Results in Engineering 23 (2024) 102356

Contents lists available at ScienceDirect

Results in Engineering

journal homepage: www.sciencedirect.com/journal/results-in-engineering

Research paper

Predicting LAN switch failures: An integrated approach with DES and machine learning techniques (RF/LR/DT/SVM)

Ali Myrzatay [a,b], Leila Rzayeva [c,*], Stefania Bandini [d,e], Ibraheem Shayea [f,*], Bilal Saoud [g,f,*], Ilhami Çolak [h], Korhan Kayisli [i]

[a] *Computer science department, Korkyt Ata Kyzylorda University, 120000, Kyzylorda, Kazakhstan*
[b] *M. Narikbayev KAZGUU University, 010000, Astana, Kazakhstan*
[c] *Department of Intelligent Systems and Cybersecurity, Astana IT University, 010000, Astana, Kazakhstan*
[d] *Department of Informatics, Systems and Communication (DISCo), University of Milano-Bicocca, Viale Sarca 336, 20126, Milan, Italy*
[e] *RCAST - Research Center for Advanced Science and Technology, The University of Tokyo, Komaba Campus, 4-6-1 Meguro-ku, 153-8904, Tokyo, Japan*
[f] *Electronics & Communications Engineering Department, Faculty of Electrical and Electronics Engineering, Istanbul Technical University (ITU), 34469, Istanbul, Turkey*
[g] *LISEA Laboratory, Faculty of Sciences and Applied Sciences, University of Bouira, 10000, Bouira, Algeria*
[h] *Department of Electrical and Electronics engineering, Faculty of Engineering and Natural Sciences, Istinye University, Istanbul, Turkey*
[i] *Department of Electrical-Electronic Engineering, Engineering Faculty, Gazi University, 06560, Ankara, Turkey*

ARTICLE INFO

Keywords:
BPMN
Double exponential smoothing
Decision-making systems
LAN
Machine learning
Network

ABSTRACT

This research paper introduces an innovative approach to predicting failures in Local Area Network (LAN) switches, combining Double Exponential Smoothing (DES) with a suite of Machine Learning (ML) algorithms including Random Forest (RF), Logistic Regression (LR), Decision Trees (DT), and Support Vector Machines (SVM). The primary objective of this study is to enhance the accuracy and timeliness of LAN switch failure predictions, thereby facilitating more proactive and effective network management. Our methodology involves the integration of DES for trend analysis and forecasting in time-series data, with the advanced predictive capabilities of the aforementioned ML algorithms. This hybrid approach not only leverages the strengths of DES in identifying underlying patterns in failure data but also capitalizes on the diverse predictive models to handle various aspects of failure prediction more robustly. The paper details the process of data collection, preprocessing, and the specific application of DES and each ML algorithm to the dataset. A notable contribution of this research is the development of a framework that effectively combines the output of DES with ML models, leading to a significant improvement in predictive accuracy as compared to traditional methods. Through rigorous testing and validation; the proposed approach demonstrated a marked increase in the precision and reliability of failure predictions. The results indicate that the integration of DES with ML algorithms can substantially aid in preemptive maintenance and decision-making processes in LAN management. The implications of these findings are profound, suggesting that such a combined approach can greatly enhance network stability and efficiency. While the focus of this study is on LAN switches, the methodology has the potential for broader applications in various fields of network management and predictive maintenance.

1. Introduction

The rapid advancement of information technology has led to a significant expansion in the complexity and size of Local Area Networks (LANs) within enterprises. This expansion has brought forth numerous challenges in the maintenance and monitoring of network infrastructures, particularly concerning the reliability and performance of active network components such as LAN switches. This paper focuses on enhancing the prediction of LAN switch failures, a critical aspect for IT service providers who manage extensive LAN systems and are responsible for ensuring uninterrupted network services [1–3].

These service providers, equipped with various active equipment including switches, routers, hubs, and servers, face a daunting task in maintaining optimal service quality and technical performance in an

* Corresponding authors.
E-mail addresses: mirzatay_ali@korkyt.kz (A. Myrzatay), l.rzayeva@astanait.edu.kz (L. Rzayeva), stefania.bandini@unimib.it (S. Bandini), shayea@itu.edu.tr (I. Shayea), bilal340@gmail.com (B. Saoud), icolak@gazi.edu.tr (I. Çolak), korhankayisli@gazi.edu.tr (K. Kayisli).

https://doi.org/10.1016/j.rineng.2024.102356
Received 22 March 2024; Received in revised form 19 May 2024; Accepted 3 June 2024
Available online 6 June 2024

ever-growing network environment. The scale of network deployment and the increasing dependency on these networks necessitate advanced methods for predicting equipment failures to minimize downtime and ensure service reliability.

Existing automated systems for enterprise management are often challenged by the growing complexity and scale of network operations. Real-time monitoring systems are employed to record incidents such as equipment breakdowns, traffic delays or losses, and failures due to external factors. These incidents are promptly registered from network devices or through end-user communication channels. However, the sheer volume and complexity of these incidents make it increasingly difficult for traditional incident management systems to effectively predict and prevent equipment failures [4,5].

The primary business objective of predicting LAN switch failures is to reduce financial losses and improve technical availability. Traditional methods, while useful, have limitations in handling the dynamic nature of network data and the intricacies of failure patterns. This research paper aims to bridge this gap by introducing a novel approach that combines Double Exponential Smoothing (DES) for trend analysis in time-series data with advanced Machine Learning (ML) algorithms including Random Forest (RF), Logistic Regression (LR), Decision Trees (DT), and Support Vector Machines (SVM). This hybrid method leverages the strengths of both statistical and ML techniques to provide a more accurate and reliable prediction of LAN switch failures.

While there is extensive research in data mining and ML applications in IT, many studies often focus on specific cases or lack comprehensive approaches that consider the multifaceted nature of network environments. Our research not only applies these advanced predictive methods but also integrates disparate data into a coherent framework suitable for failure prediction. Moreover, this paper discusses strategies for preemptive actions to mitigate predicted failures, such as environmental control measures, which can significantly reduce the risk of catastrophic network failures [6–8].

This research paper is organized into distinct sections to provide a comprehensive exploration of the innovative approach developed for predicting failures in LAN switches. The Methods and Materials section 2 meticulously details the systematic processes undertaken in the study, including data collection procedures, data preprocessing techniques and the application of both DES and various ML algorithms. This section establishes the foundation for a robust analysis of the results. The subsequent Results and Outcomes section 3 unveils the findings derived from the implemented methodology, presenting descriptive statistics, performance metrics for individual models and a comparative analysis of predictive accuracy. Finally, the Conclusion section 5 synthesizes the research journey, summarizing key findings, highlighting contributions to the field and discussing the practical implications.

2. Related works

The research methodologies used to evaluate the condition and analyze faults of surge arrestees mainly consist of the following techniques. ML-based methods, anomaly detection algorithms, time series analysis, proactive monitoring systems and data fusion techniques.

Several academics have investigated the application of anomaly detection techniques such as isolation forests, k-nearest neighbor (k-NN), and one-class SVM for predicting failures in LANs. These methods are designed to detect abnormal patterns or behaviors in network traffic that may indicate an imminent failure. Furthermore, LAN failure prediction tasks have utilized deep neural networks, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs). These models are highly proficient at capturing complex connections in time-series network data, which makes them well-suited for forecasting subtle anomalies. In addition, certain research has integrated different ML algorithms or models into ensembles in order to enhance the precision and resilience of LAN failure forecasts.

RF and gradient boosting are often employed ensemble methods in this particular application. In addition, researchers have investigated the scalability and efficiency of LAN failure prediction models with the emergence of big data technologies and cloud computing. Cloud-based technologies enable the instantaneous analysis of substantial amounts of network data. Ultimately, doing research on LAN failure prediction is of utmost importance in guaranteeing the stability and dependability of contemporary networks. The use of ML methodologies, in conjunction with advancements in data acquisition and analysis, persistently propels innovation in this domain, aiming to minimize network downtime and optimize overall performance. This section will highlight various works that are relevant to our subject.

The methodology outlined in reference [9] comprises a series of procedures for forecasting the remaining useful life of hydraulic components. The method involves data preparation, selecting relevant features, training the model, and evaluating its performance. Moreover, data preparation entails the act of purifying and converting the unprocessed data to render it appropriate for analysis. This study employed many strategies for data preprocessing, including data normalization, outlier reduction, and missing value imputation. Feature selection involves identifying and choosing the most significant features that have a significant impact on the accuracy of a prediction model. This step is crucial and enhances the precision of the model's outcomes. Several researches employed various feature selection strategies, including correlation-based feature selection and recursive feature elimination, to identify the most pertinent features [10,9].

In their study [11], authors proposed the utilization of sophisticated forecasting algorithms, such as ARIMA and neural networks, to predict future outcomes. The authors emphasized the significance of model evaluation and selection, which entails assessing the performance of various forecasting models using metrics such as Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). The suggestion was made to utilize a remote monitoring and control system, which would allow for the immediate monitoring and control of the textile production process. The participants deliberated on the utilization of sensors and other monitoring devices to gather data, which can be forwarded to a central control system for the purpose of analysis and decision-making.

The study [12] entails the selection of a suitable ML algorithm and the subsequent training of that algorithm using preprocessed data. DT has the capability to address both classification and regression problems. A decision tree is created by dividing it into separate subsets, referred to as leaf nodes. The branches symbolize distinct potential outcomes derived from the dataset and present a clearly defined objective, whilst the root node represents the most favorable selection. Various techniques, including linear regression, DT, and RF, were employed for model training. Performance evaluation entails the assessment of the precision of the trained models. The authors employed various metrics, including mean absolute error, root mean-squared error, and the coefficient of determination, to assess the efficacy of the model. In addition, they employed a k-fold cross-validation technique to assess the model's capacity to generalize.

Several researchers have applied Bayesian Networks (BN) and dynamic BN methods to predict failures in cellular communication networks [13–15]. Additionally, there are studies utilizing Neural Networks to evaluate the reliability of wireless 2G networks [16]. While it's widely acknowledged that Neural Networks undergo intensive training with analytical and empirical datasets, their effectiveness in real-world scenarios can be constrained. This limitation is primarily due to the variable topology of wireless networks. Furthermore, the accuracy of network failure predictions is intricately linked to the overall quality of the wireless network.

The studies presented in [17–19] are comprehensively described and incorporate various technological concepts pertaining to ML and predictive maintenance. Furthermore, a variety of methodologies and algorithms have been employed for the purpose of data preprocessing, feature selection, model training and performance evaluation in order to enhance the reliability of the suggested system.

In the realm of network failure prediction, particularly for LAN switches, it is imperative to discern the distinct functionalities and applicability of Data Windowing and DES. The selection of either technique hinges on the specific attributes of the data in question and the objectives of the predictive analysis [2,20–23].

Data Windowing is a method that involves analyzing a selected subset of recent data points. This approach is particularly advantageous in scenarios where the most current data is more indicative of future trends. The primary benefits of Data Windowing include its ability to rapidly adapt to recent changes and anomalies, the flexibility it offers in adjusting the size of the data window, and the reduction in computational complexity due to the smaller dataset size. However, this method is not without its drawbacks. Primarily, it may overlook critical historical trends due to its focus on a narrower data range. Additionally, the performance of models using Data Windowing is highly sensitive to the chosen size of the data window.

Conversely, DES extends beyond the capabilities of basic Exponential Smoothing by incorporating both the average level and the trend of the data series. This characteristic renders DES particularly suitable for datasets exhibiting trends. The strengths of DES lie in its ability to capture both the level and trend within the data, which is essential for identifying failures that develop gradually. Furthermore, DES effectively smooths out short-term fluctuations, thereby minimizing noise in the predictive model. However, DES has its limitations, including a delayed response to abrupt data changes due to its inherent smoothing mechanism and the assumption of linear trends, which may not be applicable for complex, non-linear data patterns.

When applying these methods to LAN switch failure prediction, the decision should be based on the nature of the failure patterns observed. If the data shows a consistent trend over time, DES is likely the more appropriate choice. In contrast, for more abrupt failure patterns or when recent data is a stronger predictor of future failures, Data Windowing may prove more effective. In large-scale network environments, where computational efficiency is a key consideration, the simplicity of Data Windowing might be advantageous. Furthermore, employing a hybrid approach that combines both Data Windowing and DES could potentially yield more accurate predictions. This involves using Data Windowing to isolate the most pertinent data segment, followed by the application of DES within this segment to discern trends [23–28].

3. Methods and materials

The predictive system's architecture, as depicted in the Fig. 1, involves an integration of real-time monitoring data and historical database records to facilitate accurate and timely predictions of LAN switch failures. The system initiates with the operational startup of the Information System (IS), pulling data from a dedicated database and a monitoring system equipped with switch sensors. This setup is crucial for capturing both historical trends and immediate operational metrics.

3.1. Case study

In our previous research [6,26], we examined a public sector service provider in a city, catering to approximately 5,000-6,000 end users through a wired data transmission system using UTP cables. The network infrastructure predominantly comprised Cisco equipment (99%), with various models like WS C2960-48P, WS-C3750-48PS, and others, totaling 115 units. These were housed in standardized cross and server rooms, with connections primarily via an optical fiber network (FOCL) in a star topology, centered around Catalyst 6509-E switches as network cores. The key infrastructure features included:

- Over half of the switches were backed by uninterruptible power supplies (UPS).
- Only 30% of the switches had cooling and ventilation systems.
- A multi-level certified protection system was in place, mitigating 95% of failure causes related to internet attacks.

Despite these measures and the integration of the PRTG network monitor for real-time monitoring via SNMP protocol and internal sensors, occasional unexplained device/switch rare failures occurred. This led us to investigate these failures and propose a ML-based predictive system for failure forecasting. Example of topology represented in Fig. 2.

Also, we automate the transformation of data from the PRTG Network Monitor into a format conducive for ML-based forecasting. We developed a script that streamlined the data structuring process, exemplified by a year's worth of operational data from network switches. This script facilitated the conversion of raw data into a format amenable for further analysis.

The dataset is derived from log files. Each log follows a consistent format that includes fourteen specifically chosen variables considered important for the current investigation. The variables are as follows:

- Date: The specific day on which the notification regarding the state of the equipment was recorded. The date format employed was "DD.MM.YYYY", exemplified by the date "10.12.2020". The column was eventually eliminated due to its lack of informative value.
- Uptime refers to the duration during which the device operated without experiencing any failures. The reported format is numeric, with "95.00" representing uninterrupted operation of the switch for 95 consecutive days without any faults.
- Downtime refers to the duration when a particular switch or device in the network is either not available or not functioning correctly. This might happen for a variety of reasons, including hardware malfunctions, software glitches, or connectivity problems. The format used for recording is based on percentages. For instance, a value of 0% shows that there has been uninterrupted operation for 24 hours. On the other hand, a value of 0.1% indicates a downtime of around $1 - 0.2\%$ of a 24-hour period, which is equivalent to around 2 minutes.
- CPU load refers to the level of use or activity of the switch's central processing unit (CPU), which is measured and expressed in numerical form.
- Available memory for Processor.
- Percentage of available memory for Processor.
- Overall available memory.
- The Response Time Index (RTI) is a metric used to quantify the response time of a network device or application.
- Temperature.

The unprocessed monitoring data acquired during the export procedure needs to undergo processing in order to be utilized as a training dataset. The issue arose from the data being exported in numerous CSV files and included duplicate features with inaccurate linguistic labels. Upon completion of data processing, a feature matrix of size was acquired.

Correlation analysis plays a crucial role in understanding the relationships between different variables and identifying potential predictors of failures in LAN devices. By examining the degree and direction of associations between various devices performance metrics and failure events, we could gain insights into which factors are most closely linked to devices failures. Before constructing predictive models, correlation analysis has been conducted to identify relevant input variables. This involves calculating correlation coefficients between each performance metric (for instance CPU usage, temperature, ...) and the occurrence of switch failures. Variables with strong correlations with failure events are typically selected as candidate predictors for the predictive models. In our predictive modeling framework, we have chosen to employ the Pearson correlation coefficient as our primary method for correlation

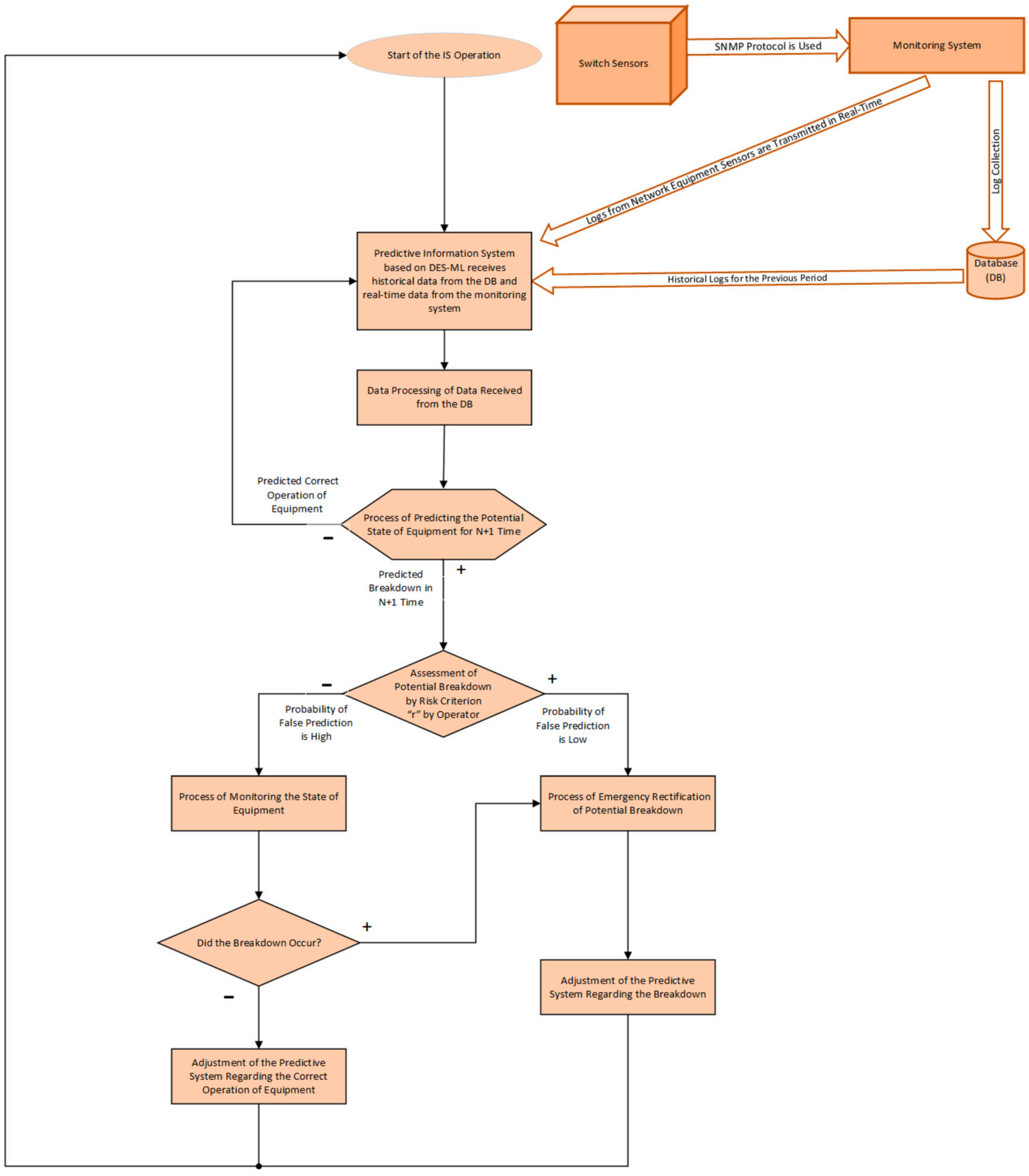

Start of the IS Operation
Switch Sensors
SNMP Protocol is Used
Monitoring System
Logs from Network Equipment Sensors are Transmitted in Real-Time
Log Collection
Database (DB)
Historical Logs for the Previous Period
Predictive Information System based on DES-ML receives historical data from the DB and real-time data from the monitoring system
Data Processing of Data Received from the DB
Predicted Correct Operation of Equipment
Process of Predicting the Potential State of Equipment for N+1 Time
Predicted Breakdown in N+1 Time
Assessment of Potential Breakdown by Risk Criterion "r" by Operator
Probability of False Prediction is High
Probability of False Prediction is Low
Process of Monitoring the State of Equipment
Process of Emergency Rectification of Potential Breakdown
Did the Breakdown Occur?
Adjustment of the Predictive System Regarding the Breakdown
Adjustment of the Predictive System Regarding the Correct Operation of Equipment

Fig. 1. The predictive system's architecture.

analysis. This decision is underpinned by the nature of our data and the specific requirements of our research. The Pearson method is particularly well-suited for datasets where the relationship between variables is presumed to be linear. Given that our study involves variables within a network environment where linear relationships are common, Pearson's correlation provides an effective means of quantifying the strength and direction of these relationships. Correlation analysis guides feature engineering efforts by highlighting which performance metrics are most informative for predicting failures. Features that exhibit significant correlations with failure events may be further processed or transformed to enhance their predictive power.

The Pearson correlation coefficient is calculated as follows:

$$r_{sy} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2\sum_{i=1}^{n}(y_i-\bar{y})}} \quad (1)$$

In this formula, $\bar{x}$ and $\bar{y}$ are the mean values of the variables x and y, respectively and n is the number of observations. This method is adept at revealing how one variable tends to change with another, providing a numerical value that ranges from -1 to 1, indicating the strength and direction of the linear relationship.

For instance, in our analysis of the "temperature" parameter with five data points, the Pearson correlation was determined as follows:

$$r_{xy} \approx 0.7935 \quad (2)$$

This approach was similarly applied to all relevant parameters, resulting in a detailed correlation map. This map is instrumental in identifying key variables that significantly influence network performance and reliability.

By employing the Pearson correlation coefficient, we can effectively discern the linear relationships among various network parameters. This insight is crucial for our predictive model, as it helps in pinpointing the variables most likely to impact network failures, thereby enhancing the model's predictive accuracy and reliability (see Fig. 3).

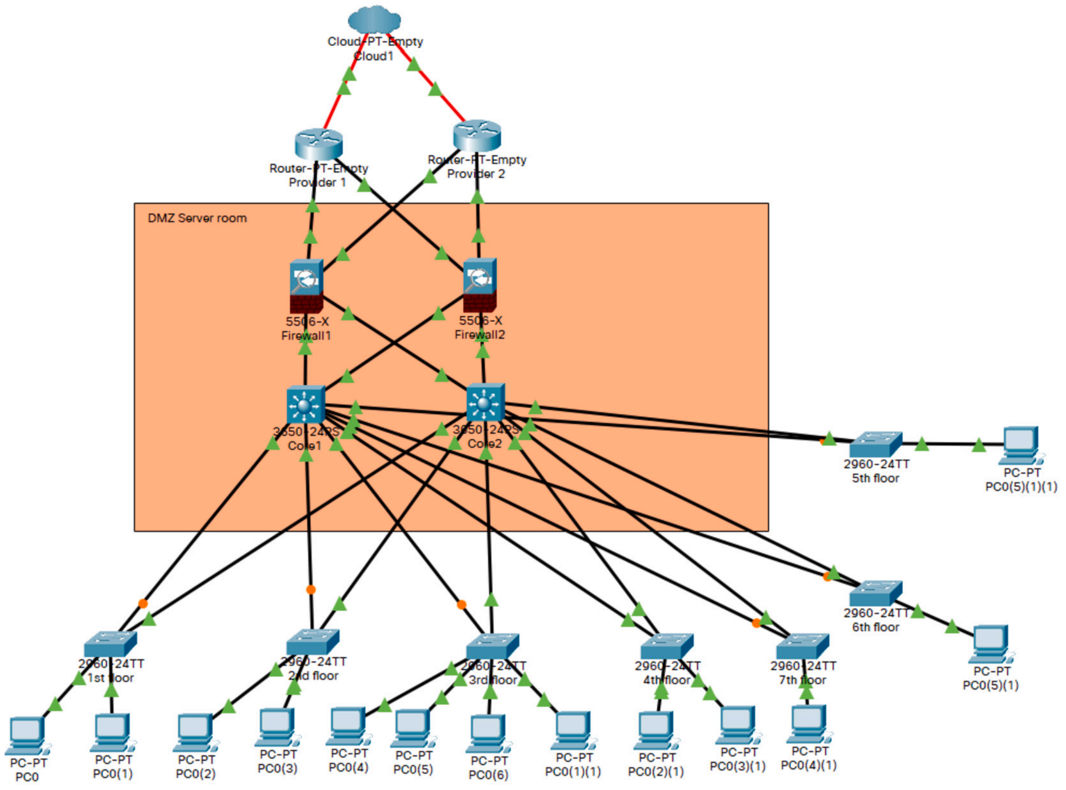


Fig. 2. Example of examined topology.

3.2. *Methods and mathematical parts*

The dataset in its general form is represented as follows:

$$X \in R^{l\times n}, y \in R^{l} \tag{3}$$

such as:

$$X = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ & \ddots & & \\ x_{l1} & x_{l2} & \cdots & x_{ln} \end{pmatrix}$$

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_l \end{pmatrix}$$

The loss function (used for training the model) is defined as:

$$L(\hat{f}) = \sum_{i=1}^{l} (\hat{f}(x_{i1}, \cdots, x_{in}) - y_i)^2 \tag{6}$$

The loss function (6) described is the Mean Squared Error (MSE) loss, commonly used in regression problems. In the context of predicting failures in LAN switches, the goal is to develop a model ($\hat{f}$) that accurately estimates the target variable (y), representing failure occurrences, based on input features ($x_{i1}, \cdots, x_{in}$), which could include various performance metrics or operational parameters of LAN switches. The MSE loss is often preferred for regression tasks due to several reasons. It penalizes large errors quadratically, placing greater emphasis on reducing outliers or large discrepancies between predicted and actual values. It is differentiable, facilitating optimization using gradient-based techniques such as gradient descent, which is commonly used in training ML models. It has a clear geometric interpretation, representing the average squared distance between predicted and actual values, making it easy to interpret and evaluate model performance.

In the context of LAN switch failure prediction, the MSE loss function is suitable because it aligns with the objective of minimizing prediction errors when estimating failure occurrences. Minimizing the squared differences between predicted and actual failure events helps to ensure that the model provides accurate estimations of switch failure probabilities. In addition, LAN switch failure prediction is essentially a regression problem where the goal is to predict a continuous variable based on input features. The MSE loss is well-suited for such tasks, providing a measure of how well the model captures the underlying relationships between input variables and failure occurrences.

$$\hat{f}(x_1, x_2, \cdots, x_9) = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_9 x_9 \tag{7}$$

Where $\hat{f}$ is the predicted value of $y = breaking$, calculated by the fault prediction model.

In our case, the sample consisting of more than 2500 rows is defined as (x_i, y_i) where:

$X =$

$$\begin{pmatrix} x_1 & \texttt{CPU Load} \\ x_2 & \texttt{CPU 1} \\ x_3 & \texttt{Available Memory 1 (Processor)} \\ x_4 & \texttt{Available Memory 2} \\ x_5 & \texttt{Percent Available Memory} \\ x_6 & \texttt{Response Time Index} \\ x_7 & \texttt{Traffic Index} \\ x_8 & \texttt{Alarms} \\ x_9 & \texttt{Temperature} \end{pmatrix}$$

and $y = 0, 1 - Breaking$ as the target variable y. It suggests that the target variable y is derived from a binary classification problem, where Breaking represents the occurrence of a failure event (e.g., a LAN switch breaking down). By subtracting Breaking from 0.1, the target variable is transformed into a numerical value that likely ranges between -0.9 and 0.1, with negative values indicating the absence of a failure event and positive values indicate its occurrence. Typically, in binary classification tasks, a threshold is chosen to differentiate between positive and negative classes. In our case, the threshold is set at 0.1, suggesting that

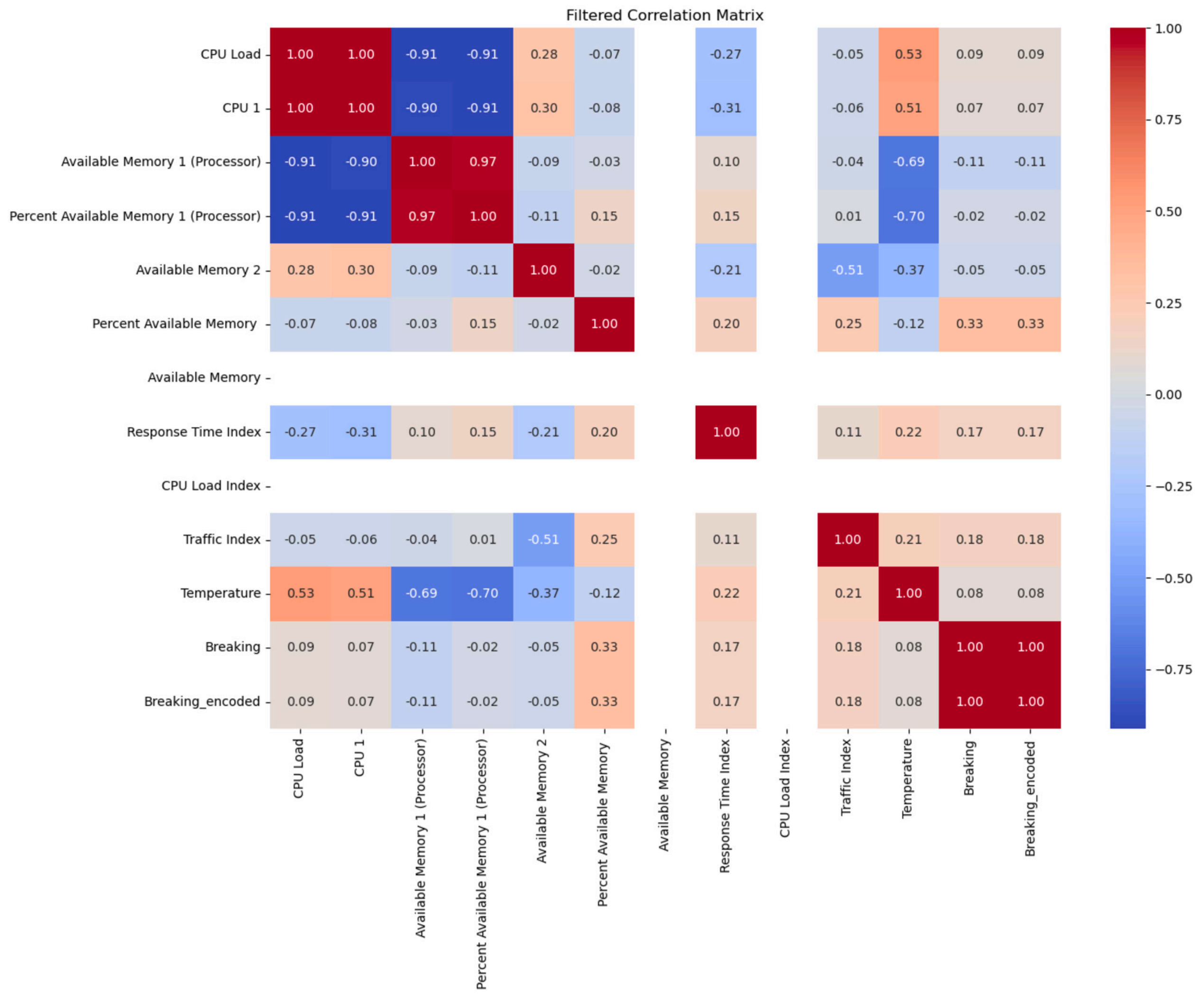


Fig. 3. Data correlation matrix.

any value greater than or equal to 0.1 corresponds to a predicted failure event, while values less than 0.1 correspond to non-failure instances.

For our case, for training purposes, $l = 80\%$ of n, $l = 2000$ rows, we express what we want to obtain in the form of a scalar product: $X \times \omega = y$. In expanded form, this looks as follows:

$$\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{19} \\ x_{21} & x_{22} & \cdots & x_{29} \\ & & \ddots & \\ x_{2000\ 1} & x_{2000\ 2} & \cdots & x_{2000\ 9} \end{pmatrix} \times \begin{bmatrix} \omega_1 \\ \omega_2 \\ \cdots \\ \omega_9 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_9 \end{bmatrix}$$

In our LR framework, we define the loss function as the cumulative average of individual losses across all training instances. This is mathematically represented as:

$$ln(L(\omega)) = -\sum_{i=1}^{l} y_i ln(\hat{f}(x_i)) + (1 - y_i) ln(1 - \hat{f}(x_i)) \to \min_{\omega} \tag{8}$$

This formulation (see equation (8)) is pivotal in optimizing our model's performance, ensuring that it accurately reflects the underlying patterns in the dataset.

Our approach to predicting the operational state of LAN switches is grounded in the principles of linear regression, as delineated in our model formulation. By overlaying the results of this linear prediction with a sigmoid function, we transition to a LR framework. This transition is crucial for producing outcomes as binary values: 0 indicating stable operation and 1 signifying a failure.

The essence of our training process lies in the model's ability to fine-tune its weight coefficients (ω). This fine-tuning is instrumental in accurately forecasting the subsequent state of the network switches and effectively classifying them into either operational or failure categories. Through this method, we harness the predictive power of LR to anticipate and classify network switch states, thereby enhancing our predictive model's efficacy in real-world network management scenarios.

We also leverage the Support Vector Machine (SVM) algorithm, a robust binary classification tool, to differentiate between operational and failure states of LAN switches. SVM operates by identifying critical data points, known as support vectors, from our training dataset. These vectors are instrumental in defining a decision boundary that effectively segregates the dataset into two distinct categories: instances of equipment failures and normal operational data.

In our analysis, we have utilized 3D scatter plots to visualize the complex relationships between pairs of input features and the target variable, "Breaking" (see Fig. 4).

The unique strength of SVM lies in its ability to determine the most optimal decision boundary, known as the hyperplane. This hyperplane is characterized by the maximum margin between the two classes, ensuring that the separation is as clear and distinct as possible. The objective of SVM is to locate this hyperplane, thereby maximizing the classification accuracy. The mathematical formulation of this decision function in SVM is given by:

$$y(x) = w^T \times x + b \tag{9}$$

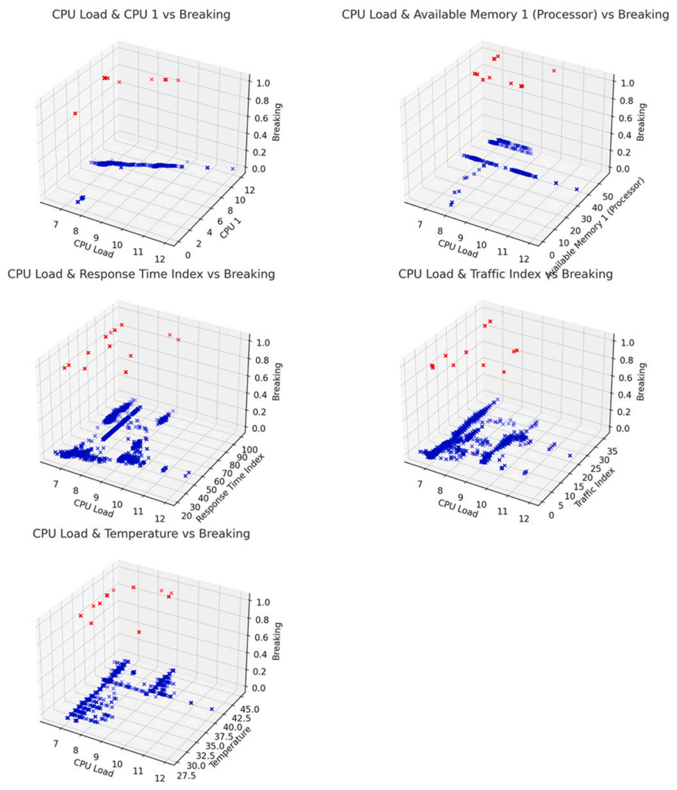


Fig. 4. SVM model Feature Combination and Target Variable Plots.

In our application of SVM, we focus on the optimization of the hyperplane, represented by the coefficients w and b. The input features, denoted as x, are derived from various operational parameters of the network equipment, such as CPU load, memory usage, response times, and internal temperatures, among others. These parameters are systematically transformed into feature vectors, each corresponding to a specific state of the network equipment.

Our dataset, comprising N feature vectors, is utilized to train the SVM model. The objective is to find the optimal hyperplane that maximizes the margin between the two classes (operational and failure states). This is achieved by ensuring that the closest points to the hyperplane, the support vectors, are as distant as possible from it. The mathematical goal of the SVM in our context is to maximize the margin, which is represented by the equation:

$$\underset{w,b}{\operatorname{argmax}} \left\{ \frac{2}{\|w\|} \right\} \min_{n} \left[w^T \cdot x_n + b \right] \tag{10}$$

Solving this optimization problem involves transforming it into a more tractable form. We utilize the method of Lagrange multipliers to address the constraints in the optimization process. The dual problem, derived using Lagrange multipliers, allows us to find the optimal values of "w" and "b" that maximize the margin between the classes. The optimization problem is then expressed as:

$$\underset{w,b}{\operatorname{argmax}} \left\{ \sum_{i=1}^{N} a_i - \frac{1}{2} \sum_{j=1}^{N} \sum_{i=1}^{N} l_j \times l_i \times a_j \times a_i \langle x_j, x_i \rangle \right\} \tag{11}$$

such as, *argmax* denotes the argument that maximizes the function, N is the total number of samples in the dataset; a_i is the Lagrange multiplier associated with the i^{th} data point or sample, l_i is the label of the i^{th} data point, x_i and x_j are feature vectors representing the i^{th} and j^{th} data points respectively, $\langle x_j, x_i \rangle$ is the inner product (dot product) between the feature vectors x_j and x_i.

The optimization is subject to constraints ensuring non-negativity of the Lagrange multipliers and the satisfaction of the Karush-Kuhn-Tucker conditions:

$$a_i \geq 0, i = 1, 2, \cdots, N \tag{12}$$

$$\sum_{i=1}^{N} a_i l_i = 0 \tag{13}$$

Solving this dual formulation allows us to determine the optimal values of w and b, essential for defining the SVM's decision boundary. The weight vector w is obtained as a linear combination of the training data points, weighted by their corresponding Lagrange multipliers:

$$w = \sum_{i=1}^{N} a_i l_i x_i \tag{14}$$

The bias term b is then derived based on the support vectors:

$$b = l_j - \sum_{i=1}^{N} a_i l_i \langle x_j, x_i \rangle \tag{15}$$

At this stage, these feature vectors are referred to as training data, and the computational process is known as the training process. As we can see, the choice of training data directly affects the solution for w and b, then impacts the accuracy of classification. To diagnose equipment failure with high precision, a sufficient amount of data on failures and normal operations is necessary.

In practice, data groups may overlap, leading to overfitting and poor generalization if we precisely separate the data. In this case, SVM uses slack variables, which allow for the misclassification of training data. The constraint coefficient C is used to control the balance between accuracy and generalization of the hyperplane. With the introduction of the constraint C, the formula (see equation (12)) is modified as $C \geq a_i \geq 0, i = 1, \cdots, N$.

The above discussion is based on the assumption that data groups are linearly separable; however, this is not often the case in practical tasks. To solve the nonlinear problem, a kernel function is introduced to map complex, linearly inseparable data from a low-dimensional feature space to a higher-dimensional feature space where the data are linearly separable. After this transformation, we can use the solution of the linear problem to solve the nonlinear problem in a higher dimension.

We define our input space as X and the transformed feature space as H. The kernel function $K(x, z)$, which satisfies $K(x, z) = \Phi(x) \cdot \Phi(z)$, facilitates this transformation. In our dual formulation of the SVM problem, we replace the scalar products in the equations with kernel functions, leading to a new optimization problem:

$$\underset{w,b}{\operatorname{argmax}} \left\{ \sum_{i=1}^{N} a_i - \frac{1}{2} \sum_{j=1}^{N} \sum_{i=1}^{N} l_j \times l_i \times a_j \times a_i K \langle x_j, x_i \rangle \right\} \tag{16}$$

$$\sum_{i=1}^{N} a_i l_i = 0 \tag{17}$$

This function, transforms our input data into a higher-dimensional feature space where linear separation is feasible. The inner product in our original space is thus converted into an inner product in this transformed feature space, as per equations (11) and (13). By identifying an appropriate kernel function $K\langle x_j, x_i \rangle$, we can effectively tackle nonlinear classification problems using equations (16) and (17).

This kernel transformation enables SVM to effectively handle binary classification tasks, such as determining the operational state of network equipment. In this context, SVM serves as a tool for intelligent pattern recognition, identifying potential failures based on operational data.

The study was employed the Radial Basis Function (RBF) for the SVM model. This choice of kernel function is pivotal due to its proficiency in handling nonlinear classification challenges. The RBF kernel effectively projects the original data into a higher dimensional space, enabling linear separation even for complex data structures. This characteristic enhances the model's capability to discern intricate patterns and relationships within the data, leading to more robust and accurate predictions of LAN switch failures.

In our application of RF for predicting LAN switch failures, we approach the concept of loss function L from a perspective that diverges from explicit minimization. Instead, we evaluate the model's effectiveness through the lens of average entropy, calculated across the ensemble of DT. This is expressed as:

$$L(\hat{f}) = \frac{1}{T}\sum_{i=1}^{T} H(T_i) \tag{18}$$

In our model, T representing the total number of trees in the forest, is set to 100. This choice balances computational efficiency with the robustness of the model. The entropy H for each tree, a critical measure in classification tasks, is calculated using the formula:

$$H(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-) \tag{19}$$

Here, p_+ and p_- denote the probabilities of a switch being operational (positive class) or failing (negative class) within a particular node of the tree.

Each tree T_i, is constructed using a bootstrapped sample drawn from the original dataset. The construction of these trees involves a critical decision-making process at each node, where we select a subset of features and then choose the feature that optimally reduces entropy, thereby maximizing information gain. The entropy $H(S)$ is calculated as:

$$H(S) = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{20}$$

where p_i represents the probability of occurrence of class i in the dataset S and n is the total number of distinct classes.

For prediction, each tree T_i in the forest contributes its individual prediction y_i for a new observation x. The final prediction $\hat{y}$ is determined through a majority voting mechanism among all the trees, as expressed in equation (21).

$$\hat{y} = majority(y_1, y_2, \cdots, y_{100}) \tag{21}$$

Selecting $T = 100$ trees for the RF ensemble strikes a balance between model complexity, computational efficiency and predictive performance. However, the choice may vary depending, $T = 100$ is a common and reasonable choice that often yields satisfactory results.

For node splitting in a decision tree, when using entropy as the loss function, the following formula is applied:

$$L(\hat{f}) = Entropy(Parent) - \sum_{i} \frac{N_i}{N} \times Entropy(Child_i) \tag{22}$$

$Entropy(S)$ often denotes the entropy of node S, which can be either a parent or a child node, depending on the context. $Entropy(S)$ is defined by formula (20). Here, N is the total number of samples in the parent node and N_i is the number of samples in the $i-th$ child node. The trained decision tree $f(x)$ represents a recursive function that, at each level d, $0 \le d < D$. Where D is the maximum depth of the tree, in our case, a depth of 3 or 4 branches.

$$f_d(x) = \begin{cases} f_{d+1,left}(x), & if\ x_i \le t \\ f_{d+1,right}(x), & otherwise \end{cases} \tag{23}$$

Here, x_i is the feature based on which the split occurs, t is the threshold value for x_i, $f_(d+1, left)$ and $f_(d+1, right)$ are recursive functions for the left and right child nodes, respectively.

However, SVM or other ML algorithms alone does not suffice for predictive analysis. As illustrated in Fig. 5, while we can analyze historical data up to time $t-1$, this does not constitute forecasting the future state of the equipment. To address this, we integrate ML with additional predictive algorithm. This algorithm forecast future operational indicators at time $t+T$, enabling us to predict potential equipment failures at this future time point. This integrated approach enhances our model's predictive capability, allowing for more accurate and timely identification of equipment failures.

DES improves upon traditional exponential smoothing by incorporating two key components: the level and the trend of the time series. This dual-component structure allows DES to adapt more responsively to sudden shifts in the data, a feature that is crucial for accurately capturing the dynamics of time series with sharp variations.

The application of DES in our predictive models is especially beneficial in scenarios where rapid changes in network performance metrics are observed. By employing DES, we can generate more accurate and timely forecasts, which are essential for proactive network management and failure prevention strategies. The DES algorithm, characterized by its layered smoothing technique, applies exponential smoothing twice: initially to the actual data and subsequently to the smoothed values. This process is encapsulated in the equations:

$$S_t^{(1)} = a y_t + (1-a) S_{t-1}^{(1)} \tag{24}$$

$$S_t^{(2)} = a S_t^{(1)} + (1-a) S_{t-1}^{(2)} \tag{25}$$

Formula (25) represents the second component (trend component) of the DES algorithm. And there is a description:

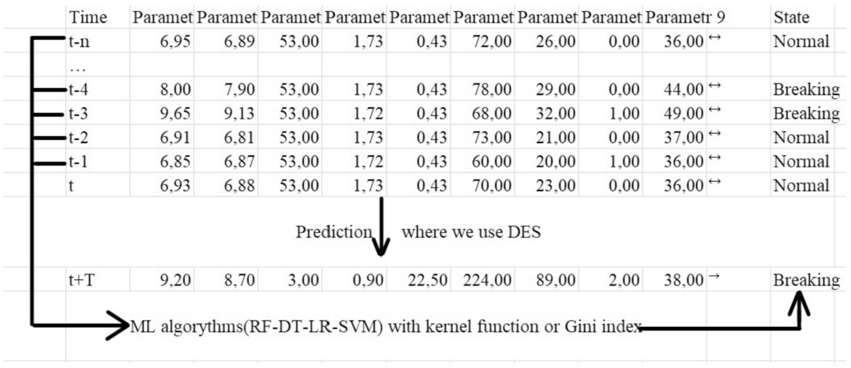
- $S_t^{(2)}$: This term represents the estimated trend component of the time series at time t. It is the output of this part of the formula, indicating the smoothed estimate of the trend at the current time step.
- a: This is the smoothing parameter for the trend component. It is a value between 0 and 1 that determines how much weight is given to the most recent observation in the time series. A higher value of a gives more weight to recent changes in the trend, making the algorithm more responsive to new trends.
- $S_t^{(1)}$:This term is the estimated level component of the time series at time t, obtained from the first equation of the DES algorithm. It represents the smoothed estimate of the series' value at the current time step.
- $(1-a)$: This part of the formula gives weight to the previous estimate of the trend. It complements the smoothing parameter a such that the sum of the weights is 1. This ensures that the entire range of past data is considered, with a focus on the most recent trend.
- $S_(t-1)^{(2)}$: This is the estimated trend component from the previous time step. It carries forward the previously estimated trend to the current calculation, ensuring continuity in the trend estimation.

In summary, this formula updates the trend component of the DES algorithm. It combines the most recent trend change (captured in $S_t^{(1)}$) with the previously estimated trend (captured in $S_(t-1)^{(2)}$), weighted by the smoothing parameter a. This approach allows the DES method to adaptively smooth the trend of the time series, making it a powerful tool for forecasting data that exhibits trends over time.

We propose a novel DES-ML forecasting method, where DES is utilized to predict future values of each indicator, and these predictions are then fed into an ML model. The ML model evaluates the likelihood of equipment failure at future time points based on these forecasts. This synergistic approach leverages the strength of DES in handling time series data with trends and the robust classification capabilities of ML algorithms, offering a comprehensive tool for proactive maintenance and failure prediction in network equipment. As an example, by equation (26), (27) and (28):

$$\hat{Y}_{t+T} = a_t + b_t \times T \tag{26}$$

$$a_t = 2S_t^{(1)} - S_t^{(2)} \tag{27}$$



Time	Paramet	Paramet	Paramet	Paramet	Paramet	Paramet	Paramet	Paramet	Parametr 9		State
t-n	6,95	6,89	53,00	1,73	0,43	72,00	26,00	0,00	36,00	↔	Normal
…											
t-4	8,00	7,90	53,00	1,73	0,43	78,00	29,00	0,00	44,00	↔	Breaking
t-3	9,65	9,13	53,00	1,72	0,43	68,00	32,00	1,00	49,00	↔	Breaking
t-2	6,91	6,81	53,00	1,73	0,43	73,00	21,00	0,00	37,00	↔	Normal
t-1	6,85	6,87	53,00	1,72	0,43	60,00	20,00	1,00	36,00	↔	Normal
t	6,93	6,88	53,00	1,73	0,43	70,00	23,00	0,00	36,00	↔	Normal
t+T	9,20	8,70	3,00	0,90	22,50	224,00	89,00	2,00	38,00	→	Breaking

Prediction
where we use DES
ML algorythms(RF-DT-LR-SVM) with kernel function or Gini index

Fig. 5. DES+ML for predicting of breakings.

$$b_t = \frac{a}{1-a}(S_t^{(1)} - S_t^{(2)}) \tag{28}$$

In this equation a_t is calculated using the current estimate of the level $S_t^{(1)}$ and the trend $S_t^{(2)}$ at time t. This formula adjusts the level component by considering the current trend. b_t is the trend component at time t. It is calculated as a function of the difference between the level and trend components, scaled by the smoothing parameter a. This formula captures the rate of change in the level component, which is indicative of the trend.

In our study, it is pertinent to briefly describe the configurations in which all experiments were conducted. While the specifications of the machines do not directly influence the results, they significantly impact the duration of computations. We utilized two distinct configurations: a high-powered workstation PC designed for various software and project applications, and a personal laptop.

- Environment A: Workstation PC:
  - Processor Model: Ryzen 5 $5800X$, featuring 8 cores and 16 threads, with a base clock speed of 3.8 GHz.
  - Operating System: Windows 10.
  - RAM: 32 GB $DDR4$ at 3200 MHz.
  - Storage: An SSD file system with 4 TB of space, housing the original dataset and its various transformations.

This workstation PC is a remarkable machine with substantial computational power, enabling the running of distributed algorithms and exploring their horizontal scalability to a certain extent.

- Environment B: Personal Laptop:
  - Model: Legion $Y530$.
  - Operating System: Windows 10.
  - Processor: Intel Core $i5 - 8300h$ at 2.3 GHz with 4 cores.
  - RAM: 16 GB $DDR4$ at 2667 MHz.
  - Storage: An HDD file system with 500 GB of space, containing the original dataset and its transformations.

The laptop, used for the majority of tasks including data scanning, result compilation, and chart creation, also facilitated model training tests. Despite being less powerful than the workstation, it provided sufficient computational capability for many experiments requiring limited computations and offered easier access to results compared to a distributed environment.

In summary, the computational power of the workstation PC is approximately two to two and a half times greater than that of the laptop. This difference in computing power between the two environments was a key consideration in our experimental design, as it influenced the time efficiency of data processing and model training.

A crucial step in our research involved training combined ML models for the effective classification of potential events. We were faced with the challenge of determining the optimal amount of data required for the most efficient training process. To address this, we segmented our primary dataset into three distinct subsets: the first comprised 500 records, the second expanded to 1000 records and the final subset encompassed 2500 records. It is important to note that regardless of the sample size, the number of parameters utilized in the algorithm remained constant at 14. However, some parameters were effectively disregarded by the algorithms due to their negligible weight coefficients, almost bordering zero. This observation aligns with our preliminary correlation analysis of the input data. Additionally, all combined models employed a five-fold cross-validation technique to enhance the accuracy of event classification.

3.3. Architecture of the predictive information system

The predictive system's architecture, as depicted in the figure 1, involves an integration of real-time monitoring data and historical database records to facilitate accurate and timely predictions of LAN switch failures. The system initiates with the operational startup of the Information System (IS), pulling data from a dedicated database and a monitoring system equipped with switch sensors. This setup is crucial for capturing both historical trends and immediate operational metrics.

The core of the predictive system is based on a DES-ML model, which harnesses DES for trend analysis and forecasting of time-series data, combined with ML algorithms for enhanced predictive accuracy. Data flows into the predictive model, where it undergoes initial preprocessing to align with the input requirements of the ML models.

The predictive process involves several key steps:

1. Data Handling: The DES-ML predictive information system processes incoming data from the database (DB) for historical context and from the monitoring system for real-time updates.
2. Prediction Processes: It includes data processing to structure the input correctly and a predictive phase where potential future states of the equipment are estimated for the upcoming N+1 time period.
3. Outcome Predictions and Adjustments: Depending on the prediction, the system either flags an equipment state as normal or predicts a potential malfunction. Risk assessments are performed based on a predefined risk criterion 'r', managed by an operator. The system's ability to adapt its predictive mechanisms based on actual outcomes versus predicted states ensures continual refinement and increased accuracy of the predictions. This dynamic adjustment process will involve monitoring the actual state of the equipment and performing emergency interventions if a predicted failure does occur.

3.4. Experimental platform and tools

The computational experiments conducted in this study were facilitated by an integrated Python environment, utilizing Jupyter Notebooks

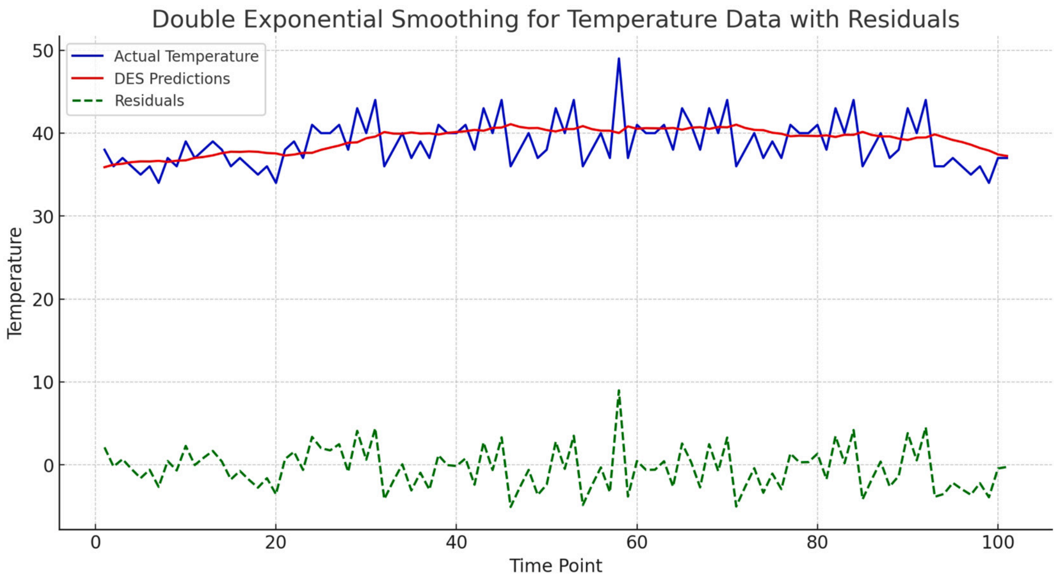


Fig. 6. Forecast of indicators of the "Temperature" parameter using the DES method.

for an interactive coding experience. Our analyses heavily relied on several robust libraries including Scikit-learn for implementing ML algorithms, Pandas for data manipulation, Numpy for numerical operations, and Matplotlib along with Seaborn for data visualization. The data preprocessing phase often employed tools such as StandardScaler for feature scaling, encapsulated within pipelines created using the make pipeline utility. The performance of our predictive models was assessed primarily using metrics like accuracy score and classification report, providing comprehensive insights into the models' prediction capabilities.

4. Results and outcomes

We applied the DES method to forecast the trend of each data parameter at the point ($\hat{Y}_{t+T}$). The forecast results for the trend were as follows: in attempting to identify the trend of the "Temperature" parameter (Fig. 6), the DES method showed that the difference between predicted and observed values was not significant, fluctuating from a minimum of 1 to a maximum of 10 values. The red line indicates the smoothed trend, reflecting the general direction and dynamics of temperature changes. The graph demonstrates that the DES model effectively identifies the main trends in the data, allowing for a better understanding of the overall direction of temperature changes. The Mean Absolute Percentage Error (MAPE) was approximately 5.25%. This indicates that, on average, the predictions deviate from the actual values by 5.25%.

MAPE measures the relative accuracy of the model and is calculated as the average of absolute percentage errors across all observations. It is computed as follows:

$$MAPE = \frac{100}{n}\sum_{i=1}^{n} |\frac{y_i - \hat{y}_i}{y_i}| \tag{29}$$

where n is the number of observations, y_i is actual value and $\hat{y}_i$ is the predicted value.

RMSE, (Root Mean Square Error) measures the average of the squares of errors, that is, the differences between the predicted values of the model and the actual data values. Its formula is:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)} \tag{30}$$

In our case, the RMSE value is approximately 2.56. This indicates that, on average, the predictions of our model deviate from the actual temperature values by 2.56 units, which is a very satisfactory result for our study. Subsequently, DES was applied to the "CPU Load" parameter for an annual record (365 entries), and as the graph demonstrates, DES effectively manages the forecasting of trends for this parameter (Fig. 7). The RRMSE in this case shows an average deviation from the actual values of 0.2486 units. The MAPE is 1.1490%. In other words, the DES method for the "CPU Load" parameter exhibited very commendable results.

The subsequent DES method to the "Traffic Index" and "Response Time Index" parameters, as shown in Fig. 8 and 9, respectively.

As evident from the visualizations (Fig. 6, 7, 8, 9) of the DES's performance for each of the parameters mentioned, the forecasting results are highly accurate when the parameters change smoothly. However, accuracy temporarily decreases with abrupt changes in the parameters. The primary reason for this is the insufficient number of data points, which hinders the algorithm's ability to smoothly adapt to changes. Nonet heless, DES still demonstrates high predictive accuracy. The MAPE and RMSE indicators for these parameters were not excessively high, indicating that the model generally performs well in forecasting, albeit with some deviations from the actual values.

In the context of our study, we observed notable results depicted in Fig. 10, focusing on three distinct models employing varied methodological approaches: one based on the traditional Decision Tree method with DES, another utilizing the RF with DES, and the third employing LR combined with DES.

1. For the 500-row dataset:
   - DES-The TR method yielded a prediction accuracy of 0.51.
   - DES-Decision Tree methodology achieved a slightly higher forecast quality of 0.52.
   - DES-The RF approach demonstrated superior predictive performance, with an accuracy of 0.858.
2. With the 1000-row dataset:
   - DES-DT showed an improvement in forecast quality, reaching an accuracy of 0.786.
   - Conversely, DES-LR exhibited a decrease in predictive accuracy to 0.43.
   - The DES-RF method maintained its previous efficiency level at 0.858.
3. For the 2500-row dataset:
   - DES-LR significantly enhanced its effectiveness, achieving a forecast quality of 0.9924.

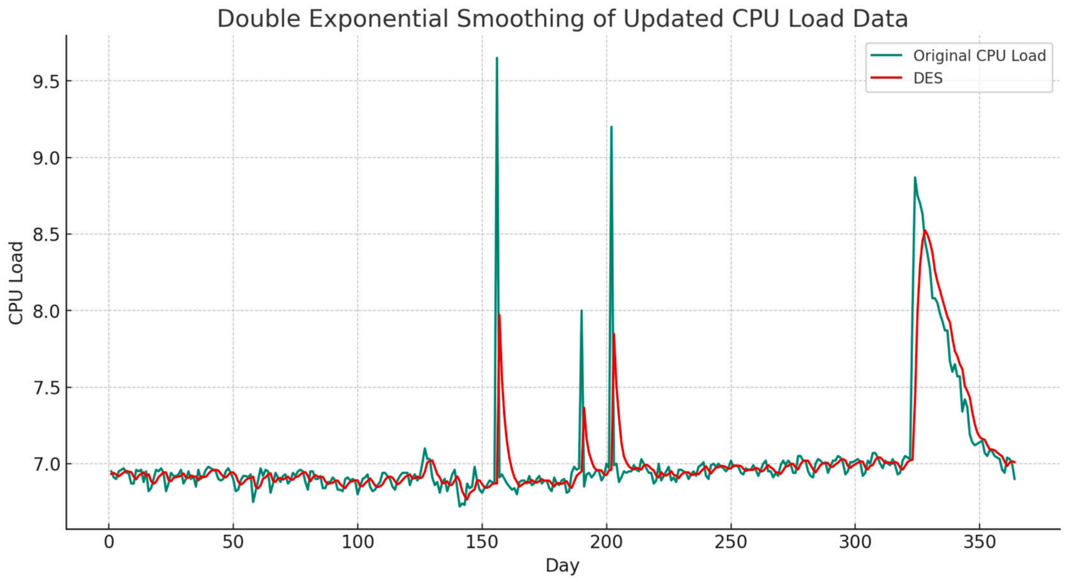


Fig. 7. Forecasting the parameters of the “CPU Load” parameter using the DES method.

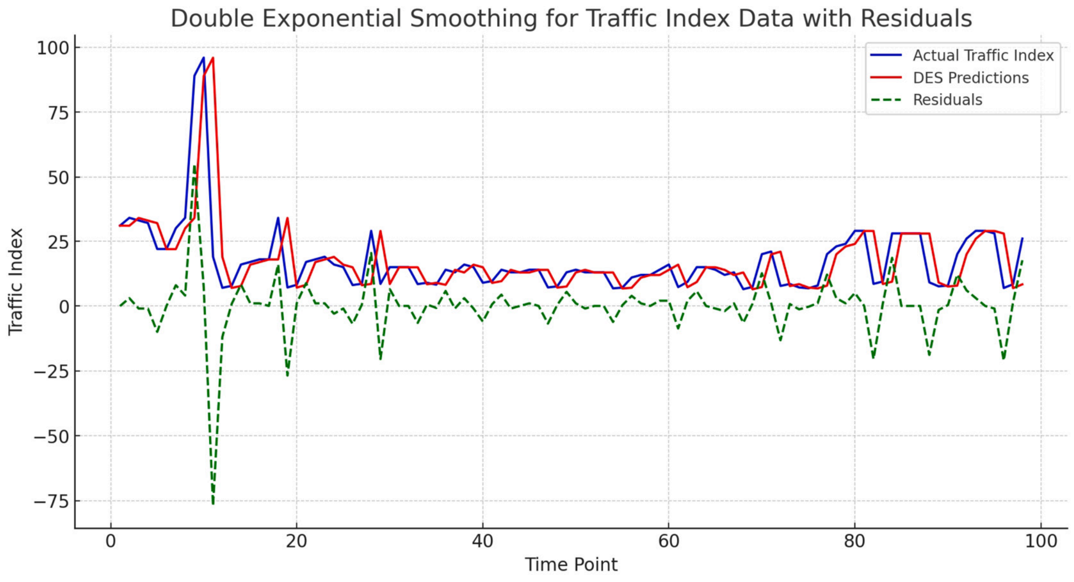


Fig. 8. Forecasting the parameters of the “Traffic index” parameter using the DES method.

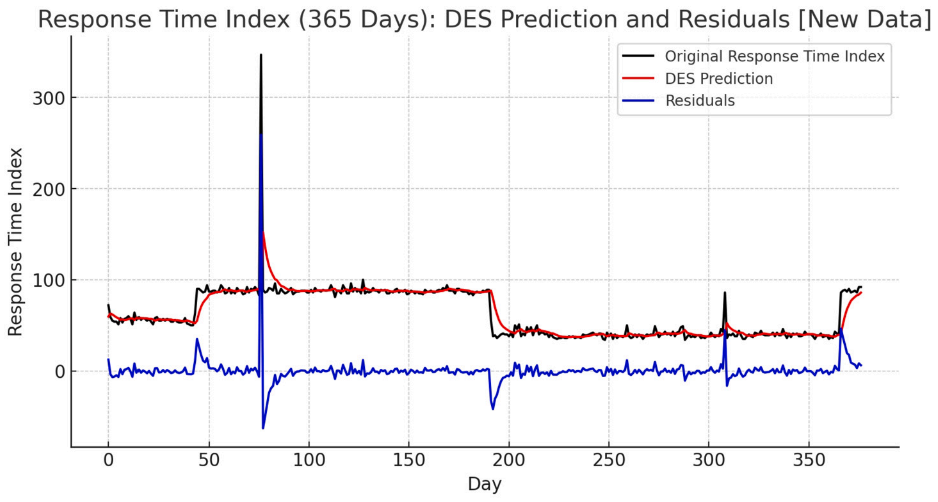


Fig. 9. Forecast of indicators of the “Response time index” parameter using the DES method.

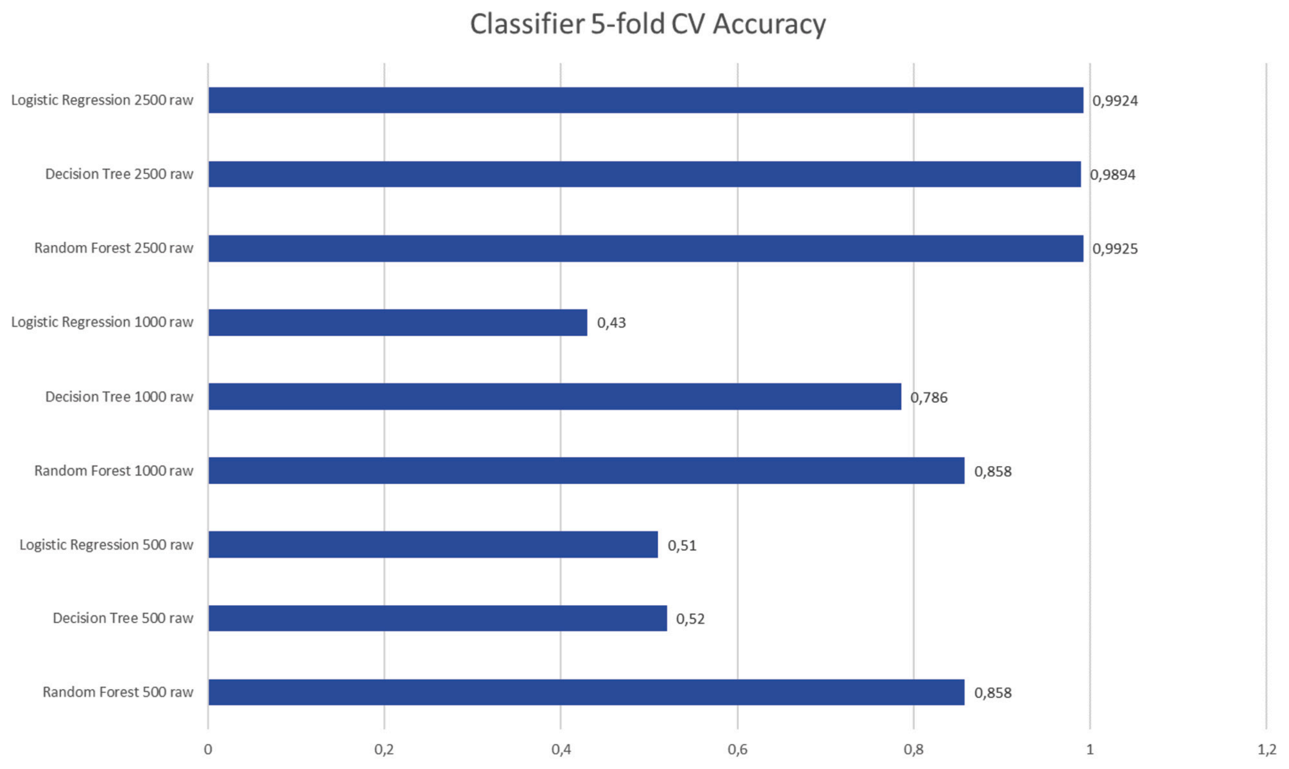


Fig. 10. General visualization of the accuracy of forecasting results.

- DES-DT also displayed high predictive accuracy, with a score of 0.9894.
- DES-RF revealed a result of 0.9925.

Analyzing the data presented in Fig. 10, it becomes evident that an increase in the volume of training data correlates with a rise in forecast accuracy.

During our investigation into the optimal data volume for training a model based on the DES-RF method, we determined that the minimum necessary data volume for effective training and reliable outcomes ranges between 500 to 1000 data rows. This finding is noteworthy as it highlights the potential for achieving acceptable forecast accuracy even with a relatively small training dataset. However, it is important to note that for the other two models examined, a significantly larger data volume, specifically more than 2000 rows, is required to achieve satisfactory accuracy.

It has been observed that with an increase in the volume of data used for training our hybrid model, there is a notable shift in the distribution of parameter weights within the model. This phenomenon is quite logical, as a larger dataset provides the model with more information to discern and account for finer dependencies among various parameters. However, it was also noted that the distribution of parameter weights varies across different models. For instance, as illustrated in Figs. 11, 12 and 13, the weight assigned to each parameter differed among models: In the DES-Decision Tree-based model, with a dataset of 1000 rows and an accuracy exceeding 78%, parameters such as “Temperature”, “Available Memory 2”, “CPU1”, “Response Time Index” and “CPU Load” were given more significance. Conversely, in the DES-RF-based model, using the same dataset size but achieving over 85% accuracy, the parameters “CPU Load”, “CPU1”, “Response Time Index”, “Temperature”, “Available Memory 2” and “Traffic Index” were more heavily weighted. Similarly, for the DES-LR model trained on 2500 data rows and achieving over 99% accuracy in quality fault prediction, the weight distribution was as depicted in Fig. 13.

Also we compared operational environments: In “Environment A”, where algorithmic experiments were conducted and these models required only 2 – 3 seconds of real-time processing for a dataset comprising 2500 records, inclusive of a five-fold cross-validation process. However, transitioning to “Environment B”, it’s noteworthy that while the initial conditions remained similar, the processing time increased

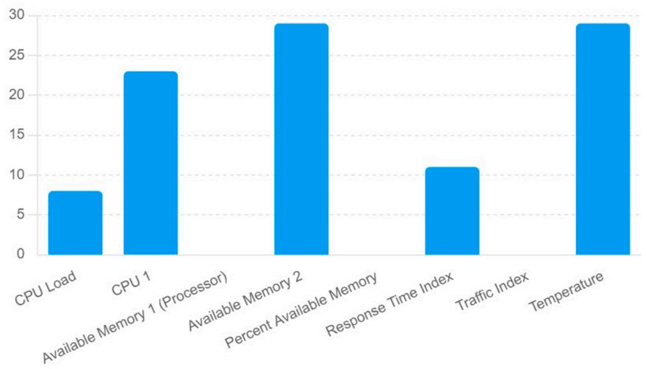


Fig. 11. DES-DT feature importances, 1000 rows of data.

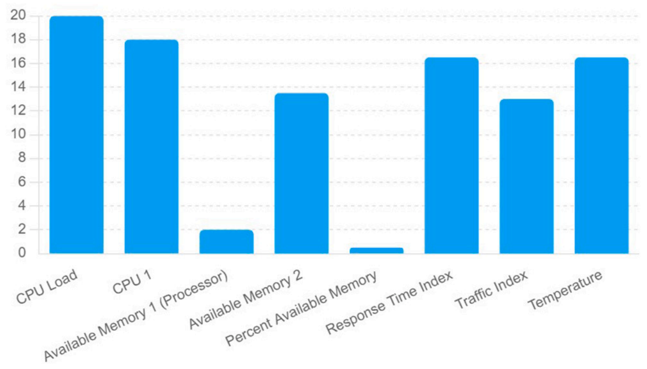


Fig. 12. DES-RF feature importances, 1000 rows of data.

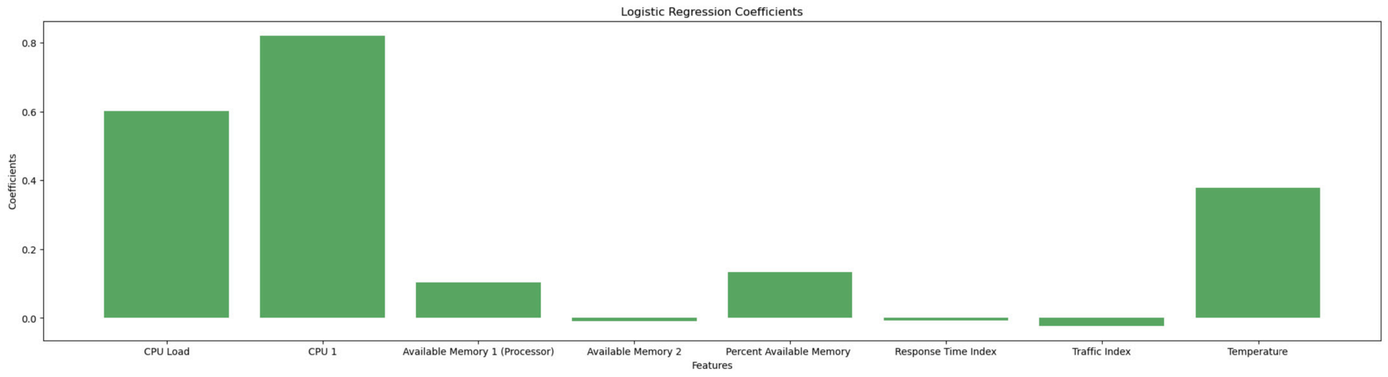


Fig. 13. DES-LR feature importances, 2500 rows of data.

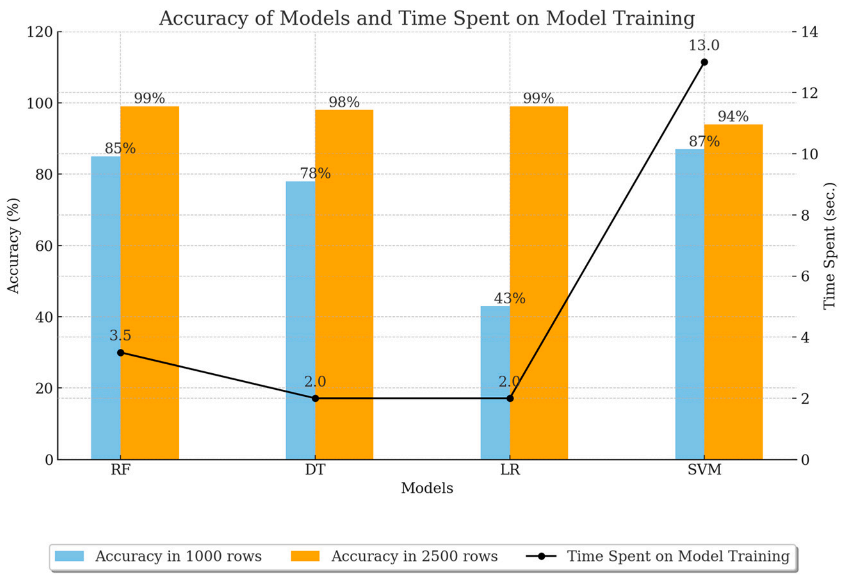


Fig. 14. Accuracy of models and time Spent on training model.

to approximately 6 seconds. This observation underscores that varying operational environments can significantly impact combined model performance.

Considering the technical specifications of the device used for testing, it is imperative to highlight the critical role of components such as the central processing unit, the size of the RAM, and the characteristics of the chosen file system. These parameters not only constitute the foundational infrastructure for combined algorithm operation but also directly influence their efficiency. Hence, it can be inferred that the quality and performance of the utilized resources are directly proportional to the time algorithms require for data processing. Optimizing these resources could, therefore, substantially accelerate the forecasting process and enhance its accuracy.

In the final phase (see Fig. 14) of our study, we conducted a detailed comparison of the three aforementioned combined algorithm models with a fourth model based on the DES - Support Vector Machine method. The DES-SVM-based algorithm was tested under similar conditions as the other three models, including datasets of 1000 and 2500 records, 14 different parameters, five-fold cross-validation, and experiments conducted in both “Environment A” and “Environment B”.

From a qualitative perspective, specifically in terms of accuracy metrics, the DES-SVM-based model demonstrated an accuracy of 87% with the 1000-record dataset and 94% with the 2500-record dataset. It is important to note that these results were lower compared to the outcomes achieved by the other models in network equipment failure prediction.

In terms of the time required for forecasting, the DES-SVM-based model also exhibited the longest duration among all models considered. Specifically, in “Environment A”, the forecasting time averaged between 12 to 14 seconds, while in “Environment B”, it varied from 17 to 19 seconds. Therefore, despite the DES-SVM-based model showing the lowest predictive accuracy and the longest processing time among the models examined, it is crucial to consider a key characteristic. The SVM method, typically exhibits high resistance to overfitting, particularly when regularization techniques are applied. This aspect can be critically important under certain conditions and should be considered when selecting a model for specific forecasting tasks.

5. Conclusion

In this paper has introduced an innovative approach to predicting failures in LAN switches by combining DES with a suite of ML algorithms including RF, LR, DT, and SVM. The primary objective of this study was to enhance the accuracy and timeliness of LAN switch failure predictions, thereby facilitating more proactive and effective network management. Our methodology involved the integration of DES for trend analysis and forecasting in time-series data with the advanced predictive capabilities of ML algorithms. This hybrid approach not only leveraged the strengths of DES in identifying underlying patterns in failure data but also capitalized on the diverse predictive models to handle various aspects of failure prediction more robustly.

Through rigorous testing and validation, our proposed approach demonstrated a marked increase in the precision and reliability of failure predictions. The results indicated that the integration of DES with ML algorithms substantially aided in preemptive maintenance and decision-making processes in LAN management.

CRediT authorship contribution statement

Ali Myrzatay: Writing – original draft, Software, Investigation. **Leila Rzayeva:** Writing – original draft, Software, Methodology. **Stefania Bandini:** Writing – original draft, Software, Methodology. **Ibraheem Shayea:** Writing – review & editing, Validation, Methodology. **Bilal Saoud:** Writing – review & editing, Visualization, Validation, Methodology. **Ilhami Çolak:** Formal analysis. **Korhan Kayisli:** Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] L.G. Rzayeva, et al., THE EFFECT OF THE AMOUNT OF DATA ARRAY ON THE RESULTS OF FORECASTING NETWORK EQUIPMENT FAILURES, Izv. Nauk Kaz. Ser. Fiz.-Mat. 6 (2021) 28–36.
[2] Zhilong Wang, et al., Failure prediction using machine learning and time series in optical network, Opt. Express 25 (16) (2017) 18553–18565.
[3] Ishkanov, S. Yu, A.A. Shelupanov, S.V. Timchenko, TUSUR Reports 25 (1) (2012) 100–103, part 2.
[4] Mohamed Abomhara, Geir M. Køien, Cyber security and the internet of things: vulnerabilities, threats, intruders and attacks, J. Cybersecurity Mob. 4 (1) (2015) 65–88.
[5] Marcin Bajer, Building an IoT Data Hub with Elasticsearch, Logstash and Kibana, 2017 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), IEEE, 2017.
[6] Leila Rzayeva, et al., Enhancing LAN failure predictions with decision trees and SVMs: methodology and implementation, Electron. 12 (18) (2023) 3950.
[7] Raouf Boutaba, et al., A comprehensive survey on machine learning for networking: evolution, applications and research opportunities, J. Internet Serv. Appl. 9 (1) (2018) 1–99.
[8] John S. Baras, et al., Automated Network Fault Management, vol. 3, IEEE, 1997.
[9] M. Yugapriya, Antony Kovilpillai, J. Judeson, S. Jayanthy, Predictive Maintenance of Hydraulic System Using Machine Learning Algorithms, 2022 International Conference on Electronics and Renewable Systems (ICEARS), IEEE, 2022.
[10] Qingzong Li, Yuqian Yang, Pingyu Jiang, Remote monitoring and maintenance for equipment and production lines on industrial internet: a literature review, Mach. 11 (1) (2022) 12.
[11] Jingfeng Shao, et al., Remote Monitoring and Control System Oriented to the Textile Enterprise, in: 2009 Second International Symposium on Knowledge Acquisition and Modeling, vol. 3, IEEE, 2009.
[12] Joanita Dsouza, Senthil Velan, Preventive Maintenance for Fault Detection in Transfer Nodes Using Machine Learning, 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), IEEE, 2019.
[13] Cynthia S. Hood, Chuanyi Ji, Proactive network-fault detection [telecommunications], IEEE Trans. Reliab. 46 (3) (1997) 333–341.
[14] Okuthe Paul Kogeda, Johnson I. Agbinya, Christian W. Omlin, A probabilistic approach to faults prediction in cellular networks, in: International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06), IEEE, 2006.
[15] P. Kogeda, Johnson I. Agbinya, Prediction of Faults in Cellular Networks Using Bayesian Network Model, International Conference on Wireless Broadband and Ultra Wideband Communication, UTS EPress, 2006.
[16] A. Snow, P. Rastogi, G. Weckman, Assessing Dependability of Wireless Networks Using Neural Networks, MILCOM 2005-2005 IEEE Military Communications Conference, IEEE, 2005.
[17] Huseyin Polat, Onur Polat, Aydin Cetin, Detecting DDoS attacks in software-defined networks through feature selection methods and machine learning models, Sustainability 12 (3) (2020) 1035.
[18] Mowei Wang, et al., Machine learning for networking: workflow, advances and opportunities, IEEE Netw. 32 (2) (2017) 92–99.
[19] Zhou Jinglong, et al., Research on Fault Prediction of Distribution Network Based on Large Data, MATEC Web of Conferences., vol. 139, EDP Sciences, 2017.
[20] Lu Xu, et al., Using Hessian Locally Linear Embedding for Autonomic Failure Prediction, 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), IEEE, 2009.
[21] Yong Wang, Margaret Martonosi, Li-Shiuan Peh, Predicting link quality using supervised learning in wireless sensor networks, Mob. Comput. Commun. Rev. 11 (3) (2007) 71–83.
[22] Hassan Hajji, Statistical analysis of network traffic for adaptive faults detection, IEEE Trans. Neural Netw. 16 (5) (2005) 1053–1063.
[23] Umair Sajid Hashmi, Arsalan Darbandi, Ali Imran, Enabling Proactive Self-Healing by Data Mining Network Failure Logs, 2017 International Conference on Computing, Networking and Communications (ICNC), IEEE, 2017.
[24] M. Viktorm, Prognoznaja analitika dlja jeffektivnogo ispol'zovanija oborudovanija, https://filearchive.cnews.ru/files/reviews/2016_03_29/2_Maltsev.pdf, 2016.
[25] M. Oliver, Predictive Maintenance & Service (PdMS), 2014.
[26] A.A. Myrzatay, L.G. Rzayeva, Creation of forecast algorithm for networking hardware malfunction in the context of small number of breakdowns, Int. J. Eng. Res. Technol. 13 (6) (2020) 1243–1248.
[27] Gulnara Abitova, et al., The study of stability control systems for the technological processes, in: Proceedings of the 2014 International Conference on Industrial Engineering and Operations Management, 2014.
[28] Batyrbai Orazbayev, et al., System concept for modelling of technological systems and decision making in their management, 2021, p. 180, Kharkiv: PC TECHNOLOGY CENTER.