applied sciences

MDPI

Article

Out-of-Stock Prediction Model Using Buzzard Coney Hawk Optimization-Based LightGBM-Enabled Deep Temporal Convolutional Neural Network

Ahmed Elghadghad, Ahmad Alzubi * and Kolawole Iyiola

Institute of Social Sciences, University of Mediterranean Karpasia, Mersin 33000, Turkey; 210634116@std.akun.edu.tr (A.E.); kolawole.iyiola@akun.edu.tr (K.I.)
* Correspondence: ahmad.alzaubi@akun.edu.tr

Abstract: Out-of-stock prediction refers to the activity of forecasting the time when a product will not be available for purchase because of an inventory deficiency. Due to difficulties, out-of-stock forecasting models now face certain challenges. Incorrect demand forecasting may result in a lack or excess of goods in stock, a factor that affects client satisfaction and the profitability of companies. Accordingly, the new approach BCHO-TCN LightGBM, which is based on Buzzard Coney Hawk Optimization with a deep temporal convolutional neural network and the Light Gradient-Boosting Machine framework, is developed to deal with all challenges in the existing models in the field of out-of-stock prediction. The role that BCHO plays in the LightGBM-based deep temporal CNNis rooted in modifying the classifier to improve both accuracy and speed. Integrating BCHO into the model training process allows us to optimize and adjust the hyperparameters and the weights of the CNN linked with the temporal DNN, which, in turn, makes the model perform better in the extraction of temporal features from time-series data. This optimization strategy, which derives from the cooperative behaviors and evasion tactics of BCHO, is a powerful source of information for the computational optimization agent. This leads to a faster convergence of the model towards optimal solutions and therefore improves the overall accuracy and predictive abilities of the temporal CNN with the LightGBM algorithm. The results indicate that when using data from Amazon India's product listings, the model shows a high degree of accuracy, as well as excellent net present value (NPV), present discounted value (PDV), and threat scores, with values reaching 94.52%, 95.16%, 94.81%, and 95.76%, respectively. Likewise, in a k-fold 10 scenario, the model achieves values of 94.81%, 95.60%, 96.28%, and 95.86% for the same metrics.

Keywords: out-of-stock prediction; LightGBM; temporal convolutional neural network; buzzard coneyhawk optimization and feature extraction

Citation: Elghadghad, A.; Alzubi, A.; Iyiola, K. Out-of-Stock Prediction Model Using Buzzard Coney Hawk Optimization-Based LightGBM-Enabled Deep Temporal Convolutional Neural Network. *Appl. Sci.* **2024**, *14*, 5906. https://doi.org/10.3390/app14135906

Academic Editor: Arkadiusz Gola

Received: 21 May 2024
Revised: 18 June 2024
Accepted: 25 June 2024
Published: 5 July 2024

1. Introduction

Maximizing profits is a key objective for retail stores like supermarkets and convenience stores, and avoiding missed sales opportunities is crucial to achieving this goal. Monitoring on-shelf availability is essential for boosting profits, as product shortages, also known as out-of-stock (OOS) situations, are a major cause of revenue loss [1]. Formally, OOS events occur when a product requested by customers is not available on the market store shelves, and are addressed when the product(s) is received at the market store [2]. Crisis prediction has been used in the banking sector, as well as in business, investments, and other areas. Crisis prediction is critical for the financial market, and has attracted many researchers and academicians [3]. Stock markets were normally predicted by financial experts in the past. However, data scientists have started solving prediction problems with the progress of learning techniques [4]. Numerous approaches and different data can be used to address the problem of OOS detection [5]. To enhance profitability, retail establishments must focus on minimizing missed sales opportunities. On-shelf availability,

which refers to the availability of products for purchase in their expected location and at the desired time, is a significant indicator of lost sales opportunities [6]. When customers find a desired product out of stock, they may opt to purchase a similar item from a competitor, resulting in lost revenue for the store [7–9]. Having products readily available on shelves is essential for boosting profits in retail stores. To further increase profitability, it is important to place products in areas that are easily seen by shoppers, such as at the front of shelves, rather than towards the back [10]. The commercial sector is experiencing new challenges and opportunities as a result of advancements in modern technology [11]. In particular, the use of recent predictive technologies grounded in deep learning is improving the ability to effectively address the issue of OOS products that are not available for purchase when requested by customers [12–14].

Reliable prediction systems accurately anticipate and manage instances of product unavailability, ensuring adequate shelf stocking to meet customer demand. The occurrence of an OOS event arises when a customer's desired product is unavailable in-store and is resolved once the replenishment of the product(s) is completed [15]. During this specific time frame, out-of-stock events lead to economic losses based on the level of product demand. Despite efforts by large companies to minimize the impact of out-of-stock events, their repercussions on the retail industry remain significant [16]. Studies suggest that around 4% of the total annual revenue in the commercial sector is lost as a result of out-of-stock occurrences, leading to more than USD 900 billion in financial losses. Some examples of unexpected market behavior are educational webinars, persona-based eBook marketing, and the use of branded schedule magnets. The study referenced found that out-of-stock events distort actual commercial demand, affecting brand loyalty and market dynamics. This method takes into account a process with a large number of connections in a multi-stage and parallel assembly. Priority relationships are established between the assembly units, and the assembly sequence is represented by a directed acyclic graph. Originally, a two-part crossover and mutation operators for assembling sequences were proposed. Due to the complexity of the assembly sequence planning problem, its optimization is required in order to obtain an efficient computational approach [17].

Utilizing deep learning is the next step in improving prediction models and achieving better performance. Even with advancements, predicting the stock market remains a difficult job for data scientists because of the market's intricate and unpredictable nature, as well as the connection between investors' emotions and market movements. Large datasets and machine learning advancements have led to the creation of several effective object detection algorithms. However, these algorithms typically necessitate detailed labeling of a significant quantity of data to train effectively in a different field [18]. Given the time-consuming nature of the annotation process, weak supervised learning methods and sparse data labeling have gained popularity recently.

Various proposed solutions focus on tackling OOS issues from the manufacturer's standpoint. These solutions rely on point-of-sale data and other distinct characteristics identified through statistical analysis or machine learning methods. The methods usually talked about in research papers are statistical models and deep learning strategies, especially supervised ones, to figure out how likely it is that an OOS event will happen and how to manage product inventories well to avoid them [19]. However, these scientific approaches suffer a decline in effectiveness due to the unpredictability of factors that significantly impact OOS events, making them difficult to model based on previous data. In certain instances, the dynamics of OOS events exhibit novelty, and didn't utilize historical market data. Also, the existing predictive models for OOS events may not take into account all the random factors that impact this occurrence. This study suggests a new model to address the limitations experienced by current models [20]. The practical application of artificial intelligence methods is particularly important for products that are described by numerous and different parameters. This group includes automotive heat exchangers, which also have to meet specific exploitative requirements resulting from their working conditions. Constant artificial neural network parameters are utilized for network learning in the Broyden–

Fletcher–Goldfarb–Shanno algorithm, in which the number of input neuron connections is 52, and the activation functions are linear, logistic, exponential, and hyperbolic [21].

The new model tackles challenges in a distinct way compared to current solutions by utilizing a special blend of BCHO, a deep temporal CNN, and LightGBM. Instead of solely depending on statistical models or shallow learning techniques like traditional methods, this model incorporates BCHO, which draws inspiration from natural behaviors, to enhance model training efficiency. With the inclusion of a deep temporal CNN, the model is able to detect complex temporal patterns in data, leading to a more detailed analysis of stock market dynamics. Moreover, the partnership with LightGBM allows for the efficient management of large datasets and sparse data labels, effectively overcoming the restrictions of data annotation requirements. Furthermore, the proactive approach to managing stocks made possible by the model allows for the improved forecasting and management of stockouts, leading to more precise and dependable predictions in stock market analysis. In general, the model presents a thorough and creative solution that tackles the obstacles in stock market forecasting by merging cutting-edge optimization strategies with deep learning technologies and effective data management abilities.

The aim of this research is to increase the accuracy of stock market predictions by combining BCHO with LightGBM and a deep temporal CNN. BCHO improves model training by using unique reinforcement and evasion tactics inspired by natural behavior. Incorporating BCHO with a deep temporal CNN efficiently identifies temporal patterns in stock market data, enabling a more thorough analysis of market trends. Moreover, working with LightGBM allows the model to effectively handle large datasets and sparse data labels. With this integrated approach, the model intends to offer more precise and dependable predictions through proactive stock management techniques, ultimately overcoming the obstacles of stock market forecasting. The major contributions are as follows:

- Buzzard Coney Hawk Optimization: BCHO is a heuristic algorithm inspired by the hunting behaviors of buzzards, coney hawks, and conies. It aims to efficiently search for the best solutions in complex optimization problems. This optimization strategy is specifically developed by assembling the features of vultures, rabbits, and Harris hawks to produce a strong computational optimization agent. BCHO contributes to the creation of new pathways through its innovative application of the unique qualities attributed to BCHO.BCHO serves as a tool in the optimization process thatincreases the accuracy and speed of model training. These cooperative approaches, in combination with evasion tactics, create an advanced accelerator to make classifiers perform better.
- LightGBM: LightGBM utilizes a fast and scalable algorithm capable of handling large datasets. It helps to increase training speed and contributes to effective memory utilization. The combination of BCHO with a TCN helps in the efficient extraction of temporal aspects and leads to better out-of-stock prediction.
- BCHO-TCN LightGBM: To achieve this efficient out-of-stock prediction model, utilize BCHO along with a TCN and LightGBM to detect and predict out-of-stock products. The cooperation of BCHO with a TCN allows for a more efficient extraction of temporal aspects, leading to more accurate out-of-stock prediction.

We have structured the manuscript as follows: Section 2 details existing works, their methods, and challenges. Section 3 outlines the process, including the proposed out-of-stock prediction model using BCHO-TCN LightGBM. Section 4 presents the mathematical equation for Buzzard Coney Hawk Optimization. Section 5 examines the findings from the experiment, while Section 6 provides a summary of the conclusions.

2. Motivation

The research aims to optimize out-of-stock predictions as the traditional methods employed for out-of-stock predictions have some limitations, such as variations in sizes, the unavailability of products, lacking performance, overfitting issues, and low resolution. To overcome these challenges, this research proposes a Buzzard Coney Hawk Optimization-Based LightGBM-Enabled Deep Temporal Convolutional Neural Network (BCHO-TCN

LightGBM) model. LightGBM utilizes a fast and scalable algorithm capable of handling large datasets. It helps to increase training speed and contributes to effective memory utilization. The combination of BCHO with a TCN helps in the efficient extraction of temporal aspects and leads to better out-of-stock prediction.

2.1. Literature Review

Dario Allegra et al. [1] emphasize the issue of identifying OOS items from a first-person perspective by introducing then ovel method of annotating the Ego Cart dataset. Their strategy involves training a model to generate attention maps that can help detect OOS items in retail environments, enabling retailers to take the necessary actions. The results show that their approach has the potential to improve restocking processes in retail settings. The success of the method is significantly influenced by the level of accuracy of the annotations.

Concetta Giaconia and Aziz Chamas [2] introduced a novel out-of-stock prediction system that utilizes deep learning and historical data to predict the remaining stock of retail products in the food distribution industry. The results of the experiments demonstrated the system's accuracy, with predictions exceeding 90%. Yet, there are still obstacles to overcome because of variations in the size, lighting, and background of areas where products are unavailable, which affect the precision of forecasts.

Nagaraj Naik and Biju R. Mohan [3] introduced a model for predicting stock crises using the Hybrid Feature Selection (HFS) technique. They first developed the HFS algorithm to eliminate unnecessary financial parameters and identify strong fundamental stocks to address overfitting problems. However, the model's performance was still lacking, necessitating further improvements and adjustments to the eXtreme Gradient Boosting (XG-Boost) method with a different optimizer. Mojtaba Nabipour et al. [4] performed research on forecasting stock market trends by utilizing machine learning and deep learning techniques. Initially, they calculated indicators based on continuous stock trading values and then converted these indicators into binary data for analysis. The results indicated that deep learning methods performed the best when using binary data for evaluation. However, the difference in performance between the methods lessened due to the significant improvement in model performance using the second approach.

Franko Sikic et al. [5] suggest a new DL-driven technique for detecting out-of-distribution samples, using a two-step training approach and a post-processing method to filter out inaccurate detections. The two-step training process enhanced the models' ability to generalize and improved upon the baseline results. However, incorporating depth estimation into the current solution could further enhance the outcomes. Saud S. Alotaibi [6] introduced a novel stock market forecasting model with three key stages: extracting features, selecting optimal features, and making predictions. During neural network training, the model fine-tunes the optimal weights to achieve accurate predictions with high performance. However, the model encountered challenges related to overfitting. KyotaHiga and Kota Iwamoto [7] introduced a technique for effectively monitoring shelves in retail establishments by utilizing supervised learning to enhance on-shelf availability. This method allows store employees to uphold a high level of on-shelf availability, thereby boosting profits for retail stores. Nonetheless, the challenge lies in the low resolution and frame rate of the videos, hindering the ability to track moving objects across successive images.

2.2. Challenges

- ➢ The main obstacles presented by the stock market's volatility and the connection between investment psychology and market trends are complexity and nonlinearity [5].
- ➢ It should be emphasized that because acquisition is conducted in actual retail store settings, manually annotating OOS items is quite difficult. This is because of factors such as perspective, shelf design, lighting, and product placement. Therefore, identifying OOS items in images proves to be a very subjective task [1].

- Due to the unpredictable, loud, and disorderly nature of data in stock markets, making accurate predictions is a major challenge. This makes it difficult for investors to allocate funds effectively in order to generate profits [6].
- Utilizing various architectural designs and incorporating depth information can be complex, but these methods are essential for achieving higher performance [1].
- Similarly, enhancing model accuracy by developing diverse fundamental stock and technical parameters poses a challenge [3].

3. Proposed Methodology for Out-of-Stock Prediction Model Using BCHO-TCN LightGBM

The main focus of this research is developing a novel out-of-stock prediction system that combines deep learning techniques with historical data analysis. This process entails the collection of data from the input dataset [22] and the subsequent preprocessing of this data for it to be ready and organized for analysis. Next, feature extraction is conducted, which involves using both statistical features and Autoregressive Fractionally Integrated Moving-Average (ARFIMA) features that combine to give a holistic image of the nature of the data. The LightGBM-enabled deep temporal CNN receives these extracted features as input. The LightGBM-based deep temporal CNN helps to modify the classifier to improve both accuracy and speed. Incorporating BCHO with a deep temporal CNN efficiently identifies temporal patterns in stock market data, enabling a more thorough analysis of market trends. Moreover, working with LightGBM allows the model to effectively handle large datasets and sparse data labels. With this integrated approach, the model intends to offer more precise and dependable predictions through proactive stock management techniques, ultimately overcoming the obstacles of stock market forecasting. To make sure that the model becomes faster and even more precise, a unique strategy is applied whereby the classifier is fine-tuned using the Buzzard Coney Hawk Optimization algorithm. This optimization strategy is specifically developed by assembling the features of vultures [23], rabbits [24], and Harris hawks [25], which produce a strong computational optimization agent. When the model is finely tuned, it is then trained using the existing dataset and finally tested against new data to determine its accuracy and predictive capabilities. This whole process finally helps to predict stocks in an accurate way, which makes proactive stock management possible in order to prevent or minimize out-of-stock circumstances. The architecture of the proposed out-of-stock prediction model is shown in Figure 1.

3.1. Input

The data utilized in the out-of-stock prediction model are gathered from [22], and are presented in mathematical form as follows:

$$K = \sum_{n=1}^{y} K_r \tag{1}$$

Here, K denotes the database and K_r denotes the total number of data present in the dataset, which is from 1 *to* y.

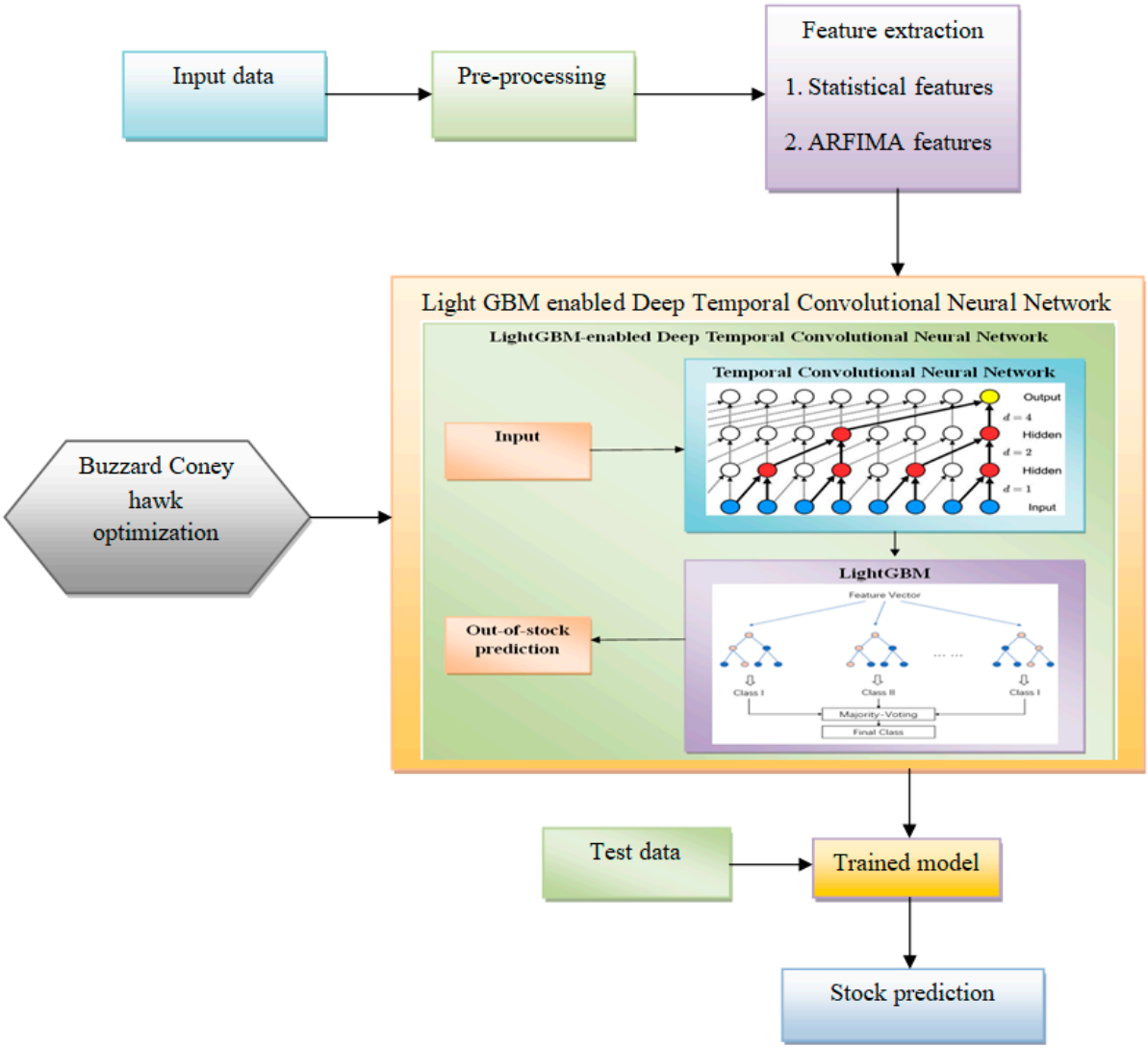

Figure 1. Architecture of the proposed out-of-stock prediction model.

3.2. Pre-Processing

The process of data cleansing involves identifying and correcting or removing incorrect or noisy data from a dataset. It typically focuses on detecting and replacing incomplete, inaccurate, irrelevant, or noisy data, or other inappropriate entries. The specific problems this research focuses on are as follows.

a. Removing irrelevant or duplicate data: Duplications can arise in the dataset when data are gathered from various sources, including scraped data and data received from multiple clients. This is a common occurrence in data collection and consolidation processes.
b. Structural error fixing: Inaccuracies arise from the incorrect labeling of categories or classes. These discrepancies can also stem from unconventional naming practices, spelling errors, or improper capitalization.
c. Missing values: Missing values are a common occurrence in datasets, often caused by data validation rules or errors in data collection. However, it is important to address these missing values as they can impact the overall quality of a model. If only a small number of values are missing, simple interpolation methods can be used to fill in the gaps.

This research uses very thorough methods to clean the data in order to make sure that the dataset is of high quality and is reliable. To start, deduplication methods are used to get

rid of any unnecessary or copied data, which helps in finding and removing redundant entries from different data sources. Structural errors, like inconsistencies in labeling and unusual naming conventions, are fixed through standardized processes to make sure that the categories and classes are consistent and accurate. Also, missing values are filled in using interpolation techniques to maintain the completeness and integrity of the dataset. By methodically tackling these obstacles, the data cleaning procedure improves the dataset for training models, resulting in more precise and dependable forecasts of out-of-stock occurrences in the retail setting.

Deduplication: This is the process of removing duplicate data from a dataset. During a secure data deduplication process, a tool assesses the data to find and remove extra copies, allowing for the storage of a single instance.

Standardization: Standardization in data processing involves converting data into a uniform format or scale for greater consistency and comparability. This entails updating the values of variables in a dataset to have a mean of zero and a standard deviation of one. By standardizing the variables, differences in scales are eliminated, making data interpretation and analysis more straightforward [5].

Interpolation: Interpolation methods involve using certain approaches to predict the values of missing data points in a dataset by referencing the values of neighboring data points. These methods are utilized when there are holes or missing values in a dataset, with the goal of filling in these gaps without altering the overall properties of the data [3]. So, after the completion of data cleansing, the preprocessed output is denoted as K_r^*.

3.3. Feature Extraction

The main objective of feature extraction is to recognize important traits within the input data. This study utilizes the following techniques to extract features.

Statistical Features

In general, statistical characteristics are features such as summary statistics derived from the dataset that describe its distribution, central tendency, variability, and shape. These elements offer information about the hidden patterns and features of the data, which can add value to predictive modeling. Common statistical features include the following:

(a) Mean: The first raw moment that informs about average stock levels during the period under consideration is a mean. The mean is defined as

$$M = \frac{1}{N}\sum_{r=1}^{N} K_r^* \tag{2}$$

Here, the preprocessed data point present in the dataset is denoted as K_r^*, and the overall number of preprocessed data points is denoted as N.

(b) Variance: Variance can be described as the square of the difference between a random variable and its average value in arithmetic terms. In Equation (3), the variance formula itself is given.

$$\sigma^2 = \frac{1}{N}\sum_{r=1}^{N}(K_r^* - M)^2 \tag{3}$$

(c) Standard deviation: The standard deviation defines the spread or variability of the levels of the stock surrounding the mean value. A greater value of standard deviation demonstrates larger fluctuations, which signifies variations in stock supply over time.

$$\sigma = \sqrt{\sigma^2} \tag{4}$$

(d) Skewness: This is a statistical concept that measures the asymmetry of a probability distribution of a random variable compared to its mean. In the field of probability theory and statistics, the skew of returns may be positive, zero, negative, or undetermined.

$$SK = \frac{\frac{1}{N}\sum_{r=1}^{N}(K_r^* - M)^3}{\sigma^3} \tag{5}$$

(e) Kurtosis: This is a mathematical computation that shows that the temporal tails of a distribution go further than the ones of nominal distribution. This implies that kurtosis is a measure of the extremes in the tails of a given distribution.

$$KU = \frac{\frac{1}{N}\sum_{r=1}^{N}(K_r^* - M)^4}{\sigma^4} \tag{6}$$

(f) Entropy: Entropy quantity is used to measure the level of disorder in a random variable. This implies that higher entropy values tend to occur in the presence of greater unpredictability in stock behavior.

$$E_n = -\sum_{r=1}^{N} p(K_r^*)\log_2(p(K_r^*)) \tag{7}$$

Here, in the preprocessed dataset, the probability occurrence of each unique value is denoted as $p(K_r^*)$.

(g) Min and max: This feature shows us the maximum and minimum inventory levels of this dataset. They provide information concerning stock levels and assist in the comprehension of the best upper and lower bounds of stock availability.

$$\min = \min(K^*{}_1, K^*{}_2, \ldots\ldots K^*{}_N) \tag{8}$$

$$\max = \max(K^*{}_1, K^*{}_2, \ldots\ldots K^*{}_N) \tag{9}$$

(h) Sum: Stock is presented as a sum of stock levels over a period of time. This provides an overview of stock availability in a given time span, and allows for the analysis of trends.

$$S = \sum_{r=1}^{N} K_r^* \tag{10}$$

The final statistical feature output is denoted as s.

3.4. ARFIMA Features

The ARFIMA attribute is applied to the dataset in this research, which stands for features derived from Autoregressive Fractionally Integrated Moving-Average models. This involves long-range dependencies and memory effects, making the model more accurate, especially for complicated tasks where deep learning is involved.

Grange, Joyeux, and Hosking suggested the ARFIMA (q, e, r) model as a way to study the long-memory property by combining fractional differenced noise and the autoregressive moving average. The formula for the ARFIMA (q, e, r) model for the time series z_t is as follows:

$$\varphi(Z)(1-Z)^e z_t = \Theta(Z)\varepsilon_t, \qquad t = 1, 2, \ldots\ldots U \tag{11}$$

The time series z_t consists of the autoregressive and moving-average polynomials, $\varphi(Z) = 1 - \varphi_1 Z - \varphi_2 Z^2 - \ldots - \varphi_q Z^q$ and $\Theta(Z) = 1 - \theta_1 Z - \theta_2 Z^2 - \ldots - \theta_q Z^r$, respectively, where $\varepsilon_t \approx (0, \sigma^2)$ represents the white noise process with a mean of zero and a variance of σ^2. The backward shift operator Z is used to define the fractional differencing operator $(1-Z)^e$ through a binomial expansion.

$$(1-Z)^e = \sum_{k=0}^{\infty} \binom{e}{k} (-1)^k Z^k \tag{12}$$

$$\begin{pmatrix} e \\ k \end{pmatrix}(-1)^k = \frac{\Gamma(e+1)(-1)^k}{\Gamma(e-k+1)\Gamma(k+1)} = \frac{\Gamma(-e+k)}{\Gamma(-e)\Gamma(k+1)} \tag{13}$$

The gamma function is denoted by $\Gamma(*)$ and e represents the number of differences needed for a series to become stationary in an ARFIMA (q, e, r) process. If $e \in (0, 0.5)$ is the number of differences required for the process to be stationary, then the process exhibits long-memory behavior. The concatenated statistical and *ARFIMA* features are denoted as $T = [SF, ARFIMA]$, and form the input for the LightGBM-enabled deep temporal convolutional neural network model.

3.5. Working of TCN LightGBM in Out-of-Stock Prediction

The concatenated feature extraction output T forms the input for the out-of-stock prediction model. The deep temporal CNN, which has been designed to support time-series data, visualizes the temporal conditions of the data through causal convolution and dilation, and is thus able to capture long-term dependencies without the leakage of future information. In this way, the entire process aims to provide feedback on these features in the form of temporal patterns and trends in the LightGBM model. LightGBM, which is a powerful platform with high efficiency and accuracy, fine-tunes the feature representations and predicts the occurrence possibility of out-of-stock situations based on the extracted features. This hybrid approach integrates the strengths of deep learning techniques with those of the gradient boosting method, which helps to preserve reliability and accuracy in prediction, and leads to a proactive stock management strategy aimed at preventing or reducing out-of-stock situations.

Time-series data are effectively handled by TCNs, specialized convolutional neural networks designed for this purpose. TCNs follow two important principles: maintaining the output length consistent with the input sequence, like long short-term memory (LSTM) networks, and avoiding information leakage from the future to the past through the use of causal convolutions. Unlike standard convolutions, causal convolutions utilized in TCNs do not take future values as inputs when calculating the output at a given time step. This means that the output at time t is generated using a kernel size s and the values of $T_{t-(s-1)}, T_{t-(s-2)}, ..., T_{t-1}, T_t$. Additionally, zero padding of length s is implemented at each layer to maintain consistency in the input sequence length.

In addition, in order to capture more extensive patterns over time, temporal convolutional networks utilize one-dimensional dilated convolutions. These convolutions help expand the network's field of perception without the need for pooling operations, ensuring that resolution is not sacrificed. Dilation involves skipping certain values between input points in the convolutional process. The overall process of dilated causal convolution across successive layers can be expressed as follows:

$$TCN = h\left(\sum_{s=o}^{s-1} w_m^s y_{(m-1)}^{(t-(s\times e))} + b_m\right) \tag{14}$$

At position t in the mth layer of a neural network, TCN represents the neuron output, s denotes the convolutional kernel width, w_m^s is the weight at position s, e is the convolution dilation factor, and b_m is the bias term. Here, h in the equation refers to the activation function used on the result of the convolution. Rectified Linear Units (ReLU) are commonly used as the activation function $(h(y) = \max(0, y))$. To increase the network's receptive field, multiple TCN blocks can be concatenated, but these result in more parameters and complicate learning. To address this, a residual connection is added to each TCN block's output, following the concept introduced to improve performance in deep architectures, where the input of the TCN block is added to its output $(o = h(y + F(y)))$.

The characteristics possessed by TCNs make them well suited for tackling complex time-series issues. One key advantage of TCNs is their ability to manage inputs of varying lengths through the use of a one-dimensional causal convolutional kernel, similar to RNNs. Additionally, TCNs are more efficient in terms of memory compared to recurrent

networks, as they can process long sequences simultaneously due to their shared convolution architecture. Unlike RNNs, which process input sequences sequentially and result in increased computation time, TCNs are trained using the standard backpropagation algorithm, eliminating the gradient issues associated with the backpropagation through time (BPTT) algorithm used in RNNs.

3.5.1. LightGBM

LightGBM is a distributed gradient-boosting framework used for machine learning that stands out for its implementation of decision trees in tasks such as classification, ranking, and regression. The purpose of LightGBM is to provide a fast and scalable algorithm capable of handling large datasets. It is recognized for its fast training speed and effective memory utilization. Compared to XGBoost, LightGBM has been found to be faster and more accurate in various benchmarks and experiments. LightGBM's unique approach to leaf-wise tree growth sets it apart from other tree-based learning algorithms, like XGBoost, which typically use level-wise tree growth. While level-based tree growth advances the structure of a tree step by step, leaf-based tree growth prioritizes expanding the tree according to the nodes, resulting in the largest reduction in loss.

The leaf-wise tree method has smaller tree nodes compared to the level-wise tree method of the same depth, resulting in faster training on large datasets. LightGBM utilizes two unique techniques: Gradient One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). GOSS removes data instances with minimal gradients and keeps the rest of the data to calculate information gain, as instances with higher gradients have a bigger influence on information gain calculations. By using a smaller dataset, GOSS can accurately estimate information gain. EFB, the second technique, reduces the number of features by combining mutually exclusive features. The decision to use LightGBM is based on its considerable success, which is attributed to its advantages over other algorithms. LightGBM is well known for its rapid training speed, accurate results, and efficient use of memory. Additionally, it is praised for its ability to support parallel, distributed, and GPU learning, as well as its capacity to effectively handle extensive datasets. These advantages have made LightGBM a popular choice in many winning solutions in machine learning competitions.

3.5.2. Concatenation of Both Models

The input value T is first processed by the TCN, after which it goes to LightGBM. Now, the output of the TCN is denoted as T_{TCN}. The TCN manipulates spatial input using causal convolutions and dilation to capture temporal patterns using an efficient approximation. It prevents the release of data at the next time step and can handle variable-length inputs. The output T_{TCN} encodes the learned characteristics of the input sequence and then feeds the LightGBM model for the subsequent prediction. Let us call the function that learns for the LightGBM model $l_{GBM}(T_{TCN})$, which is used to predict the target variable based on the information the TCN has extracted. Mathematically, the process can be represented as follows:

$$T_{TCN} = l_{TCN}(T) \text{ \#TCN} \tag{15}$$

$$PO = l_{GBM}(T_{TCN}) \text{ \#LGBM} \tag{16}$$

So, the final output is

$$PO = l_{GBM}(l_{TCN}(T)) \tag{17}$$

This procedure involves the TCN extracting informative temporal attributes from the input sequence. These are further integrated into the LightGBM model, which then makes the predictions. This combination uses the positive features of both models to optimize the model's forecasting abilities. The architecture of the proposed LightGBM-enabled deep temporal convolutional neural network is shown in Figure 2. Here, the input layer is represented in blue circles, the red circles correspond to hidden layers, and the output layer is indicated as yellow circles in the TCN. The configuration of the neural network layers is included in Table 1.

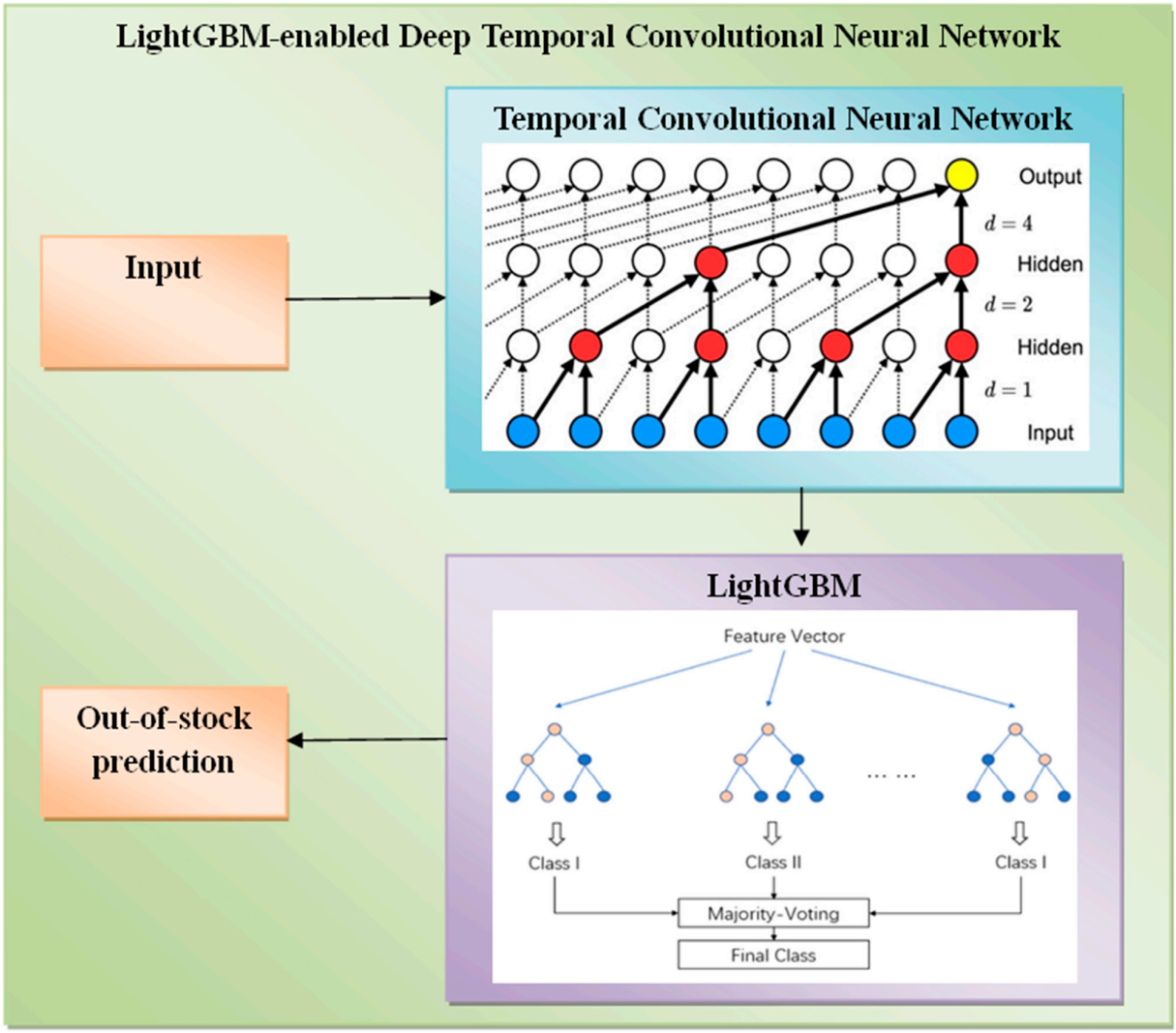

Figure 2. Architecture of the proposed TCN LightGBM.

Table 1. Configurations of the neural network layers.

Layers	Dimensions	Trainable Weights	Output Details
input_1 (Input Layer)	[(None, 25, 1, 1)]	0	[]
conv2d (Conv2D)	(None, 25, 1, 16)	160	[i'nput_1[0][0]']
activation (Activation)	(None, 25, 1, 16)	0	[c'onv2d [0][0]']
batch normalization (Batch Normalization)	(None, 25, 1, 16)	64	[a'ctivation [0][0]']
max_pooling2d (MaxPooling2D)	(None, 25, 1, 16)	0	[b'atch_normalization [0][0]']
dropout (Dropout)	(None, 25, 1, 16)	0	[m'ax_pooling2d [0][0]']
conv2d_1 (Conv2D)	(None, 25, 1, 32)	4640	[d'ropout [0][0]']
activation_1 (Activation)	(None, 25, 1, 32)	0	[c'onv2d_1[0][0]']
batch_normalization_1 (Batch Normalization)	(None, 25, 1, 32)	128	[a'ctivation_1[0][0]']
conv2d_2 (Conv2D)	(None, 25, 1, 64)	18,496	[b'atch_normalization_1[0][0]']

Table 1. *Cont.*

Layers	Dimensions	Trainable Weights	Output Details
activation_2 (Activation)	(None, 25, 1, 64)	0	[c'onv2d_2[0][0]']
batch_normalization_2 (Batch Normalization)	(None, 25, 1, 64)	256	[a'ctivation_2[0][0]']
max_pooling2d_1 (MaxPooling2D)	(None, 25, 1, 64)	0	[b'atch_normalization_2[0][0]']
dropout_1 (Dropout)	(None, 25, 1, 64)	0	[m'ax_pooling2d_1[0][0]']
conv2d_3 (Conv2D)	(None, 25, 1, 128)	73,856	[d'ropout_1[0][0]']
activation_3 (Activation)	(None, 25, 1, 128)	0	[c'onv2d_3[0][0]']
batch_normalization_3 (Batch Normalization)	(None, 25, 1, 128)	512	[a'ctivation_3[0][0]']
max_pooling2d_2 (MaxPooling2D)	(None, 25, 1, 128)	0	[b'atch_normalization_3[0][0]']
(Temporal Attention)			
global_average_pooling2d (Glob alAveragePooling2D)	(Glob (None, 128)	0	[m'ax_pooling2d_2[0][0]']
global_max_pooling2d_1 (GlobalMaxPooling2D)	(None, 128)	0	[g'lobal_average_pooling2d [0][0]'
reshape (Reshape)	(None, 1, 1, 128)	0	[g'lobal_average_pooling2d [0][0]'
reshape_2 (Reshape)	(None, 1, 1, 128)	0	[g'lobal_max_pooling2d_1[0][0]']
dense (Dense)	(None, 1, 1, 16)	2064	[r'eshape [0][0]', r'eshape_2[0][0]']
dense_1 (Dense)	(None, 1, 1, 128)	2176	[d'ense [0][0]', d'ense [2][0]']
add (Add)	(None, 1, 1, 128)	0	[d'ense_1[0][0]', d'ense_1[2][0]']
activation_4 (Activation)	(None, 1, 1, 128)	0	['add [0][0]']
multiply (Multiply)	(None, 25, 1, 128)	0	[m'ax_pooling2d_2[0][0]', a'ctivation_4[0][0]']
dropout_2 (Dropout)	(None, 25, 1, 128)	0	['multiply [0][0]']
Flatten (Flatten)	(None, 3200)	0	[d'ropout_2[0][0]']
Features --------->LightGBM----------> Output			

4. Proposed Buzzard Coney Hawk Optimization

Additionally, the BCHO algorithm adjusts the hyperparameters of the classifier in order to achieve an accurate out-of-stock prediction model. BCHO is an innovative optimization algorithm founded on the hunting strategies and the food-finding behaviors of rabbits, African vultures, and Harris hawks. The rabbit's behaviors of instinctual detours during foraging and random hiding from enemies are adopted as the approach to avoiding complex search areas. This is incorporated into BCHO. Also, the method fuses the cooperative ability and surprise-pounce hunting style of Harris hawks, which allows them to successfully locate and take down speedy prey. Combining these two distinctive features, BCHO not only avoids the inborn limitations of other existing optimization algorithms, but also boosts the speed of convergence. Thus, BCHO is one of the finest candidates for various applications to solve complex optimization problems, regardless of the field.

BCHO is a heuristic algorithm inspired by the hunting behaviors of buzzards, Coney hawks, and conies. It aims to efficiently search for the best solutions in complex optimization problems. The algorithm starts by creating a group of possible solutions, which are then assessed using a specific objective function. In the buzzard phase, individuals work together to explore the solution space and find promising areas with high-quality solutions. On the other hand, during the Coney hawk phase, individuals compete to further exploit these areas. By going through rounds of selection, crossover, and mutation, BCHO strikes a

balance between exploring new possibilities and exploiting known solutions, ultimately moving towards the best solutions. Pairing BCHO with LightGBM and a temporal CNN enhances its effectiveness in different situations, especially in predicting stock market outcomes. For instance, BCHO has the capability to fine-tune the hyperparameters of LightGBM in order to steer the model towards settings that result in better accuracy and generalization. Moreover, BCHO can assist in feature engineering and selection by pinpointing meaningful predictors from the input data to enhance the efficiency and predictive capability of the model. When used in conjunction with temporal CNNs, BCHO directs the investigation of temporal patterns in stock market data, helping to pinpoint crucial dynamics and trends that impact market fluctuations.

Inspiration:

A remarkable observation of the survival tactics employed in nature, especially evasive and pursuing behaviors, inspired the development of BCHO. Researchers were greatly inspired by the adaptability of rabbits, African vultures, and Harris hawks. They observed the synergy of these characteristics, and thus the thought-out framework of BCHO was developed. The rabbit adroitly navigates its world through detour foraging and evasive absconds, and the African vulture maintains a precise searching technique for its food, while the Harris hawk utilizes a cooperative surprise-pounce technique to secure its prey. The correlation of these attributes in BCHO portrays a superb amalgamation of nature's finest machinations. By combining these characteristics with interdisciplinary synthesis, BCHO is a demonstration of biomimicry intelligence, offering a novel strategy for optimization that utilizes the experiences of the animal kingdom to solve complicated problems with unprecedented accuracy and efficiency.

4.1. Initialization

The starting point of the solution, consisting of a randomly selected population, is expressed mathematically as

$$X^t = X_{t-1} - r_1(u - l)V_{t-1} \tag{18}$$

Here, X_t, X_{t-1}, and V_{t-1} denote the position at t iterations and the velocity at $t-1$ iterations; u and l denote the upper and lower bounds; and r_1 represents the step-by-step solution.

4.2. Fitness Evaluation

Fitness evaluation allows problems to be quantified in accordance with predefined metrics, and hence, the optimization algorithm will move closer to optimal solutions.

4.3. Parameter Update

The parameter update includes parameter tuning, which means that the values of the parameters in the model or algorithm are adjusted to improve its performance or adapt to new conditions.

(i) Searching phase:

$if|G| \geq 1$. Here, G represents the satiation factor of the solution.

This above condition shows the solution has a satiation value greater than or equal to 1. This shows the solution has been satiated and it should be in the exploration phase. Finding food for the solution may become difficult due to the increased search space. Thus, it may need to travel long distances to search for food. Thus, it uses a technique to save energy levels, which is known as a detour foraging characteristic. This involves tracking the food traces of other leading solutions and approaching the food.

$$X^{t+1} = \frac{\left[(X_{rand} - r_2|X_{rand}(t) - 2r_3X_t|) + \left(X_t + R\middle|X_g - X_t\right| + round(0.5(0.05 + r_4)).n)\right]}{2}, if\ q \geq 0.5 \tag{19}$$

The equation seems to describe a way of shifting the position X_{t+1} of a solution during the search process, and encompasses a detour foraging model, the determination of the exploration phase, the approach of other solutions' food traces, and energy conservation. It is a strategy in which solutions change their location in response to random factors, current positions, and the positions of other solutions in the search space when the goal is to ultimately find the best solution.

$$X^{t+1} = \frac{X_{prey} - X^t_{mean} - r_3(l + r_5(u-l)) + \left[X_t + R\left|X_g - X_t\right| + rand(0.5(0.05 + r_4))\right]}{2}, if\ q \geq 0.5 \tag{20}$$

Here, q denotes the perching factor and X_{rand} denotes the random position.

$$r_2 = \left[1 - \left(\frac{e^{t-1}}{e^t}\right).t\right],\ r_3\varepsilon(0,2) \tag{21}$$

Here, $2r$ is an error adjustment factor that is considered important in optimization procedures. This equation calculates $2r$ with the help of a relative change of error between two sequential time steps, t and $t-1$, by multiplying $\left(\frac{e^{t-1}}{e^t}\right)$ with t and subtracting it from 1,adjusting r_2, and affecting the optimization. The parameter r_3 is constrained to the interval $(0,2)$, so there are adjustments within certain limits. Therefore, r_2 controls the behavior of the optimization by controlling the relative error change, while r_3 controls the magnitude of this charge. This formulation gives a simplified effective way of adjusting the optimization process according to new error trends, which is vital when it comes to optimizing the optimization results.

$$R = \left(e - e\left(\frac{t-1}{t_{\max}}\right)^2\right).\sin(2\pi) \tag{22}$$

Here, the term R represents a foraging character with respect to direction, r_4 denotes the anonymous factor, n denotes the random population number, N denotes the total population, *round* means rounding factor, and r_5 means hectic value $(\varepsilon\ 2\ to\ 1)$.

(ii) CNN

$if\ |G| < 1$. This condition shows the predator is in a hunger state and it prefers exploitation to exploration.

Subcase: (i) $if|E_l| \geq |The_l|$. This is the aggressive attack phase, where E_l, denotes the energy level of prey, The_l denotes the energy level of prey, and $The_l \cong 0.5$.

The previous condition shows the prey has an energy level greater or equal to the required threshold level. Then, the solution uses the aggressive safe-fight strategy to slow down the movement and energy level of the prey. Due to continuous surprise-pounce attacks by the solution, the movement of the prey is eventually stopped.

$$X^{t+1} = \begin{cases} 0.5\left[\frac{B_1+B_2}{2}\right] + \\ 0.5\left[\left(X_{prey} - X_t\right) - E_l/J\left(X_{prey} - X_t\right)\right] \end{cases} \tag{23}$$

Here, E_l denotes the energy level of the prey. $J = 2 - \frac{k}{t_{\max}}$, where J means the escape factor and the k term represents the conditional movement of the prey. The t in X_{t+1} is literally a subscript showing that we are at a certain step of the iteration. The equation formulates the update rule for the position X of an entity at time $t+1$, depending on X_{prey}, X_t, E, J and constants B_1 and BB_2.

$$B_1 = X^t_g - \frac{X^t_g.X_t}{X^t_g - (X_t)^2}.G \tag{24}$$

$$B_2 = X_g^{t-1} - \frac{X_g^{t-}.X_t}{X_g^{t-1} - (X_t)^2}.G \tag{25}$$

Here, B_1 and B_2 are the fractional global best-based positional vectors (X_g^t and X_g^{t-1}).
Subcase (ii): $if|E_l| < |The_l|$. This is the rotational flight phase.

The condition shows the prey energy condition is less than the needed threshold level to escape. Thus, the solution is to undertake rotational flight frequency until the energy level of the solution prey is exhausted. Thus, the energy consumption for the solution is decreased in manner. Thus,

$$X^{t+1} = \frac{\left[X_g - (s_1 + s_2)\right] + \left[X_{prey} - E_l \middle| JX_{prey}^t - X_{mean}\right]}{2} \tag{26}$$

Here, X_{mean} denotes the average position of the solution.

$$s_1 = X_g.\left(\frac{r_6.X_t}{2\pi}\right).\cos(X_t) \tag{27}$$

$$s_2 = X_g.\left(\frac{r_7.X_t}{2\pi}\right).\sin(X_t) \tag{28}$$

s_1 and s_2 are the angular function-based positional vectors; r_6 and r_7 represent the clockwise and anticlockwise movements of the solution $\varepsilon(-1, 1)$.

The flowchart for the proposed Buzzard Coney Hawk Optimization method is shown in Figure 3. The arrow mark indicates the flow of optimizations and if the condition is satisfied the process flows through the 'yes' side otherwise, the process flows through the 'no' side.

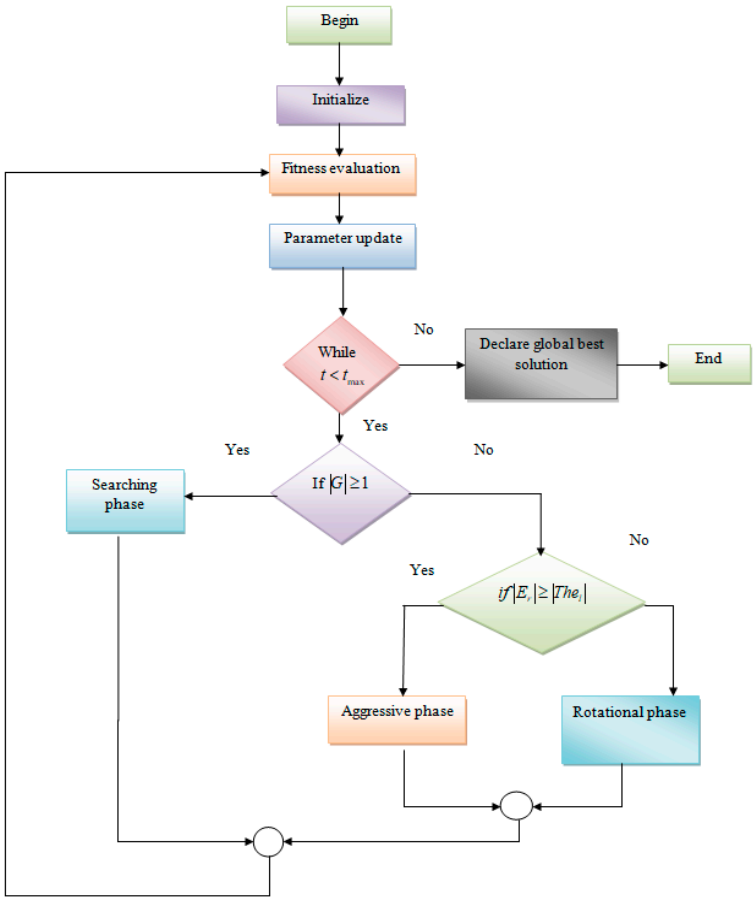

Figure 3. Flow chart for the proposed Buzzard Coney Hawk Optimization.

5. Result and Discussion

After implementing the BCHO-TCN LightGBM model for forecasting out-of-stock items, the model's efficiency was evaluated by measuring its performance against other top models.

5.1. Experimental Setup

To conduct the out-of-stock experiment, a Python program was run on a Windows 10 system with an 8 GB memory capacity.

5.2. Dataset Description

Product listing from Amazon India dataset [22]: This dataset contains approximately 30,000 records and includes information such as total record count (553,724), domain name (amazon.in), date range (1–31 October 2019), and file extension (CSV). Fields that are available include unique ID, crawl timestamp, category, title of product, description of product, brand, quantity or pack size, MRP, price, name of site, discounts, bundle deals, stock availability, ASIN of product, and URLs of images.

5.3. Performance Analysis Based on TP

Figure 4 showcases the results of using the BCHO-TCN LightGBM model for predicting out-of-stock occurrences. Figure 4a shows the accuracy for epochs 100, 200, 300, 400, and 500 as 89.78%, 89.8%, 93.92%, 94.09%, and 94.52%, respectively, with a TP of 90. Likewise, Figure 4b displays the outcomes of the BCHO-TCN LightGBM model in terms of NPV with a TP of 90, with values of 86.04%, 86.76%, 92.68%, 93.91%, and 95.16%. The results shown in Figure 4c for epoch values of 100, 200, 300, 400, and 500 are 87.90%, 91.43%, 92.60%, 94.60%, and 94.81% for PDV at a TP of 90. Similarly, Figure 4d illustrates the findings of the BCHO-TCN LightGBM model in terms of threat score with a TP of 90, with values of 86.63%, 91.68%, 93.00%, 93.79%, and 95.76%.

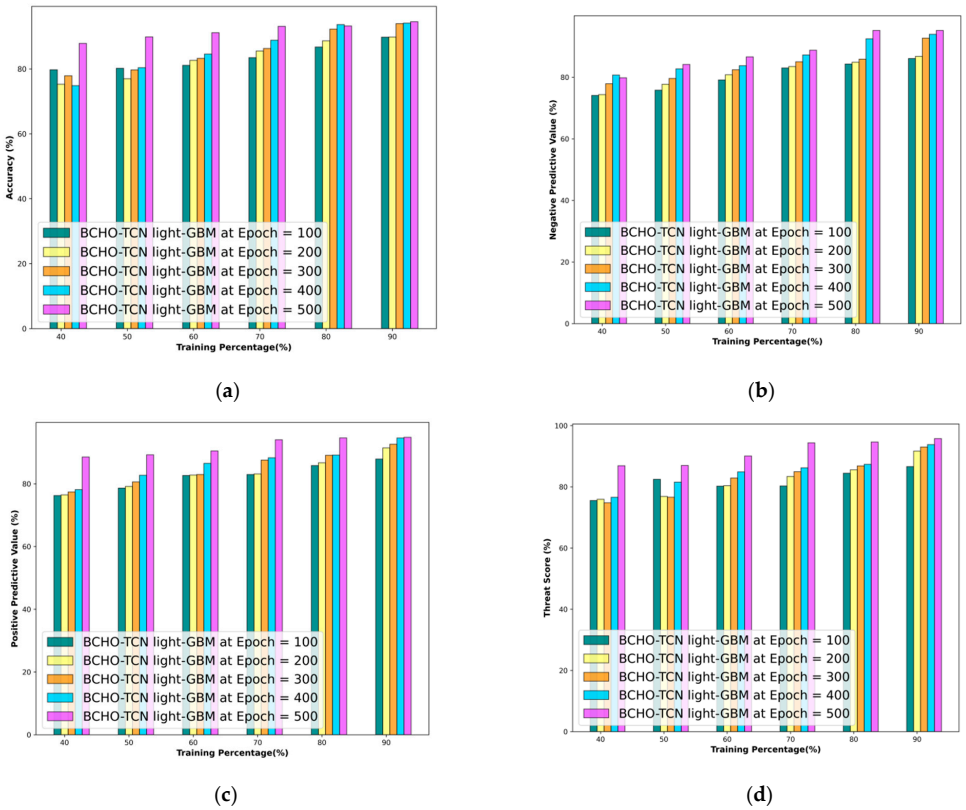

Figure 4. Performance analysis based on TP: (**a**) accuracy, (**b**) NPV, (**c**) PDV, and (**d**) threat score.

5.4. Performance Analysis Based on K-Fold

Figure 5 showcases the results of the BCHO-TCN LightGBM model for predicting out-of-stock occurrences. Figure 5a shows that the accuracy increases with each epoch value, with percentages of 83.35%, 88.01%, 90.26%, 92.51%, and 94.81% for epochs of 100, 200, 300, 400, and 500, respectively, using a 10-fold method. Likewise, Figure 5b displays the outcomes of the BCHO-TCN Light GBM model for NPV with a k-fold of 10, with values of 93.34%, 93.48%, 93.49%, 94.76%, and 95.60%. The results shown in Figure 5c for the epoch values of 100, 200, 300, 400, and 500 are 91.29%, 91.76%, 92.86%, 93.63%, and 96.28% for PDV at a k-fold of 10. Similarly, Figure 5d illustrates the findings of the BCHO-TCN LightGBM model in terms of threat score with a k-fold of 10, with values of 88.25%, 93.93%, 94.14%, 94.34%, and 95.86%.

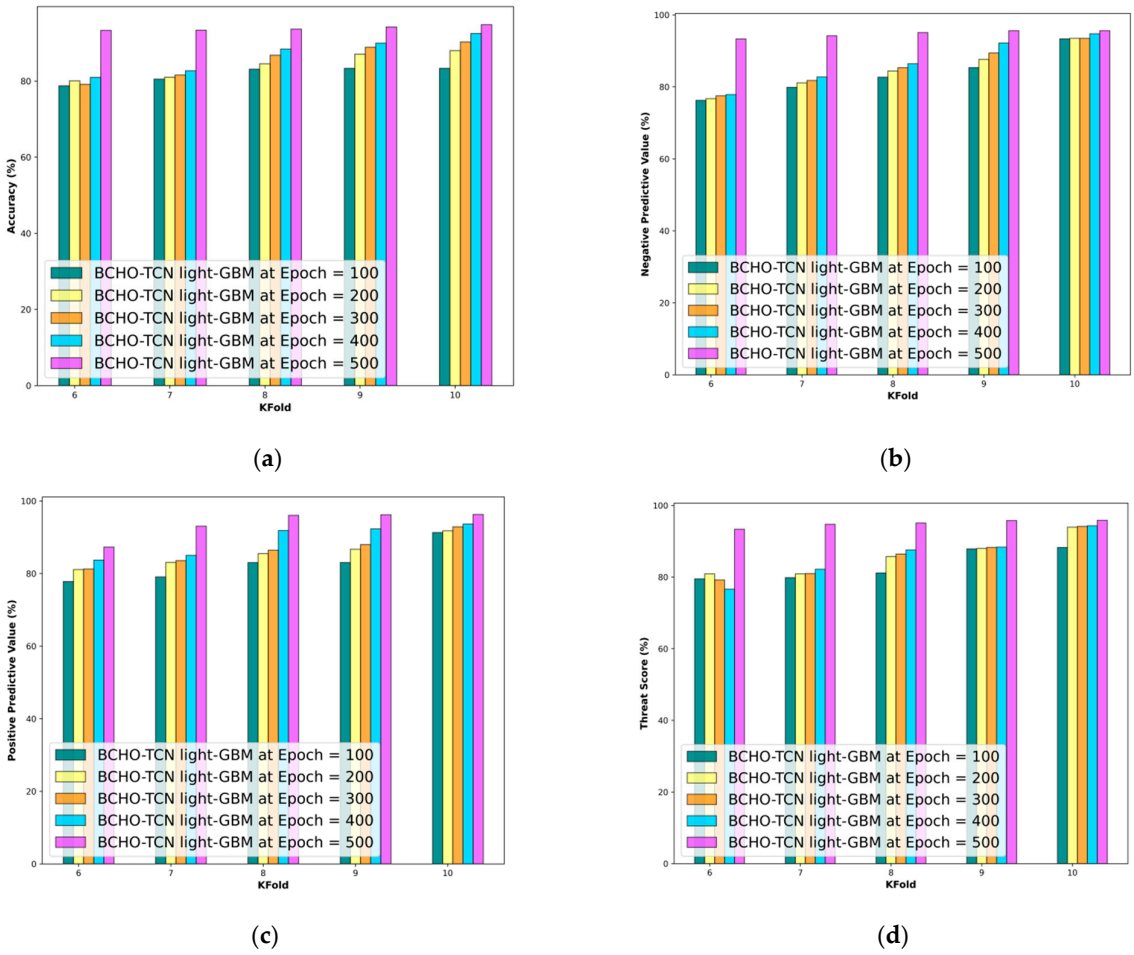

Figure 5. Performance analysis based on k-fold: (**a**) accuracy, (**b**) NPV, (**c**) PDV, and (**d**) threat score.

5.5. Comparative Methods

To highlight the accomplishments of the BCHO-TCN LightGBM model, a comparison was conducted. This assessment included an analysis of different approaches like perceptron [26], CNN-LSTM [27], BiLSTM [28], rabbit-TCN Light GBM, hawk-TCN LightGBM, and vulture-TCN LightGBM.

5.5.1. Comparative Analysis Based on TP

Figure 6a–d show the comparative analysis based on TP for accuracy, NPV, PDV, and threat score. The BCHO-TCN LightGBM model demonstrates superior accuracy in predicting out-of-stock events, with a 2.13% improvement over the vulture-TCN LightGBM model, achieving an accuracy of 94.52%.

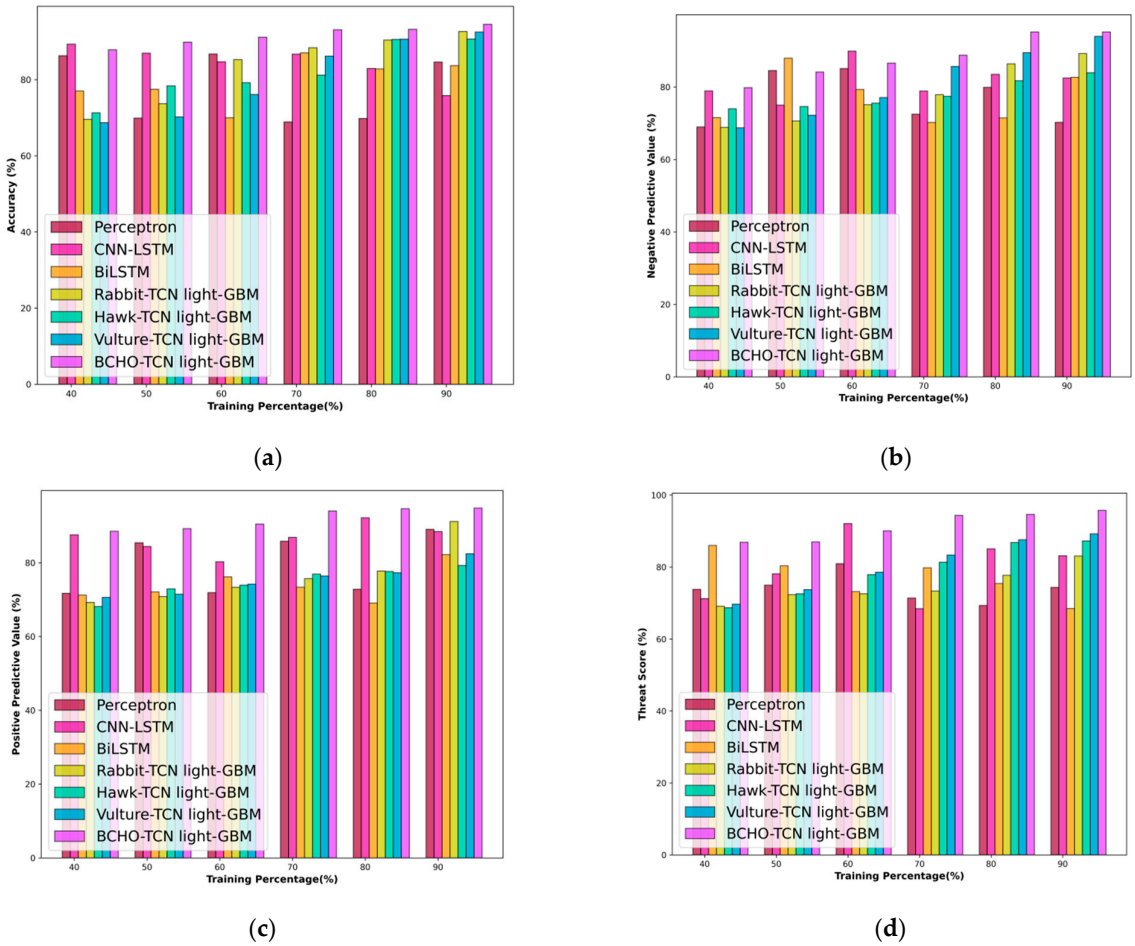

Figure 6. Comparative analysis based on TP: (**a**) accuracy, (**b**) NPV, (**c**) PDV, and (**d**) threat score.

The BCHO-TCN LightGBM model shows better performance in predicting out-of-stock situations compared to the vulture-TCN LightGBM model. It surpasses the vulture-TCN model by 1.28% and achieves an NPV of 95.16% with a TP of 90.

The BCHO-TCN LightGBM model outperforms the vulture-TCN LightGBM model in out-of-stock prediction by a margin of 13.08%. With a TP of 90, it boasts a PPV of 94.81%, surpassing existing methods.

The BCHO-TCN LightGBM model shows better performance in predicting out-of-stock events compared to the vulture-TCN LightGBM model. It surpasses the vulture-TCN model by 6.81%, achieving a threat score of 95.76% with TP 90.

5.5.2. Comparative Analysis Based on K-Fold

Figure 7a–d show the comparative analysis based on k-fold for accuracy, NPV, PDV, and threat score. The BCHO-TCN LightGBM model demonstrates superior accuracy in predicting out-of-stock events, with a 10.09% improvement over the vulture-TCN LightGBM model, achieving an accuracy of 94.81%.

The BCHO-TCN LightGBM model shows better performance in predicting out-of-stock situations compared to the vulture-TCN LightGBM model. It surpasses the vulture-TCN model by 1.04% and achieves an NPV of 95.60% with a TP of 90.

The BCHO-TCN LightGBM model outperforms the vulture-TCN LightGBM model in out-of-stock prediction by a margin of 0.40%. With a TP of 90, it boasts a PPV of 96.28%, surpassing existing methods.

The BCHO-TCN LightGBM model shows better performance in predicting out-of-stock events compared to the vulture-TCN LightGBM model. It surpasses the vulture-TCN model by 5.84%, achieving a threat score of 95.86% with a TP of 90.

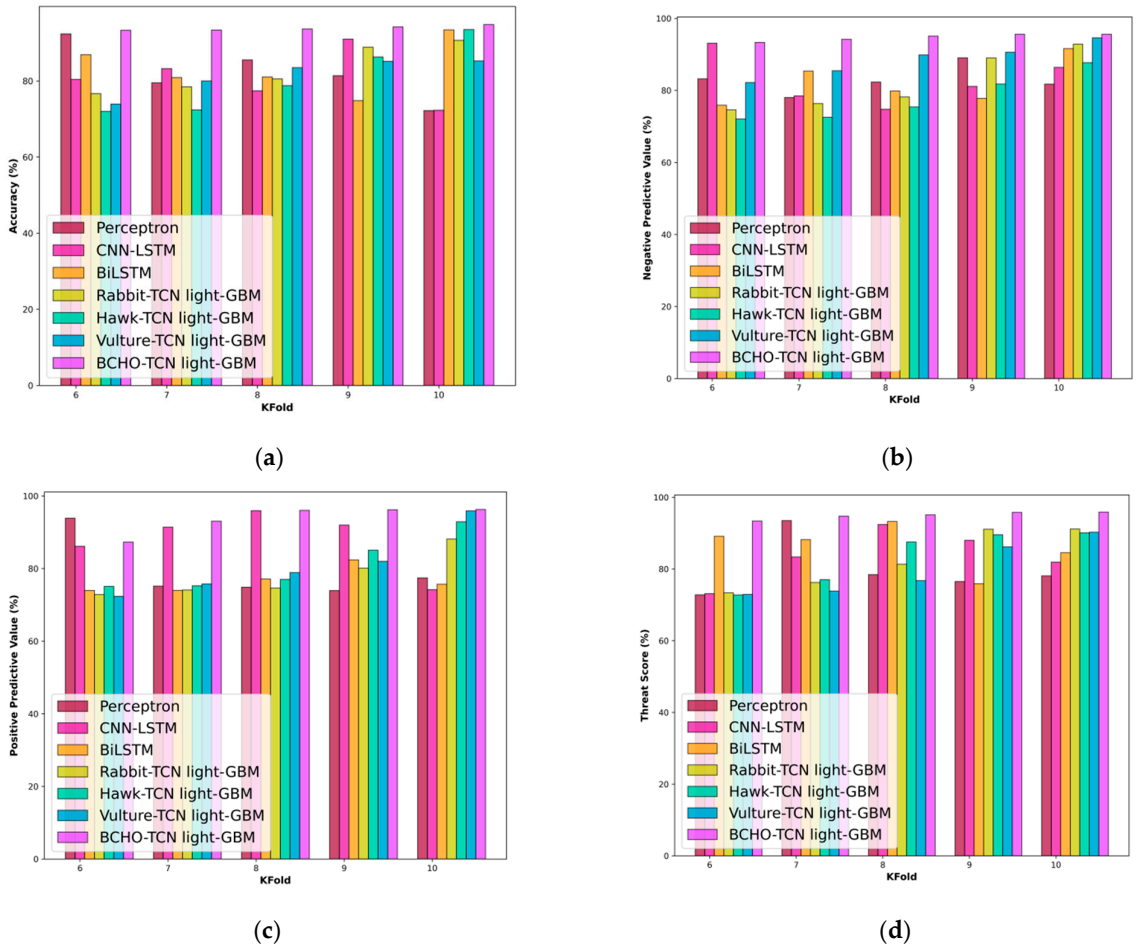

Figure 7. Comparative analysis based on TP:(**a**) accuracy, (**b**) NPV, (**c**) PDV, and (**d**) threat score.

5.5.3. Statistical Analysis Based on TP

Statistical analyses based on TP for accuracy, NPV, PDV, and threat score are included in Tables 2–5. The BCHO-TCN LightGBM model demonstrates the best accuracy in predicting out-of-stock prediction, with a 2.13% improvement over the vulture-TCN LightGBM model, achieving a maximum of 94.52%.

The BCHO-TCN LightGBM model demonstrates the best NPV when predicting out-of-stock prediction, with a 1.28% improvement over the vulture-TCN LightGBM model, achieving a maximum of 95.16%.

The BCHO-TCN LightGBM model demonstrates the best PPV when predicting out-of-stock prediction, with a 13.08% improvement over the vulture-TCN LightGBM model, achieving a maximum of 94.81%.

The BCHO-TCN LightGBM model demonstrates the best threat score in predicting out-of-stock prediction, with a 6.81% improvement over the vulture-TCN LightGBM model, achieving a maximum of 95.76%.

Table 2. Statistical analysis based on TP for accuracy.

Method	Best	Mean	Variance
Perceptron	86.74	77.70	67.32
CNN-LSTM	89.31	84.40	18.65
BiLSTM	87.03	79.67	30.99
Rabbit-TCN Light GBM	92.66	83.33	74.69
Hawk-TCN Light GBM	90.65	81.87	47.56
Vulture-TCN Light GBM	92.51	80.72	90.60
BCHO-TCN Light GBM	94.52	91.62	5.13

Table 3. Statistical analysis based on TP for NPV.

Method	Best	Mean	Variance
Perceptron	85.07	76.86378	43.44928
CNN-LSTM	89.88	81.43829	21.82321
BiLSTM	87.95	77.17706	43.94623
Rabbit-TCN Light GBM	89.22	78.00761	57.04131
Hawk-TCN Light GBM	83.91	77.85161	13.83371
Vulture-TCN Light GBM	93.95	81.17866	84.12889
BCHO-TCN Light GBM	95.16	88.26374	31.11267

Table 4. Statistical analysis based on TP for PPV.

Method	Best	Mean	Variance
Perceptron	89.05	79.4469	54.95356
CNN-LSTM	92.16	86.61442	13.39274
BiLSTM	82.18	74.00379	17.94495
Rabbit-TCN Light GBM	91.17	76.33485	52.0083
Hawk-TCN Light GBM	79.25	74.79213	13.57551
Vulture-TCN Light GBM	82.41	75.39529	15.62542
BCHO-TCN Light GBM	94.81	91.95131	6.807469

Table 5. Statistical analysis based on TP for threat score.

Method	Best	Mean	Variance
Perceptron	80.93	74.12895	12.94691
CNN-LSTM	92.08	79.67916	65.8623
BiLSTM	85.99	77.19707	31.61515
Rabbit-TCN Light GBM	83.07	74.68913	20.41043
Hawk-TCN Light GBM	87.25	79.09748	47.22807
Vulture-TCN Light GBM	89.24	80.36782	50.166
BCHO-TCN Light GBM	95.76	91.45962	13.36229

5.5.4. Statistical Analysis Based on K-Fold

Statistical analyses based on K-fold for accuracy, NPV, PDV, and threat score are included in Tables 6–9. The BCHO-TCN LightGBM model demonstrates the best accuracy

in predicting out-of-stock prediction, with a 10.09% improvement over the vulture-TCN LightGBM model, achieving a maximum of 94.81%.

The BCHO-TCN LightGBM model demonstrates the best NPV when predicting out-of-stock prediction, with a 1.04% improvement over the vulture-TCN LightGBM model, achieving a maximum of 95.60%.

The BCHO-TCN LightGBM model demonstrates the best PDV when predicting out-of-stock prediction, with a 0.40% improvement over the vulture-TCN LightGBM model, achieving a maximum of 96.28%.

The BCHO-TCN LightGBM model demonstrates the best threat score when predicting out-of-stock prediction, with a 5.84% improvement over the vulture-TCN LightGBM model, achieving a maximum of 95.86%.

Table 6. Statistical analysis based on k-fold for accuracy.

Method	Best	Mean	Variance
Perceptron	92.37	82.20	44.41
CNN-LSTM	91.00	80.87	38.75
BiLSTM	93.40	83.41	39.55
Rabbit-TCN Light GBM	90.67	83.04	31.98
Hawk-TCN Light GBM	93.49	80.59	68.66
Vulture-TCN Light GBM	85.25	81.56	18.25
BCHO-TCN Light GBM	94.81	93.87	0.32

Table 7. Statistical analysis based on k-fold for NPV.

Method	Best	Mean	Variance
Perceptron	89.06	82.88	12.68
CNN-LSTM	93.09	82.77	41.04
BiLSTM	91.61	82.11	32.71
Rabbit-TCN Light GBM	92.84	82.20	53.54
Hawk-TCN Light GBM	87.70	77.90	36.03
Vulture-TCN Light GBM	94.61	88.54	18.49
BCHO-TCN Light GBM	95.60	94.76	0.79

Table 8. Statistical analysis based on k-fold for PDV.

Method	Best	Mean	Variance
Perceptron	93.88	79.06	56.27
CNN-LSTM	95.95	87.93	57.12
BiLSTM	82.38	76.63	9.67
Rabbit-TCN Light GBM	88.12	77.96	32.00
Hawk-TCN Light GBM	92.88	81.05	48.38
Vulture-TCN Light GBM	95.89	80.97	65.97
BCHO-TCN Light GBM	96.28	93.77	12.00

Table 9. Statistical analysis based on k-fold for threat score.

Method	Best	Mean	Variance
Perceptron	93.49	79.86	50.45
CNN-LSTM	92.42	83.77	41.86
BiLSTM	93.24	86.19	34.28
Rabbit-TCN Light GBM	91.15	82.63	54.56
Hawk-TCN Light GBM	90.09	83.39	50.63
Vulture-TCN Light GBM	90.26	79.98	48.54
BCHO-TCN Light GBM	95.86	94.98	0.80

5.6. Comparative Discussion

The findings from comparing the BCHO-TCN LightGBM model with existing methods highlight its significant implications for predicting out-of-stock events and the results are presented in Table 10. By surpassing traditional techniques like perceptron, CNN-LSTM, and BiLSTM, the BCHO-TCN LightGBM model demonstrates the potential of combining models to enhance prediction accuracy and dependability. Despite its achievements, we should consider potential limitations such as data quality, scalability, and model interpretability. Nevertheless, the model's potential impact on industry practices is substantial. By integrating BCHO for fine-tuning, a TCN for feature extraction, and LightGBM for prediction, the method effectively captures complex temporal patterns and achieves superior accuracy. Compared to traditional and advanced models like perceptron, CNN-LSTM, and BiLSTM, the BCHO-TCN LightGBM model demonstrates enhanced performance. Additionally, it surpasses other hybrid models like rabbit-TCN LightGBM, hawk-TCN LightGBM, and vulture-TCN LightGBM, showcasing its effectiveness in proactive stock management. In order to showcase its excellence, the BCHO-TCN LightGBM model was tested against other models to assess its performance. The evaluation involved analyzing metrics with a TP of 90 and a k-fold of 10. The findings demonstrate that when utilizing the product listings from the Amazon India dataset, the model exhibits high levels of accuracy, in addition to excellent NPV, PDV, and threat scores, reaching values of 94.52%, 95.16%, 94.81%, and 95.76%, respectively. Similarly, at ak-fold of 10, the model achieves values of 94.81%, 95.60%, 96.28%, and 95.86% for the same metrics.

Table 10. Comparative discussion table for TP and K-fold.

Models	TP 90				k-Fold 10			
	Accuracy	NPV	PDV	Threat Score	Accuracy	NPV	PDV	Threat Score
Perceptron	84.61	70.24	89.05	74.35	72.21	81.75	77.44	78.09
CNN-LSTM	75.81	82.48	88.46	83.14	72.30	86.41	74.17	81.92
BiLSTM	83.69	82.63	82.18	68.44	93.40	91.61	75.68	84.52
Rabbit-TCN LightGBM	92.66	89.22	91.17	83.07	90.67	92.84	88.12	91.15
Hawk-TCN LightGBM	90.65	83.91	79.25	87.25	93.49	87.70	92.88	90.09
Vulture-TCN LightGBM	92.51	93.95	82.41	89.24	85.25	94.61	95.89	90.26
BCHO-TCN LightGBM	94.52	95.16	94.81	95.76	94.81	95.60	96.28	95.86

6. Conclusions

This study presents the innovative BCHO-TCN LightGBM model and offers valuable new insights and practical benefits for industry use, especially in predicting out-of-stock situations. Our research highlights its superior accuracy and reliability compared to conventional methods. The model has the best predictive performance of any that combines

Buzzard Coney Hawk Optimization with LightGBM and convolutional neural networks. LightGBM utilizes a fast and scalable algorithm capable of handling large datasets. It helps to increase training speed and contributes to effective memory utilization. This advancement shows great potential for industries that rely heavily on inventory management, such as retail, manufacturing, and supply chain logistics. By implementing this model in real-world scenarios for out-of-stock prediction, businesses can improve their inventory management efficiency, reduce instances of stockouts, and enhance overall customer satisfaction. Using advanced methods and strategies, the model allows for proactive decision-making, the optimization of inventory levels, and the simplification of supply chain operations. Ultimately, the BCHO-TCN LightGBM model has practical benefits that include improving business performance, reducing revenue losses caused by stock shortages, and promoting operational efficiency in different industry sectors. The results show that when using product listingsfrom the Amazon India dataset, the models achieve high levels of accuracy, as well as excellent NPV, PDV, and threat scores, with values of 94.52%, 95.16%, 94.81%, and 95.76%, respectively. Likewise, at k-fold of 10, the model attains values of 94.81%, 95.60%, 96.28%, and 95.86% for the same metrics. While these achievements are very promising, we should also consider the potential limitations of the proposed method, such as data quality, scalability, and model interpretability. Moreover, in the future, improving the interpretability of the model and developing post-processing techniques for actionable insights are important aspects to consider.

Author Contributions: Conceptualization, A.E.; methodology, A.E.; software, A.E.; writing—original draft preparation, A.E.; review, editing and supervision, A.A. and K.I. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: This study was conducted in accordance with the ethical standards of the University of Mediterranean Karpasia Institutional Review Board (IRB), confirming adherence to ethical guidelines and protocols for research involving human subjects.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data used in this study are available at: Product Listing From Amazon India dataset: https://www.kaggle.com/datasets/promptcloud/product-listing-from-amazon-india (accessed on 20 May 2024).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Allegra, D.; Litrico, M.; Spatafora, M.A.N.; Stanco, F.; Farinella, G.M. Exploiting egocentric vision on shopping cart for out-of-stock detection in retail environments. In Proceedings of the IEEE/CVF International Conference on Computer Vision 2021, Montreal, QC, Canada, 10–17 October 2021; pp. 1735–1740.
2. Giaconia, C.; Chamas, A. Innovative Out-of-Stock Prediction System Based on Data History Knowledge Deep Learning Processing. *Computation* **2023**, *11*, 62. [CrossRef]
3. Naik, N.; Mohan, B.R. Novel stock crisis prediction technique—a study on indian stock market. *IEEE Access* **2021**, *9*, 86230–86242. [CrossRef]
4. Nabipour, M.; Nayyeri, P.; Jabani, H.; Shahab, S.; Mosavi, A. Predicting stock market trends using machine learning and deep learning algorithms via continuous and binary data; a comparative analysis. *Ieee Access* **2020**, *8*, 150199–150212. [CrossRef]
5. Šikić, F.; Kalafatić, Z.; Subašić, M.; Lončarić, S. Enhanced Out-of-Stock Detection in Retail Shelf Images Based on Deep Learning. *Sensors* **2024**, *24*, 693. [CrossRef] [PubMed]
6. Alotaibi, S.S. Ensemble technique with optimal feature selection for Saudi stock market prediction: A novel hybrid red deer-grey algorithm. *IEEE Access* **2021**, *9*, 64929–64944. [CrossRef]
7. Higa, K.; Iwamoto, K. Robust shelf monitoring using supervised learning for improving on-shelf availability in retail stores. *Sensors* **2019**, *19*, 2722. [CrossRef] [PubMed]
8. Aalen, O.O. A linear regression model for the analysis of life times. *Statist. Med.* **1989**, *8*, 907–925. [CrossRef] [PubMed]
9. Aditya, M.A.; Helen, A.; Suryana, I. Naïve Bayes and maximum entropy comparison for translated novel's genre classification. *J. Phys. Conf. Ser.* **2021**, *1722*, 012007. [CrossRef]
10. Alexander, S.S. Price movements in speculative markets: Trends or random walks. *Ind. Manage. Rev.* **1961**, *2*, 7.

11. Andersen, T.G.; Bollerslev, T.; Diebold, F.X.; Labys, P. Modeling and forecasting realized volatility. *Econometrica* **2003**, *71*, 579–625. [CrossRef]
12. Bag, V.; Kulkarni, U.V. Stock price trend prediction and recommendation using cognitive process. *Int. J. Rough Sets Data Anal.* **2017**, *4*, 36–48. [CrossRef]
13. Blair, B.J.; Poon, S.H.; Taylor, S.J. Forecasting S&P 100 volatility: The incremental information content of implied volatilities and highfrequency index returns. In *Handbook of Quantitative Finance and RiskManagement*; Lee, C.F., Lee, A.C., Lee, J., Eds.; Springer: Boston, MA, USA, 2010. [CrossRef]
14. Bollen, J.; Mao, H.; Zeng, X. Twitter mood predicts the stock market. *J. Comput. Sci.* **2011**, *2*, 1–8. [CrossRef]
15. Trichilli, Y.; Abbes, M.B.; Masmoudi, A. Predicting the effect of Googling investor sentiment on Islamic stock market returns: A five-state hidden Markov model. *Int. J. Islam. Middle East. Financ. Manag.* **2020**, *13*, 165–193. [CrossRef]
16. Oueslati, A.; Hammami, Y. Forecasting stock returns in Saudi Arabia and Malaysia. *Rev. Account. Financ.* **2018**, *17*, 259–279. [CrossRef]
17. Suszyński, M.; Peta, K.; Černohlávek, V.; Svoboda, M. Mechanical Assembly Sequence Determination Using Artificial Neural Networks Based on Selected DFA Rating Factors. *Symmetry* **2022**, *14*, 1013. [CrossRef]
18. Hung, N.T. Stock market volatility and exchange rate movements in the Gulf Arab countries: A Markov-state switching model. *J. Islam. Account. Bus. Res.* **2020**, *11*, 1969–1987. [CrossRef]
19. Polamuri, S.R.; Srinivas, K.; Mohan, A.K. Multi model-based hybrid prediction algorithm (MM-HPA) for stock market prices prediction framework (SMPPF). *Arab. J. Sci. Eng.* **2020**, *45*, 10493–10509. [CrossRef]
20. Zhang, X.; Li, Y.; Wang, S.; Fang, B.; Yu, P.S. Enhancing stockmarket prediction with extended coupled hidden Markov model over multisourced data. *Knowl. Inf. Syst.* **2019**, *61*, 1071–1090. [CrossRef]
21. Peta, K.; Żurek, J. Prediction of air leakage in heat exchangers for automotive applications using artificial neural networks. In Proceedings of the 2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 8–10 November 2018; pp. 721–725.
22. Product Listing From Amazon India dataset. Available online: https://www.kaggle.com/datasets/promptcloud/product-listing-from-amazon-india (accessed on 18 June 2024).
23. Abdollahzadeh, B.; Gharehchopogh, F.S.; Mirjalili, S. African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems. *Comput. Ind. Eng.* **2021**, *158*, 107408. [CrossRef]
24. Wang, L.; Cao, Q.; Zhang, Z.; Mirjalili, S.; Zhao, W. Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems. *Eng. Appl. Artif. Intell.* **2022**, *114*, 105082. [CrossRef]
25. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. [CrossRef]
26. Namdari, A.; Durrani, T.S. A Multilayer Feedforward Perceptron Model in Neural Networks for Predicting Stock Market Short-term Trends. *SN Oper. Res. Forum* **2021**, *2*, 38. [CrossRef]
27. Wu, J.M.T.; Li, Z.; Herencsar, N.; Vo, B.; Lin, J.C.W. A graph-based CNN-LSTM stock price prediction algorithm with leading indicators. *Multimed. Syst.* **2023**, *29*, 1751–1770. [CrossRef]
28. Fudholi, D.H.; Nayoan, R.A.N.; Rani, S. Stock prediction based on Twitter sentiment extraction using BiLSTM-attention. *Indones. J. Electr. Eng. Inform.* **2022**, *10*, 187–198. [CrossRef]