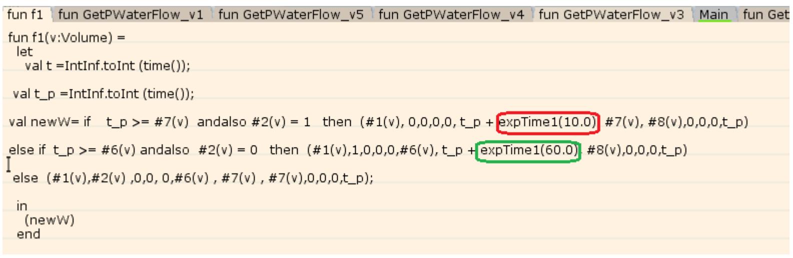
Table 3: Discrete Transitions and Places.

#	specification	Control condition associated with the continuous transitions
1	$Open_1, Close_1$ c_2, c_2	T_1
2	$Open_2, Close_2$ c_6, c_5	T_2
3	$Open_3, Close_3$ c_4, c_3	T_3
4	$Open_4, Close_4$ c_{11}, c_{10}	T_4
5	$Open_5, Close_5$ c_8, c_7	T_5

For transitions T_1 to T_5, five fault functions fT_1 to fT_5 can be considered sequentially, assuming the asynchrony of faults (according to Figure (5)). For example, in the normal state, in a time interval of 60 hours, equivalent to 2.5 days (consistent with the time variable α_2 with a random exponential distribution), the instantaneous speed of T1 is considered constant at $20m^3/h$ (according to Figure (5)(a)). After this period elapses, transition T_1 incurs the fault fT_1, and the system enters the fault state (according to Figure (5)(b)). In this case, a time interval of 10 hours

Table 4: Conditions of Valve Open or Closed (Continuous Transitions).

#	specification	description
1	$Normal_1, fault_1$ n_1, f_1	Discrete places and transitions with exponential time distribution of 10 and 60 hours, respectively, to simulate the normal and fault states for T1
2	$Normal_2, fault_2$ n_2, f_2	Discrete places and transitions with exponential time distribution of 10 and 60 hours, respectively, to simulate the normal and fault states for T2
3	$Normal_3, fault_3$ n_3, f_3	Discrete places and transitions with exponential time distribution of 10 and 60 hours, respectively, to simulate the normal and fault states for T3
4	$Normal_4, fault_4$ n_4, f_4	Discrete places and transitions with exponential time distribution of 10 and 60 hours, respectively, to simulate the normal and fault states for T4
5	$Normal_5, fault_5$ n_5, f_5	Discrete places and transitions with exponential time distribution of 10 and 60 hours, respectively, to simulate the normal and fault states for T5



```
fun f1(v:Volume) =
 let
   val t =IntInf.toInt (time());

 val t_p =IntInf.toInt (time());

val newW= if   t_p >= #7(v)  andalso #2(v) = 1  then  (#1(v), 0,0,0,0, t_p + expTime1(10.0) #7(v), #8(v),0,0,0,t_p)

else if  t_p >= #6(v) andalso  #2(v) = 0  then  (#1(v),1,0,0,0,#6(v), t_p + expTime1(60.0) #8(v),0,0,0,t_p)

 else  (#1(v),#2(v) ,0,0, 0,#6(v) , #7(v) , #7(v),0,0,0,t_p);

 in
   (newW)
 end
```

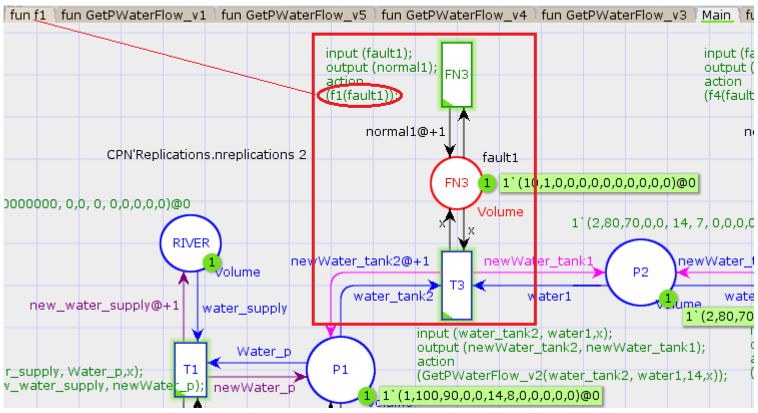


Figure 4: (Upper figure) Function F1 for simulating fault states (red rectangle represents an exponential function with a time variable of 10 hours) and normal states (green rectangle represents an exponential function with a time variable of 60 hours). (Bottom figure) Function F1 associated with each continuous transition.

(consistent with the time variable α_1 with a random exponential distribution) is considered. Therefore, the output speed of T_1 becomes zero (after this period elapses, the system returns to the normal state). The same assumptions apply to the other transitions.

According to Figure (4), the function F_1 simulates the fault of each transition according to the above definition.

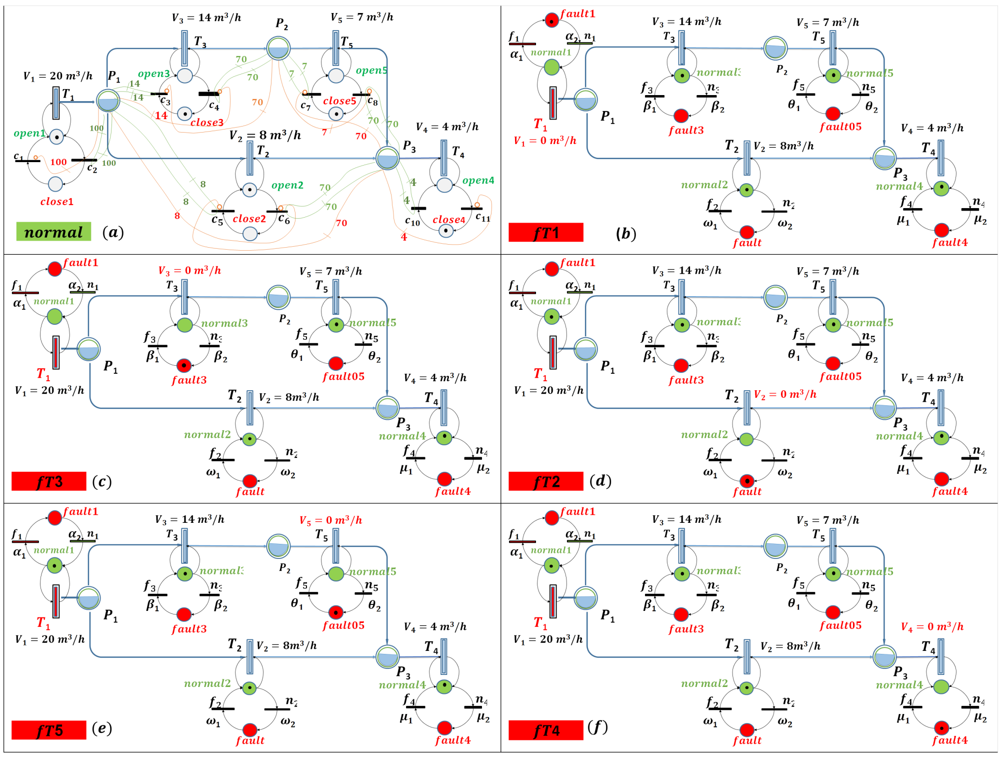


Figure 5: Various fault states in the five fault classes: fT_1 to fT_5 and the normal state.

4.1.1 Explanation of Concepts: Fault Detection, Fault Classification, and Fault Tolerance Evaluation

Fault Detection Concepts: For transition T_i, in this case, if the liquid volumes P_i and P_{i+1} are respectively greater and less than V_i and $70m^3$, and $V_i = 0$, then this state is considered an fault; otherwise, the system is in a normal state.

$\exists P_i \mid P_i \subset \{P_1, P_2, \dots, P_n\}, i = 1; i, j \subset Z^+$

$\exists T_j \mid T_j \subset \{T_0, T_1, \dots, T_n\}, j = 0,$

$So:\ if\ (V_i = 0\, and\, m(P_i) > V_i\, and\, m(P_{i+1}) < 70m^3)\, then\, the\, state\, is\, fault\, else\, normal.$

Classification Concepts for Fault fT_i: For transition T_i, if the liquid volumes P_i and P_{i+1} are respectively greater and less than V_i and $70m^3$, and $V_i = 0$, then this state is considered a fault fTi; otherwise, the system is in a normal state. If the following conditions are met, the fault is classified as a transition fault fT_i.

$\exists P_i \mid P_i \subset \{P_1, P_2, \dots, P_n\}, 1 \leq i < n; i, j \subset Z^+$

$\exists T_j \mid T_j \subset \{T_0, T_1, \dots, T_n\}, 0 < j < n, i = j,$

$So: if(V_i = 0\, and\, m(P_i) > V_i\, and\, m(P_{i+1}) < 70m^3)\, then\, fT_i\, else\, normal.$

Tolerance Concepts for Fault fT_i: For transition T_i, if the liquid volume P_i is greater than V_i and the volume P_{i+1} is between $50m^3$ and $70m^3$, and $V_i = 0$, then the fault is considered tolerable; otherwise, it is considered intolerable, and the system must respond to it.

$\exists P_i \mid P_i \subset \{P_1, P_2, \dots, P_n\}, 1 \leq i < n; i, j \subset Z^+$

$\exists T_j \mid T_j \subset \{T_0, T_1, \dots, T_n\}, 0 < j < n, i = j,$

$So:\ if(V_i = 0\, and\, m(P_i) > V_i\, and\, 50m^3 < m(P_{i+1}) < 70m^3)\, then$

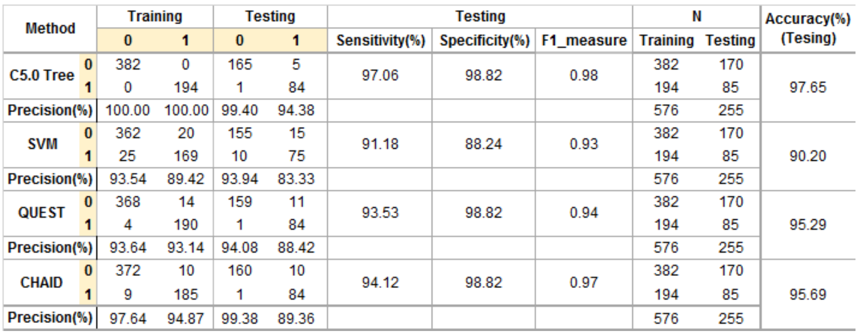
$the\, fault\, is\, tolerable\, else\, fault\, is\, intolerable.$

4.1.2 Decision Tree Training

In this section, considering the fault detection concept presented in 4.1.1, decision tree algorithms (C5.0, CHAID, and QUEST) and SVM have been applied to 576 samples from the tested dataset. The decision tree algorithms, including nodes (C5.0, CHAID, and QUEST) and SVM nodes, were implemented in Clementine 12.0 software.

4.1.3 Numerical Results for a Fault Detection Model

Among the trained models, the designed C5.0 model accurately identified 97.65% of the samples in the test dataset. However, the developed models CHAID and QUEST failed to classify 11 and 12 samples in the test dataset correctly. The developed SVM model with the RBF kernel achieved 90.20% prediction accuracy in the test dataset. Details of the identified faults and the accuracy values of the developed models using decision tree methods are provided in Table (5). Table (5) results indicate that the parameters (Sensitivity, F1_measure, and Accuracy) in the proposed method using the C5.0 decision tree outperform other machine learning methods in the case study.



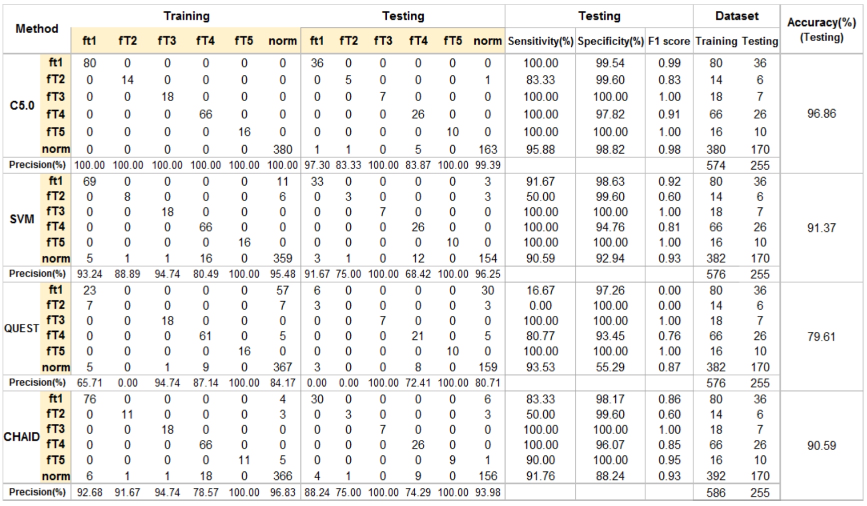
Method		Training		Testing		Testing			N		Accuracy(%) (Tesing)
		0	1	0	1	Sensitivity(%)	Specificity(%)	F1_measure	Training	Testing	
C5.0 Tree	0	382	0	165	5	97.06	98.82	0.98	382	170	97.65
	1	0	194	1	84				194	85	
Precision(%)		100.00	100.00	99.40	94.38				576	255	
SVM	0	362	20	155	15	91.18	88.24	0.93	382	170	90.20
	1	25	169	10	75				194	85	
Precision(%)		93.54	89.42	93.94	83.33				576	255	
QUEST	0	368	14	159	11	93.53	98.82	0.94	382	170	95.29
	1	4	190	1	84				194	85	
Precision(%)		93.64	93.14	94.08	88.42				576	255	
CHAID	0	372	10	160	10	94.12	98.82	0.97	382	170	95.69
	1	9	185	1	84				194	85	
Precision(%)		97.64	94.87	99.38	89.36				576	255	

Table 5: Comparison of Sensitivity, F1_measure, and Accuracy metrics for machine learning methods in fault detection.

4.1.4 Numerical Results for a Fault Classification Model

Among the trained models, the created model using C5.0 accurately classified 96.86% of the samples in the test dataset. However, the developed CHAID and QUEST models failed to classify 24 and 49 samples in the test dataset correctly. The developed SVM model with the RBF kernel achieved 91.37% prediction accuracy in the test dataset. Details of the identified faults and the accuracy values of the developed models using decision tree methods are provided in Table (6). The results in this table indicate that the parameters (Sensitivity, F1_measure, and Accuracy) in the proposed approach using the C5.0 decision tree outperform other machine learning methods in the case study.

Table 6: Comparison of Sensitivity, F1_measure, and Accuracy metrics for machine learning methods in fault classification.



Method		Training						Testing						Testing			Dataset		Accuracy(%) (Testing)
		ft1	fT2	fT3	fT4	fT5	norm	ft1	fT2	fT3	fT4	fT5	norm	Sensitivity(%)	Specificity(%)	F1 score	Training	Testing	
C5.0	ft1	80	0	0	0	0	0	36	0	0	0	0	0	100.00	99.54	0.99	80	36	96.86
	fT2	0	14	0	0	0	0	0	5	0	0	0	1	83.33	99.60	0.83	14	6	
	fT3	0	0	18	0	0	0	0	0	7	0	0	0	100.00	100.00	1.00	18	7	
	fT4	0	0	0	66	0	0	0	0	0	26	0	0	100.00	97.82	0.91	66	26	
	fT5	0	0	0	0	16	0	0	0	0	0	10	0	100.00	100.00	1.00	16	10	
	norm	0	0	0	0	0	380	1	1	0	5	0	163	95.88	98.82	0.98	380	170	
Precision(%)		100.00	100.00	100.00	100.00	100.00	100.00	97.30	83.33	100.00	83.87	100.00	99.39				574	255	
SVM	ft1	69	0	0	0	0	11	33	0	0	0	0	3	91.67	98.63	0.92	80	36	91.37
	fT2	0	8	0	0	0	6	0	3	0	0	0	3	50.00	99.60	0.60	14	6	
	fT3	0	0	18	0	0	0	0	0	7	0	0	0	100.00	100.00	1.00	18	7	
	fT4	0	0	0	66	0	0	0	0	0	26	0	0	100.00	94.76	0.81	66	26	
	fT5	0	0	0	0	16	0	0	0	0	0	10	0	100.00	100.00	1.00	16	10	
	norm	5	1	1	16	0	359	3	1	0	12	0	154	90.59	92.94	0.93	382	170	
Precision(%)		93.24	88.89	94.74	80.49	100.00	95.48	91.67	75.00	100.00	68.42	100.00	96.25				576	255	
QUEST	ft1	23	0	0	0	0	57	6	0	0	0	0	30	16.67	97.26	0.00	80	36	79.61
	fT2	7	0	0	0	0	7	3	0	0	0	0	3	0.00	100.00	0.00	14	6	
	fT3	0	0	18	0	0	0	0	0	7	0	0	0	100.00	100.00	1.00	18	7	
	fT4	0	0	0	61	0	5	0	0	0	21	0	5	80.77	93.45	0.76	66	26	
	fT5	0	0	0	0	16	0	0	0	0	0	10	0	100.00	100.00	1.00	16	10	
	norm	5	0	1	9	0	367	3	0	0	8	0	159	93.53	55.29	0.87	382	170	
Precision(%)		65.71	0.00	94.74	87.14	100.00	84.17	0.00	0.00	100.00	72.41	100.00	80.71				576	255	
CHAID	ft1	76	0	0	0	0	4	30	0	0	0	0	6	83.33	98.17	0.86	80	36	90.59
	fT2	0	11	0	0	0	3	0	3	0	0	0	3	50.00	99.60	0.60	14	6	
	fT3	0	0	18	0	0	0	0	0	7	0	0	0	100.00	100.00	1.00	18	7	
	fT4	0	0	0	66	0	0	0	0	0	26	0	0	100.00	96.07	0.85	66	26	
	fT5	0	0	0	0	11	5	0	0	0	0	9	1	90.00	100.00	0.95	16	10	
	norm	6	1	1	18	0	366	4	1	0	9	0	156	91.76	88.24	0.93	392	170	
Precision(%)		92.68	91.67	94.74	78.57	100.00	96.83	88.24	75.00	100.00	74.29	100.00	93.98				586	255	

4.2 Training ANFIS for Fault Tolerance Assessment

In this section, considering the introduced concept of fault tolerance in 4.1.1, the ANFIS model has been trained on the fault states of the dataset, which includes 258 training data samples and 118 test data samples. The training process is performed using MATLAB software. For system states identified as faults by the fault detection tree and following the concept mentioned in 4.1.1, if the tolerance conditions for the identified faults are met, the fault is characterized at three fuzzy levels of tolerance: high, medium, and low. Table (7) presents the features of the neural-fuzzy model used in the proposed approach.

Table 7: Features of a neural fuzzy network.

Variable	Name
Number of input	4
Number of membership function for each input	3, 2, 2, 2
Epoch number	3000
Learning data set	258
testing data set	118
Membership function type	gbellmf

4.2.1 Numerical Results for the Fault Tolerance Assessment Model

The model demonstrates an average test fault of 0.16. Considering the well-matched predicted results of ANFIS, it is observed that the expected output has been accurately obtained (according to Figure (6)).

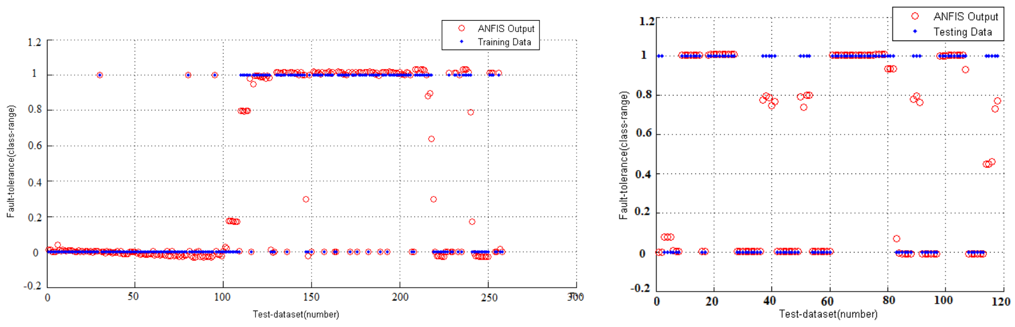


Figure 6: (a) Comparison between test data and ANFIS output. (b) Comparison between training data and ANFIS output.

5 Conclusion

This article focuses on predicting and locating faults and evaluates the tolerance level of identified faults. This is at the stages preceding the actual system construction and implementation. To achieve this, the decision tree algorithm C5.0 is modeled for fault detection on the dataset obtained from the simulated system. This provides a high accuracy estimate compared to other machine learning methods. Ultimately, the modeled ANFIS is used to assess identified fault tolerance. The results demonstrate detection efficiency and fault tolerance evaluation. In future studies, an attempt will be made to train the proposed hybrid method through online rather than offline learning. Unlike the offline method, which requires an offline dataset, the dataset is continuously updated in real-time states in the online method and is used for training at specified intervals. The advantage of this approach over the proposed method in this study will be better adaptability to various system states and more accurate predictions.

References

[1] B. Barzegar, *Fuzzy logic controller for traffic signal controller unit system and modeling with colored petri net*, Indian J. Sci. Technol. **4** (2011), 1420–1428.

[2] B. Barzegar, S. Ghanbari, H. Bozorgi, and M. Rahmani, *Modeling and simulation of traffic lights and controller unit systems by colored Petri net*, Int. J. Phys. Sci. **6** (2011), 7760–7770.

[3] B. Barzegar, M. Mehrabanian, and S. Bandegan, *Fuzzy logic for a traffic signal control with colored Petri net*, Aust. J. Basic Appl. Sci. **5** (2011), 2961–2964.

[4] B. Barzegar and H. Motameni, *Modeling and simulation firewall using colored Petri net*, World Appl. Sci. J. **15** (2011), 826–830.

[5] B. Barzegar, H. Motameni, and H. Bozorgi, *Solving flexible job-shop scheduling problem using gravitational search algorithm and colored Petri net*, J. Appl. Math. **2012** (2012).

[6] V.G. Costa and C.E. Pedreira, *Recent advances in decision trees: An updated survey*, Artific. Intel. Rev. **56** (2023), 4765–4800.

[7] R.B. Damala, R.K. Patnaik, and A.R. Dash, *A simple decision tree-based disturbance monitoring system for VSC-based HVDC transmission link integrating a DFIG wind farm*, Protect. Control Mod. Power Syst. **7** (2022), 1–19.

[8] R. David and H. Alla, *Discrete, Continuous, and Hybrid Petri Nets*, Springer, 2010.

[9] E. Dubois and H. Alla, *Hybrid Petri nets with a stochastic discrete part*, Proc. 2nd ECC Eur. Control Conf., 1993, pp. 144–149.

[10] C.-L. Fan, *Data mining model for predicting the quality level and classification of construction projects*, J. Intel. Fuzzy Syst. **42** (2022), no. 1, 139–153.

[11] Z. Hou, C. Lee, Y. Lv, and K. Keung, *Fault detection and diagnosis of air brake system: A systematic review*, J. Manufact. Syst. **7** (2023), 34–58.

[12] J.-S. Jang, *ANFIS: Adaptive-network-based fuzzy inference system*, IEEE Trans. Syst. Man Cybernet. **23** (1993),665-685.

[13] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, *Neuro-fuzzy and soft computing-a computational approach to learning and machine intelligence [Book Review]*, IEEE Trans. Automatic Control **42** (1997), 1482–1484.

[14] D. Karaboga and E. Kaya, *Adaptive network based fuzzy inference system (ANFIS) training approaches: A comprehensive survey*, Artific. Intel. Rev. **52** (2019), 2263–2293.

[15] M. Mafarja, T. Thaher, M.A. Al-Betar, J. Too, M.A. Awadallah, I. Abu Doush, and H. Turabieh, *Classification framework for faulty-software using enhanced exploratory whale optimizer-based feature selection scheme and random forest ensemble learning*, Appl. Intel. **53** (2023), no. 15, 18715–18757.

[16] H. Motameni, V. Mohammad, B. Shirazi, and B. Barzegar, *Modeling and evaluation of web services in mobile networks using stochastic colored Petri nets*, J. Adv. Comput. Res. **4** (2013), no. 4, 37–50.

[17] B. Qi, J. Liang, and J. Tong, *Fault diagnosis techniques for nuclear power plants: a review from the artificial intelligence perspective*, Energies **16** (2023), no. 4, 1850.

[18] K. Renganathan and V. Bhaskar, *Modeling, analysis and performance evaluation for fault diagnosis and Fault Tolerant Control in bottle-filling plant modeled using hybrid Petri nets*, Appl. Math. Model. **37** (2013), 4842–4859.

[19] F.M. Shakiba, S.M. Azizi, M. Zhou, and A. Abusorrah, *Application of machine learning methods in fault detection and classification of power transmission lines: A survey*, Artific. Intel. Rev. **56** (2023), 5799–5836.

[20] Y. Sheng and S.M. Rovnyak, *Decision tree-based methodology for high impedance fault detection*, IEEE Trans. Power Delivery **19** (2004), 533–536.

[21] Y.-Y. Song and L. Ying, *Decision tree methods: applications for classification and prediction*, Shanghai Arch. Psych. **27** (2015), 130.

[22] J. Wang, Y. Gao, Y. Cao, T. Tang, and Y. Zhu, *The investigation of data voting algorithm for train air-braking system based on multi-classification SVM and ANFIS*, Chinese J. Electron. **24** (2024), 274–281.

[23] J. Zhang and S. Tong, *Event-triggered fuzzy adaptive output feedback containment fault-tolerant control for non-linear multi-agent systems against actuator faults*, Eur. J. Control **75** (2023), 100887.

[24] Y. Zhao, L. Yang, B. Lehman, J.-F. de Palma, J. Mosesian, and R. Lyons, *Decision tree-based fault detection and classification in solar photovoltaic arrays*, Twenty-Seventh Ann. IEEE Appl. Power Electron. Conf. Expos., 2015, pp. 93–99.