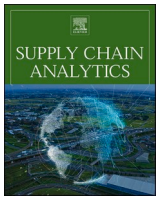
Supply Chain Analytics 9 (2025) 100091

Contents lists available at ScienceDirect

Supply Chain Analytics

journal homepage: www.sciencedirect.com/journal/supply-chain-analytics



A conceptual digital twin framework for supply chain recovery and resilience

Oluwagbenga Victor Ogunsoto [a], Jessica Olivares-Aguila [b,*,1], Waguih ElMaraghy [a]

[a] Intelligent Manufacturing Systems (IMS) Centre, University of Windsor, Windsor, Canada
[b] Shannon School of Business, Cape Breton University, Sydney, Canada

ARTICLE INFO

Keywords:
Supply chain
Digital twin
Supply chain resilience
Supply chain recovery indicator
Digital supply chain twin

ABSTRACT

Amidst escalating global supply system risks and interruptions, the imperative for fortified supply networks is evident. Organizations striving for competitiveness and resilience must adeptly recognize, comprehend, and address disruptions. This study presents a three-phase digital supply chain twin framework, leveraging discrete event simulation and neural networks to anticipate floods—a typical natural catastrophe and disruptive event—and predict recovery indicators. This aids supply chain (SC) managers in making informed decisions. In the first phase, machine learning algorithms, including logistic regression and Long Short-Term Memory (LSTM), were trained on Kerala India's precipitation data to predict floods. LSTM outperforms logistic regression, achieving flood prediction with 73 % recall, 75 % accuracy, and 84 % Area Under Curve-Receiver Operating Characteristics score. In the second phase, simulations replicate value chain breakdowns. A process flow logic-driven discrete event simulation within a real-world SC network emulates operational disruptions. FlexSim is employed to model service-level failures, influencing SC model performance based on the distribution center service level. The third phase employs simulated case scenario data to train a multilayer neural perceptron network (MLPNN) for predicting production network recovery post-disruptions. The MLPNN monitors the mean squared error (MSE) and disruptive inputs throughout training and validation, revealing consistent MSE reduction over recovery periods. The number of epochs needed to achieve a minimum MSE is used as a recovery indicator to predict service restoration time. Consequently, this study introduces a conceptual digital twin framework for catastrophic operations chain breakdowns and recovery prediction. The framework's output assists SC planners in shaping robust strategies by foreseeing disruptions and facilitating recovery.

1. Introduction

Supply Chain (SC) disruptions could have a low probability of occurrence but disproportionately significant effects on businesses. Black swan events or unanticipated occurrences can severely impede distribution networks' effectiveness [1]. To prepare for disruptions within complex multi-tiered systems, SC planners must consider it necessary to implement preventative and corrective measures [2]. However, it is essential to note that disruptions, including natural disasters, biological pandemics, and armed conflicts, have achieved global prominence as causes of corporate losses. Alarmingly, reported losses nearly doubled within two years, from 28 % in 2011 to 48 % in 2013 [3].

This research focuses on floods, a natural calamity notorious for disrupting SC operations. An overwhelming proportion of reported natural disasters, including coastal, riverine, ice jam, and flash floods, demonstrates that flooding is one of the most pervasive disruptions. Flooding accounts for approximately one-third of total damage, making it, along with storms, a common natural disaster. Instances such as the devastating Thailand flood in October 2011, which severely disrupted the supply network operations of computers and Japanese automobile manufacturers operating in Thailand [4], illustrate the significance of this phenomenon causing global repercussions.

Consequently, developing a digital twin (DT) framework is a compelling study area. This digital replica attempts to predict disruptions caused by natural disasters, particularly in flooding and recovery parameters. This tool has the potential to improve SC disruption mitigation and decision-making processes. A digital SC twin (DSCT), a digital representation of a physical SC, emerges as an effective strategy for

* Corresponding author.
E-mail addresses: ogunsot@uwindsor.ca (O.V. Ogunsoto), jessica_olivares@cbu.ca (J. Olivares-Aguila), wem@uwindsor.ca (W. ElMaraghy).
[1] ORCID: 0000–0003-3986–7894

https://doi.org/10.1016/j.sca.2024.100091
Received 6 June 2024; Received in revised form 30 October 2024; Accepted 14 November 2024
Available online 19 November 2024

enhancing SC resilience. DTs can revolutionize SC management by providing predictive insights into disruptive elements, forecasting recovery parameters, and acting as a decision-support tool. Their technological capabilities enable real-time monitoring and SC analytics, providing invaluable insights and responses [5].

Hence, this study uses historical precipitation data and machine learning algorithms alongside TensorFlow, Keras, and Python for neural network training to predict floods. Moreover, SC data is simulated to train an inventory model based on a machine learning controller, providing a recovery indicator. SC planners will be proactive instead of reactive with this disruption prediction tool. Therefore, this framework for the DSCT will assist managers in developing SC plans. The enterprise resource program (ERP), connected to the cloud server, and the trained algorithms provide the production network team with the necessary insights to predict flooding, which has been selected as this study's disruptive value chain element.

The remainder of this article is organized as follows: Section 2 presents the theoretical foundations of the research, the research gap, and the theoretical contributions to the literature body. Section 3 discusses the research methodology and implementation. Section 4 introduces the results, analyses, discussion, and methodology outcomes. The paper concludes by summarizing the research findings and recommendations in Section 5.

2. Literature review

SC disruptions and resilience have attracted significant attention in recent years. Moreover, leveraging digital technologies such as DTs to tackle SC challenges represents opportunities to predict, mitigate, and recover from such disruptions. Hence, this article reviews related works on SC disruptions, resilience, and DSCTs.

2.1. Supply chain disruptions and resilience

SC risks can be classified into two broad categories: operational and disruptive risks [6]. Disruption risks could highly impact SC performance, underscoring resilience as a key determinant of long-term success while integrating agility, flexibility, and collaboration [7]. Tang [8] provided a seminal work that unified SC risk management and proposed strategies that balance efficiency with risk mitigation. That work also highlighted the scarcity of models for managing disruption risks.

SC resilience is defined as the adaptive capability of the SC to be prepared for unexpected events and respond and recover to its original state [9]. Hence, Christopher et al. [10] laid a foundational framework for building resilient SCs, identifying visibility, velocity, and flexibility as crucial components. That research highlighted the necessity of incorporating resilience into SC design to ensure quick responses to disruptions. Further advancing the discourse, Blackhurst et al. [11] identified critical research areas in managing SC disruptions, emphasizing the need for strategic frameworks to enhance resilience.

The resilience of essential infrastructure components was investigated in [12], focusing on the interaction between SCs and core infrastructure and promoting the use of integrated risk management methodologies. Moreover, Petit et al. [13] examined the integration of interdependencies in critical infrastructure protection, emphasizing the requirement for comprehensive approaches to handle cascading impacts across interconnected systems networks. Meanwhile, the U.S. government's policy framework for Critical Infrastructure Security and Resilience [14], emphasized the protection of vital assets, directly influencing SC continuity.

Disruption management starts with resilience evaluation involving network performance analysis and recovery effort calculation [15]. Therefore, awareness and preparedness are imperative to overcoming and quickly recovering from disruption risks by exploring mitigating techniques to achieve resilience [16]. Several authors have identified key strategies such as redundancy and network reconfiguration [17], inventory management, and dual sourcing [18], flexible manufacturing [3], and supplier diversification [19] as essential for building resilience and effective recovery. In the same vein, Aldrighetti et al. [20] investigated the balance between preparedness and recovery investments in designing a resilience portfolio, proposing a model that optimizes these investments to enhance SC resilience.

Resilience is not just about bouncing back from disruptions but also about anticipating potential disruptions and adapting to changing circumstances in real-time [21]. Several quantitative methods for measuring SC resilience (e.g., optimization, simulation, and network analysis) have been proposed in the literature [22]. For instance, Lücker et al. [18] stated that the current optimization and simulation models aided in stress testing existing SC designs and deploying contingency and recovery plans. Moreover, with the advent of Industry 4.0 technologies, resilience capabilities could be enhanced [23] by examining the role of DTs through effective information processing and risk management [24]. Hence, decision-making support tools to recognize disruptions in real-time, monitor and identify them, and plan for the time of interruption and recovery could be developed [25].

2.2. Digital supply chain twins

The SC landscape has been transformed by the introduction of DT technologies, which allow the interaction of optimization, simulation, and data analytics capabilities to enhance the decision-making process. Grieves and Vickers [26] proposed the DT concept to reduce unpredictable and unwanted behavior in complex systems by generating precise virtual reproductions of physical processes, setting the foundations for DT research in SCs. Similarly, ElMaraghy and ElMaraghy [27], while introducing the Cognitive DT concept, affirmed that cognitive systems might duplicate the functioning of the human brain by applying self-learning algorithms that use data mining, pattern recognition, and language management.

The use of DTs to enhance operational effectiveness and decision-making in SCs was discussed by Stark and Damerau [28] who proposed a methodology for predicting SC performance that integrates the SC Operations Reference (SCOR) metrics with machine learning (ML), utilizing multilayer perceptron neural networks. Their method shows how predicted accuracy can be increased by fusing cutting-edge neural network techniques with conventional performance indicators. In the same vein, Zhao et al. [29] proposed a multi-mediation model to assess the impact of digitalization on SC resilience, emphasizing the critical role of DTs in enhancing overall performance.

In recent years, the development and implementation of DTs for SCs have attracted increasing interest, and implementations for different SC functions (e.g., procurement, logistics, retail) have been developed [30], [31], and [32]. For instance, Peron [33] introduced a vision for a DT for SC spare parts enabled by additive manufacturing. Negueroles et al. [34] introduced a blockchain-integrated DT framework for logistics and transportation to enhance operational transparency, security, and efficiency.

There is no denying that DTs are urgently needed for mapping supply networks and ensuring visibility, considering SC collapses and adaptations during disruptions [35]. Therefore, Ivanov and Dolgui [5] proposed a DT framework for managing disruption risks in SCs, illustrating how these technologies can simulate disruptions and optimize recovery strategies to enhance resilience. Similarly, Barykin et al. [36] contributed with a conceptual framework for implementing DTs in SCs, emphasizing their potential to improve transparency, collaboration, and risk management. Ivanov [37] also introduced the intelligent DT for SC stress-testing, resilience, and viability assessment, presenting it as an advanced model capable of simulating various stress scenarios. Moreover, Schleifenheimer et al. [38] analyzed the response of the pharmaceutical retail SC to the COVID-19 pandemic, showcasing how DTs can bolster resilience during global crises. Furthermore, Fogli et al. [39] introduced chaos engineering to assess DT resilience, offering a

framework for stress-testing DTs to identify potential vulnerabilities.

In a more limited scope, Hong Lim et al. [40] proposed a DT-enhanced approach for managing disruptions on manufacturing shop floors, arguing that real-time decision-making enabled by DTs can more effectively mitigate the impact of disruptions. Guo and Mantravadi [32] studied the integration of DT technology, highlighting its potential to enhance efficiency and waste reduction in the lean SC management context, focusing on optimizing lean processes through real-time data and predictive analytics.

DTs have been studied to improve sustainability in different sectors. For example, Rigó et al. [30] investigated the benefits of DTs in optimizing operations, reducing waste, and minimizing environmental impact in logistics processes. Likewise, Yevu et al. [41] explored the application of DTs in prefabrication SCs for smart construction, emphasizing their potential to optimize processes and contribute to sustainability by monitoring real-time carbon emissions. In the agri-food sector, Guidani et al. [42] presented a transparent, sustainable DT structure to improve resource utilization, monitor environmental impacts, and ensure sustainability compliance using data analytics and virtual simulations. Also, Maheshwari et al. [31] showed that DT-driven real-time planning improved operational efficiency, waste reduction, and demand responsiveness in a food SC. That study stressed the need for DTs for SC visibility and collaboration. The interaction between SC efficiency, resilience, and sustainability has been recently explored in the literature. Cimino et al. [43] proposed a cyclic and holistic methodology for exploiting DTs to achieve resilience and sustainability in SCs, advocating for continuous improvement and adaptation.

Table 1 shows an overview of papers discussing DSCTs for resilience. Despite the increasing interest in crafting DSCTs for disruption and resilience, the literature review reveals that considering a DSCT framework that studies simultaneous disruption prediction and recovery forecasting for flooding has received little attention.

This research aims to fill these gaps using a conceptual DT framework to predict flooding and the system's recovery. Through the integration of simulation software, ERP, ML, and the Internet of Things (IoT), the DT has some enablement that could help forecast the risk factors and advise. Specifically, attention would be focused on predicting flooding and recovering SC echelons using a DT framework. The possibility of a DT performing these tasks forms the basis of this research. Hence, the main research question is as follows:

(i) How can a DT enhance SC resilience?

In a bid to answer the main research question, another sub-question was asked:

(ii) How can a DT predict a SC disruption and recovery?

2.3. Theoretical contributions to the body of literature

This research contributes to the DSCTs literature in two key areas: First, it identifies gaps in existing approaches to improving SC resilience with DTs, particularly in addressing hydrological disruptions like flooding. Existing research has focused more on biological and operational disruptions, neglecting hydrological forms. This article addresses this gap by utilizing real-world data to model ML algorithms for predicting hydrological disruptions in SCs. Secondly, the research introduces the novel concept of concurrently predicting disruption and SC performance recovery indicators using real-world data to train ML algorithms. This approach involves a real-world SC network and a discrete-event simulation (DES) model built in FlexSim. The recovery algorithm is trained on data generated from FlexSim, marking a significant contribution to the discourse on enhancing SC resilience.

3. Research methodology

An Integrated Definition Method (IDEF) provides the general flow for the research methodology, as illustrated in Fig. 1. This study consists of two levels of IDEF, each focusing on a distinct phase of the research. For the first level, the inputs include historical precipitation data. ML algorithms, along with the TensorFlow and Keras platforms for training neural networks and the Python programming language, are utilized for the analysis. The control factors include data preprocessing, training of the neural networks, and hyperparameter tuning. The outputs from this level are the trained model and performance visuals for flood prediction.

The second level's inputs include the flood prediction trained model, precipitation, and SC data. The prediction model and precipitation data are used to predict floods. Based on the prediction, SC data is analyzed through simulations of various potential scenarios. Hence, the SC performance based on service level is collected and used to train an ML algorithm to evaluate recovery indicators. As a result, a conceptual DT to enhance SC resilience is proposed by integrating the different research phases.

Section 3.1 details the methods for predicting disruptions. Section 3.2 introduces what-if scenarios based on the flood prediction results. Additionally, Section 3.3 discusses the prediction of a recovery indicator.

3.1. First phase: prediction of disruption

The forecasting of flooding is the starting point for this study's first phase. Precipitation data from Kerala, a state in the Republic of India, was employed for scenario building, and real-time disruption data is needed for quick bottleneck identification and recovery deployment. Two ML algorithms, logistic regression and Long Short-Term Memory (LSTM), were used to train the dataset, and a training validation was done to understand the model's performance. Sections 3.1.1 to 3.1.4 explain the steps taken in achieving the first phase.

3.1.1. Logistics regression

As illustrated in Fig. 2, the flood prediction process was implemented using logistic regression, leveraging historical precipitation data. The computational procedures were executed by applying key ML functions, including the *Sigmoid* function, hypothesis function, cost function, and gradient descent, which collectively enabled the training and

Table 1
Representative articles on DSCTs for resilience.

Author	Research strategy	Application	Tool
[5]	Conceptual framework for DSCT	Industry 4.0 environments	-
[36]	Conceptual framework for SC risks	General SCs	-
[37]	Conceptual framework for intelligent DSCT - stress testing	General SCs	-
[38]	Case study - COVID−19	Pharmaceutical SCs	Simulation (AnyLogistix)
[40]	Case study	Manufacturing shop floor	Simulation
[44]	Case study - post-disruption recovery for COVID−19	Food retail	Simulation (AnyLogistix)
[45]	Case study - operational & disruption risks	Semiconductor industry	Simulation (AnyLogistix)
[35]	Case study - epidemic outbreaks	Global SCs	Simulation (AnyLogistix)
[46]	Conceptual framework for DSCT - supplier reliability	Perishable Food SCs	Simulation (SimPy) and ML algorithms
[47]	Case study – supplier management	Solar water pump system	Simulation and multi-criteria model
[43]	DT strategy for resilience and sustainability	Agri-food sector	Simulation (AnyLogistix)
This paper	Conceptual DSCT framework for resilience	Floods affecting SCs	Simulation (FlexSim) and ML algorithms

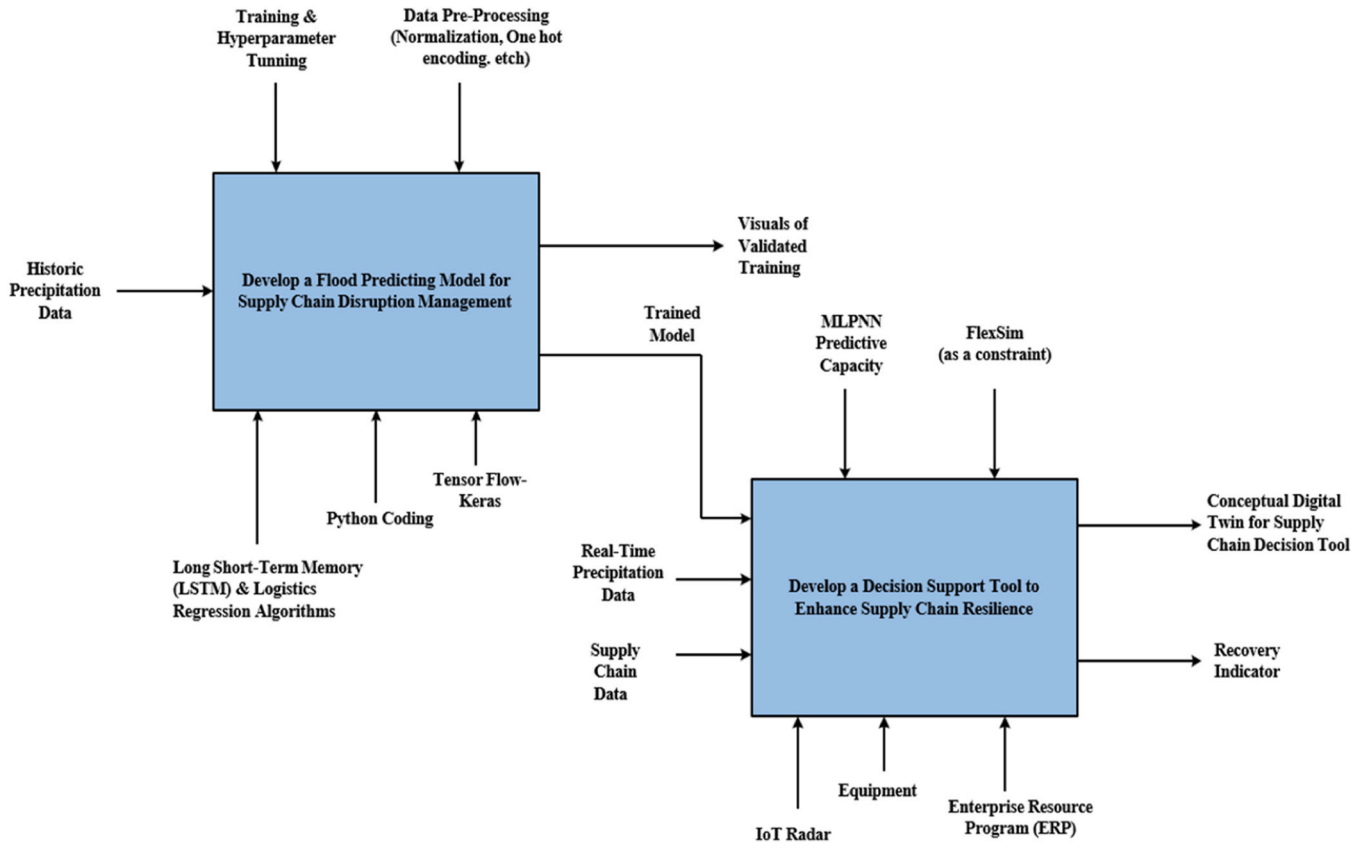


Fig. 1. IDEF diagram of the current investigation.

optimization of this predictive model.

The *Sigmoid* function transforms the linear combination of the input features and coefficients into a probability. The equation for the *Sigmoid* function is:

$$\partial(x) = \frac{1}{(1 + e^{(-x)})} \tag{1}$$

Where x is the linear combination of the input features and coefficients.

The hypothesis function represents the predicted probability of flood occurrence based on the input data. In this study, it is defined as:

$$h(x) = \delta(\theta_0 + \theta_1 * x) \tag{2}$$

Where $h(x)$ represents the predicted probability of flood occurrence, x is the precipitation value, and θ_0 and θ_1 are the coefficients (parameters) that need to be learned.

The logistic regression cost function (Log Loss) measures the error between the predicted probabilities and the actual flood occurrence labels. The equation for the log loss is:

$$J(\theta) = -\left(\frac{1}{m}\right)\Sigma[y \quad *\log\ (h(x))*(1-y)*\log(1-h(x))] \tag{3}$$

Where $J(\theta)$ represents the cost function, m is the number of training examples, y is the actual flood occurrence labels, and $h(x)$ is the predicted probability of flood occurrence.

Gradient descent minimizes the cost function and finds the optimal coefficients. The update rule for the coefficients is:

$$\theta j = \theta_j - \alpha * \frac{\partial J(\theta)}{\partial \theta_j} \tag{4}$$

Where θ_j represents the jth coefficient, α is the learning rate, and $\partial J(\theta)/\partial\theta_j$ is the partial derivative of the cost function concerning the jth coefficient.

During the training phase, the logistic regression model learns the optimal coefficients by iteratively updating them using gradient descent until the cost function, $J(\theta)$, is minimized. The outputs are trained models and performance visuals. Appendix 1 presents the pseudocode for predicting floods using logistics regression.

3.1.2. Long short-term memory

Historical precipitation data was the input to the LSTM, as shown in Fig. 3, represented by a sequence of time steps. Each time step consists of a precipitation value at that time. The LSTM cell is the key component of the LSTM model. It consists of different gates and memory units that allow it to capture long-term dependencies. The input sequence is represented as $X = [x_1, x_2, \ldots, x_t]$, where t is the number of time steps. The weights matrices and bias terms for the forget gate, input gate, candidate cell gate, output gate, and output layer are represented using W_f, W_i, W_c, W_o, W_y, *bf, bi, bc, bo*, and *by,* respectively.

The hidden layers consist of the following with the computation of the training process:

(a) Forget Gate: The forget gate determines how much of the previous cell state to retain. It takes the current input (x_t) and the previous hidden state (h_{t-1}) as input and produces a forget gate value, *f(t)*, between 0 and 1 using the *Sigmoid* activation function:

$$f(t) = Sigmoid(W_f * [h_{t-1}, x_t] * bf) \tag{5}$$

The *Sigmoid* function is a mathematical function that maps any real-valued number into the range between 0 and 1, which is particularly useful in various applications such as binary classification and neural networks. It is defined by the formula: $\delta(x) = 1/(1 + e^{-x})$. It is used to determine the forget gate's activation, which regulates the extent to which the previous cell state should be retained or discarded in the

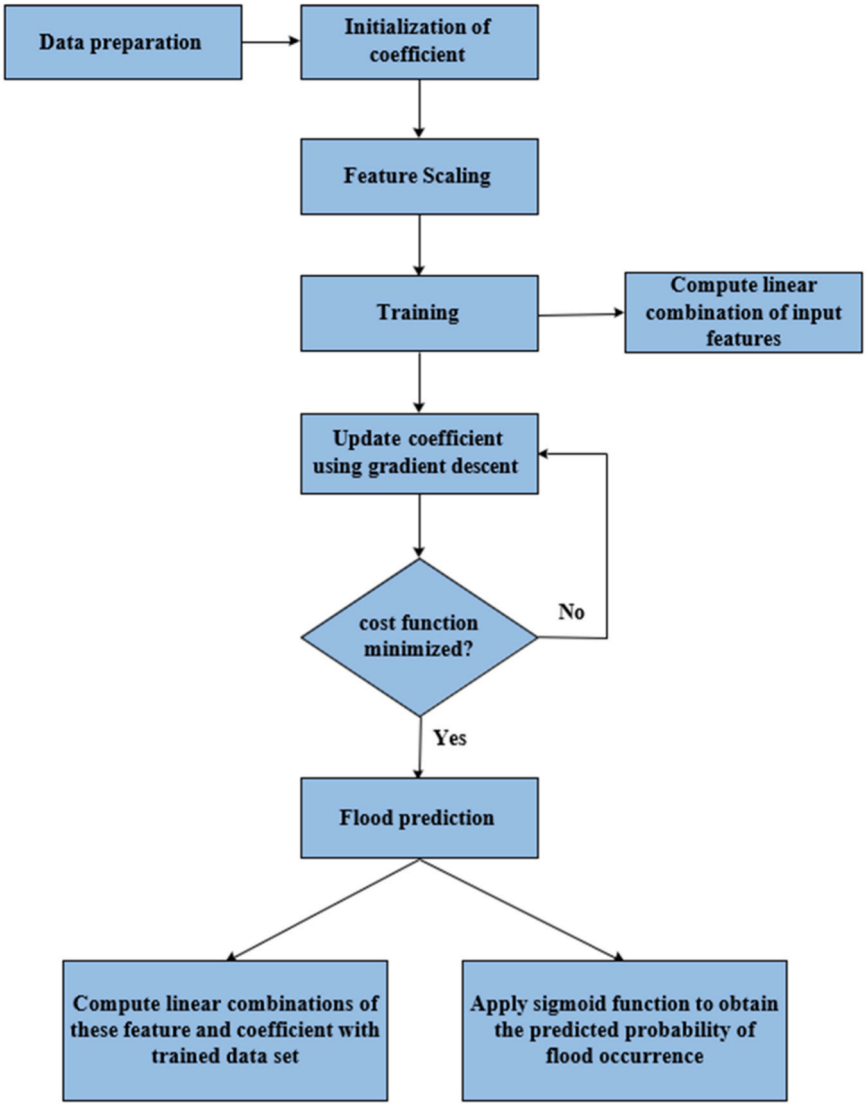


Fig. 2. Illustration of the logistics regression for flood prediction.

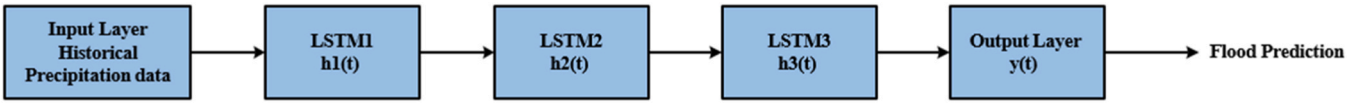


Fig. 3. Illustration of the LSTM for flood prediction.

current cell state by outputting a value between 0 and 1 for each component of the cell state.

(b) Input Gate: The input gate determines how much new information can be stored in the cell state. It takes the current input (x_t) and the previous hidden state (h_{t-1}) as input and produces an input gate value between 0 and 1 using the *Sigmoid* activation function:

$$i(t) = Sigmoid(W_i{*}[h_{t-1}, x_t]{*}bi) \tag{6}$$

(c) Candidate Cell State: The candidate cell state $C'(t)$ represents the new information that can be added to the cell state. It takes the current input (x_t) and the previous hidden state (h_{t-1}) as input and produces a candidate cell state value using the Rectified Linear Unit (ReLU) activation function, defined as ReLU(x) = max (x, 0). This function outputs the input directly if it is positive; otherwise, it outputs zero. ReLU is used to introduce non-linearity into the model, enabling it to learn complex patterns:

$$C'(t) = ReLU(W_c{*}[h_{t-1}, x_t]{*}bc) \tag{7}$$

(d) Update Cell State: The update cell state $C(t)$ combines the forget gate, input gate, and candidate cell state:

$$C(t) = f_t{*}C_{t-1} + i(t){*}C'(t) \tag{8}$$

(e) Output Gate: The output gate determines how much of the cell state should be exposed as the hidden state. It takes the current input (x_t) and the previous hidden state (h_{t-1}) as input and produces an output gate value, $O(t)$, between 0 and 1 using the *Sigmoid* activation function:

$$O(t) = Sigmoid(W_o{*}[h_{t-1}, x_t]{*}bo) \tag{9}$$

(f) Hidden State: The hidden states, $h(t)$, are obtained by applying the output gate to the updated cell state using the ReLU activation function.

$$h(t) = ReLU(C(t))^{*}O(t) \quad (10)$$

(g) Output Layer: The final hidden state, $h(t)$, is fed into the output layer, which produces the prediction $y(t)$ for flood occurrence using a fully connected layer and the *Sigmoid* activation function.

$$y(t) = Sigmoid(W_y)^{*}h(t) + by \quad (11)$$

Appendix 2 presents the pseudocode for the flood prediction using LSTM.

3.1.3. Source of precipitation data

The precipitation data set was obtained from Kerala, India, which has recorded several historic flooding events. The data set contains rainfall data from 1901 to 2015, covering 114 years. Some of these recorded rainfalls led to flooding, while others did not. The input data set covers three parameters: the average rainfall from March to May, the average rainfall in the first ten days of June, and the increase in rainfall from May to June for all the years from 1901 to 2015.

3.1.4. Validation of algorithms performance

The performance of logistic regression and LSTM algorithms was evaluated using 34 % of the dataset, with a confusion matrix and Area Under Curve - Receiver Operating Characteristic (AUC-ROC) curves. Key metrics like recall, precision, and specificity were analyzed to assess model accuracy. Recall measures the ability to predict flooding, precision reflects correct positive classifications, and specificity shows the model's ability to avoid false alarms. High values for these metrics are preferred for optimal model performance.

A benchmark of the performance of the algorithms was done with the work presented in [48], who used five different algorithms to predict flooding. Saravanan et al. [48] used Adaboost, gradient boosting, extreme gradient boosting (XGB), CatBoost, and stochastic gradient boosting (SGB) to predict flooding in the Idukki district of Kerala. The results of the performance evaluation indicators are presented in Section 4.

3.2. Second phase: simulating what-if-scenarios planning

This research uses simulation to assess the impact of flooding on SC performance, particularly service level (SL), by modeling disruptions to infrastructure like power, transportation, and labor. After predicting flood risks, the focus is on evaluating how these disruptions affect SC nodes.

3.2.1. Supply chain structure

This research's SC network structure was selected from one of the 38 real-world multi-echelon SC models in the data set provided in [49]. This data set describes 38 different SCs implemented in practice. The third SC network (i.e., SC-3) from the 38 was chosen for this research because it is a four-level multi-echelon, processes several products (i.e., products A, B, and C), incorporates product assembly and distribution, and has crucial nodes necessary to produce all three products.

Fig. 4 shows the adapted SC network, which has 13 nodes owned and operated by the same company. The SC's ownership by one business enterprise will be the guiding principle of this research. The key activities of the SC—supplying parts, manufacturing and packaging operations, and the distribution of finished goods—have been represented by the letters SP, M, PKG, and DC, respectively, with numbers allotted to each. The SC model has five suppliers (SP1, SP2, SP3, SP4, and SP5) providing parts for the manufacturing and processing plants. The processing plants are divided into manufacturing (M1 and M2) and packaging (PKG1 and PKG2) operations. The last echelon of the SC network consists of four distribution centers (DCs) (DC1, DC2, DC3, and DC4). The original SC had transportation segments between the second and third echelons but has been modified into Fig. 4, with the transportation lead time added to the preceding stages. In this SC model, three final items are sold at four sites (i.e., DC1, DC2, DC3, and DC4). Sites DC3 and DC4 sell the same product, A, while sites DC1 and DC2 sell items B and C, respectively.

3.2.2. Safety stock placement and model formulation

This study used the Guaranteed-Service Model for inventory policy by Graves & Willems [50]. This methodology guarantees client service at each stage before disruption. A supply stage sets a service time for its downstream customer, who must have enough inventory. This model assumes demand is bound to assure service time. Thus, a finite list can ensure service time. These servicing periods guarantee reliable and deterministic downstream replenishment. Safety stock placement allowed the downstream stage to organize its inventory and assure service time. Moreover, Graves & Willems [50] provided a policy and computational logic that defined the determination of the safety stock policy of this SC model. Notations to explain the terms used in determining the guaranteed service model are contained in Table 2.

The inventory at every node, j, is calculated as follows:

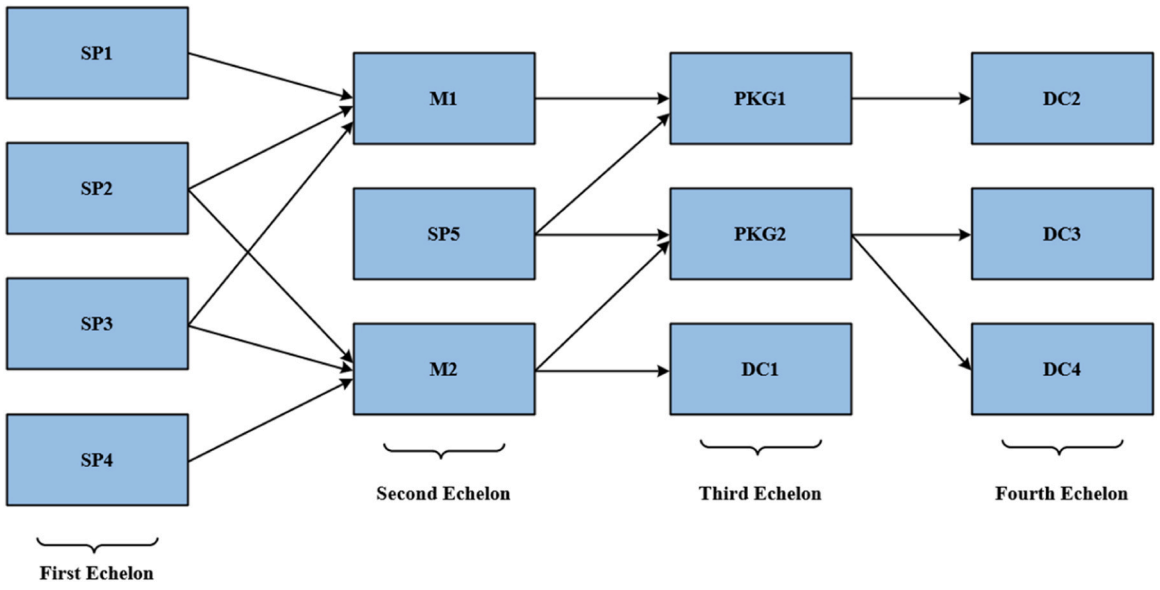


Fig. 4. Four Echelon SC (adapted from [49]).

Table 2
Definition of parameters.

Notations	Definition
S_j^{out}	Outbound service time to the immediate downstream stage(s)
S_j^{in}	Inbound service time to a stage j
$D_{j(t)}$	Maximum demand over t consecutive periods
$I_{j\ (t)}$	Inventory position of node *j* at time *t*.
$F_j\ (t)$	Quantity of finished goods at node *j* at time *t*.
$WPj(t)$	Work-in-process inventory for node *j* at time *t*.
$PQj(t)$	Inventory waiting production for node *j* at time *t*.
B_j	Calculated base stock level for stage *j*.
σ_j, μ_j	Mean and Standard Deviation of demand, respectively
T_j	Net replenish time at node j.

$$I_j(t) = F_j(t) + WP_j(t) + PQj(t) \tag{12}$$

According to [50], the order quantity from downstream node j to upstream node (j-1) is given as:

$$Q(t)(j \rightarrow j-1) = Bj - Ij(t) \tag{13}$$

Where, $B_j = D_j(Sj^{in} + Lj - Sj^{out})$

The net replenishment time for stage j is the replenishment time minus the stage's outbound service, and it is represented as:

$$T_j = Sj^{in} + L_j - Sj^{out} \tag{14}$$

The data set referenced in Section 3.2.4 provides both mean and standard deviations for stage lead times and a set of discrete stage times with corresponding probabilities.

3.2.3. Modeling of the supply chain disruption

The SC modeling was carried out by considering the following parameters: the duration of the disruption, the severity of the supply disruption, and the node location of trouble. The duration of the disruption used in this study to carry out simulations was pegged at 600 simulation days. The days mentioned in this research were converted into hours and seconds for simulation. For example, the disruption in this model started on day 200, which corresponded to hour 4800. Concerning supply disruption severity, reduced operating capacity is a more feasible and practical study option to pursue during a disruption caused by flooding; hence, this situation was investigated. As a result, this study compared the severity ratings of partial SC disruptions at two levels: a moderate level of 65 % and a severe level of 35 % operational capacity.

Blackhurst et al. [11] used "node criticality" to choose SC disruption test nodes for the site of disruption within the SC. Node criticality requires selecting nodes using SC functionality. Hence, nodes that affect several products or require multiple successor nodes were chosen for this research. The suppliers were not considered for any disturbance based on the initial assumption that they have an infinite supply of items. However, the manufacturing and packaging plants, M1, M2, PKG1, and PKG2, were chosen as the nodes where the stress test will occur.

3.2.4. Simulation setup

M1, M2, PKG1, and PKG2 are critical to this network. Hence, FlexSim was used to simulate the performance responses when these nodes could not deliver at total capacity due to disruptions. Initial values were set as starting inventories for all thirteen nodes. The initial base values are seen in Table 3.

A minimum inventory level (represented as *Ij(t)* in Table 3) that must be maintained during the regular operation was set. At the beginning of the simulation, these various values were pushed to each node through FlexSim's process flow logic. The demands at nodes DC1, DC2, DC3, and DC4 followed a normal distribution, with values of means and standard deviation obtained from [49] are shown in Table 4.

First, a Monte Carlo simulation was run on the demand values using the Excel Add-in (Analyze), a normal distribution, and six seeds to make 600 random demand data points for DC1, DC2, DC3, and DC4 based on the average and standard deviation of the demand values from Table 4. These values were uploaded into FlexSim's global table for the simulation's onward random pulling of items. Some demand values yielded negative values during the Monte Carlo simulation, which were shifted to zero since part of the assumption of this work is that there is no backorder. Service inbound and outbound times are expressed in days for nodes DC1 to DC2 and were obtained from [49].

Table 5 shows the combination of service times and demand per node. It also describes the supply schedule from each preceding stage to the following location of the SC network. The process flow customized logic was programmed in C++ to implement the supply schedule, considering each node's service times and demand. The DES model, using FlexSim, started with a warm-up period of 100 days. The steady-state period came at the end of the warm-up period, which lasted another 100 days. During this time, the performance measures at the four DCs were captured using FlexSim's statistics tool. These measures were delivered products, average output per hour, content per hour, and SL. The SL was calculated using process flow to compute the ratio of the delivered products to the demanded items per day.

After the steady-state period, the capacity of location nodes M1, M2, PKG1, and PKG2 was reduced to impose disruption on them. The capacity reduction levels were 35 %, which translated to an operational capacity of 65 % and 65 %, which amounted to 35 % operating capacity for each of the chosen nodes. Table 6 shows the different scenarios.

The imposed disruption was carried out for 200 days of simulation time to observe the effects these will have on the SL for DC1, DC2, DC3, and DC4. On day 400, the capacity constraint representing disruption was removed from the nodes, and the simulation was made to run for another 200 days of simulation time. A total of 15 replications for each combination of scenarios were carried out using the experimental feature of FlexSim. The results, showing the graphs for the different combinations of scenarios, are presented and analyzed in Section 4.2. Fig. 5 provides a summary of the simulation setup.

3.3. Third phase: prediction of recovery indicator

Having simulated the what-if scenarios to determine the impacts of different disruption levels on SL, the data generated was used to predict a recovery indicator for SC planners, which is one of the significant contributions of this research. The projection of recovery indicators involves training a multilayer neural perceptron network (MLPNN) using the data generated in FlexSim. The architecture of the MLPNN for this

Table 4
Average and standard deviation demand data.

Nodes	Average Demand (Mean)	Standard Deviation
DC1	70	35
DC2	126	32.3
DC3	57	51.3
DC4	46	45

Table 3
Base stock and minimum inventory values at the nodes.

Nodes	SP1	SP2	SP3	SP4	SP5	M1	M2	PKG1	PKG2	DC1	DC2	DC3	DC4
B_j	1300	2400	950	1700	10000	11000	9000	250	200	300	1900	1000	800
$I_j(t)$	1200	2300	800	1500	895	9850	7500	196	150	265	1460	766	571

Table 5
Schedule supply at the stages of the network.

Stages	SP1	SP2	SP3	SP4	SP5	M1	M2	PKG1	PKG2	DC1	DC2	DC3	DC4
Inbound Service Time, S_j^{in} (days)	0	0	0	0	0	56	56	2	2	2	10	10	10
Outbound Service Time, S_j^{out} (days)	47	39	56	28	33	2	2	10	12	0	0	0	0
Demand at Each node, μ_j	120	150	150	173	115	120	150	120	52	70	126	57	46

Table 6
Scenario definitions based on node location and percentage of disruption.

Scenarios	Location (Node)	Operational Capacity
Scenario 1 (S1)	All the nodes	100 %
Scenario 2 (S2)	Manufacturing Plant 1 (M1)	65 %
Scenario 3 (S3)	Manufacturing Plant 1(M1)	35 %
Scenario 4 (S4)	Manufacturing Plant 2 (M2)	65 %
Scenario 5 (S5)	Manufacturing Plant 2 (M2)	35 %
Scenario 6 (S6)	Packaging Plant 1(PKG1)	65 %
Scenario 7 (S7)	Packaging Plant 1(PKG1)	35 %
Scenario 8 (S8)	Packaging Plant 2(PKG2)	65 %
Scenario 9 (S9)	Packaging Plant 2(PKG2)	35 %

research is shown in Fig. 6.

The first layer is dense with 16 neurons and a ReLU activation function. The model has a batch size of 16, meaning that 16 samples of the data features are fed into the input layer simultaneously, hence the 16 neurons at the input layer. The input to the network is denoted as X, a vector of shape (3,). In this context, the term "shape" represents the dimensions of a tensor, and the tuple (3,) specifically indicates that the vector has three elements along a single axis.

The elements denote the three input features (obtained from the DES model, which include average demand, average SL during disruption, and average level of disruption) used in training the algorithm. The computation of the policy of this algorithm in the first layer is represented by Eqs. (15) and (16) thus:

$$Z1 = W1{*}X + W2{*}X + \ldots + W16{*}X + b1 \tag{15}$$

$$A1 = ReLU(Z1) \tag{16}$$

Here, W1 represents the weight matrix of shape (16, 3), b1 represents the bias vector of shape (16,), and ReLU(Z1) applies the ReLU activation function element-wise to the values in Z1.

The second layer is a dense layer with eight neurons and a ReLU activation function, and the forward computation process is expressed by Eqs. (17) and (18).

$$Z2 = W17{*}X + W18{*}X + \ldots + W32{*}X + b2 \tag{17}$$

$$A2 = ReLU(Z2) \tag{18}$$

In this case, W17 represents the weight matrix of shape (8, 16), b2

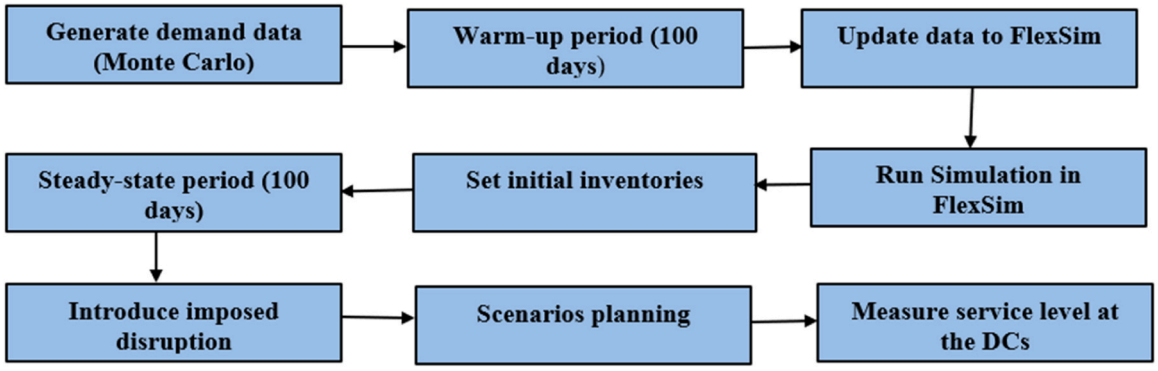


Fig. 5. Flowchart illustrating the simulation set-up.

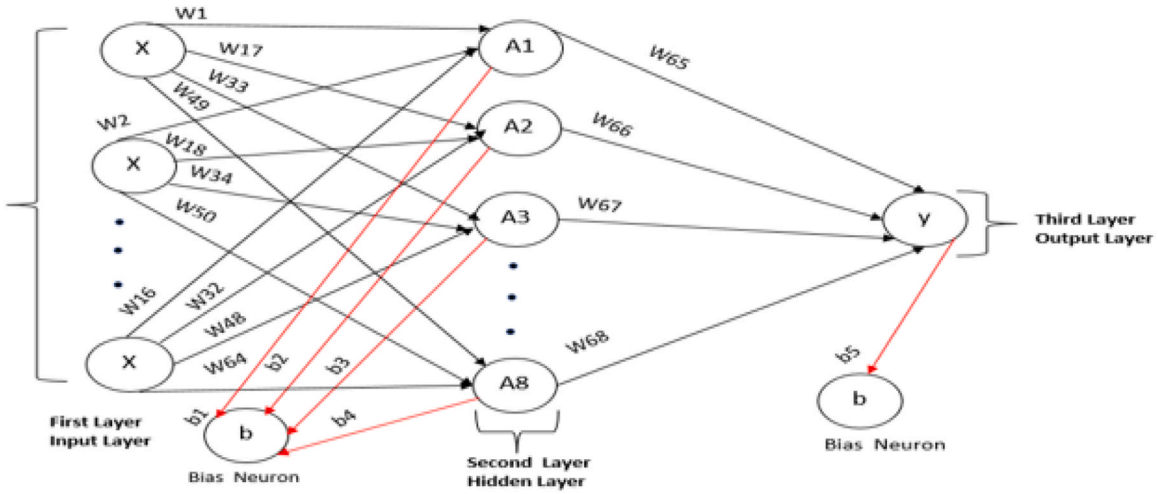


Fig. 6. MLPNN architecture for implementing recovery indicator prediction, showing the weights, biases, activation functions, and inputs.

represents the bias vector of shape (8,), X denotes the inputs (average demand, average SL during disruption, and average level of disruption), and ReLU(Z2) applies the rectified linear unit activation function element-wise to the values in Z2. Also, A3 is calculated in Eqs. (19) and (20).

$$Z3 = W33*X + W34*X + . \quad . \quad . + W48X + b3 \tag{19}$$

$$A3 = ReLu(Z3) \tag{20}$$

Similarly, A8 is calculated as follows:

$$Z8 = W49*X + W50*X + . \quad . \quad . + W64*X + b4 \tag{21}$$

$$A8 = ReLU(Z8) \tag{22}$$

The output layer is a dense layer with one neuron and a linear activation function derived computationally through Eqs. (23) and (24) thus:

$$Z9 = W65*A1 + W66*A2 + W67*A3 + . \quad . \quad . + W68*A8 + b5 \tag{23}$$

$$A9 = Z9 \tag{24}$$

In the third layer, W65, W66, W67, and W68 represent the weight matrixes of shape (1, 8), b5 represents the bias vector of shape (1,), and the linear activation function is applied directly, resulting in A9 equal to Z9.

In summary, the MLPNN can be represented mathematically as a composition of the above equations, with the ReLU activation function applied to the first two layers and the linear activation function applied to the output layer. The mean absolute error (MSE), commonly used for regression problems, is the loss function employed. Adam, an efficient stochastic gradient descent algorithm, is the optimizer, with MSE and root means squared error as evaluation metrics.

During the training of MLPNN, three input features were used: the average and standard deviation of the SL at 65 % and 35 % interruptions, the average percentage of total disruption, and the 600 variable demand data points. Regarding the interruption portion, the same standard deviation of SL was utilized to generate 600 random data points using Monte Carlo simulation in the Excel add-in. The target of prediction was the average SL obtained during the pre-disruption period when there was no disturbance. The average SL of 93.69 % for DC1 and 90.38 % for DC2, as shown in Table 7, were set as the target outputs. Hence, the training and test losses were monitored as these input features were trained to predict those values.

As partial disruptions of 65 % and 35 % were imposed on node M2, the SL at DC1 and DC2 dropped from 93.69 % and 90.38–83.9 % and 82.13 %, respectively, with standard deviation values of 2.99 and 2.84. Table 8 shows the summary of SL reductions during the imposed disruptions.

During the training process, the model learns from the training data. Its performance (based on the MSE reduction) gradually improves as it adjusts its weights and biases to minimize the loss function. The model fitting to the training set took place using 100 epochs and 16-epoch batches. Appendix 3 presents the pseudocode to obtain the recovery indicator prediction using MLPNN.

Table 7
Average SL at DCs 1–4 before disruption.

Distribution centers output	Amount Delivered	SL
DC1 customers demand	15507	93.69
Amount DC1 delivered to customers	14529	
DC2 customers demand	11765	90.38
Amount DC2 delivered to customers	10633	
DC3 customers demand	6208	89.66
Amount DC3 delivered to customers	5566	
DC4 customers demand	5733	87.82
Amount DC4 delivered to customers	5035	

The learning curve in Fig. 7 shows how the model's performance for S5 changed over time as it was trained over the epoch. Section 4.2.2 shows the remaining performance plots for S4, S6, and S7. A hyperparameter known as the number of epochs determines the number of times the learning algorithm will iterate through the whole training dataset and allows the update of the internal model parameters. Hence, the plot indicates if the error rate decreases over epochs and at which epoch the model performance is optimal. The lower the loss becomes, the better the model's performance.

In this study, the epoch—a hyperparameter typically used to determine how many times a model passes through a dataset during training—was employed as a recovery indicator due to different datasets' varying characteristics and complexities. The optimal number of epochs required for model convergence can differ across datasets based on factors like data distribution, variability, size, and complexity. For instance, datasets with distinct patterns may converge faster, whereas more complex or noisy datasets might require additional epochs. This study trained the algorithm using outputs from what-if scenario simulations, reflecting diverse input datasets.

The relationship between resilience and performance indicators, such as the epoch, is crucial. When assessing an SC's resilience, it focuses on its ability to absorb shocks, recover swiftly, and maintain operations efficiently during disruptions. Hence, the number of epochs needed for model convergence, derived from training and test data obtained through simulations, directly correlates with the SC's time to reach its pre-disruption state. For example, different test MSE values were observed for the same epoch, indicating that the data from DES influenced model convergence.

Understanding the recovery indicator (epoch) allows SC planners to simulate optimal data parameters, including the extent of disruption, represented by reduced operational capacity in this study. The algorithm's resulting epoch directly relates to the predicted pre-disruption SL, a critical performance indicator. By leveraging this knowledge, SC planners can make informed decisions to mitigate disruption impacts and enhance SC resilience.

4. Results and analyses

This section will divide the results and analysis into three phases: prediction of flooding, simulation of real-world SC network disruption scenarios, and prognosis of recovery indicators.

4.1. Analysis of flooding forecasting using logistic regression and long short-term memory

The prediction performance of disruptions due to flooding, using logistic regression and LSTM models, was evaluated through the confusion matrix and AUC-ROC analysis. This evaluation involved comparing the models' predictions against actual outcomes. The confusion matrix contains four key metrics: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). In this context, TP reflects correctly identified flooding events, TN represents accurately predicted non-flood events, FP indicates non-flood events wrongly classified as floods, and FN denotes missed flood predictions. These metrics provide the basis for calculating accuracy, precision, and recall, comprehensively evaluating the model's predictive capabilities. The confusion matrices for logistic regression and LSTM are illustrated in Fig. 8. Using the matrices in Fig. 8, performance metrics such as true positive rate (TPR) or recall, false negative rate (FNR), true negative rate (TNR), false positive rate (FPR), and accuracy were computed. A total of 116 data points were used, of which 34 % were used as the test data. The total numbers on the confusion matrix represented how the test data, 68 in total, performed. Table 9 shows the performance analysis of the logistics regression and LSTM models.

Also, the AUC-ROC for logistics regression and LSTM, shown in Fig. 9, was used to evaluate the models' performance. LSTM performs

Table 8
SL at DC1 when M2 was operating at 65 % and 35 % capacities.

Scenario	Mean (95% Confidence Interval)	Standard Deviation	Min	Max
S1	90.53 ± 1.76	3.18	85.31	95.88
S2	90.53 ± 1.76	3.18	85.31	95.88
S3	90.53 ± 1.76	3.18	85.31	95.88
S4	83.90 ± 1.65	2.99	78.80	88.51
S5	82.13 ± 1.57	2.84	77.95	86.58
S6	92.36 ± 1.82	3.29	86.95	97.80
S7	93.24 ± 1.82	3.29	88.66	98.54
S8	93.12 ± 1.63	2.95	87.88	97.76
S9	94.41 ± 1.72	3.11	89.63	99.91

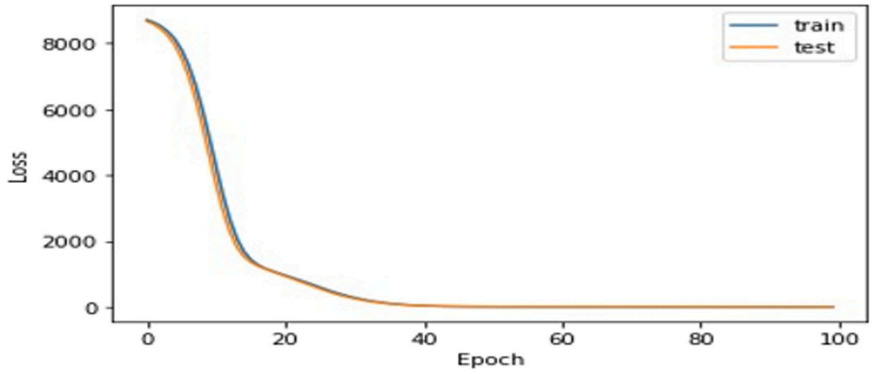


Fig. 7. Learning curve showing the training and test over the epoch.

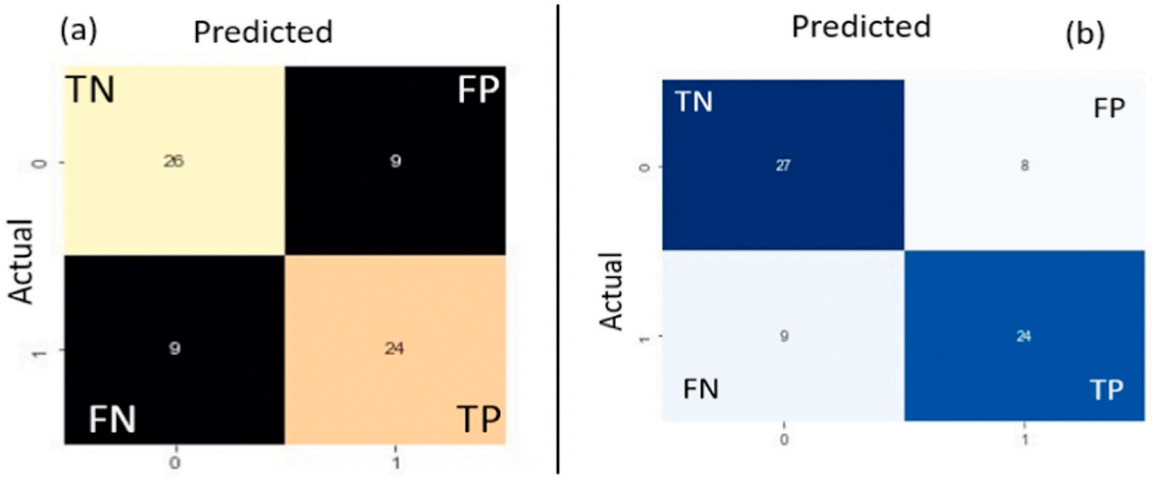


Fig. 8. Confusion matrices for (a) the logistics regression model and (b) LSTM.

slightly better than logistic regression, achieving 73 % recall, 75 % accuracy, and 84 % AUC-ROC score in flood prediction.

4.2. Analysis of case-study SC network and imposed disruptions

Fig. 4 illustrates an SC network modeled using a DES model in FlexSim to evaluate its behavior under full and partial operational capacities. As detailed in Section 3.2.4, operational disruptions of 65 % and 35 % capacity were introduced into the SC model. The primary performance metric assessed was the SL, defined as the ratio of delivered goods to requested quantities. This metric was measured at DC1, DC2, DC3, and DC4. The Experimenter imposed these disruption levels by adjusting the number of finished inventories and the processing time at the affected nodes according to the customized instructions from the process flow. Sections 4.2.1 and 4.2.2 analyze the disruptions' impact on SL at 65 % and 35 % operational capacities across the DCs. Before

Table 9
Performance comparison for logistics regression and LSTM.

Classification	Formula	Logistics Regression		LSTM	
TPR	TP/ (TP +FN)	24/ (24+9)=	0.73	24/(24+9) =	0.73
FNR	FN/ (TP +FN)	9/(24+9)=	0.27	9/ (24+9)=	0.27
TNR	TN/ (TN +FP)	26/ (26+9)=	0.74	27/ (27+8)=	0.77
FPR	FP/ (TN +FP)	9/(26+9)=	0.26	8/ (27+8)=	0.23
Accuracy	(TP + TN) / (TP + FP + FN + TN)	(24+26)/ 68 =	0.74	(24+27)/ 68=	0.75
AUC-ROC		**0.78**		**0.84**	

introducing these disruptions, Table 7—extracted from FlexSim's simulation environment—presents the baseline average SL at all distribution centers. A total of 9 scenarios, with 15 replications, were simulated. This process accounted for a total of 135 experiments conducted by the experimenter.

4.2.1. Evaluation of performance at DC1, DC2, DC3 and DC4

The earlier assumption in this research was that suppliers 1–5 have unlimited inputs and are always available to deliver items. Therefore, it is possible to extract from the SC network in Fig. 4 the flow of services to DC1, DC2, DC3, and DC4, which are represented in Fig. 10 (a), (b), and (c). All three separate product lines are sub-diagrams of the main SC network shown in Fig. 4.

The flow of services from the upstream echelons before it gets to DC1 would first be shown to fully capture the effects of disruption on the concerned node. Hence, disruption was imposed on M2 (scenarios 4 and 5 shown in Fig. 11), directly feeding DC1 and PKG2.

Fig. 11 illustrates the significant impact of scenarios S4 and S5 on the SL performance of DC1. Each scenario in Fig. 11 has 15 dot symbols in different colors. For example, in Fig. 11, S4 and S5 have 15 olive green and aqua blue dots, indicating 15 replications for each case. As observed in Fig. 10 (a), M2 is the immediate upstream supplier to DC1, resulting in a marked reduction in SL under these scenarios. The SL for DC1 in response to S4 and S5 decreased to 83.90 and 82.13, respectively, as presented in Table 10.

Fig. 12 captures DC2's performance during the different disruption levels. For DC2, the SL dropped from 90.38 (as seen in Table 7) to 81.70 in S6 and 79.72 in S7 (as seen in Table 10).

DC3 and DC4 had the same flow path, as shown in Fig. 11(c). Therefore, disruption at PKG2 reduced their performance in S8 and S9, as shown in Fig. 13 and Fig. 14, respectively. Although DC3 and DC4 had the same flow path, the effects of this disruption were not the same based on the statistical values of S8 and S9 in Table 10.

4.2.2. Analysis of the prediction of the recovery indicator

Section 3.3 provided detailed information regarding building the model for predicting the recovery indicator. Data from FlexSim was used to train an MLPNN. Data from DC1 and DC2 have been used to forecast the recovery indicator. DC1 and DC2 have a total of four scenarios. S4 and S5 were simulated for DC1, while S6 and S7 were carried out for DC2.

4.2.3. Recovery indicator prediction using data from scenarios 4 and 5

This section used data from DC1, specifically from S5 and S4, to train the MLPNN model. For S5, the model was trained using the input combinations of the operational capacity of 35 % and an average SL of 82.13 % during the disruption, with a standard deviation of 2.84, as detailed in Section 3.3. The objective was to predict the average SL of DC1 before the disruption, with performance assessed by monitoring the learning curve across epochs. During the training phase, the model iteratively improved its performance by adjusting its weights and biases to minimize the loss function. The learning curve, depicted in Fig. 15 (a), plots the loss function (MSE) against the number of training epochs. The learning curve demonstrated significant performance gains during the initial epochs, eventually reaching a plateau at approximately epoch 30. The training loss continued to decrease, while the validation loss stabilized around epoch 60. The optimal model performance was observed at epoch 100 with a final MSE of 2.06, suggesting that the model achieved a prediction accuracy of 93.69 %, with minimal error levels.

Similarly, the MLPNN was trained with data from S4, which also had a 65 % operational capacity but maintained an average SL of 83.90 % during the disruption, with a standard deviation of 1.65. The demand data remained consistent with that used for S5. The model's performance was monitored over the epochs, as illustrated in Fig. 15 (b), yielding an MSE of 1.34.

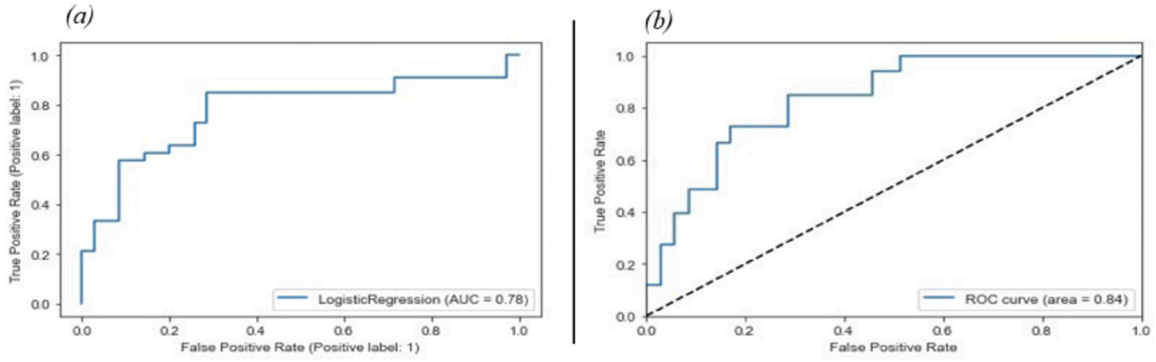


Fig. 9. AUC-ROC for the disruption prediction model for a) logistics regression and b) LSTM.

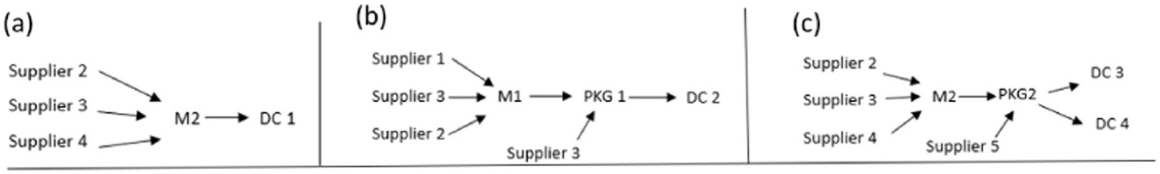


Fig. 10. The flow path of items and activities to a) DC1, b) DC2, and c) DC3 and DC4.

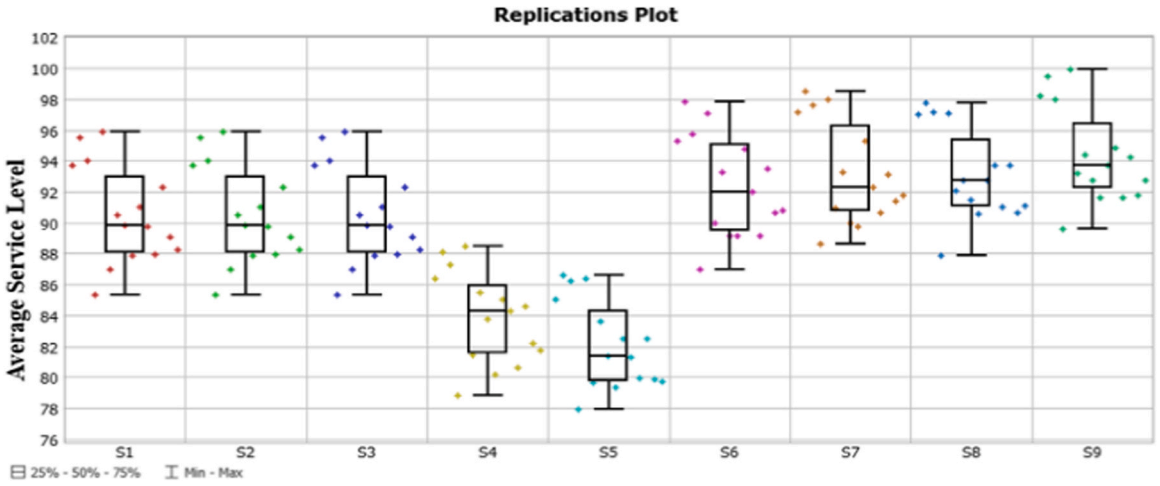


Fig. 11. Box Plot showing the performance (average SL) of DC1 for scenarios S4 and S5.

Table 10
Statistical breakdown of average SL at the DCs for all the scenarios.

DC	Scenario	Mean (95 % Confidence Interval)	Standard Deviation	Min	Max
DC1	S4	83.90 ± 1.65	2.99	78.80	88.51
	S5	82.13 ± 1.57	2.84	77.95	86.58
DC2	S6	81.70 ± 0.57	1.04	79.98	83.70
	S7	79.72 ± 0.58	1.04	78.03	81.87
DC3	S8	85.00 ± 0.51	0.92	83.67	87.37
	S9	82.79 ± 0.51	0.94	81.66	85.53
DC4	S8	82.66 ± 0.45	0.82	81.35	83.67
	S9	80.17 ± 0.38	0.69	79.01	81.38

For S4 and S5, the hyperparameter values were kept constant to observe the test MSE's convergence rate. Table 11 compares test MSE at epoch 100 for S5 and S4. From Table 11, the optimal test MSE for S5 was an MSE test value of 2.06 at epoch 100. At this same epoch, the test MSE value for scenario S4 was 1.34. The approximate test MSE value of 2.06 was achieved at epoch 87 for S4.

4.2.4. Validation of the recovery indicator using data from scenarios 6 and 7

The sensitivity analysis for predicting the recovery indicator in Section 4.2.3 was performed using data from DC2, representing scenarios 6 and 7. The same hyperparameter used in S4 and S5 was used in training the MLPNN for S6 and S7. Based on validation MSE, the learning curve results were consistent with the findings from S4 and S5.

In Fig. 16(a), the learning curve for S6 shows that the validation MSE decreases as the number of epochs increases, indicating that the model is improving over time. The test MSE at epoch 100 was 0.14 for S6, as shown in Table 11. The learning curve for S7 is shown in Fig. 16 (b). This was based on simulation data for 35 % operational capacity. The test or validation MSE at epoch 100 was 1.11 for scenario S7, as shown in Table 11. Detailed MSE values across the 100 iterations can be accessed at https://osf.io/u75y6.

At 100 epochs, S6 and S7 exhibited test MSE values of 0.14 and 1.11, respectively, indicating a superior convergence rate for S6. Notably, while S7 reached a test MSE of 1.11 at epoch 100, S6 achieved a lower MSE value at an earlier epoch, specifically at 78, further highlighting its more rapid convergence. This faster reduction in error suggests that S6 more efficiently approaches the pre-disruption target value of 90.38 for

Table 11
Comparison of Test MSE at Epoch 100 for S4, S5, S6 and S7.

Nodes	Scenario	Value of Test MSE	Number of Epoch
DC1	S4	1.34	100
	S5	2.06	100
DC2	S6	0.14	100
	S7	1.11	100

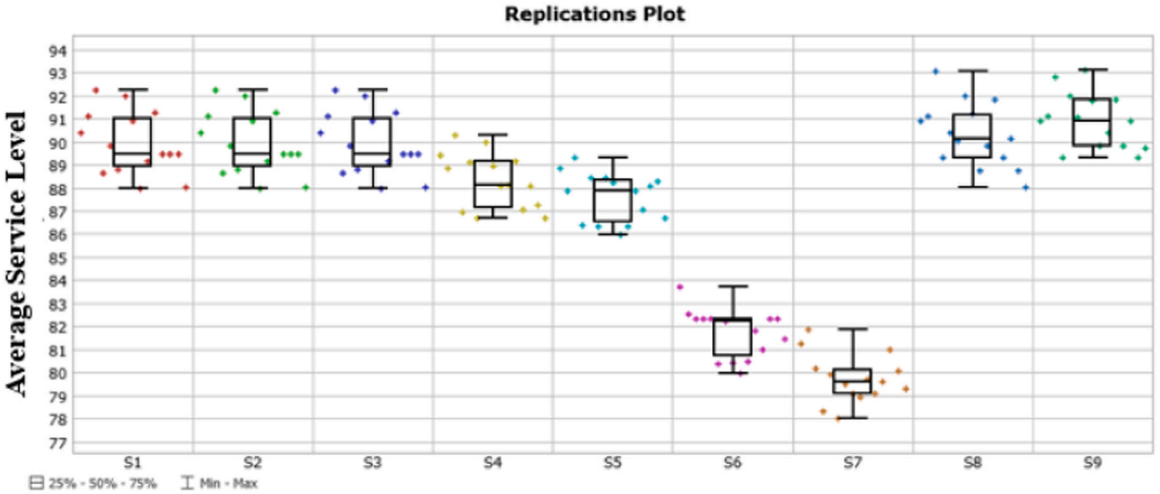


Fig. 12. Box Plot showing the performance (average SL) of DC2 based on scenarios S6 and S7.

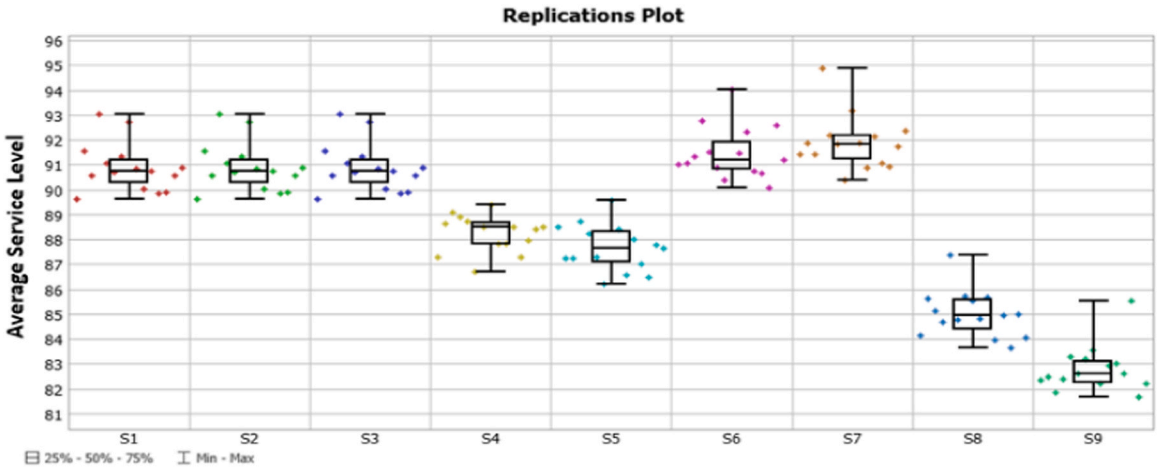


Fig. 13. Box Plot showing the performance (average SL) of DC3 based on S8 and S9.

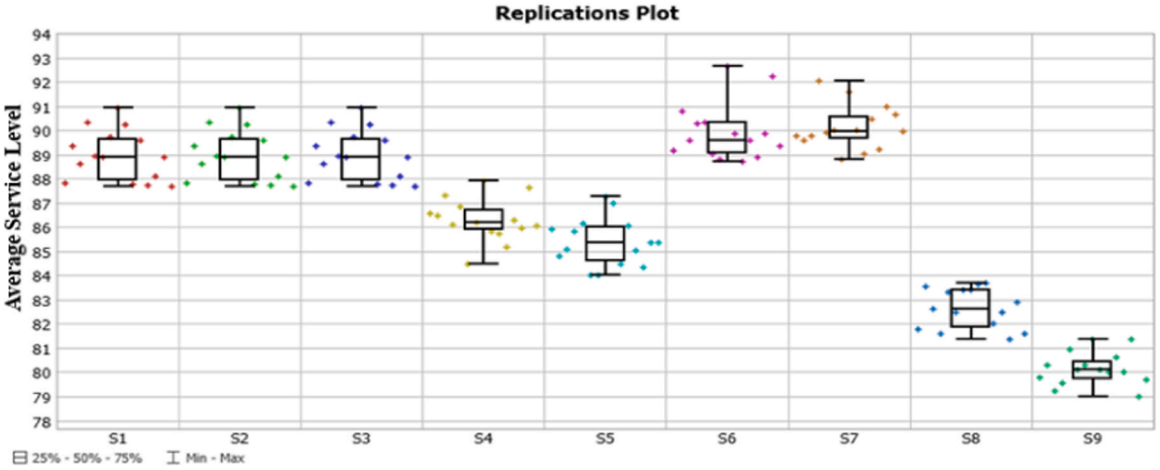


Fig. 14. Box Plot showing the performance (average SL) of DC4 for scenarios S8 and S9.

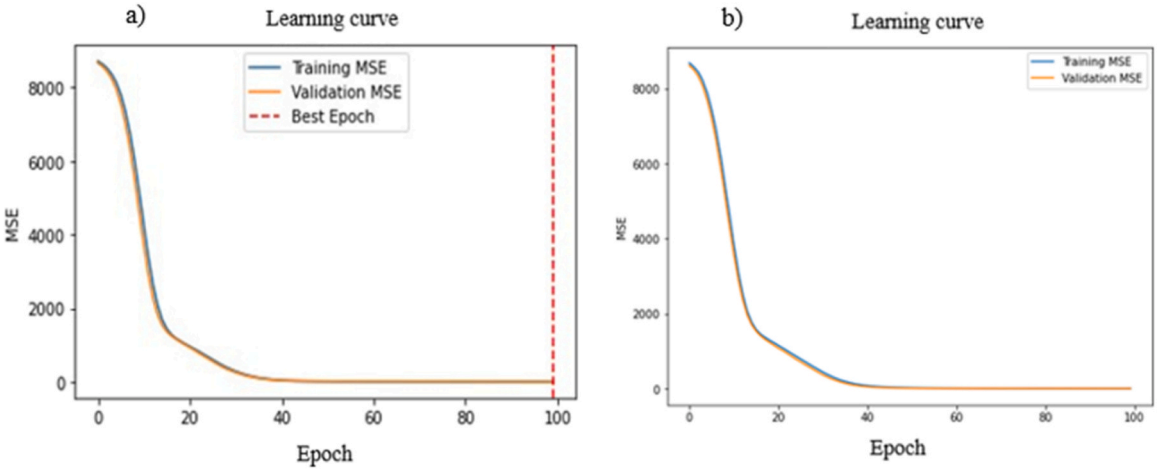


Fig. 15. Learning curve showing MSE values over the epoch for a) S5 and b) S4.

DC2.

It is important to note that the MSE is calculated by averaging the squared differences between predicted and actual values, ensuring a positive result regardless of the sign of the deviations. Hence, the MSE value ideally should be as close to zero as possible. The squaring of errors means that larger discrepancies are penalized more heavily than smaller ones, underscoring the importance of minimizing significant errors in model predictions.

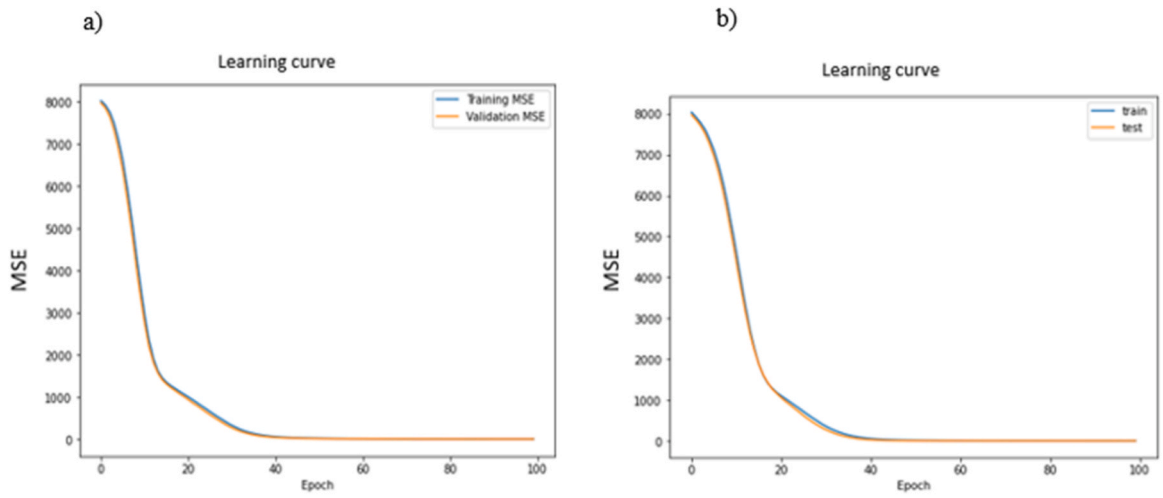


Fig. 16. Learning curve showing MSE values over the epoch for a) S6 and b) S7.

4.2.5. *Observations from the recovery indicator prediction*

The learning curves depicted the error rate decreased during training over epochs to achieve optimal performance. In this research, the target output was 93.69 % and 90.38 % for DC1 and DC2, respectively. The lower the loss function value (MSE), the closer the predicted output is to the actual production pre-disruption SL, which means the model performs well. The implication of this is that the performance of the regression model is improving, as it is becoming better at predicting the target variable.

4.2.6. *Discussion*

This research provides valuable insights into the performance and robustness of logistic regression and LSTM models in predicting flooding-induced disruptions within SCs. Both models demonstrated competent predictive capabilities, with LSTM slightly outperforming logistic regression, achieving a higher AUC-ROC score. The DES model of SC disruptions revealed significant impacts on SL across various DCs under reduced operational capacities. The prognosis of recovery indicators using an MLPNN showed promising results, effectively forecasting recovery with minimal error. The study highlights opportunities for further model refinement, such as integrating more advanced ML algorithms and expanding the disruption scenarios to improve prediction accuracy and resilience forecasting. Additionally, using DES outputs to train ML models for predicting SC recovery indicators is an innovative approach that warrants further exploration.

4.2.7. *Benchmarking of the models*

The study by Saravanan et al. [48] serves as a benchmark for evaluating the algorithms employed in the initial stage of this research. Their work predicted flooding in Kerala's Idukki region using five distinct techniques: Adaboost, Gradient Boosting (GB), Extreme Gradient Boosting (XGB), CatBoost, and Stochastic Gradient Boosting (SGB). The reported AUC scores for SGB and GB were 92 %, while Adaboost and CatBoost achieved 87 % and 79 %, respectively. The logistic regression and LSTM models developed in this research achieved AUC scores of 78 % and 84 %, respectively. These results demonstrate the comparative efficacy of the models, although further data could enhance their performance.

This research compared the model's SL predictions at DC1 against the actual SL of a real-world SC network to evaluate the key performance indicator, particularly the pre-disruption SL. The real system's SL at DC1 was 95 %, while the simulated model slightly underperformed with a 94 % SL. Regarding the validation of the proposed model for predicting recovery indicators, no prior study has, to the best of the authors' knowledge, utilized DES outputs to design an ML model for predicting SC recovery indicators using an MLPNN. The four distinct data scenarios used to train the MLPNN revealed varying error reduction rates at different epoch levels. The model's performance was rigorously assessed using MSE as the evaluation metric. Lima and Carpinetti [51] have previously noted the adequacy of MLPNN for predicting SC performance, which supports its use in this research.

4.3. *Resultant framework for DSCT for disruption and recovery prediction indicator*

The proposed DSCT framework, as illustrated in Fig. 17, outlines the three-phase methodology employed in this research. Each phase is indicated in bold to denote its implementation within the study. Initially, historical precipitation data was utilized to train ML algorithms, specifically LSTM and logistic regression models, with real-time data transmitted via IoT precipitation devices to predict flooding. It is important to note that the logistic regression model does not appear in the framework, given its inferior performance compared to the LSTM model.

A DES model was executed using FlexSim to develop a what-if scenario simulation in the event of a predicted flood. The simulation outputs were then used to train an MLPNN model to predict the recovery indicator. Here, the recovery indicator is defined as the epoch at which training and test errors (MSE) significantly decrease and stabilize. Although an epoch typically serves as a hyperparameter that indicates the learning duration of a model, in this context, it is leveraged as a recovery indicator due to the variations in dataset characteristics such as distribution, variability, size, and complexity. SC planners can determine the most appropriate epoch for the algorithm by simulating data parameters related to disruption percentages. Identifying the recovery indicator allows planners to make more informed decisions, enhancing their ability to mitigate SC disruptions effectively.

This DT framework integrates ERP systems with cloud-connected servers and trained algorithms to streamline SC management. The built-in DES using FlexSim enables managers to evaluate the potential impact of predicted capacity reductions during scenario planning. The model calculates the impact by considering disruption, proposed mitigation strategies, and average SL. By utilizing different feature levels, managers can determine the optimal epoch value, with a higher epoch indicating a longer recovery time from disruptions.

The study emphasizes the practical application of a predictive tool in proactive SC planning, highlighting the significance of understanding the relationship between disruption, resilience, and the recovery indicator (epoch). Equipped with this insight, SC planners can strategically enhance network resilience by experimenting with various feature

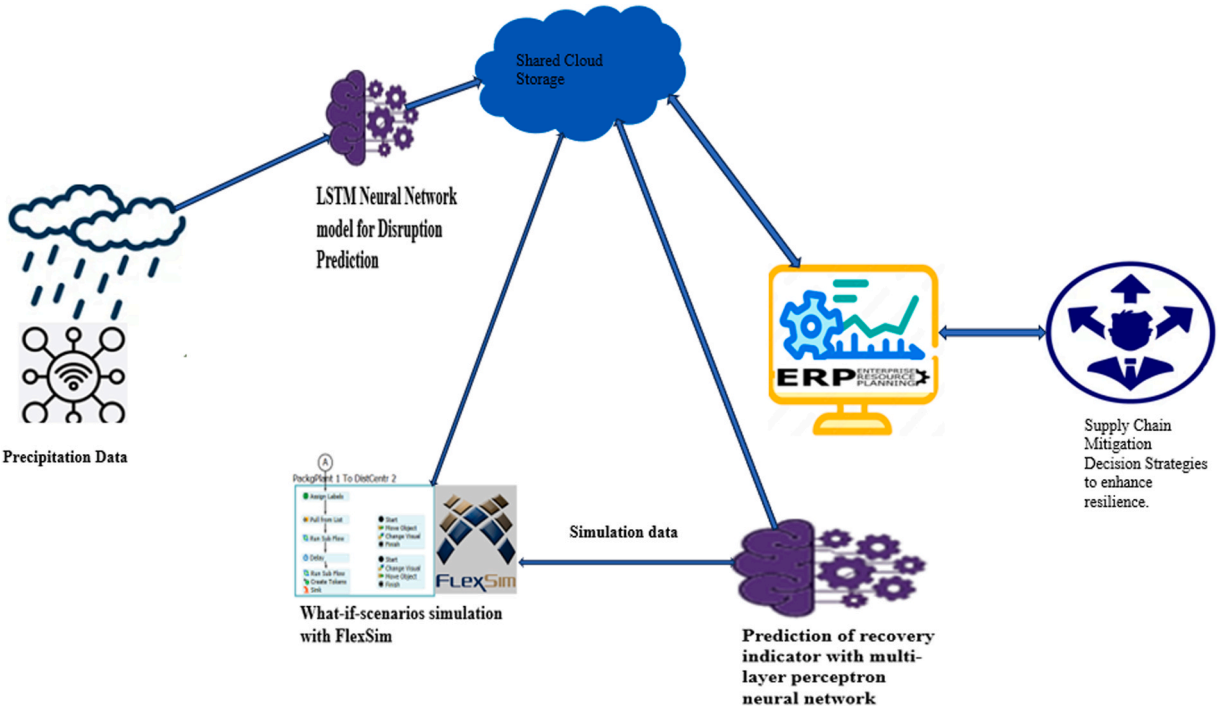


Fig. 17. Proposed Framework (design) for Digital SC Twin for Disruptions and Recovery Prediction.

combinations and levels to facilitate a return to normal operations following a disruption.

5. Conclusions, managerial insights, and future research

This work introduced a DT framework to tackle the impacts of flooding on SCs. The framework utilizes robust tools such as ML algorithms and simulations to enable the prediction of floodings disrupting SC operations and its possible recovery. Hence, DSCT literature gains two major contributions from this study. First, it finds flaws in current DT methods for SC resilience, particularly in the context of hydrological disruptions like flooding. Prior research focused on biological and operational disturbances, ignoring hydrological forms. Second, this research models ML techniques for SC hydrological disruption prediction using real-world data to fill this gap.

The proposed framework serves as a decision-support tool to enhance resilience against floods, a significant source of SC disruptions. It facilitates the analysis of disruption impacts and the effectiveness of mitigation strategies, enabling managers to anticipate potential challenges, make informed decisions, and implement appropriate measures to maintain service SL and reduce recovery time. Furthermore, various SC industries can embrace this conceptual framework to develop both proactive and reactive strategies, ensuring business continuity in both the short and long term. Hence, this research's conceptual DT framework addresses the first research question.

This study utilized ML algorithms and scenario planning to enhance SC resilience through disruption prediction, real-world SC simulation, and ML training using simulated SC data to develop recovery indicators. By integrating an ERP system with a connected cloud server and the trained algorithms, disruptions such as flooding can be anticipated effectively. A predictive recovery tool was created based on results from FlexSim simulations conducted during disruptions. This tool incorporates features like the average disruption level, average demand during disruptions, and average SL during disruptions to train an MLPNN. In this context, the recovery indicator is defined as the epoch, which is the point where both training and testing errors stabilize. If the SC takes a longer time to reach this epoch, as determined by what-if scenario simulations, it indicates a prolonged recovery period. Understanding and leveraging this recovery indicator enables SC planners to make informed decisions, helping them mitigate the effects of disruptions. Hence, the methods for implementing the conceptual DT framework address the second research question.

The conceptual framework introduced in Fig. 17 could have a broader range of potential applications than simply addressing flooding-related disruptions. It could also apply to natural disasters like earthquakes, storms, and wildfires. These natural occurrences negatively impact production infrastructures and contribute to employee absenteeism. In the event of other natural disasters, such as wildfires, scenario planning can be conducted similarly to understand how it would be for flooding. It is possible to make accurate predictions of future natural disasters by verifying the variables that caused the damage in the affected areas and then using these variables as training features for neural networks to make accurate predictions.

Future research should consider a broader range of disruptive events, such as earthquakes, and explore additional factors like flood magnitude and timing. The current study assumes a single-vendor SC, neglecting backorder costs and potential complexities of multi-vendor networks. It relies on one performance indicator, which could be expanded to include multiple metrics. The study also assumes demand patterns remain unchanged during disruptions; future work should investigate variations in capacity and demand. The method of using epoch numbers for recovery prediction could be refined by developing models to forecast precise recovery durations. Lastly, exploring additional ML algorithms beyond logistic regression, MLPNN, and LSTM may yield improved predictive results.

Funding

This work was partially funded by an NSERC Discovery Grant, University of Windsor, and Cape Breton University through project number 4081205.

CRediT authorship contribution statement

Waguih ElMaraghy: Writing – review & editing, Supervision, Funding acquisition, Conceptualization. **Jessica Olivares-Aguila:** Writing – review & editing, Supervision, Methodology, Funding acquisition, Conceptualization. **Oluwagbenga Victor Ogunsoto:** Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization.

Declaration of Competing Interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Jessica Olivares Aguila reports financial support was provided by Cape Breton University. Waguih ElMaraghy reports financial support was provided by NSERC Discovery Grant. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix 1. : Pseudocode for logistics regression algorithm

ALGORITHM 1. : FLOOD PREDICTION USING LOGISTICS REGRESSION

Inputs:
- *x: feature matrix (m x n), where n is the number of features (1 for rainfall data)*
- *y: target vector (m x 1) indicating flood occurrence (1 for flood, 0 for no flood)*
- *train_ratio: ratio of data to be used for training (e.g., 0.66 for 66%)*
- *learning_rate: learning rate for gradient descent*
- *num_iterations: number of iterations for gradient descent*

Outputs:
- *theta: learned parameters (n x 1)*
- *x_test: test feature matrix*
- *y_test: test target vector*

Procedure:

1 ***Initialization of theta**: Initialize theta as a vector of zeros(n x1)*
2 ***Define** ReLu*
ReLu [f(x] = max(0,x)
3 ***Define** the cost function:*
*cost_function(X, y, theta) = (-1 / m) * sum(y * log(ReLu(X * theta)) + (1 - y) * log(1 - ReLu(X * theta)))*
4 ***Define** the gradient descent update rule:*
*theta = theta - (learning_rate / m) * X' * (ReLu(X * theta) - y)*
5 ***Read** rainfall data from 1901 to 2015 for Kerala state and store it in an array or matrix of size (m x 1), X*
***Read** flood occurrence data from 1901 to 2015 for Kerala state and store it in an array or matrix of size (m x 1), y*
6 ***Set** train_size as train_ratio * m (number of data points)*
Set X_train as the first train_size rows of X
Set y_train as the first train_size rows of y
Set X_test as the remaining rows of X
Set y_test as the remaining rows of y
7 ***Perform** gradient descent to learn the parameters theta:*
***For** iteration = 1 to num_iterations:*
*theta = theta - (learning_rate / m) * X_train' * (ReLu(X_train * theta) - y_train)*
8 ***Return** theta, X_test, y_test*

Appendix 2. : Pseudocode for LSTM algorithm

ALGORITHM 2. : FLOOD PREDICTION USING LSTM

```
INPUT:
- rainfall_data[1:150]
X_train = rainfall_data[1:TRAINING_DATA_SIZE]
y_train = rainfall_data[2:TRAINING_DATA_SIZE+1]
X_test = rainfall_data[TRAINING_DATA_SIZE+1:]
y_test = rainfall_data[TRAINING_DATA_SIZE+2:]
DEFINE LSTM_model
DEFINE train_parameters
DEFINE evaluate_parameters
FUNCTION LSTM_model():
   CREATE LSTM model
   ADD LSTM layer(s) with desired number of units and activation functions
   ADD output layer with one unit
   RETURN model
FUNCTION train_parameters(model, X_train, y_train, num_epochs, learning_rate):
   FOR epoch = 1 TO num_epochs:
      FOR i = 1 TO TRAINING_DATA_SIZE:
         # Forward propagation
         Compute predicted_output using model and input value X_train[i]
         # Backpropagation
         Compute error using predicted_output and y_train[i]
         Update model weights using gradient descent
   RETURN model
FUNCTION evaluate_parameters(model, X_test, y_test):
   correct_predictions = 0
   FOR i = 1 TO TESTING_DATA_SIZE:
      Compute predicted_output using model and input value X_test[i]
      IF predicted_output >= threshold AND y_test[i] = 1 OR predicted_output < threshold AND y_test[i]
= 0:
         correct_predictions = correct_predictions + 1
   accuracy = (correct_predictions / TESTING_DATA_SIZE) * 100
   RETURN accuracy
DEFINE num_epochs = epoch_num
DEFINE learning_rate = 0.001

model = LSTM_model()
model = train_parameters(model, X_train, y_train, num_epochs, learning_rate)
accuracy = evaluate_parameters(model, X_test, y_test)
```

Appendix 3. : Pseudocode for recovery indicator

ALGORITHM 3. : RECOVERY INDICATOR PREDICTION USING MLPNN

```
INPUT:
- percentage_of_disruption[1:600]
- service_level_at_percentage_of_disruption[1:600]
- demand_data[1:600]
- service_level_target_output[1:600]
TRAINING_DATA_SIZE = 0.8 * 600
TESTING_DATA_SIZE = 0.2 * 600
X_train = [percentage_of_disruption[1:TRAINING_DATA_SIZE],
        service_level_at_percentage_of_disruption[1:TRAINING_DATA_SIZE],
        demand_data[1:TRAINING_DATA_SIZE]]
y_train = service_level_target_output[1:TRAINING_DATA_SIZE]

X_test = [percentage_of_disruption[TRAINING_DATA_SIZE+1:],
       service_level_at_percentage_of_disruption[TRAINING_DATA_SIZE+1:],
       demand_data[TRAINING_DATA_SIZE+1:]]
y_test = service_level_target_output[TRAINING_DATA_SIZE+1:]
DEFINE MLP_model
DEFINE train_parameters
DEFINE evaluate_parameters
FUNCTION MLP_model():
   CREATE MLP model.
   ADD fully connected input layer with three input units.
   ADD one or more fully connected hidden layers with desired number of units and activation functions.
   ADD fully connected output layer with one output unit.
   RETURN model
FUNCTION train_parameters(model, X_train, y_train, num_epochs, learning_rate, error_tolerance):
   error_stabilized_epoch = -1
   previous_error = MAX_VALUE
   FOR epoch = 1 TO num_epochs:
      mean_squared_error = 0
          FOR i = 1 TO TRAINING_DATA_SIZE:
            # Forward propagation
            Compute predicted_output using model and input values X_train[i]
                    # Backpropagation
            Compute error using predicted_output and y_train[i]
              Update model weights using gradient descent
          mean_squared_error = mean_squared_error + error^2
          mean_squared_error = mean_squared_error / TRAINING_DATA_SIZE
          IF ABS(previous_error - mean_squared_error) < error_tolerance AND error_stabilized_epoch = -
   1:
            error_stabilized_epoch = epoch
          previous_error = mean_squared_error
   IF error_stabilized_epoch != -1:
            BREAK
   RETURN model, error_stabilized_epoch
   DEFINE num_epochs = nth_epoch
   DEFINE optimizer = Adam
   model, error_stabilized_epoch = train_parameters(model, X_train, y_train, num_epochs, learning_rate,
   error_tolerance)
```

Data Availability

All data generated or analyzed during this study are included in this published article. Upon a reasonable request, the authors will provide the source code and other research materials for reproducibility.

References

[1] J. Olivares-Aguila, W. ElMaraghy, System dynamics modelling for supply chain disruptions, Int J. Prod. Res 59 (6) (2021) 1757–1775.

[2] W. Ho, T. Zheng, H. Yildiz, S. Talluri, Supply chain risk management: a literature review, Int J. Prod. Res 53 (16) (Aug. 2015) 5031–5069, https://doi.org/10.1080/00207543.2015.1030467.

[3] J. Thun, An empirical analysis of supply chain risk management in the German automotive industry, D. H.-I. journal of production economics, and 2011, in: JH Thun (Ed.), D HoenigInternational Journal of Production Economics, 2011, Elsevier, 2024, https://www.sciencedirect.com/science/article/pii/S0925527309003715.

[4] M. Haraguchi, U. Lall, Flood risks and impacts: a case study of Thailand's floods in 2011 and research questions for supply chain decision making, Int. J. Disaster Risk Reduct. 14 (Dec. 2015) 256–272, https://doi.org/10.1016/J.IJDRR.2014.09.005.

[5] D. Ivanov, A. Dolgui, A digital supply chain twin for managing the disruption risks and resilience in the era of Industry 4.0, Prod. Plan. Control 32 (9) (2020) 775–788, https://doi.org/10.1080/09537287.2020.1768450.

[6] P.R. Kleindorfer, G.H. Saad, Managing disruption risks in supply chains, Prod. Oper. Manag 14 (1) (2005) 53–68, https://doi.org/10.1111/J.1937-5956.2005.TB00009.X.

[7] S.M. Wagner, C. Bode, An empirical examination of supply chain performance along several dimensions of risk, J. Bus. Logist. 29 (1) (Mar. 2008) 307–325, https://doi.org/10.1002/J.2158-1592.2008.TB00081.X.

[8] C.S. Tang, Perspectives in supply chain risk management, Int. J. Prod. Econ. 103 (2) (2006) 451–488, https://doi.org/10.1016/j.ijpe.2005.12.006.

[9] S.Y. Ponomarov, M.C. Holcomb, Understanding the concept of supply chain resilience, Int. J. Logist. Manag. 20 (1) (May 2009) 124–143, https://doi.org/10.1108/09574090910954873/FULL/XML.

[10] M. Christopher, H. Peck, Building the resilient supply chain, Int. J. Logist. Manag. 15 (2) (2004) 1–14, https://doi.org/10.1108/09574090410700275.

[11] J. Blackhurst, C.W. Craighead, D. Elkins, R.B. Handfield, An empirically derived agenda of critical research issues for managing supply-chain disruptions, Int J. Prod. Res 43 (19) (Oct. 2005) 4067–4081, https://doi.org/10.1080/00207540500151549.

[12] D. Rehak, P. SenovskyResilience of critical infrastructure elements and its main factors," mdpi.comD Rehak, P Senovsky, S SlivkovaSystems, 2018•mdpi.com, vol. 6, no. 2, doi: 10.3390/systems60200212018.

[13] F. Petit, D. Verner, J. Phillips, L.P. Lewis, Critical infrastructure protection and resilience—integrating interdependencies, Adv. Sci. Technol. Secur. Appl. (2018) 193–219, https://doi.org/10.1007/978-3-319-78021-4_10.

[14] PPD-21, "Presidential Decision Directive: Critical Infrastructure Security and Resilience," Washington, 2013.

[15] J.O. Aguila, W. ElMaraghy, Supply chain resilience and structure: an evaluation framework, Procedia Manuf. 28 (2019) 43–50, https://doi.org/10.1016/J.PROMFG.2018.12.008.

[16] K. Scholten, P.S. Scott, B. Fynes, Mitigation processes – antecedents for building supply chain resilience, Supply Chain Manag. 19 (2) (2014) 211–228, https://doi.org/10.1108/scm-06-2013-0191.

[17] D. Ivanov, A. Dolgui, B. Sokolov, M. Ivanova, Literature review on disruption recovery in the supply chain, Int J. Prod. Res. 55 (20) (Oct. 2017) 6158–6174, https://doi.org/10.1080/00207543.2017.1330572.

[18] F. LückerR. S.- Omega, and 2017, "Building up resilience in a pharmaceutical supply chain through inventory, dual sourcing and agility capacity," Elsevier F Lücker, RW SeifertOmega, 2017•Elsevier,2024, https://www.sciencedirect.com/science/article/pii/S0305048317300294.

[19] A. Diabat, K. Govindan, V.V. Panicker, Supply chain risk management and its mitigation in a food industry, Int J. Prod. Res. 50 (11) (Jun. 2012) 3039–3050, https://doi.org/10.1080/00207543.2011.588619.

[20] R. Aldrighetti, D. Battini, D. Ivanov, Efficient resilience portfolio design in the supply chain with consideration of preparedness and recovery investments, Omega (U. Kingd.) 117 (Jun. 2023), https://doi.org/10.1016/J.OMEGA.2023.102841.

[21] J. Olivares-Aguila, A. Vital-Soto, Supply chain resilience roadmaps for major disruptions, Logistics 5 (4) (Nov. 2021) 78, https://doi.org/10.3390/LOGISTICS5040078.

[22] S. Hosseini, D. Ivanov, A. Dolgui, Review of quantitative methods for supply chain resilience analysis, Transp. Res. E Logist. Transp. Rev. 125 (May 2019) 285–307, https://doi.org/10.1016/J.TRE.2019.03.001.

[23] P. Centobelli, R. Cerchione, A. Maglietta, E. Oropallo, Sailing through a digital and resilient shipbuilding supply chain: an empirical investigation, J. Bus. Res. 158 (Mar. 2023), https://doi.org/10.1016/J.JBUSRES.2023.113686.

[24] A. Rashid, R. Rasheed, A.H. Ngah, M.D.R. Pradeepa Jayaratne, S. Rahi, M.N. Tunio, Role of information processing and digital supply chain in supply chain resilience through supply chain risk management, J. Glob. Oper. Strateg. Sourc. 17 (2) (2024) 429–447, https://doi.org/10.1108/JGOSS-12-2023-0106/FULL/HTML.

[25] S.F. Nimmy, O.K. Hussain, R.K. Chakrabortty, F.K. Hussain, M. Saberi, Explainability in supply chain operational risk management: a systematic literature review, Knowl. Based Syst. 235 (Jan. 2022), https://doi.org/10.1016/J.KNOSYS.2021.107587.

[26] M.W. Grieves, J. Vickers, Digital Twin: mitigating unpredictable, undesirable emergent behavior in complex systems. eBooks, Springer, 2016, pp. 85–113, https://doi.org/10.1007/978-3-319-38756-7_4.

[27] H.A. ElMaraghy, W. ElMaraghy, Adaptive cognitive manufacturing system (ACMS) – a new paradigm, Int J. Prod. Res 60 (24) (2022) 7436–7449, https://doi.org/10.1080/00207543.2022.2078248.

[28] R. Stark, T. Damerau, Digital Twin. in In Springer eBooks, Nature, Springer, 2019, pp. 1–8.

[29] N. Zhao, J. Hong, K.H. Lau, Impact of supply chain digitalization on supply chain resilience and performance: a multi-mediation model, Int J. Prod. Econ. 259 (May 2023), https://doi.org/10.1016/J.IJPE.2023.108817.

[30] L. Rigó, J. Fabianová, M. Lokšík, N. M.- Sustainability, and undefined 2024, "Utilising Digital Twins to Bolster the Sustainability of Logistics Processes in Industry 4.0," *mdpi.comL Rigó, J Fabianová, M Lokšík, N MikušováSustainability, 2024•mdpi.com*, Accessed: Aug. 27, 2024. [Online]. Available: ⟨https://www.mdpi.com/2071-1050/16/6/2575⟩.

[31] P. Maheshwari, S. Kamble, A. Belhadi, M. Venkatesh, M.Z. Abedin, Digital twin-driven real-time planning, monitoring, and controlling in food supply chains, Technol. Forecast Soc. Change 195 (Oct. 2023), https://doi.org/10.1016/J.TECHFORE.2023.122799.

[32] D. GuoThe role of digital twins in lean supply chain management: review and research directions," Taylor & FrancisD Guo, S MantravadiInternational Journal of Production Research, 2024•Taylor & Francis, 2024, doi:10.1080/00207543.2024.23726552024.

[33] M. Peron, A digital twin-enabled digital spare parts supply chain, Int J. Prod. Res. (2024), https://doi.org/10.1080/00207543.2024.2338878.

[34] S. Cuñat Negueroles, et al., A Blockchain-based Digital Twin for IoT deployments in logistics and transportation, Future Gener. Comput. Syst. 158 (Sep. 2024) 73–88, https://doi.org/10.1016/J.FUTURE.2024.04.011.

[35] D. Ivanov, Predicting the impacts of epidemic outbreaks on global supply chains: a simulation-based analysis on the coronavirus outbreak (COVID-19/SARS-CoV-2) case, Transp. Res. E Logist. Transp. Rev. 136 (Apr. 2020), https://doi.org/10.1016/J.TRE.2020.101922.

[36] S.Y. Barykin, A.A. Bochkarev, O.V. Kalinina, V.K. Yadykin, Concept for a supply chain digital twin, Int. J. Math. Eng. Manag. Sci. 5 (6) (2020) 1498–1515, https://doi.org/10.33889/IJMEMS.2020.5.6.111.

[37] D. Ivanov, Intelligent digital twin (iDT) for supply chain stress-testing, resilience, and viability, Int J. Prod. Econ. 263 (Sep. 2023), https://doi.org/10.1016/J.IJPE.2023.108938.

[38] M. SchleifenheimerD. I.-A. of O. Research, and 2024, Pharmaceutical retail supply chain responses to the COVID-19 pandemic, M Schleifenheimer, D IvanovAnnals of Operations Research. Springer, doi:10.1007/s10479-024-05866-02024.

[39] M. Fogli, C. Giannelli, F. Poltronieri, C. Stefanelli, M. Tortonesi, Chaos engineering for resilience assessment of digital twins, IEEE Trans. Ind. Inf. 20 (2) (Feb. 2024) 1134–1143, https://doi.org/10.1109/TII.2023.3264101.

[40] K.Y. Hong Lim, A. Seif, N. Agarwal, N.T. Le, Digital twin-enhanced approach for supply chain disruption management in manufacturing shop floors, 2021 IEEE Int. Conf. Ind. Eng. Eng. Manag., IEEM 2021 (2021) 848–852, https://doi.org/10.1109/IEEM50564.2021.9673020.

[41] S.K. Yevu, E.K. Owusu, A.P.C. Chan, S.M.E. Sepasgozar, V.R. Kamat, Digital twin-enabled prefabrication supply chain for smart construction and carbon emissions evaluation in building projects, J. Build. Eng. 78 (Nov. 2023), https://doi.org/10.1016/J.JOBE.2023.107598.

[42] B. Guidani, M. Ronzoni, R. Accorsi, Virtual agri-food supply chains: a holistic digital twin for sustainable food ecosystem design, control and transparency, Sustain Prod. Consum 46 (May 2024) 161–179, https://doi.org/10.1016/J.SPC.2024.01.016.

[43] A. Cimino, F. Longo, G. Mirabelli, V. Solina, A cyclic and holistic methodology to exploit the supply chain digital twin concept towards a more resilient and sustainable future, Clean. Logist. Supply Chain 11 (2024), https://doi.org/10.1016/J.CLSCN.2024.100154.

[44] D. Burgos, D. Ivanov, Food retail supply chain resilience and the COVID-19 pandemic: A digital twin-based impact analysis and improvement directions, Transp. Res E Logist. Transp. Rev. 152 (2021), https://doi.org/10.1016/j.tre.2021.102412.

[45] P. Nguyen, D. Ivanov, F. Sgarbossa, A digital twin–based approach to reinforce supply chain resilience: simulation of semiconductor shortages, in: IFIP Adv Inf Commun Technol, 692, AICT, 2023, pp. 563–576, https://doi.org/10.1007/978-3-031-43688-8_39.

[46] P.S. Sutar, J. Olivares-Aguila, and A. Vital-Soto, An Offline Digital Twin for Resilience and Supplier Reliability in Perishable Food Supply Chains, Proceedings

of the 2nd Australian International Conference on Industrial Engineering and Operations Management, Nov. 2023, doi: 10.46254/AU02.20230085.
[47] Y.L. Li, Y.P. Tsang, C.H. Wu, C.K.M. Lee, A multi-agent digital twin–enabled decision support system for sustainable and resilient supplier management, Comput. Ind. Eng. 187 (Jan. 2024) 109838, https://doi.org/10.1016/J.CIE.2023.109838.
[48] S. Saravanan, et al., Flood susceptibility mapping using machine learning boosting algorithms techniques in Idukki district of Kerala India, Urban Clim. 49 (10150) (2023) 3.
[49] S.P. Willems, Data Set—Real-world multiechelon supply chains used for inventory optimization, Manuf. Serv. Oper. Manag. 10 (1) (2008) 19–23, https://doi.org/10.1287/msom.1070.0176.
[50] S.E. Graves, S.P. Willems, Supply chain design: safety stock placement and supply chain configuration, Elsevier eBooks, 2003, pp. 95–132.
[51] F.R. Lima-Junior, L.C.R. Carpinetti, Predicting supply chain performance based on SCOR ® metrics and multilayer perceptron neural networks, Int J. Prod. Econ. 212 (Jun. 2019) 19–38, https://doi.org/10.1016/J.IJPE.2019.02.001.