Integrating predictive and hybrid machine learning approaches for optimizing solar still performance: a comprehensive review.

ELSHEIKH, A., FAQEHA, H., HAMMOODI, K.A., BAWAHAB, M., FUJII, M., SHANMUGAN, S., ESSA, F.A., ABD-ELAZIEM, W., RAMESH, B., SATHYAMURTHY, R. and EGIZA, M.

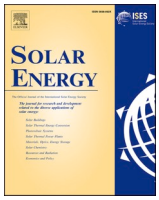
2025

Solar Energy 295 (2025) 113536

Contents lists available at ScienceDirect

Solar Energy

journal homepage: www.elsevier.com/locate/solener



Integrating predictive and hybrid Machine Learning approaches for optimizing solar still performance: A comprehensive review

Ammar Elsheikh [a,b,*], Hosam Faqeha [c], Karrar A. Hammoodi [d], Mohammed Bawahab [e], Manabu Fujii [f], S. Shanmugan [g], Fadl A. Essa [h], Walaa Abd-Elaziem [m,n], B. Ramesh [j], Ravishankar Sathyamurthy [k,l], Mohamed Egiza [h,i,*]

[a] Department of Production Engineering and Mechanical Design, Faculty of Engineering, Tanta University, Tanta 31527, Egypt
[b] Faculty of Engineering, Pharos University in Alexandria, Alexandria 21648, Egypt
[c] Mechanical Engineering Department, Umm Al-Qura University, Makkah 21955, Saudi Arabia
[d] College of Engineering, University of Al Maarif, Al Anbar, 31001, Iraq
[e] Department of Mechanical and Materials Engineering, University of Jeddah, P.O. Box 80327, Jeddah 21589, Saudi Arabia
[f] Institute of Science Tokyo, Meguro-ku, Tokyo 152-8552, Japan
[g] Research Centre for Solar Energy, Department of Integrated Research and Discovery-Physics, Koneru Lakshmaiah Education Foundation, Green Fields, Vaddeswaram, Guntur 522502, Andhra Pradesh, India
[h] Mechanical Engineering Department, Faculty of Engineering, Kafrelsheikh University, Kafrelsheikh 33516, Egypt
[i] School of Computing, Engineering and Technology, Robert Gordon University, Garthdee Road, Aberdeen AB107GJ, UK
[j] Department of Mechanical Engineering, School of Engineering and Technology, Dhanalakshmi Srinivasan University, Samayapuram, Tiruchirappalli - 621112, Tamil Nadu, India
[k] IRC Sustainable Energy Systems (IRC-SES), King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia
[l] Department of Mechanical Engineering, King Fahd University of Petroleum & Minerals, Dhahran 31621, Saudi Arabia
[m] Department of Mechanical Design and Production Engineering, Faculty of Engineering, Zagazig University, P.O. Box 44519, Egypt
[n] Department of Materials Science and Engineering, Northwestern University, Evanston, IL 60208, USA

ARTICLE INFO

Keywords:
Water desalination
Solar stills
Prediction
Machine learning
Metaheuristic optimization

ABSTRACT

The increasing global need for freshwater, coupled with the imperative for sustainable and energy-efficient solutions, has fueled interest in solar distillation technologies. Solar stills (SSs) offer a simple, low-cost, and environmentally friendly approach to desalination. However, their performance can be significantly influenced by various factors, including climatic conditions, design parameters, and operational variables. To address these challenges and predict SS performance, machine learning (ML) techniques have emerged as a powerful tool. This review explores the application of various ML models, including Support Vector Machines (SVM), Multi-Layer Perceptrons (MLP), Adaptive Neuro-Fuzzy Inference Systems (ANFIS), Decision Trees (DT), and hybrid ML/metaheuristic optimizer models, such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Simulated Annealing (SA), in predicting water production rates, managing energy consumption, and providing decision support for operators. The review highlights the potential of these models to enhance the efficiency and sustainability of solar desalination systems. By leveraging data-driven insights and predictive modeling, ML-based approaches enable the prediction of performance metrics, identification of optimal operating conditions, and real-time monitoring and control. Furthermore, hybrid ML/metaheuristic models, which combine algorithms like SVM, MLP, and ANFIS with optimization techniques, offer enhanced reliability and resilience in complex scenarios. This review emphasizes the significant potential of ML in advancing solar distillation technologies, showing that integrating ML techniques into SS systems can lead to more efficient, sustainable, and cost-effective solutions to address global water scarcity challenges.

* Corresponding authors at: Department of Production Engineering and Mechanical Design, Tanta University, Tanta 31527, Egypt (A. Elsheikh); Mechanical Engineering Department, Faculty of Engineering, Kafrelsheikh University, Kafrelsheikh 33516, Egypt (M. Egiza).
E-mail addresses: ammar_elsheikh@f-eng.tanta.edu.eg (A. Elsheikh), Mohamed_egiza@eng.kfs.edu.eg (M. Egiza).

https://doi.org/10.1016/j.solener.2025.113536
Received 27 October 2024; Received in revised form 25 March 2025; Accepted 17 April 2025
Available online 21 April 2025

1. Introduction

For years, water desalination has been a prominent research focus to address domestic, industrial, and agricultural water needs. Global water scarcity affects a substantial portion of the population, particularly during drought seasons [1]. The rise in global water demand is influenced not only by population growth but also by industrial expansion. In regions like the Arab world, freshwater scarcity is particularly acute due to limited natural water sources [2]. Over the past three decades, the Arab population has tripled, reaching 150 million, with projections indicating it could surpass 500 million within the next five years. This rapid population growth, coupled with limited water resources and challenges such as the Grand Ethiopian Renaissance Dam issue, has positioned Arab countries among the most severely affected by water scarcity globally [3]. Consequently, Arab governments have devised long-term plans and implemented comprehensive policies to address this pressing issue.

Seawater desalination techniques, such as membrane distillation [4], reverse osmosis [5], humidification-dehumidification [6], and multi-stage flash distillation [7], have been widely used to produce drinkable water but they are known for their huge energy consumption. Solar Stills (SSs) have been also extensively employed to distillate seawater in arid and coastal regions due to their simplicity, ease of operation, and eco-friendly nature [8]. However, its main drawback is its low water yield and efficiency [9]. Many research attempts have been made to enhance the performance of SSs such as applications of nanofluids and/or energy storage materials [10], as well as incorporating additional thermal devices like condensers [11], heat exchangers [12], reflectors [13], solar collectors [14] and photovoltaic panels [15]. Many studies have focused on assessing the performance of SSs over short periods, typically spanning only a few days. However, there is a crucial need to investigate their performance over longer durations under various operational and meteorological conditions. Conducting long-term tests on SSs can be both time-consuming and costly. Modeling SSs presents an alternative approach, allowing for the forecasting and prediction of their performance over extended periods with minimal experimental effort. Traditional mathematical modeling techniques, such as numerical or analytical approaches, often involve complex and cumbersome processes, requiring numerous assumptions to simplify the modeling task. Machine Learning (ML) methods offer a promising solution for modeling SSs under diverse conditions with reasonable accuracy, potentially overcoming the limitations of conventional modeling techniques [16].

ML models leverage experimental data from SS operations, encompassing environmental factors (such as solar radiation, temperature, humidity, and wind speed) and system parameters (like inclination angle, condensation surface material, and airflow rates), to construct predictive models of SS performance. These models capture the complex relationships between input variables and water production rates, enabling predictions of SS output variables based on current or future environmental conditions and system configurations. Operators can use these predictive models to anticipate water production rates under various scenarios and optimize operational strategies to enhance efficiency and output. Thus, ML models act as decision support tools for SS operators and engineers, offering actionable insights for informed decision-making and identifying opportunities for process optimization. This review explores the utilization of ML approaches in SS modeling, covering different ML models, statistical performance measures for prediction accuracy assessment, applications of ML in SS performance modeling, and the use of advanced metaheuristic optimizers to enhance ML model accuracy. It concludes with key findings and future prospects in the field.

2. Types of solar distillers

Solar Stills (SSs) are very simple thermal devices composed of a trough with a glass cover in which a basin containing seawater is placed as shown in Fig. 1. The incoming solar radiation penetrates the solar still (SS) by passing through the glass cover. The basin absorbs most of the solar radiation and reflects thermal radiation with a long wavelength that cannot pass through the glass cover. The absorbed and confined energies are converted into thermal energy which helps in heating the seawater in the basin. Once the seawater is heated up, it begins to evaporate. The vapor released from the surface of seawater condenses on the interior surface of a glass cover, transforming into distilled water. This condensed water is gathered from the distiller via a collection trough. Various designs of SSs have been documented in the literature, and they will be briefly outlined in this section.

The main parameters influencing the performance of SSs can be grouped into three main categories [17,18]: design parameters, meteorological parameters, and operating parameters as shown in Fig. 2. Setting the optimal design parameters of SSs, such as the space between the glass cover and basin, glass cover thickness, glass cover inclination, type and material of absorber plate, insulation thickness, and SS type, is a critical issue that has been extensively studied in the literature [19]. There are main types of SSs, namely passive and active distillers [20]. Both types may be single slope [21], single basin [22], triangular [23], inclined [24], stepped [25], wick-based [26], tubular [27], pyramid [28 29], pyramid with inverted pyramid basin [30], cascade [31], concave [32], or hemispherical [33] distillers. However, the main difference between them is that the active distiller must integrate with other thermal devices such as solar collectors [34], PV panels [35], or solar ponds [36]; while passive distillers are considered standalone distillers [37]. Operating parameters like water depth, glass cover cooling, water salinity, and color of water, may be adjusted during the operation of SSs. However, meteorological conditions such as solar intensity, wind speed, air temperature, cloud cover, and humidity, could not be controlled during the operation of SSs. These conditions vary not only during the same day but also during the year.

It is a critical issue to predict the SS performance in terms of water yield, thermal efficiency, and exergy efficiency, for different operating and meteorological parameters. This may help to assess the application of a certain distiller design to distillate water in a certain region. This review paper aims to shed light on the utilization of ML methods in the modeling of SSs.

3. Statistical performance measures

Various statistical measures are employed in the literature to evaluate the prediction accuracy of ML models. Statistical measures that are frequently used include mean error (ME), mean relative error (MRE), mean absolute error (MAE), mean absolute percentage error (MAPE), root mean square error (RMSE), normalized root-mean-square error (NRMSE), coefficient of residual mass (CRM), index of agreement (IA), and coefficient of determination (R^2) [38–41].

ME quantifies the average difference between predicted and actual target values. It can be positive or negative, indicating overestimation or underestimation, respectively. It is computed as [42]:

$$ME = \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i) \tag{1}$$

where y_i is the observed value, x_i is the predicted value, n is the number of observations.

MAE measures the average of the absolute errors between predicted and target response values. It gives an idea of the average magnitude of the errors in the predictions. It is computed as [43]:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - x_i| \tag{2}$$

MRE quantifies the average of the relative errors by comparing the predicted and actual target response values. It is calculated by dividing

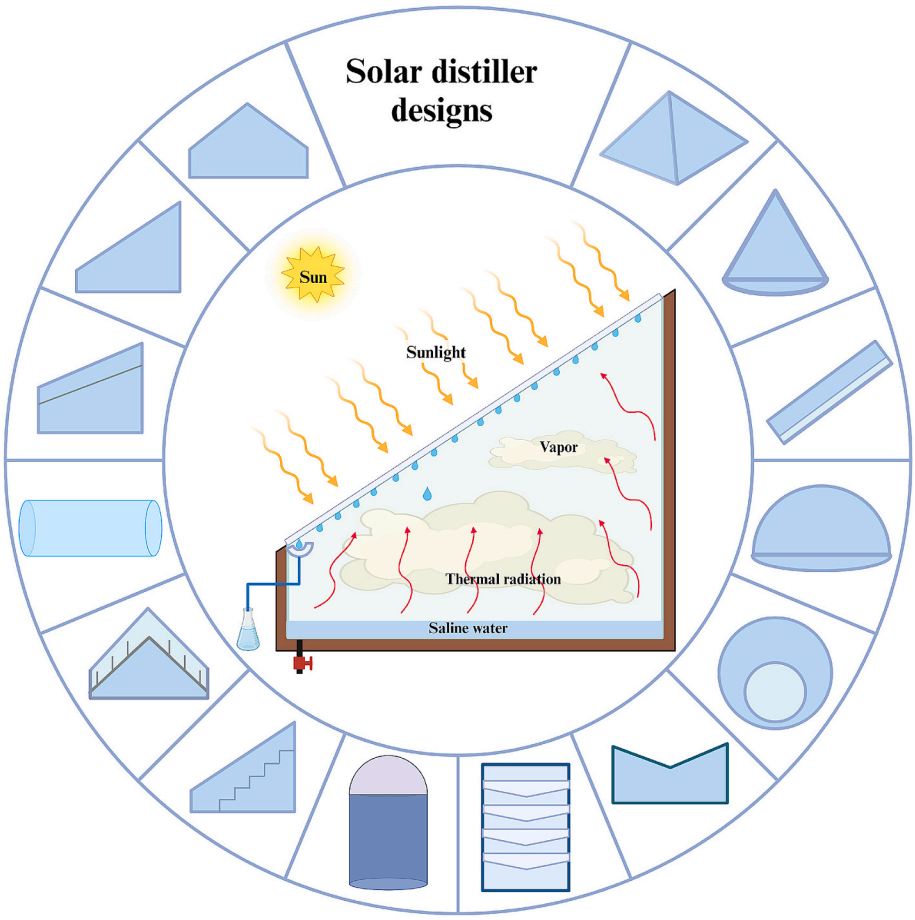


Fig. 1. A schematic of a typical Solar Stills (SS) and different designs.

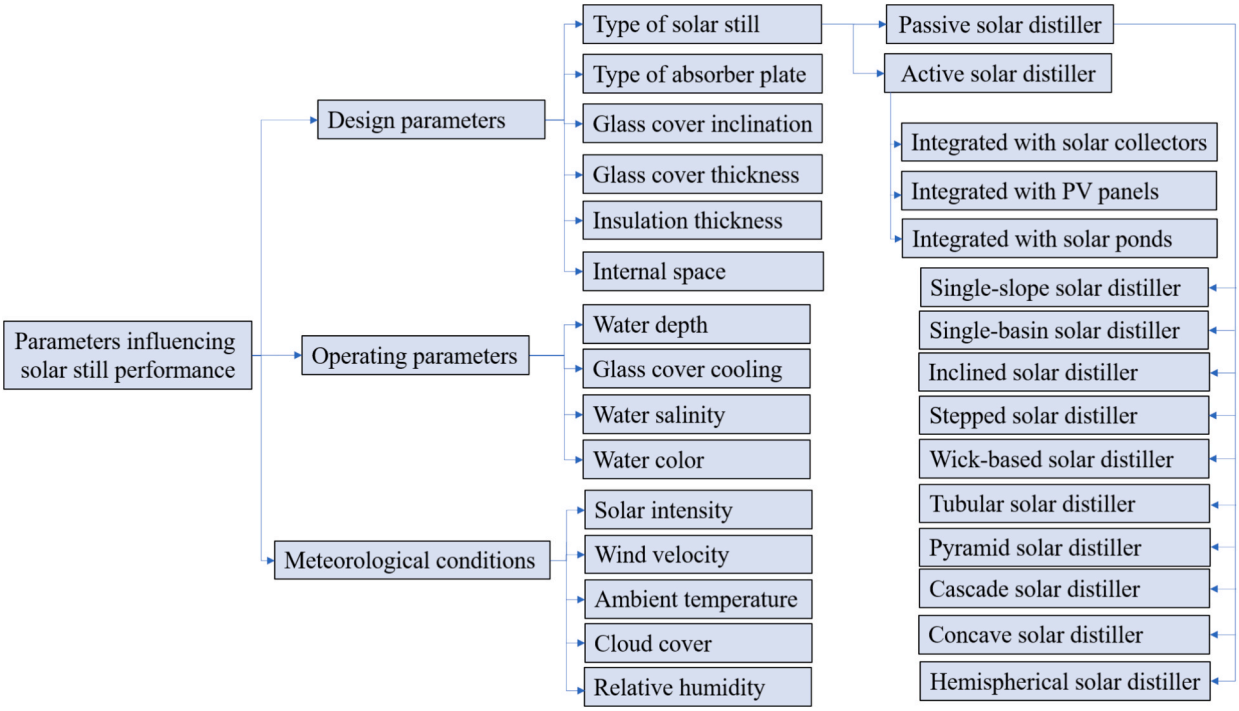


Fig. 2. Parameters influencing SS performance.

the absolute error by the actual value and then averaging these ratios. It is computed as [16]:

$$MRE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - x_i}{y_i}\right| \tag{3}$$

MAPE represents the average of the absolute percentage errors between predicted and actual values. It provides a percentage representation of the average magnitude of the errors relative to the actual values. It is computed as [44]:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - x_i}{y_i}\right| \times 100 \tag{4}$$

RMSE quantifies the disparity between actual and predicted data points. A lower RMSE value signifies a higher accuracy of the model. *RMSE* can be calculated using the following formula [38]:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2} \tag{5}$$

The function of *NRMSE* is to provide a normalized measure of the RMSE, facilitating comparisons of model accuracy across diverse datasets or scales. It's determined by dividing the RMSE by the range of observed values. This normalization aids in interpreting the error relative to data magnitude, enabling more meaningful model comparisons. It is defined as [38]:

$$NRMSE = \frac{\sqrt{n\sum_{i=1}^{n}(y_i - x_i)^2}}{\sum_{i=1}^{n} y_i} \tag{6}$$

CRM is a statistical metric employed to assess the goodness of fit between observed and predicted values in a model. It assesses the magnitude of residual errors, which represent the disparities between target values and predicted outcomes. A lower CRM value indicates better agreement between observed and target response values, suggesting a more accurate model. Conversely, a higher CRM value suggests larger discrepancies between observed and target response values, indicating poorer model performance. The formula for calculating the CRM is [45]:

$$CRM = \frac{\sum_{i=1}^{n}(y_i - x_i)}{\sum_{i=1}^{n}(y_i)} \tag{7}$$

IA is a statistical measure employed to evaluate the agreement between observed and predicted values in a model. It evaluates the predictive performance of a model by comparing the deviation of predicted values from the observed values relative to the deviation of the observed values from their mean. The IA ranges from 0 to 1, with a value closer to 1 indicating better agreement between observed and predicted values. A value of 1 indicates perfect agreement, while lower values indicate poorer agreement. The formula for calculating the IA is [46]:

$$IA = 1 - \frac{\sum_{i=1}^{n}(y_i - x_i)^2}{\sum_{i=1}^{n}\left(|x_i - \bar{y}| + |y_i - \bar{y}|\right)^2} \tag{8}$$

where $\bar{y}$ is the mean of the observed values.

R^2 is used to assess the fit goodness of a regression process. R^2 value ranges from 0 to 1, where:

$R^2 = 0$ suggests that the regression process does not explain any of the variability in the target response around its computed mean.

$R^2 = 1$ suggests that the regression process accounts for all the variance in the target response relative to its computed mean.

$0 < R^2 < 1$ indicates the proportion of the variability in the dependent variable that is accounted for by the independent variables included in the regression process.

Higher R^2 values signify that the model aligns more closely with the data. It is calculated by [47];

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - x_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{9}$$

4. Machine learning algorithms

In recent years, there has been a notable increase in the utilization of machine learning (ML) methods in modeling different engineering systems such as heat exchangers [48], thermoacoustic devices [49], solar collectors [50], pumping systems [51], and wastewater treatment plants [52]. The most commonly used ML methods are; linear regression (LR), random forest (RF), support vector machine (SVM), fuzzy logic (FL), radial basis function neural network (RBFNN), decision tree (DT), artificial neural network (ANN), multilayer perceptrons (MLP), Gaussian process regression (GPR), adaptive neuro-fuzzy inference system (ANFIS), long short-term memory (LSTM), and random vector functional link (RVFL). These ML techniques have been successfully applied to model and predict the performance of different types of SSs.

4.1. Linear regression

LR is a classical ML algorithm with supervised characteristics employed to forecast a continuous target variable using one or more input features [53]. It represents the correlation between the input features and the target variable through a linear equation. The procedures of linear regression are explained as follows:

i Model Representation: In linear regression, we assume that the connection between the input features (denoted as x) and the predicted outcome (denoted as y) can be represented by a linear equation of the form

$$y = \delta_0 + \delta_1 x_1 + \delta_2 x_2 + \cdots + \delta_m x_m + \varepsilon \tag{10}$$

where: y is the target variable we want to predict. $x_1, x_2, \cdots, x_m$ are the input features.$\delta_0, \delta_1, \cdots, \delta_m$ are the coefficients or the linear model weights.ε is the error term representing the variance between the observed and forecasted outcomes.

ii. Training the Model: The objective of LR is to determine the optimal parameters for the coefficients $\delta_0, \delta_1, \cdots, \delta_m$ that minimize the error between the actual target values and the predicted values. This is typically done by minimizing a cost function, such as the sum of squared errors (SSE) or the mean squared error (MSE).

iii Gradient descent or analytical Solution: There are two main approaches to finding the optimal coefficients

Gradient Descent: This iterative optimization algorithm updates the coefficients in the direction of the steepest descent of the cost function. It repeats this process until convergence to find the optimal coefficients that minimize the error.

Analytical Solution (Normal Equation): This closed-form solution directly calculates the optimal coefficients by solving a system of linear equations. While it might incur significant computational costs when dealing with extensive datasets, it provides an exact solution without the need for iterative optimization.

iv Model Evaluation: Once the model is trained, it can be evaluated using metrics such as R^2, MSE, or RMSE. These metrics measure the accuracy of the fit between the actual data and predicted outcomes, providing insights into the model's performance

v Prediction: After training and evaluation, the linear regression model can be used to make predictions on new unseen data by plugging in the values of the input features into the learned linear equation. The model generates the predicted outcomes of the target variable using the learned coefficients

LR is widely applied across diverse domains, like engineering, finance, economics, and social sciences, for tasks such as sales forecasting, risk assessment, and trend analysis. Its popularity stems from its interpretability, simplicity, and effectiveness in modeling linear relationships between variables. However, LR has limitations. It assumes a linear relationship between input features and the target variable, which can lead to biased predictions if the actual relationship is non-linear. LR is sensitive to outliers, which can skew model parameters and degrade performance. It also assumes homoscedasticity, where the variance of residuals remains constant across all levels of independent variables; violations can lead to inefficient estimates. Additionally, LR assumes errors are independent, but correlations among errors can bias estimates. LR's inflexibility in capturing complex relationships compared to polynomial or spline regression and ML algorithms like decision trees or neural networks is another drawback. LR can suffer from overfitting or underfitting, failing to generalize well to unseen data if the model is too simple or too complex. Despite these limitations, LR remains valuable due to its simplicity and interpretability, serving as a baseline for comparing more complex ML algorithms.

4.2. Multivariate Adaptive regression Splines (MARS)

MARS is a flexible regression algorithm that can figure out non-linear relationships between the target variables and the input features by fitting piecewise linear functions [54]. MARS builds a model by recursively partitioning the input space into segments and fitting simple linear functions within each segment. This algorithm is implemented in four main steps as follows:

Basis Functions:

MARS starts with a set of basis functions, typically including constants and hinge functions (also known as "tent" functions), which are defined as:

$$h(x,\beta,\gamma) = \max(0, \beta \bullet x - \gamma) for \beta > 0 \tag{11}$$

These basis functions allow MARS to create piecewise linear segments by setting breakpoints (split points) based on the input feature values.

Model Construction:

MARS iteratively builds the model by adding basis functions and creating new segments to capture non-linear relationships. At each iteration, MARS considers all possible combinations of existing basis functions and potential split points to determine the best model improvement. The algorithm selects the most significant basis function and split point combination based on a chosen criterion, such as minimizing the MSE or maximizing the R^2.

Model Representation:

The final MARS model is represented as a sum of basis functions, where each basis function is multiplied by a coefficient:

$$f(x) = \sum_{j=1}^{J} \beta_j \bullet h_j(x) \tag{12}$$

where, $f(x)$ is the predicted outcome, β_j are the coefficients, and $h_j(x)$ are the basis functions.

Model Interpretation:

MARS provides interpretable models by representing complex relationships between input features and the target variable as a series of simple linear segments. The breakpoints (split points) in each segment indicate the values of the input features where the relationships change direction or slope. Coefficients associated with each basis function offer insights into the direction and strength of the relationships between input features and the target outcomes.

MARS is an effective regression algorithm capable of capturing intricate non-linear relationships while preserving interpretability. However, it may suffer from overfitting if the number of basic functions or segments is not appropriately controlled. Regularization techniques, such as limiting the maximum number of basic functions or applying penalty terms to the coefficients, can aid in boosting generalization performance and averting overfitting.

4.3. Evolutionary polynomial regression

EPR is a regression ML algorithm that combines the concepts of genetic algorithms with polynomial regression to evolve mathematical expressions that best fit the given dataset [55]. EPR aims to discover the polynomial equation that represents the relationship between the target variable and the input features by evolving a population of candidate models over multiple generations. This algorithm is implemented in eight main steps as follows:

Initialization:

EPR begins by initializing a population of candidate polynomial models. Each candidate model represents a potential polynomial equation capable of forecasting the target variable using the input features.

Evaluation:

Each candidate model in the population is evaluated based on its fitness, which is typically determined by how well it fits the training data. Common fitness metrics include the RMSE, MAE, or R^2.

Selection:

A selection process is used to choose the most promising candidate models from the population to proceed to the next generation. Various selection strategies can be employed, such as roulette wheel or tournament methods, where models with higher fitness scores have an increased likelihood of being selected.

Genetic Operators:

Genetic operators, including mutation, crossover, and reproduction, are applied to the selected candidate models to create offspring for the next generation. These genetic operators introduce variation and diversity into the population, allowing new models to be explored.

Evolution:

The population undergoes multiple generations of evolution, where candidate models are repeatedly evaluated, selected, and subjected to genetic operators. Through this iterative process, EPR aims to improve the fitness of the candidate models and discover polynomial equations that better fit the training data.

Termination:

The evolution process proceeds until a termination condition is met, such as reaching a maximum number of generations, achieving the desired fitness level, or experiencing stagnation in improvement over consecutive generations.

Best Model Selection:

Once the evolution process is complete, the best-performing candidate model, typically based on its fitness on a validation set, is selected as the final model for prediction.

Model Representation:

The final model discovered by EPR is represented as a polynomial equation, where the coefficients of the polynomial terms are determined through the evolution process:

$$y = \delta_0 + \delta_1 x_1 + \delta_2 x_2 + \delta_3 x_1^2 + \delta_4 x_1 x_2 + \delta_5 x_2^2 + \cdots \quad (13)$$

where: y is the predicted target variable we want to predict. $x_1, x_2, \cdots, x_m$ are the input features.$\delta_0, \delta_1, \cdots, \delta_m$ are the coefficients determined by EPR.

EPR is a flexible and powerful regression algorithm that can discover complex polynomial equations to represent the relationships between input features and the target variable. It is particularly well-suited for problems where conventional regression techniques may face challenges in capturing non-linear relationships or where the underlying functional form of the data is unknown. By combining genetic algorithms with polynomial regression, EPR can explore a wide range of models, adapting to the complexity of the data and revealing hidden patterns. However, despite its strengths, EPR has some limitations. One of the key drawbacks is its computational complexity, especially when dealing with large datasets or high-dimensional input spaces. The evolutionary process requires evaluating multiple candidate models across several generations, which can be time-consuming. Additionally, EPR is prone to overfitting, particularly when higher-degree polynomials are used or when the training data is noisy. Another limitation is its sensitivity to outliers, which can skew the model's performance. Furthermore, EPR may not always provide an easily interpretable model, as the discovered polynomial equations can become quite complex, making them difficult to understand and explain.

4.4. K-Nearest neighbors (KNN)

KNN is a straightforward ML algorithm utilized for regression problems [56]. In KNN regression, the predicted value for a new data point is computed according to the average of the target outcomes of its k adjacent neighbors in the feature space. This model operates during training and prediction stages as follows:

During the training phase:

- Given a training dataset with features $X = \{x_1, x_2, x_3, \cdots, x_n\}$ and corresponding target values $Y = \{y_1, y_2, y_3, \cdots, y_n\}$
- The algorithm stores the training data to be used for predicting new data points.

During the prediction phase:

- For a new data point x_{new} for which we want to predict the target value y_{new}.
- Calculate the distance between x_{new} and all other points in the training set using a distance metric such as Euclidean distance (ED):

$$ED(x_{new}, x_i) = \sqrt{\sum_{j=1}^{d} \left(x_{new,j} - x_{i,j}\right)^2} \quad (14)$$

- Select the k data points with the smallest distances to x_{new}.
- Retrieve the target values y_i corresponding to these k nearest neighbors.
- Predict the target value y_{new} for x_{new} as the average of the target values of the k nearest neighbors:

$$y_{new} = \frac{1}{k} \sum_{i=1}^{k} y_i \quad (15)$$

- Alternatively, weighted averaging can be used, where closer neighbors contribute more to the prediction:

$$y_{new} = \frac{\sum_{i=1}^{k} w_i \bullet y_i}{\sum_{i=1}^{k} w_i} \quad (16)$$

Here, w_i is a weight assigned to each neighbor based on its distance to x_{new}. Common choices for weights include inverse distance or Gaussian kernel weights.

KNN regression is a straightforward algorithm with few parameters to tune, such as the number of neighbors k and the choice of distance metric. Yet, it can be computationally demanding, particularly with extensive datasets, as it necessitates computing distances between the new data point and other training points. Additionally, KNN regression may not perform well in high-dimensional feature spaces or with noisy data. Proper preprocessing, feature scaling, and careful selection of k are essential for optimal performance.

4.5. Decision tree (DT)

DT algorithm is a widely-used ML technique employed for both regression and classification problems [57]. It functions by iteratively splitting the input space into smaller sections according to the feature values of the data. Here's how the decision tree algorithm works:

i. Tree Structure: A DT has a hierarchical structure composed of nodes, where each node signifies a decision based on the value of a particular feature. The top node of the tree is called the root node, and it represents the whole dataset. The intermediary nodes embody choices determined by feature values, while the terminal nodes indicate the ultimate output.
ii. Splitting Criteria: At each node of the tree, the DT selects a feature and a corresponding threshold value to divide the data into two or more subsets. The goal is to maximize the homogeneity (or purity) of the subsets in terms of the target variable. For classification tasks, common purity measures include entropy and Gini impurity, while for regression tasks, mean squared error or variance reduction is often used.
iii. Recursive Partitioning: The dataset undergoes recursive partitioning into smaller subsets, determined by specific splitting criteria at every node. This procedure continues until a termination condition is fulfilled, such as achieving a minimum number of samples per leaf node, reaching a maximum tree depth, or no further improvement in purity can be achieved.
iv. Prediction: After the tree is built, it can be utilized to forecast outcomes on fresh, unseen data. For classification tasks, the algorithm traverses the tree from the root node to a leaf node based on the feature values of the input data and assigns the majority class label of the corresponding leaf node. For regression tasks, the algorithm follows a similar process but assigns the average or median value of the target variable in the leaf node.
v. Handling Categorical Features: Decision trees can handle both numerical and categorical features. For categorical features, the algorithm can either perform binary splits (e.g., is the feature value equal to a specific category?) or use techniques like one-hot encoding to represent categorical variables as binary vectors.
vi. Pruning: To prevent overfitting, decision trees can be pruned after construction by removing nodes that do not significantly improve predictive performance on a validation dataset. Pruning helps simplify the tree structure and improve generalization to unseen data.

DTs are renowned for their intuitive nature, interpretability, and

capacity to capture intricate data relationships. However, they face challenges like overfitting, especially with noisy or high-dimensional datasets. To counter this, ensemble techniques like Gradient Boosting and Random Forest are recommended. DTs exhibit high variance, prone to overfitting by capturing irrelevant patterns or noise easily, resulting in poor generalization. They're sensitive to variations in training data, leading to unstable models. DTs struggle with complex relationships compared to neural networks or ensemble methods and can't handle non-linearly separable data well. They're prone to overfitting with noisy or outlier-rich datasets, creating complex structures that hinder generalization. Despite limitations, DTs remain popular for their simplicity and ability to handle various data types, with ensemble methods offering solutions to their shortcomings.

4.6. Random Forest

RF stands as a versatile and robust ensemble learning method employed for both classification and regression assignment tasks [58]. Its operation involves building numerous decision trees during the training phase, and upon completion, it delivers the average prediction (for regression) or the mode (for classification) derived from the individual trees. The procedures of random forest are explained as follows:

i. Bootstrapped Sampling: RF begins by randomly selecting subsets of the training data (with replacement), referred to as bootstrapped samples. These samples are used to train each decision tree in the ensemble.
ii. Random Feature Selection: In every decision tree within the ensemble, a random subset of features is chosen at each node to identify the optimal split. This process helps introduce diversity among the trees and prevents overfitting by reducing the correlation between them.
iii. Decision Tree Construction: Every decision tree within the Random Forest (RF) is built using the bootstrapped sample and a random selection of features. These trees are usually expanded until they reach their maximum depth or until they attain a minimum number of samples per leaf node.
iv. Voting (Classification) or Averaging (Regression): After the construction of all decision trees, predictions are generated by combining the outputs of each individual tree. In classification tasks, the final prediction is determined by selecting the mode (the most frequent class label) among the predictions of all trees. In regression tasks, the final prediction is obtained by calculating the average prediction of all trees.
v. Ensemble Aggregation: The final prediction of the RF is obtained by aggregating the predictions of all decision trees. This ensemble technique enhances the accuracy, resilience, and generalization ability of the model when contrasted with individual decision trees.
vi. Parameter Tuning: Random Forest provides various hyperparameters that can be adjusted to enhance its performance, including the maximum depth of individual trees, the number of trees within the ensemble, and the number of features examined for splitting at each node. Employing cross-validation methods aids in identifying the most effective combination of hyperparameters.
vii. Feature Importance: RF offers a metric for feature importance, showcasing the relative significance of each feature in prediction-making. Feature importance scores are computed based on the decrease in impurity (Gini impurity or entropy) achieved by each feature when used for splitting nodes in the trees.

RF has a robust predictive accuracy, good resistance to overfitting, and good ability to handle extensive datasets across various domains. It is widely used in classification, regression, feature selection, and anomaly detection tasks. However, while RF models are generally resistant to overfitting, they can still overfit noisy or outlier-rich data, leading to poor generalization performance. In such cases, the model may memorize the noise rather than capturing meaningful patterns. Additionally, RF models might experience performance degradation when dealing with highly complex nonlinear relationships between input features and the target variable, where other models, such as MLP or SVM, might be more effective. Although RF models are easier to interpret than more complex models like neural networks, interpreting individual trees within the ensemble can still present challenges due to the large number of trees involved. The training process of RF models can also demand significant computational resources, especially when using large datasets or a high number of trees, as it builds multiple decision trees concurrently. This requirement for substantial memory and processing power can become a constraint in memory-constrained environments or when working with large-scale datasets. Furthermore, RF models require hyperparameter tuning (such as selecting the tree depth and number of trees), which involves extensive experimentation and computational resources to find the optimal settings. Despite these limitations, RF remains a popular choice for regression and classification tasks due to its ability to handle high-dimensional data, nonlinear relationships, and missing values, while offering robustness against overfitting in many practical applications..

4.7. Support vector Machines

SVMs are primarily known as classification algorithms, but they can also be used for regression tasks, where the target is to predict a continuous target variable [59]. When SVM is applied to regression, it's known as Support Vector Regression (SVR). SVR works by finding the hyperplane that best fits the data while minimizing the margin violations, similar to how SVM determines the best hyperplane in the classification to obtain separate classes. The procedures of SVM are explained as follows:

i Model Representation: In SVR, the objective is to identify a hyperplane that optimally fits the training data while simultaneously minimizing the error (or deviation) of the data points from this hyperplane. The hyperplane is represented as

$$f(x) = \delta' x + \beta \tag{17}$$

where: $f(x), x, \delta, and \beta$ are the predicted outcomes, the input feature vector, the weight vector (coefficients), and the bias term.

ii Loss Function: SVR minimizes a loss function that penalizes points for being outside a certain margin (ε) around the hyperplane. The loss function typically takes the form of the ε-insensitive loss function

$$L(y, f(x)) = max(0, |y - f(x)| - \varepsilon) \tag{18}$$

where y is the true target variable and ϵ is a predefined tolerance level. This loss function ignores errors within the margin ϵ and penalizes errors outside the margin.

iii Optimization: The objective of SVR is to minimize the combined loss function and regularization term, aiding in managing the model's complexity to avoid overfitting. This optimization problem can be expressed as follows:

$$min_{w,b} \frac{1}{2}\|\delta\|^2 + C\sum_{i=1}^{m} L(y_i, f(x_i)) \tag{19}$$

where C is the regularization parameter, which controls the balance between maximizing the margin and minimizing the error, is essential in SVM models.

iv. Kernel Trick: Just like SVM for classification, SVR can utilize kernel functions to map the input features into a higher-dimensional space, enabling the creation of more intricate decision boundaries. Typical kernel functions encompass linear, polynomial, radial basis function (RBF), and sigmoid kernels.

v Prediction: After training, SVR can be employed to predict

outcomes on new, unseen data by applying the learned hyperplane to the input feature vectors

SVR excels in capturing complex relationships in data, making it invaluable for high-dimensional or non-linear datasets. However, SVR's performance relies heavily on hyperparameter selection, particularly the regularization parameter C and kernel parameters, necessitating careful tuning for optimal results. SVR can be memory-intensive and computationally demanding, especially with large datasets or complex kernels, requiring substantial computational resources and time for training. It may struggle with imbalanced datasets, favoring classes with more samples and potentially leading to biased predictions and poor performance in minority classes. Nonetheless, SVMs remain popular and effective, especially for binary classification tasks, where finding the optimal decision boundary is critical. With proper parameter tuning and feature engineering, SVMs can achieve high performance across a variety of datasets.

4.8. Multilayer Perceptron

The MLP model involves multiple layers of neurons [60]. In regression scenarios, the MLP architecture typically comprises an output layer, an input layer, and one or several hidden layers as shown in Fig. 3 (a). Forward propagation entails calculating the output of every neuron in the network by performing a weighted sum of its inputs and then applying an activation function to the result.

For a neuron j in the hidden layer l, the output $h_j^{(l)}$ can be calculated as:

$$h_j^{(l)} = \Delta\left(\sum_{i=1}^{m} \omega_{ij}^{(l)} h_i^{(l-1)} + \beta_j^{(l)}\right) \tag{20}$$

where $\omega_{ij}^{(l)}$ is the weight connecting the i^{th} neuron in the $(l-1)^{th}$ layer to the j^{th} neuron in the l^{th} layer; $\beta_j^{(l)}$ is the bias term for the j^{th} neuron in the l^{th} layer; $h_j^{(l)}$ is the output of the j^{th} neuron in the l^{th} hidden layer and Δ is the activation function.

Typical activation functions encompass the sigmoid function $\Delta(x) = {}^{1}/_{1+e^{-x}}$ and hyperbolic tangent function $\Delta(x) = tanh(x)$. x is the input feature vector.

In the output layer, we typically use a linear activation function since we're dealing with regression tasks. So, the predicted output y can be computed as:

$$y = \sum_{i=1}^{n} \omega_i^{(L)} h_i^{(L-1)} + \beta^{(L)} \tag{21}$$

where $\omega_i^{(L)}$ is the weight connecting the i^{th} neuron in the last hidden layer to the output neuron, and $\beta^{(L)}$ is the bias term for the output neuron.

During training, we adjust the weights and biases of the network to minimize the chosen loss function, typically the Mean Squared Error (MSE) between predicted and actual outputs, using backpropagation. This involves computing the gradients of the loss function with respect to the network's parameters and using them to update the parameters iteratively. Once training is complete, the MLP can make predictions for new input data by passing it through the network with the updated parameters. MLPs are prone to overfitting, especially when trained on small datasets or with overly complex architectures relative to available data. Sufficient data is crucial for effective generalization and to mitigate overfitting. The effectiveness of MLPs can be affected by the initialization of model parameters. Selecting appropriate initializations is challenging as poor choices may lead to slow convergence or suboptimal results. MLPs have multiple hyperparameters requiring tuning for optimal performance, including the number of hidden layers, neurons per layer, learning rate, and activation functions. Finding the best combination often requires extensive experimentation and computational resources. Preprocessing of input data, such as normalization, is often necessary to ensure stable and efficient training. Failure to preprocess data appropriately can result in slow convergence or inferior performance. Despite these challenges, MLPs remain widely used for their ability to learn complex patterns across various domains. With careful design, tuning, and training, MLPs can achieve state-of-the-art performance in various ML tasks.

4.9. Adaptive Neuro-Fuzzy Inference systems

ANFIS is a hybrid ML model that merges the adaptive capabilities of neural networks with the interpretability of fuzzy logic systems [61]. ANFIS is particularly useful for regression tasks where the target is to

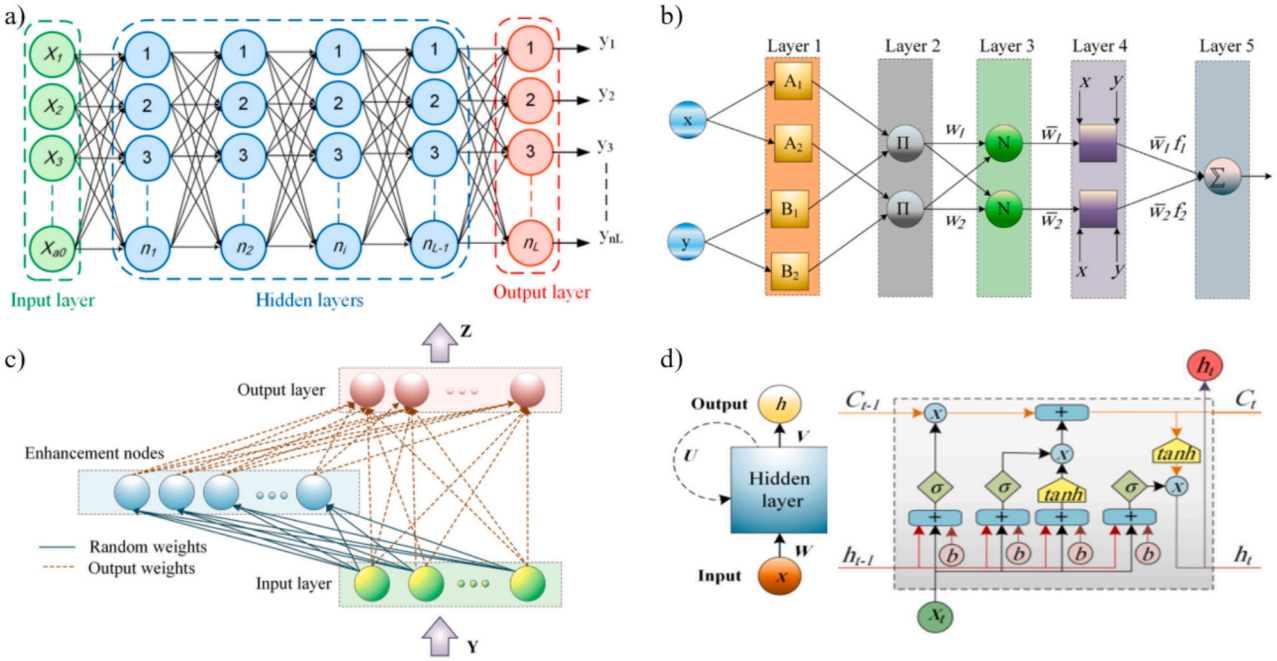


Fig. 3. The structure of well-known ML methods: a) MLP; b) ANFIS; c) RVFL; d) LSTM.

predict a continuous target variable. ANFIS consists of five layers as shown in Fig. 3 (b).:

Layer 1 (Input Layer): The input layer consists of nodes representing the input features $x_1, x_2, \cdots, x_n$.
Layer 2 (Fuzzy Layer): The fuzzy layer computes the membership degree of each input variable to fuzzy sets. The membership degree $\left(\mu_{ij}\right)$ of the i^{th} input variable to the j^{th} fuzzy set is determined using Gaussian or bell-shaped membership functions.
Layer 3 (Rule Layer): The rule layer computes the firing strength of each rule by taking the product of the membership degrees of the input variables associated with that rule.
Layer 4 (Normalization Layer): The normalization layer normalizes the firing strengths of the rules to ensure that they sum up to 1.
Layer 5 (Output Layer): The output layer computes the predicted output $(\hat{y})$ by taking a weighted sum of the normalized firing strengths of the rules.

The forward propagation process in ANFIS involves computing the outputs of each layer based on the inputs:

Layer 1 (Input Layer): The inputs $x_1, x_2, \cdots, x_n$ are passed directly to the next layer.
Layer 2 (Fuzzy Layer): The membership degrees μ_{ij} are computed using the Gaussian or bell-shaped membership functions.
Layer 3 (Rule Layer): The firing strengths of the rules are computed as the product of the membership degrees associated with each rule.
Layer 4 (Normalization Layer): The firing strengths of the rules are normalized to ensure that they sum up to 1.
Layer 5 (Output Layer): The predicted output $(\hat{y})$ is computed as a weighted sum of the normalized firing strengths of the rules.

ANFIS parameters, including membership functions and output layer parameters, are trained using gradient-based optimization algorithms like backpropagation or least squares. During training, the model minimizes a loss function, typically MSE, between predicted and true outputs. Once trained, ANFIS predicts new data by passing it through the network and computing output using learned parameters. ANFIS combines neural networks' adaptability with fuzzy logic's linguistic modeling power, making it suitable for complex regression tasks. However, training ANFIS can be computationally demanding, especially with intricate fuzzy rule sets or large datasets. Parameter initialization and hyperparameter selection are critical, affecting model performance and convergence. ANFIS may struggle to generalize with sparse or highly nonlinear data, and scalability issues arise with high-dimensional data due to exponential growth in model complexity. Despite challenges, ANFIS offers a flexible framework for complex systems and uncertainty modeling. With careful design and training, ANFIS has proven effective in prediction, classification, and control applications.

4.10. Long Short-Term memory networks

LSTM networks belong to the class of recurrent neural network (RNN) architectures designed to figure out dependencies with long-term characteristics within sequential data [62]. While LSTM networks are often used for sequence prediction tasks like time series forecasting or natural language processing, they can also be applied to regression tasks where the objective is to predict a continuous target variable. Herein a simple mathematical representation of LSTM as a regression machine learning algorithm is introduced:

An LSTM network consists of multiple memory cells, each of which contains a set of gates to control the flow of information: an input gate (i_t), a forget gate $\left(f_t\right)$, an output gate (o_t), and a cell state (C_t). The network also has a hidden state (h_t) which is passed between time steps and an output $\left(y_t\right)$ as shown in Fig. 3 (d).

At each time step t, the LSTM network receives an input (x_t) and the previous hidden state (h_{t-1}) and cell state (C_{t-1}). The key equations governing the LSTM cell dynamics are as follows:

Input gate

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}C_{t-1} + \beta_i) \tag{22}$$

Forget gate

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}C_{t-1} + \beta_f\right) \tag{23}$$

Output gate

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}C_t + \beta_o) \tag{24}$$

Cell state update

$$\check{C}_t = tanh(W_{xc}x_t + W_{hc}h_{t-1} + \beta_c) \tag{25}$$

$$C_t = f_t \bullet C_{t-1} + i_t \bullet \check{C}_t \tag{26}$$

Hidden state update

$$h_t = o_t \bullet tanh(C_t) \tag{27}$$

Output layer

$$y_t = W_{hy}h_t + \beta_y \tag{28}$$

where W, β, σ, and $tanh$ are weight matrices, bias vectors, sigmoid activation function, and the hyperbolic tangent activation function.

During training, the network learns the parameters (W and b) by optimizing a loss function, typically MSE between the predicted outputs $\left(y_t\right)$ and the true targets. Backpropagation through time is employed to calculate the gradients of the loss function concerning the network's parameters. These gradients are subsequently utilized to adjust the parameters through an optimization algorithm such as stochastic gradient descent or its variations. Once trained, the LSTM network can be used to make predictions on new input sequences by passing them through the network and obtaining the predicted output $\left(y_t\right)$ at each time step.

LSTM networks excel in regression tasks with sequential data and long-term dependencies, accurately forecasting outcomes over time. However, training LSTM networks can be complex and time-consuming, especially for large datasets or deep architectures, due to backpropagation through time (BPTT) and vanishing or exploding gradients. Optimizing hyperparameters like hidden units, layers, and learning rates requires extensive experimentation and computational resources. Poor initialization of model parameters can hinder convergence, requiring techniques like orthogonal initialization for better results. Overfitting is a concern with small datasets or complex architectures, necessitating regularization techniques such as dropout or weight decay. LSTMs are sensitive to irrelevant features and noisy data, requiring preprocessing methods like feature scaling or noise reduction. Despite challenges, LSTMs are widely used for tasks like speech recognition, natural language processing, time series forecasting, and sequence generation, offering state-of-the-art performance with careful design and training.

4.11. Random vector functional link

The RVFL network is a type of neural network architecture designed for regression tasks [63]. It consists of a single hidden layer with randomly generated input weights and biases, followed by a linear output layer. RVFL networks are simple and computationally efficient, making them suitable for regression problems with large datasets.

The RVFL network consists of three layers: an input layer, a hidden layer, and an output layer as shown in Fig. 3 (c).

Input Layer: The input layer consists of n input neurons representing the features of the input data.

Hidden Layer: The hidden layer consists of m hidden neurons. The weights connecting the input neurons to the hidden neurons (W) and the biases of the hidden neurons (β) are randomly generated.

Output Layer: The output layer consists of a single neuron, representing the predicted output $(\widehat{y})$. There are no weights or biases associated with the output neuron.

At each hidden neuron j, the weighted sum of the inputs (x) is computed using the randomly generated weights (w_{ij}) and biases $\left(\beta_j\right)$. The output of the hidden layer is then passed through an activation function φ. The output of the i^{th} hidden neuron (h_j) can be calculated as follows:

$$h_j = \varphi\left(\sum_{i=1}^{n} w_{ij}x_i + \beta_j\right) \tag{29}$$

where x_i denotes i^{th} input feature, w_{ij} denotes the weight connecting the i^{th} input neuron to the j^{th} hidden neuron, and β_j denotes the bias of the j^{th} hidden neuron.

The output of the hidden layer (H) can be represented as a matrix:

$$H = [h_1 h_2 .. h_m]' \tag{30}$$

The output of the output layer $\widehat{y}$) is computed as a linear combination of the outputs of the hidden layer:

$$\widehat{y} = H' \bullet W_{out} \tag{31}$$

where W_{out} denotes the weight matrix connecting the hidden layer to the output layer.

During training, the weights and biases of the hidden layer are randomly generated. The weight matrix (W_{out}) connecting the hidden layer to the output layer is then computed using the Moore-Penrose pseudoinverse method:

$$W_{out} = (H'H + \mu I)^{-1} H'y \tag{32}$$

where $y, \mu, andI$ denote the vector of true target values, the regularization parameter to prevent overfitting, and the identity matrix.

Once trained, the RVFL network predicts new input data by passing it through the network and computing the output using the learned weights. It's a simple and efficient regression algorithm suitable for handling large datasets. RVFL randomly generates weights and biases for the hidden layer and computes the output weights using the Moore-Penrose pseudoinverse method, ensuring computational efficiency. However, its fixed architecture lacks hidden layers or non-linear transformations, potentially limiting its performance on complex datasets with non-linear relationships. RVFL may struggle to capture complex feature interactions or hierarchical representations due to the absence of hidden layers. Random initialization of weights and biases can affect performance, requiring multiple runs for robustness. RVFL networks rely solely on linear combinations of input features, which may hinder their ability to capture non-linear correlations, necessitating more complex models or feature engineering techniques. Despite these limitations, RVFL networks remain useful for tasks prioritizing interpretability or computational efficiency, offering competitive performance with careful design and preprocessing of input data.

5. Machine learning for solar distiller modeling

Machine learning approaches in solar distillation modeling represent a powerful toolkit for predicting performance and optimizing operational parameters. These models, especially those based on regression techniques, analyze large datasets from solar stills (SSs), including inputs such as solar irradiance, ambient temperature, wind speed, and humidity, to estimate outputs like water yield. By leveraging historical data and real-time inputs, regression models can enable accurate predictions of optimal SS operating conditions, which helps maximize water output while minimizing energy usage. For instance, Mashaly and Alazba [64] utilized Multilayer Perceptron (MLP) and Linear Regression (LR) models to forecast water productivity, finding MLP models superior to LR, with minimum average R^2 of MLP and LR was 0.917 and 0.688, respectively. The higher predictive capacity of MLP over LR is attributed to its ability to model complex non-linear relationships, making it more resilient to overfitting issues frequently observed in simpler LR models.

In comparative analyses, various studies demonstrate the unique advantages of machine learning models for specific SS configurations and datasets. Santos et al. [65] modeled SS performance using an MLP model, achieving an R^2 range of 0.909 to 0.966, with 78 % of predicted values within 10 % of experimental data. Murugan et al. [66] investigated several ML models, including Decision Trees (DTs), MLP, and Random Forest (RF), noting that DTs outperformed other models with optimized cross-validation scores due to their robustness to complex, non-linear input–output relationships. The DT model's capability to generalize well on new data arises from its hierarchical nature, enabling accurate predictions even with minimal feature transformations. This feature contrasts with LR, which assumes linearity and often underperforms when input variables exhibit complex interdependencies.

In other studies, ensemble methods and hybrid models further refine SS predictive capabilities. Maddah [67] compared ensemble bagged-trees DTs to LR, revealing that ensemble methods (DTs with $R^2 = 0.93$ compared to LR's $R^2 = 0.68$) significantly boost accuracy by reducing overfitting through the aggregation of multiple model outputs. Similarly, Saravanan et al. [68] observed that DT models provided the highest R^2, RMSE, and MAE values compared to LR and KNN models in thermal performance prediction, affirming that DT's structure is particularly well-suited for capturing non-linear interactions without requiring extensive preprocessing. This capacity for implicit feature interaction makes DT advantageous over simpler models, but it also introduces susceptibility to overfitting, which must be managed through pruning or ensemble techniques.

For advanced machine learning models, such as SVM and RF, robustness against overfitting and sensitivity to non-linearities are evident benefits. Bamasag et al. [69] found SVM (with R2 = 0.99) performed better than MLP and Adaptive Neuro-Fuzzy Inference Systems (ANFIS) due to SVM's ability to manage noisy datasets and capture non-linear dynamics. RF models, as demonstrated by Gao et al. [70] in their prediction of SS performance across Chinese cities, achieved high accuracy (R^2 up to 0.939), benefiting from their ensemble structure, which aggregates multiple decision trees to enhance model stability and reduce overfitting. The RF's ensemble design promotes resilience to variability in data, unlike individual MLP or LR models, which may suffer from biases if dataset complexity is high or the data are sparse.

When considering regression models suited for non-linear relationships, MLPs and RF exhibit distinct advantages over traditional LR. Elgendi and Atef [71] utilized MLP and LR for a pyramid SS, finding that MLP ($R^2 = 0.976$) surpassed LR in predictive accuracy due to MLP's multi-layer structure, which learns hierarchical feature representations. However, while MLP's deep learning capacity is advantageous for intricate regression tasks, RF offers superior performance in high-dimensional spaces or with noisy datasets, as its ensemble framework mitigates overfitting while capturing multi-level interactions.

Other studies explored alternative ML methods with evolutionary and probabilistic components to manage noisy data effectively. Nazari et al. [72] employed Evolutionary Polynomial Regression (EPR) and Multivariate Adaptive Regression Splines (MARS) for SS performance prediction. The EPR model achieved superior results, attributed to its evolutionary optimization mechanism that increases resistance to overfitting and improves generalization across diverse conditions. Additionally, Sohani et al. [73] utilized feedforward (FF) and radial basis function (RBF) neural networks, finding that FF was ideal for hourly water production prediction ($R^2 = 0.963$), whereas RBF performed best in water temperature estimation ($R^2 = 0.977$) due to its localized learning.

Finally, studies utilizing Long Short-Term Memory (LSTM) networks for time series forecasting reveal advantages for applications requiring the analysis of sequential data. Elsheikh et al. [74] found LSTM outperformed the traditional ARIMA model (R^2 up to 0.997 versus 0.0017 for ARIMA) due to LSTM's ability to capture long-term dependencies in time series data, a crucial feature for water productivity forecasting in SSs. LSTM's recurrent structure, equipped with memory cells, enables it to learn from past data while adapting to dynamic future inputs, unlike ARIMA, which relies strictly on linear patterns.

Table 1 summarizes various studies on the application of machine learning models for predicting SS performance, including details on the SS design, meteorological variables, system parameters, target variables, machine learning models used, and the corresponding results. The table highlights the comparative performance and suitability of different models for various SS configurations and datasets.

Machine learning in solar distillation modeling demonstrates marked performance improvements across various applications, including regression, ensemble, and recurrent neural network models. Model selection, however, remains contingent upon specific SS design, input data characteristics, and desired performance outcomes, necessitating careful consideration of each method's strengths, limitations, and potential for data generalization.

6. Machine learning/metaheuristic optimizers hybrid models

Hybrid models that integrate machine learning techniques, such as Artificial Neural Networks (ANN), Adaptive Neuro-Fuzzy Inference Systems (ANFIS), Long Short-Term Memory (LSTM), and Random Vector Functional Link (RVFL), with metaheuristic optimization algorithms have been widely applied to model solar stills (SS). Notable examples include ANNs optimized by Particle Swarm Optimization (PSO) and Humpback Whale Optimizer (HWO) [75], ANFIS and ANN optimized by PSO [76]. LSTM optimized by the Great Wall Construction Algorithm (GWCA) [77], and RVFL optimized by the Sine Cosine Algorithm (SCA), Manta Ray Foraging Optimizer (MRFO), and Heap-Based Optimizer (HBO) [78]. Despite these advancements, determining the optimal hyperparameters for each model remains crucial to achieving high predictive accuracy. Effective hyperparameter tuning is vital in machine learning regression to enhance model performance, prevent overfitting, improve generalization, increase algorithm sensitivity, optimize computational efficiency, and ensure model robustness and stability.

Hyperparameters govern the operational intricacy of regression models. Adjusting these parameters facilitates identifying configurations that enhance performance metrics like accuracy, recall, precision, and mean squared error. This tuning process not only improves predictive accuracy but also mitigates overfitting by preventing models from learning the noise within the training data. A well-tuned regression model is more adept at generalizing to unseen data, effectively balancing bias and variance to avoid underfitting or overfitting. Hyperparameter tuning allows the adaptation of algorithms, such as SVM, to specific dataset characteristics, optimizing key parameters like kernel function selection and regularization, which significantly impact model efficacy.

In addition to enhancing model accuracy, optimized hyperparameters can improve computational efficiency. For instance, reducing tree depth in decision-tree-based models or limiting iterations in iterative algorithms can accelerate training without sacrificing performance. Systematic hyperparameter tuning, through extensive exploration of the hyperparameter space and cross-validation experiments, yields models that are robust and stable across diverse datasets and real-world conditions.

Metaheuristic optimization algorithms play a crucial role in efficiently exploring hyperparameter spaces to find near-optimal configurations, as depicted in Fig. 4. Algorithms such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Simulated Annealing (SA), and Differential Evolution (DE) are designed to conduct global searches through population-based strategies, mutation, crossover, and stochastic sampling. These techniques enable these algorithms to identify promising regions in the search space without exhaustively evaluating all possible configurations. They dynamically adjust key parameters, such as mutation rates and crossover probabilities, adapting their search strategies to prioritize promising hyperparameter regions. Many metaheuristic algorithms also support parallelization, allowing concurrent evaluation of multiple hyperparameter configurations and expediting the tuning process. Additionally, these algorithms can be customized to target specific performance metrics, computational resources, or

Table 1
Summary of studies on machine learning models for solar still performance prediction.

Study	SS Design	Meteorological Variables	Target Variable	ML Models Used	Results
Mashaly and Alazba [62]	Not specified	Solar Irradiance, Ambient Temperature	Water Yield	MLP, LR	MLP: $R^2 = 0.917$, LR: $R^2 = 0.688$, MLP superior to LR due to non-linear relationship handling
Santos et al. [63]	Not specified	Solar Irradiance, Ambient Temperature, Wind Speed	Water Yield	MLP	R^2 range: 0.909 to 0.966, 78 % of predicted values within 10 % of experimental data
Murugan et al. [64]	Not specified	Temperature, Humidity, Wind Speed	Water Yield	DT, MLP, RF	DT outperformed others with optimized cross-validation, hierarchical nature enabling robust predictions
Maddah [65]	Not specified	Solar Irradiance, Ambient Temperature	Water Yield	Ensemble Bagged Trees (DT), LR	DT $R^2 = 0.93$, LR $R^2 = 0.68$, Ensemble methods reduce overfitting
Saravanan et al. [66]	Not specified	Temperature, Humidity	Thermal Performance	DT, LR, KNN	DT: Highest R^2, RMSE, MAE compared to LR and KNN, suited for non-linear interactions
Bamasag et al. [67]	Not specified	Solar Irradiance, Ambient Temperature	Water Yield	SVM, MLP, ANFIS	SVM $R^2 = 0.99$, outperformed MLP and ANFIS due to noise handling and non-linear dynamics
Gao et al. [68]	Not specified	Solar Irradiance, Temperature	SS Performance	RF	$R^2 = 0.939$, RF outperformed MLP and LR, robust against dataset variability
Elgendi and Atef [69]	Pyramid SS	Solar Irradiance, Temperature	Water Yield	MLP, LR	MLP $R^2 = 0.976$, superior to LR due to deep learning capacity
Nazari et al. [70]	Not specified	Solar Irradiance, Temperature	Water Yield	EPR, MARS	EPR outperformed MARS, higher generalization, resistance to overfitting
Sohani et al. [71]	Not specified	Solar Irradiance, Temperature, Humidity	Water Yield, Temperature	FF, RBF Neural Networks	FF: $R^2 = 0.963$ for water production, RBF: $R^2 = 0.977$ for temperature estimation

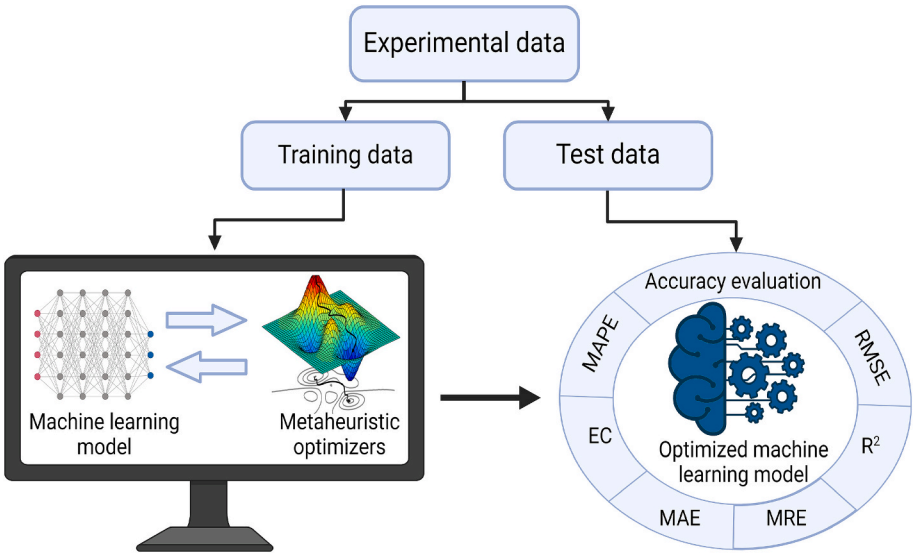


Fig. 4. Integeration between ML and metaheuristic optimizers for hyperparameter tuning.

robustness, thereby tailoring the search for optimal hyperparameter setups that meet specific requirements.

In the context of metaheuristic optimization algorithms, various methods, such as GA, PSO, SA, DE, Chimp Optimizer (CO), Tabu Search (TS), Grey Wolf Optimizer (GWO), Harmony Search (HS), Harris Hawk's Algorithm (HHO), Moth-Flame Optimizer (MFO), Sine Cosine Algorithm (SCA), Heap-Based Optimizer (HBO), Artificial Rabbit Optimizer (ARO), and Humpback Whale Algorithm (HWA), each offer unique advantages and limitations depending on the problem at hand. GA are widely used for solving complex optimization problems due to their robust global search capabilities. However, their slow convergence and high computational cost can be a drawback when applied to large datasets. PSO is known for its fast convergence and simplicity, making it efficient for continuous optimization, though it may struggle with high-dimensional or multimodal problems. SA is particularly effective for avoiding local optima by mimicking the physical annealing process, but it can be computationally expensive and slow to converge for large-scale problems. DE, similar to GA, is robust and well-suited for continuous optimization tasks, but its performance can be highly dependent on the selection of control parameters. CO is inspired by chimpanzee behavior and offers fast convergence and strong global optimization performance, though it may require careful tuning of its parameters to perform well. TS, which uses memory structures to avoid revisiting previously visited solutions, excels in local search and fine-tuning, but its computational expense and the need for proper memory management can be limiting factors. GWO, inspired by the hunting behavior of grey wolves, is known for its excellent convergence and ability to handle complex, high-dimensional optimization tasks, though it can sometimes fall prey to local optima. HS operates on the idea of musical improvisation and is simple to implement with low computational cost; however, it may not perform well in highly complex optimization tasks. HHO balances exploration and exploitation effectively, making it suitable for various problems, but it can be sensitive to parameter settings, which requires careful tuning for optimal performance. MFO, inspired by the navigation behavior of moths, is known for its ability to find global optima in complex landscapes, but like many other algorithms, it can struggle with local optima. SCA offers a simple and efficient optimization method with rapid convergence, particularly for continuous tasks, but it can have limitations when dealing with multi-modal or highly constrained problems. HBO is particularly useful in combinatorial optimization tasks, where it manages search space efficiently, but it may not be the best fit for continuous optimization problems. ARO, which simulates the behavior of rabbits searching for food, is effective in global optimization problems but can be computationally expensive and requires careful parameter tuning. Finally, the HWA mimics the bubble-net feeding behavior of humpback whales, which allows it to perform well in large and complex optimization spaces, though it too is computationally intensive and may require fine-tuning. In solar still optimization, these algorithms have been applied in various ways, such as parameter selection, system configuration, and operational efficiency maximization, with each algorithm offering distinct strengths for handling different types of optimization challenges, particularly when dealing with complex, multi-variable, and non-linear relationships. However, the choice of algorithm largely depends on the specific problem characteristics, computational resources, and the desired trade-off between exploration and exploitation in the search space.

The adaptability of metaheuristic algorithms makes them compatible with popular machine learning frameworks, facilitating seamless integration into hyperparameter tuning pipelines. While models like ANN rely on optimizers like gradient descent or stochastic gradient descent, these often encounter limitations such as high computational costs, susceptibility to local minima, and sensitivity to learning rates. Metaheuristic optimizers, such as GA, PSO, CO, TS, GWO, and HS, address these challenges by identifying optimal values for internal parameters, enhancing ANN accuracy.

Metaheuristic optimization has proven effective across various studies in improving machine learning model performance [79]. Essa et al. [80] developed a water yield prediction model using ANN optimized with the HHA, resulting in HHA-ANN, which outperformed both conventional ANN and SVM models in predicting water yield from solar stills. Moustafa et al. [75] combined a conventional ANN model with the HWA and PSO to develop HWA-ANN and PSO-ANN models. The HWA-ANN model achieved the highest R^2 values, ranging from 0.98 to 0.99, significantly outperforming pure ANN and PSO-ANN in accuracy.

Additionally, Elsheikh et al. [81] integrated a MFO with LSTM models to predict SS water yield, attaining an R2 value of 0.999, superior to traditional LSTM's 0.997–0.998, underscoring MFO-LSTM's precision. Abd Elaziz et al. [78] optimized RVFL models using several optimizers, including the SCA and HBO, with HBO-RVFL yielding the highest R^2 values, indicating its superior predictive accuracy. Alsaiari et al. [82] employed an ARO to enhance MLP models, achieving nearly perfect correlations (R^2 between 0.997–0.999) in predicting water yield across various SS designs, outperforming traditional optimizers like PSO and GA. Table 2 provides a comprehensive summary of various studies that utilized metaheuristic optimization algorithms to enhance machine learning models for predicting water yield in solar stills, showcasing the

Table 2
Summary of metaheuristic optimization techniques for enhancing machine learning models in solar still water yield prediction.

Study Reference	Machine Learning Model	Optimization Algorithm(s)	Target Variable	Key Results/ Findings
Essa et al. [74]	ANN	Harris Hawk's Algorithm (HHA)	Water yield	HHA-ANN outperformed conventional ANN and SVM in predicting water yield.
Moustafa et al. [75]	ANN	Harris Hawk's Algorithm (HWA), Particle Swarm Optimization (PSO)	Water yield	HWA-ANN achieved R^2 values from 0.98 to 0.99, significantly better than pure ANN and PSO-ANN models.
Elsheikh et al. [76]	LSTM	Moth-Flame Optimizer (MFO)	Water yield	MFO-LSTM achieved an R^2 of 0.999, outperforming traditional LSTM (0.997–0.998).
Abd Elaziz et al. [77]	RVFL	Sine Cosine Algorithm (SCA), Heap-Based Optimizer (HBO)	Water yield	HBO-RVFL yielded the highest R^2 values, indicating superior predictive accuracy.
Alsaiari et al. [78]	MLP	Artificial Rabbit Optimizer (ARO)	Water yield	ARO-MLP achieved R^2 values between 0.997 and 0.999, outperforming traditional optimizers like PSO and GA.

performance of different models and optimization techniques.

7. Discussion and prospects

ML models utilize data from SS operations, including meteorological variables (such as solar irradiance, temperature, humidity, and wind speed) and system parameters (like tilt angle, fin distribution, and airflow rates), to construct predictive models. These models analyze the relationships between input variables and water production rates to discern the factors influencing SS performance. They can predict water production rates of different SS designs, such as single basin, tubular, pyramid, stepped, inclined, hemispherical, and double slope, based on current or anticipated environmental conditions and system settings, aiding operators in optimizing strategies for efficiency and output. Additionally, ML models, such as MLP, LR, ANFIS, RVFL, LSTM, DT, and RF, can optimize operational parameters such as tilt angle and condensation surface material to maximize water production while minimizing energy consumption or cost. By offering actionable insights and recommendations, these models support decision-making and highlight opportunities for process optimization in SS operations. Thus, ML models play a vital role in modeling SS performance, empowering operators and engineers to enhance efficiency, identify faults, and make informed decisions to address water scarcity and promote environmental sustainability. The integration between ML and SSs' operation and optimization is illustrated in Fig. 5.

In modeling SSs, various ML regression tools can be employed to predict key performance parameters and optimize system efficiency. Regression models, such as LR, MARS, EPR, KNN, DT, RF, SVM, MLP, ANFIS, LSTM, and RVFL, are used to predict continuous output variables based on input features. They are commonly employed to model relationships between environmental factors (such as solar radiation, temperature, and humidity) and performance metrics of SSs (such as water production rate, energy efficiency, exergy efficiency, and water temperature). Optimization algorithms, including genetic algorithms, simulated annealing, heap-based optimizers, tree–seed algorithms, particle swarm optimization, and rabbit optimizers are employed to

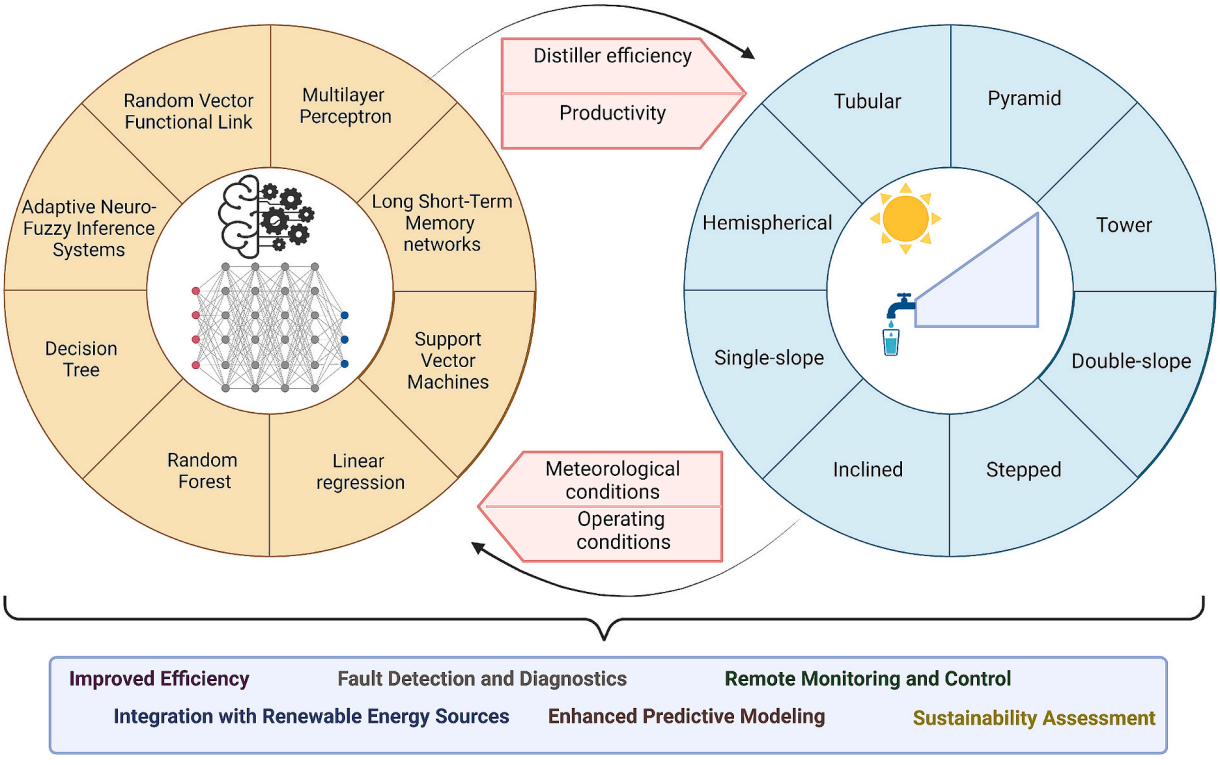


Fig. 5. The integration between ML and SSs' operation and optimization.

optimize system design parameters or operating conditions in solar distillation systems. These algorithms seek the best solution within a specified parameter space, considering constraints and objectives set by the user.

The choice of an ML tool to model a certain SS depends on factors such as the complexity of the relationship between input variables and output parameters, the size and nature of the dataset, computational resources, and the objectives of the analysis. In practice, a combination of different machine learning tools and techniques may be employed to achieve the best results in modeling solar distillation systems.

ML models provide numerous advantages that make them invaluable tools for predicting continuous outcomes based on input features. They offer flexibility, robustness, predictive accuracy, adaptability, scalability, feature importance analysis, and versatility. These attributes render them essential for predictive modeling and decision-making in the field of water desalination. One key advantage is their flexibility, allowing them to capture complex connections between input variables and output parameters. ML models can handle non-linear, time-varying, and high-dimensional data, making them suitable for modeling diverse and intricate systems. Moreover, they excel in achieving high predictive accuracy by learning from data patterns and relationships. Their ability to discern subtle patterns and dependencies in the data enables more precise predictions of continuous outcomes. ML models are also robust, often unaffected by noise and outliers in the data. They possess the capability to filter out irrelevant information and focus on the most informative features, resulting in robust and reliable predictions even in the presence of noisy data. Additionally, they provide insights into the importance of different input features through feature importance analysis. By examining feature importance scores or coefficients, practitioners can gain valuable insights into the factors driving predictions, aiding decision-making and further analysis. Furthermore, ML models offer adaptability and automation, allowing them to adjust to changing data and make predictions in real time. Their scalability enables them to handle large datasets efficiently, making them suitable for analyzing vast amounts of information commonly encountered in water desalination applications. Overall, the versatility and effectiveness of ML models make them indispensable tools for enhancing the efficiency and sustainability of water desalination processes.

ML regression models offer powerful techniques for predicting continuous outcomes based on input features. However, they also come with several drawbacks and limitations that must be carefully considered and addressed to ensure accurate and reliable predictions. One common drawback is overfitting, wherein the model grasps noise or erratic variations in the training data instead of the fundamental patterns, potentially leading to subpar performance on new data. Conversely, underfitting arises when the model is overly simplistic to capture the genuine underlying relationships within the data, resulting in insufficient predictive capability. The quality and quantity of the training data also significantly influence the performance of ML models. Noisy, incomplete, or biased training data can result in inaccurate or unreliable predictions. Certain ML regression models, such as linear regression or Gaussian process regression, may make assumptions about the data distribution or relationships between variables. If these assumptions are violated, the model's predictions may be inaccurate. Moreover, ML models may require large datasets to generalize well, particularly for complex tasks or high-dimensional data. Training complex machine learning models on large datasets can be computationally demanding and resource-intensive. Additionally, optimizing the hyperparameters of regression models, which control their learning process, is a crucial but time-consuming task that requires extensive experimentation to find the optimal settings. Overcoming these challenges of computational requirements and hyperparameter tuning is crucial to maintaining the effectiveness, reliability, and strong predictive power of machine learning models in solar desalination applications. It's essential to acknowledge that the effectiveness of different machine learning models can vary depending on factors such as the unique characteristics of the dataset, the complexity of the underlying relationships, and the optimization of model hyperparameters. Therefore, it is crucial to carry out experiments with various models and evaluate their performance using appropriate metrics to determine the most suitable model for a specific regression task.

MLPs and ANFIS excel in capturing complex non-linear relationships between input features and target variables, outperforming LR, which assumes a linear relationship. While MLPs and ANFIS autonomously model feature interactions, LR requires explicit specification. SVMs, however, are robust to overfitting and effective in high-dimensional spaces, capturing non-linearities through kernel functions. DTs offer interpretability, simplicity, and robustness to outliers, contrasting with the complexity and overfitting risks of MLP and ANFIS. RF improve generalization through ensemble learning, handling feature selection and noise without manual intervention. LSTM networks are ideal for sequential data, learning temporal dependencies with minimal feature engineering, while RVFL networks are simple and efficient, offering excellent generalization and transparency with minimal hyperparameter tuning. Ensemble RVFL models enhance prediction accuracy and robustness, addressing the limitations of individual models and improving generalization, making them powerful for complex regression tasks.

The optimal machine learning model for regression tasks is contingent on several factors, including the dataset's attributes, the intricacy of the relationship between features and target variables, computational resources available, interpretability requirements, and the specific goals of the regression task. Various models possess distinct strengths and weaknesses, and determining the optimal model typically requires experimentation and evaluation. Selecting the "best" model for regression tasks often involves comparing the performance of multiple models through techniques like cross-validation and assessing metrics such as R^2, RMSE, MAE, and CRM. Factors such as model complexity, interpretability, and computational efficiency should be taken into account when deciding on the most suitable model for a specific regression task.

ML has the potential to transform the real-time control of SSs by utilizing data-driven algorithms to optimize operational parameters and enhance freshwater production. These algorithms analyze real-time sensor data to forecast environmental conditions and predict the optimal settings for controlling parameters such as the orientation of solar concentrators, feed flow rates, and condensation surface temperatures. Adaptive control systems dynamically adjust operating conditions in response to changing environmental factors and system dynamics, ensuring continuous optimization of performance and efficiency. Moreover, ML algorithms can detect anomalies and faults in system components, facilitating early detection and intervention to prevent downtime and enhance reliability. Thus, ML enables SSs to achieve adaptive, efficient, and intelligent control, thereby advancing sustainable freshwater production in resource-constrained environments.

Hyperparameter tuning is pivotal in ML models as it directly influences their performance, generalization capability, and efficiency. Hyperparameters are settings that govern the behavior and complexity of ML algorithms, including parameters such as the learning rate, regularization strength, number of hidden neurons, depth of decision trees, or the number of hidden layers in neural networks. Optimizing these hyperparameters ensures that the model effectively learns the underlying patterns in the data, avoids overfitting, and generalizes well to unseen data. By tuning hyperparameters, the objective is to identify the optimal configuration that maximizes the model's performance metrics, such as precision and accuracy.

Without proper hyperparameter tuning, ML models may underperform, leading to inaccurate predictions. Hyperparameter tuning is crucial for optimizing performance and ensuring reliable results. Metaheuristic optimization algorithms are effective in hyperparameter tuning, as they explore high-dimensional search spaces efficiently. Unlike traditional methods, these optimizers use heuristic rules to navigate

complex, non-linear spaces. Algorithms such as genetic algorithms, particle swarm optimization, and simulated annealing can evaluate large numbers of hyperparameter configurations and identify optimal solutions. These techniques improve model performance and generalization, resulting in more accurate predictions.

Various software tools and libraries play a crucial role in modeling the performance of SSs and analyzing experimental data. Among these, Python, Matlab, and R stand out as popular choices due to their versatility and robustness in implementing ML algorithms and conducting statistical analysis. These software tools provide a comprehensive platform for researchers and engineers to develop accurate predictive models, optimize system design parameters, and evaluate the impact of different factors on SS performance. By leveraging these tools, the field of sustainable freshwater production continues to advance, offering innovative solutions to address water scarcity challenges.

The utilization of machine learning in solar desalination using SSs offers several promising prospects as follows:

- **Improved Efficiency:** ML algorithms have the potential to enhance both the design and operation of SS systems, resulting in heightened water production efficiency. Through the analysis of diverse data sources encompassing weather conditions, system parameters, and water quality, ML models can pinpoint optimal operational conditions and control tactics to amplify water yield while curbing energy usage. These algorithms can also optimize crucial operating parameters of SSs, like tilt angle, condensation surface material, and airflow rates, by drawing insights from historical data and predictive modeling. This approach enables the identification of optimal operating conditions geared toward maximizing water production efficiency and optimizing energy utilization.
- **Enhanced Predictive Modeling:** ML methodologies facilitate precise predictive modeling of SS performance across diverse environmental conditions. Leveraging historical data alongside real-time sensor readings, ML models can anticipate water production rates, forecast system malfunctions or maintenance requirements, and fine-tune operation schedules to accommodate fluctuations in water demand. Furthermore, the advancement of sophisticated predictive models, such as ensemble techniques, deep learning structures, or hybrid models amalgamating ML with physics-driven modeling methodologies, holds promise. These models excel in capturing intricate relationships and dynamics within SS systems, thereby enhancing prediction accuracy and resilience.
- **Integration of Sensor Data:** Integrating real-time sensor data collected from SSs, encompassing metrics like temperature, humidity, solar radiation, and water production rates, into machine learning frameworks enhances predictive capabilities. This incorporation allows for more precise forecasts of system performance and enables the implementation of proactive maintenance and control approaches. Through the analysis of sensor data patterns and detection of anomalies, machine learning models can highlight potential issues such as clogging, leaks, or equipment malfunctions, empowering proactive troubleshooting and remediation efforts.
- **Integration with Renewable Energy Sources:** Explore the integration of SSs with other renewable energy sources, such as photovoltaic panels or wind turbines, and develop ML-based optimization strategies for hybrid renewable energy systems. These strategies can optimize energy production and utilization, enhance system resilience, and improve overall sustainability.
- **Remote Monitoring and Control:** Implement remote monitoring and control systems for SSs using ML algorithms. These systems can leverage IoT (Internet of Things) technologies and cloud computing platforms to remotely monitor system performance, analyze data in real-time, and autonomously adjust operating parameters for optimal performance.
- **Sustainability Assessment:** Conduct sustainability assessments of SS systems using ML-based life cycle analysis and environmental impact assessment techniques. These assessments can quantify the environmental, economic, and social impacts of SS technologies and inform decision-making processes for sustainable water desalination solutions.

By implementing these recommendations for future research, scholars and professionals can propel the use of ML in SS technology, enhancing the effectiveness, dependability, and eco-friendliness of water desalination methods. The outlook for employing ML in solar desalination with stills is optimistic, presenting avenues for enhancing efficiency, dependability, and sustainability in water generation while cutting down on expenses and environmental footprints. Ongoing exploration and innovation in this domain will continue to push the boundaries of ML's role in solar desalination, ultimately resulting in more robust and accessible water supply solutions for global communities.

8. Conclusions

This review examines the integration of predictive and hybrid ML approaches for optimizing the performance of SSs. The findings indicate that ML techniques effectively forecast key parameters such as water yield, thermal efficiency, and exergy efficiency, enhancing the overall performance of solar desalination systems. Notably, the use of metaheuristic optimizers significantly improves the prediction accuracy of these models. By analyzing data on meteorological variables, system parameters, and water quality, integrated ML frameworks can identify optimal operating conditions to maximize water output while minimizing energy consumption. Furthermore, these approaches facilitate real-time adaptations of operational parameters, improving system responsiveness to environmental changes. ML also plays a critical role in fault detection and anomaly identification, enabling proactive maintenance strategies that enhance system reliability. Additionally, the integration of ML with renewable energy sources allows for dynamic energy management, further improving the resilience and sustainability of solar still operations. Future research should expand these methodologies to more complex desalination systems, such as those utilizing solar collectors, photovoltaic panels, or humidification-dehumidification techniques, to enhance the effectiveness of solar desalination technologies in addressing global water scarcity challenges.

Data Availability Statement

Not applicable

CRediT authorship contribution statement

Ammar Elsheikh: Writing – review & editing, Writing – original draft, Visualization, Formal analysis, Conceptualization. **Hosam Faqeha:** Writing – review & editing, Writing – original draft. **Karrar A. Hammoodi:** Writing – review & editing, Writing – original draft. **Mohammed Bawahab:** Writing – review & editing, Writing – original draft. **Manabu Fujii:** Writing – review & editing, Writing – original draft. **S. Shanmugan:** Writing – review & editing, Writing – original draft. **Fadl A. Essa:** Writing – review & editing, Writing – original draft. **Walaa Abd-Elaziem:** Writing – review & editing, Writing – original draft. **B. Ramesh:** Writing – review & editing, Writing – original draft. **Ravishankar Sathyamurthy:** Writing – review & editing, Writing – original draft. **Mohamed Egiza:** Writing – review & editing, Writing – original draft, Formal analysis.

Declaration of Competing Interest

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

References

[1] Z. Huang, X. Yuan, X. Liu, The key drivers for the changes in global water scarcity: Water withdrawal versus water availability, J Hydrol (amst) 601 (2021) 126658.
[2] C.D. Bernholz, Population and Development Report: Water Scarcity in the Arab World, Gov Inf Q 22 (2005) 134–136.
[3] M. Khaki, I. Hoteit, Monitoring water storage decline over the Middle East, J Hydrol (amst) 603 (2021) 127166.
[4] M. Mukherjee, S. Roy, K. Bhowmick, S. Majumdar, I. Prihatiningtyas, B. Van der Bruggen, P. Mondal, Development of high performance pervaporation desalination membranes: A brief review, Process Saf. Environ. Prot. 159 (2022) 1092–1104.
[5] S.M. Shalaby, S.W. Sharshir, A.E. Kabeel, A.W. Kandeal, H.F. Abosheiasha, M. Abdelgaied, M.H. Hamed, N. Yang, Reverse osmosis desalination systems powered by solar energy: Preheating techniques and brine disposal challenges – A detailed review, Energy Convers Manag 251 (2022) 114971.
[6] Z. Rahimi-Ahar, M.S. Hatamipour, L.R. Ahar, Air humidification-dehumidification process for desalination: A review, Prog Energy Combust Sci 80 (2020) 100850.
[7] I. Khoshrou, M.R. Jafari Nasr, K. Bakhtari, New opportunities in mass and energy consumption of the Multi-Stage Flash Distillation type of brackish water desalination process, Solar Energy 153 (2017) 115–125.
[8] F.A. Essa, A.H. Elsheikh, A.A. Algazzar, R. Sathyamurthy, M.K. Ahmed Ali, M. A. Elaziz, K.H. Salman, Eco-friendly coffee-based colloid for performance augmentation of solar stills, Process Safety and Environmental Protection 136 (2020) 259–267.
[9] D. Mevada, H. Panchal, M. Ahmadein, M.E. Zayed, N.A. Alsaleh, J. Djuansjah, E. B. Moustafa, A.H. Elsheikh, K.K. Sadasivuni, Investigation and performance analysis of solar still with energy storage materials: An energy- exergy efficiency analysis, Case Stud. Therm. Eng. 29 (2022) 101687.
[10] S.W. Sharshir, A.H. Elsheikh, E.M.A. Edreis, M.K.A. Ali, R. Sathyamurthy, A. E. Kabeel, J. Zang, N. Yang, Improving the solar still performance by using thermal energy storage materials: A review of recent developments, Desalination, Water Treat 165 (2019) 1–15, https://doi.org/10.5004/dwt.2019.24362.
[11] S.S. Adibi Toosi, H.R. Goshayeshi, S. Zeinali Heris, Experimental investigation of stepped solar still with phase change material and external condenser, J Energy Storage 40 (2021) 102681.
[12] A.H. Elsheikh, H.N. Panchal, S. Sengottain, N.A. Alsaleh, M. Ahmadein, Applications of Heat Exchanger in Solar Desalination: Current Issues and Future Challenges, Water 14 (2022), https://doi.org/10.3390/w14060852.
[13] M. Ahangar Darabi, G. Pasha, B. Ebrahimpour, A.M. Guodarzi, F. Morshedsolouk, H. Habibnejad Roshan, R. Shafaghat, Experimental investigation of a novel single-slope tilted wick solar still with an affordable channeled absorber sheet, an external condenser, and a reflector, Solar Energy 241 (2022) 650–659.
[14] M. Al-Harahsheh, M. Abu-Arabi, M. Ahmad, H. Mousa, Self-powered solar desalination using solar still enhanced by external solar collector and phase change material, Appl Therm Eng 206 (2022) 118118.
[15] K. Pansal, B. Ramani, K. kumar Sadasivuni, H. Panchal, M. Manokar, R. Sathyamurthy, A.E. kabeel, M. Suresh, M. Israr, Use of solar photovoltaic with active solar still to improve distillate output: A review, Groundw Sustain Dev 10 (2020) 100341. https://doi.org/https://doi.org/10.1016/j.gsd.2020.100341.
[16] A.H. Elsheikh, S.W. Sharshir, M. Abd Elaziz, A.E. Kabeel, W. Guilan, Z. Haiou, Modeling of solar energy systems using artificial neural network: A comprehensive review, Sol. Energy (2019), https://doi.org/10.1016/j.solener.2019.01.037.
[17] K.A. Hammoodi, H.A. Dhahad, W.H. Alawee, Z.M. Omara, A detailed review of the factors impacting pyramid type solar still performance, Alex. Eng. J. 66 (2023) 123–154.
[18] M. Egiza, M.R. Diab, N. Faisal, A.H. Elsheikh, Natural fibers for enhanced efficiency and sustainability in solar desalination: A review, Sol. Energy 282 (2024) 112963.
[19] V.P. Katekar, S.S. Deshmukh, A review on research trends in solar still designs for domestic and industrial applications, J Clean Prod 257 (2020) 120544.
[20] A. Elsheikh, K.A. Hammoodi, A.M.M. Ibrahim, A.-H.-I. Mourad, M. Fujii, W. Abd-Elaziem, Augmentation and evaluation of solar still performance: A comprehensive review, Desalination 574 (2024) 117239.
[21] R. Sahu, A.C. Tiwari, Performance enhancement of single slope solar still using nanofluids at different water depth, Desalination Water Treat 317 (2024) 100046.
[22] S. Shanmugan, K.A. Hammoodi, T. Eswarlal, P. Selvaraju, S. Bendoukha, N. Barhoumi, M. Mansour, H.A. Refaey, M.C. Rao, A.-H.-I. Mourad, M. Fujii, A. Elsheikh, A technical appraisal of solar photovoltaic-integrated single slope single basin solar still for simultaneous energy and water generation, Case Stud. Therm. Eng. 54 (2024) 104032.
[23] Sundeep. Siddula, N. Stalin, C.R. Mahesha, V.S.N.C.H. Dattu, H. S, D.P. Singh, V. Mohanavel, R. Sathyamurthy, Triangular and single slope solar stills: Performance and yield studies with different water mass, Energy Reports 8 (2022) 480–488. https://doi.org/https://doi.org/10.1016/j.egyr.2022.10.225.
[24] E.A. Tei, R.M.S. Hameed, M. Illyas, M.M. Athikesavan, Experimental investigation of inclined solar still with and without sand as energy storage materials, J Energy Storage 77 (2024) 109809.
[25] H. Amiri, Development and application of a thermal model for the improved stepped solar still with a built-in passive condenser, Sol. Energy 270 (2024) 112378.
[26] S. Pavithra, T. Veeramani, S. Sree Subha, P.J. Sathish Kumar, S. Shanmugan, A. H. Elsheikh, F.A. Essa, Revealing prediction of perched cum off-centered wick solar still performance using network based on optimizer algorithm, Process Safety and Environmental Protection 161 (2022) 188–200.
[27] R.K. Sambare, S. Joshi, N.C. Kanojiya, Improving the freshwater production from tubular solar still using sensible heat storage materials, Therm. Sci. Eng. Prog. 38 (2023) 101676.
[28] K.A. Hammoodi, H.A. Dhahad, W.H. Alawee, Z.M. Omara, T. Yusaf, Pyramid solar distillers: A comprehensive review of recent techniques, Results Eng. 18 (2023) 101157.
[29] M.E.A.E. Ahmed, S. Abdo, M.A. Abdelrahman, O.A. Gaheen, Finned-encapsulated PCM pyramid solar still – Experimental study with economic analysis, J Energy Storage 73 (2023) 108908.
[30] M.R. Diab, M. Rozza, M. Alhosary, S. Nassar, N. Faisal, A.H. Elsheikh, M. Egiza, Performance enhancement of a modified solar still with inverted pyramid aluminum basin geometry: Experimental optimization, thermal, and economic assessment, Process Saf. Environ. Prot. 193 (2025) 1173–1187.
[31] K. Kelly Freitas Sarmento, C. Barbosa Silva, D. Silva de Abreu Benedito, G. Gilvania Cavalcante, K. Machado de Medeiros, C. Antônio Pereira de Lima, Cascade type solar distiller with the use of photothermic materials applied in the treatment of surface water, Appl Therm Eng (2024) 122721. https://doi.org/https://doi.org/10.1016/j.applthermaleng.2024.122721.
[32] L.D. Jathar, S. Ganesan, S. Gorjian, An experimental and statistical investigation of concave-type stepped solar still with diverse climatic parameters, Clean Eng Technol 4 (2021) 100137.
[33] I.M. Elsawy, A. Hamoda, S.W. Sharshir, A. Khalil, Experimental study on optimized using activated agricultural wastes at hemispherical solar still for different types of water, Process Saf. Environ. Prot. 177 (2023) 246–257.
[34] A. Hemmatian, H. Kargarsharifabad, A. Abedini Esfahlani, N. Rahbar, S. Shoeibi, Improving solar still performance with heat pipe/pulsating heat pipe evacuated tube solar collectors and PCM: An experimental and environmental analysis, Solar Energy 269 (2024) 112371.
[35] S. Arora, H.P. Singh, L. Sahota, M.K. Arora, R. Arya, S. Singh, A. Jain, A. Singh, Performance and cost analysis of photovoltaic thermal (PVT)-compound parabolic concentrator (CPC) collector integrated solar still using CNT-water based nanofluids, Desalination 495 (2020) 114595.
[36] G.S. Dhindsa, M.K. Mittal, Experimental study of basin type vertical multiple effect diffusion solar still integrated with mini solar pond to generate nocturnal distillate, Energy Convers Manag 165 (2018) 669–680.
[37] G. Singh, P.K. Singh, A. Saxena, N. Kumar, D.B. Singh, Investigation of conical passive solar still by incorporating energy metrics, efficiency, and sensitivity analyses for sustainable solar distillation, J Clean Prod 434 (2024) 139949.
[38] A.H. Elsheikh, A.I. Saba, H. Panchal, S. Shanmugan, N.A. Alsaleh, M. Ahmadein, Artificial Intelligence for Forecasting the Prevalence of COVID-19 Pandemic: An Overview, Healthcare 9 (2021), https://doi.org/10.3390/healthcare9121614.
[39] E.B. Moustafa, A. Elsheikh, Predicting Characteristics of Dissimilar Laser Welded Polymeric Joints Using a Multi-Layer Perceptrons Model Coupled with Archimedes Optimizer, Polymers (basel) 15 (2023), https://doi.org/10.3390/polym15010233.
[40] M. Abd Elaziz, S. Senthilraja, M.E. Zayed, A.H. Elsheikh, R.R. Mostafa, S. Lu, A new random vector functional link integrated with mayfly optimization algorithm for performance prediction of solar photovoltaic thermal collector combined with electrolytic hydrogen production system, Appl Therm Eng (2021) 117055.
[41] A.H. Elsheikh, M. Abd Elaziz, A. Vendan, Modeling ultrasonic welding of polymers using an optimized artificial intelligence model using a gradient-based optimizer, Weld. World 66 (2022) 27–44, https://doi.org/10.1007/s40194-021-01197-x.
[42] A.H. Elsheikh, A.I. Saba, M.A. Elaziz, S. Lu, S. Shanmugan, T. Muthuramalingam, R. Kumar, A.O. Mosleh, F.A. Essa, T.A. Shehabeldeen, Deep learning-based forecasting model for COVID-19 outbreak in Saudi Arabia, Process Saf. Environ. Prot. 149 (2021) 223–233.
[43] A.H. Elsheikh, M.A. Elaziz, S.R. Das, T. Muthuramalingam, S. Lu, A new optimized predictive model based on political optimizer for eco-friendly MQL-turning of AISI 4340 alloy with nano-lubricants, J Manuf Process 67 (2021) 562–578.
[44] M. Faegh, P. Behnam, M.B. Shafii, M. Khiadani, Development of artificial neural networks for performance prediction of a heat pump assisted humidification-dehumidification desalination system, Desalination 508 (2021) 115052.
[45] M.A.A. Al-qaness, A.A. Ewees, H. Fan, L. Abualigah, A.H. Elsheikh, M. Abd Elaziz, Wind power prediction using random vector functional link network with capuchin search algorithm, Ain Shams Engineering Journal (2022) 102095. https://doi.org/https://doi.org/10.1016/j.asej.2022.102095.
[46] M.E. Zayed, J. Zhao, W. Li, A.H. Elsheikh, M.A. Elaziz, D. Yousri, S. Zhong, Z. Mingxi, Predicting the performance of solar dish Stirling power plant using a hybrid random vector functional link/chimp optimization model, Sol. Energy 222 (2021) 1–17.
[47] A.H. Elsheikh, Applications of machine learning in friction stir welding: Prediction of joint properties, real-time control and tool failure diagnosis, Eng Appl Artif Intell 121 (2023) 105961.
[48] E.M.S. El-Said, M. Abd Elaziz, A.H. Elsheikh, Machine learning algorithms for improving the prediction of air injection effect on the thermohydraulic performance of shell and tube heat exchanger, Appl Therm Eng 185 (2021) 116471.
[49] M.A. Elaziz, A.H. Elsheikh, S.W. Sharshir, Improved prediction of oscillatory heat transfer coefficient for a thermoacoustic heat exchanger using modified adaptive neuro-fuzzy inference system, Int. J. Refrig 102 (2019) 47–54.
[50] M.E. Zayed, J. Zhao, W. Li, A.H. Elsheikh, M.A. Elaziz, A hybrid adaptive neuro-fuzzy inference system integrated with equilibrium optimizer algorithm for predicting the energetic performance of solar dish collector, Energy 235 (2021) 121289.
[51] H.A. Babikir, M.A. Elaziz, A.H. Elsheikh, E.A. Showaib, M. Elhadary, D. Wu, Y. Liu, Noise prediction of axial piston pump based on different valve materials using a modified artificial neural network model, Alex. Eng. J. (2019), https://doi.org/10.1016/j.aej.2019.09.010.
[52] K. Elmaadawy, M.A. Elaziz, A.H. Elsheikh, A. Moawad, B. Liu, S. Lu, Utilization of random vector functional link integrated with manta ray foraging optimization for

effluent prediction of wastewater treatment plant, J Environ Manage 298 (2021) 113520.
[53] X. Huang, H. Wang, W. Luo, S. Xue, F. Hayat, Z. Gao, Prediction of loquat soluble solids and titratable acid content using fruit mineral elements by artificial neural network and multiple linear regression, Sci Hortic 278 (2021) 109873.
[54] M.A. Sahraei, H. Duman, M.Y. Çodur, E. Eyduran, Prediction of transportation energy demand: Multivariate Adaptive Regression Splines, Energy 224 (2021) 120090.
[55] H. Bonakdari, A. Gholami, A.M.A. Sattar, B. Gharabaghi, Development of robust evolutionary polynomial regression network in the estimation of stable alluvial channel dimensions, Geomorphology 350 (2020) 106895.
[56] G. Lin, A. Lin, D. Gu, Using support vector regression and K-nearest neighbors for short-term traffic flow prediction based on maximal information coefficient, Inf Sci (n y) 608 (2022) 517–531.
[57] H. Lu, X. Ma, Hybrid decision tree-based machine learning models for short-term water quality prediction, Chemosphere 249 (2020) 126169.
[58] L. Yang, H. Wu, X. Jin, P. Zheng, S. Hu, X. Xu, W. Yu, J. Yan, Study of cardiovascular disease prediction model based on random forest in eastern China, Sci Rep 10 (2020) 5245, https://doi.org/10.1038/s41598-020-62133-5.
[59] G.-Q. Lin, L.-L. Li, M.-L. Tseng, H.-M. Liu, D.-D. Yuan, R.R. Tan, An improved moth-flame optimization algorithm for support vector machine prediction of photovoltaic power generation, J Clean Prod 253 (2020) 119966.
[60] S. Afzal, B.M. Ziapour, A. Shokri, H. Shakibi, B. Sobhani, Building energy consumption prediction using multilayer perceptron neural network-assisted models; comparison of different optimization algorithms, Energy 282 (2023) 128446.
[61] I. Ahmadianfar, S. Shirvani-Hosseini, J. He, A. Samadi-Koucheksaraee, Z. M. Yaseen, An improved adaptive neuro fuzzy inference system model using conjoined metaheuristic algorithms for electrical conductivity prediction, Sci Rep 12 (2022) 4934.
[62] B. Lindemann, T. Müller, H. Vietz, N. Jazdi, M. Weyrich, A survey on long short-term memory networks for time series prediction, Procedia CIRP 99 (2021) 650–655.
[63] A.H. Elsheikh, T.A. Shehabeldeen, J. Zhou, E. Showaib, M. Abd Elaziz, Prediction of laser cutting parameters for polymethylmethacrylate sheets using random vector functional link network integrated with equilibrium optimizer, J Intell Manuf (2020). https://doi.org/10.1007/s10845-020-01617-7.
[64] A.F. Mashaly, A.A. Alazba, Neural network approach for predicting solar still production using agricultural drainage as a feedwater source, Desalination, Water Treat 57 (2016) 28646–28660, https://doi.org/10.1080/19443994.2016.1193770.
[65] N.I. Santos, A.M. Said, D.E. James, N.H. Venkatesh, Modeling solar still production using local weather data and artificial neural networks, Renew, Energy 40 (2012) 71–79.
[66] D.K. Murugan, Z. Said, H. Panchal, N.K. Gupta, S. Subramani, A. Kumar, K. K. Sadasivuni, Machine learning approaches for real-time forecasting of solar still distillate output, Environ. Challenges 13 (2023) 100779.
[67] H.A. Maddah, Predictive Regression Models for Solar Energy Harvesting and Sustainable, Low Energy, Highly Efficient Solar-Desalination Systems, in: 2021 5th International Conference on Power and Energy Engineering (ICPEE), IEEE, 2021: pp. 111–115.
[68] A. Saravanan, S. Parida, M. Murugan, M.S. Reddy, P. Bora, S.R. Sree, Performance estimation of tubular solar still with a wicked rotating drum using DT, LR, and KNN techniques of machine learning, Neural Comput Appl (2022), https://doi.org/10.1007/s00521-022-07293-3.
[69] A. Bamasag, F.A. Essa, Z.M. Omara, E. Bahgat, A.O. Alsaiari, H. Abulkhair, R. A. Alsulami, A.H. Elsheikh, Machine learning-based prediction and augmentation of dish solar distiller performance using an innovative convex stepped absorber and phase change material with nanoadditives, Process Saf. Environ. Prot. 162 (2022) 112–123.
[70] W. Gao, L. Shen, S. Sun, G. Peng, Z. Shen, Y. Wang, A.W. Kandeal, Z. Luo, A. E. Kabeel, J. Zhang, H. Bao, N. Yang, Forecasting solar still performance from conventional weather data variation by machine learning method, Chin. Phys. B 32 (2023) 48801, https://doi.org/10.1088/1674-1056/ac989f.
[71] M. Elgendi, M. Atef, Calculating the impact of meteorological parameters on pyramid solar still yield using machine learning algorithms, International Journal of Thermofluids 18 (2023) 100341.
[72] S. Nazari, M. Najafzadeh, R. Daghigh, Techno-economic estimation of a non-cover box solar still with thermoelectric and antiseptic nanofluid using machine learning models, Appl Therm Eng 212 (2022) 118584.
[73] A. Sohani, S. Hoseinzadeh, S. Samiezadeh, I. Verhaert, Machine learning prediction approach for dynamic performance modeling of an enhanced solar still desalination system, J Therm Anal Calorim 147 (2022) 3919–3930, https://doi.org/10.1007/s10973-021-10744-z.
[74] A.H. Elsheikh, V.P. Katekar, O.L. Muskens, S.S. Deshmukh, M.A. Elaziz, S. M. Dabour, Utilization of LSTM neural network for water production forecasting of a stepped solar still with a corrugated absorber plate, Process Saf. Environ. Prot. 148 (2021), https://doi.org/10.1016/j.psep.2020.09.068.
[75] E.B. Moustafa, A.H. Hammad, A.H. Elsheikh, A new optimized artificial neural network model to predict thermal efficiency and water yield of tubular solar still, Case Stud. Therm. Eng. 30 (2022) 101750.
[76] M. Bahiraei, S. Nazari, H. Safarzadeh, Modeling of energy efficiency for a solar still fitted with thermoelectric modules by ANFIS and PSO-enhanced neural network: A nanofluid application, Powder Technol 385 (2021) 185–198.
[77] A. Elsheikh, M. Zayed, A. Aboghazala, F.A. Essa, S. Rehman, O.L. Muskens, A. Kamal, M.A. Elaziz, Innovative solar distillation system with prismatic absorber basin: Experimental analysis and LSTM machine learning modeling coupled with great wall construction algorithm, Process Saf. Environ. Prot. 186 (2024) 1120–1133.
[78] M.A. Elaziz, E.M.S. El-Said, A.H. Elsheikh, G.B. Abdelaziz, Performance prediction of solar still with a high-frequency ultrasound waves atomizer using random vector functional link/heap-based optimizer, Adv. Eng. Softw. 170 (2022) 103142.
[79] M. Abd Elaziz, A.H. Elsheikh, D. Oliva, L. Abualigah, S. Lu, A.A. Ewees, Advanced Metaheuristic Techniques for Mechanical Design Problems: Review, Arch. Comput. Meth. Eng. 29 (2022) 695–716, https://doi.org/10.1007/s11831-021-09589-4.
[80] F.A. Essa, M. Abd Elaziz, A.H. Elsheikh, An enhanced productivity prediction model of active solar still using artificial neural network and Harris Hawks optimizer, Appl Therm Eng 170 (2020), https://doi.org/10.1016/j.applthermaleng.2020.115020.
[81] A.H. Elsheikh, H. Panchal, M. Ahmadein, A.O. Mosleh, K.K. Sadasivuni, N. A. Alsaleh, Productivity forecasting of solar distiller integrated with evacuated tubes and external condenser using artificial intelligence model and moth-flame optimizer, Case Stud. Therm. Eng. 28 (2021) 101671.
[82] A.O. Alsaiari, E.B. Moustafa, H. Alhumade, H. Abulkhair, A. Elsheikh, A coupled artificial neural network with artificial rabbits optimizer for predicting water productivity of different designs of solar stills, Adv. Eng. Softw. 175 (2023) 103315.