International Journal of Finance and Managerial Accounting, Vol.8, No.28, Winter 2023 IJFMA

# Forecasting Tehran Price Index (TEPIX) Using Novel Meta-Heuristic Algorithms

**Milad Shahvaroughi Farahani**
1Department of Finance, Faculty of Management, Khatam University, Tehran, Iran
M.shahvaroughi@khatam.ac.ir

**Mohammadreza Nejad Falatouri Moghaddam**
2Ph.d Candidate, Department of Financial Management, Faculty of Management and Economics, Science and Research Branch, Islamic Azad University, Tehran, Iran
Mr.moghadddam@gmail.com

**Ali Ramezani**
3Ph.d Candidate, Department of Financial Management, Faculty of Management and Economics, Science and Research Branch, Islamic Azad University, Tehran, Iran
Ramezani.2006@yahoo.com

**Submit: 08/05/2021 Accept: 09/04/2022**

## ABSTRACT

The stock market involves risks and returns that, if forecasted correctly, can lead to profitability, and for this forecasting, appropriate methods are needed. It is affected by various parameters and needs a way to identify these parameters well and have a dynamic nature. The main goal of this article is forecasting Tehran Price Index (TEPIX) by using hybrid Artificial Neural Network (ANN) based on Genetic Algorithm (GA), Harmony Search (HS) particle Swarm Optimization algorithm (PSO) Moth Flame Optimization (MFO) and Whale Optimization algorithms. GA is used as feature selection. So, PSO, HS MFO and WOA are used to determine the number of input and hidden layers. We use the daily values of the stock price index of the Tehran Stock Exchange from 2013 to 2018 in order to forecasting price and test it. The accuracy of ANN, hybrid Artificial Neural Network with HS, PSO MFO and WOA is evaluated based on different loss functions such as MSE, MAE and etc. the results show that the predictability of Meta-heuristic algorithms in testing period is higher than normal ANN. Also, the predictability of hybrid WOA is higher than hybrid PSO and HS algorithms and MFO.

## Keywords:

Technical Indicators, Whale Optimization Algorithm, Genetic Algorithm, Harmony Search, particle Swarm Optimization algorithm, Moth Flame Optimization Algorithm

# 1. Introduction

Technical analysis and other methods such as fundamental analysis and statistical methods are used to prediction of stock price. The main reason beside financial gains for stock market prediction is Efficient Market Hypothesis (EMH). We have discussed EMH because if the market be efficient, prediction can be effective and non-efficient market means there are some people and firms which have rent. EMH means that information has a high impact on stock prices and prices modifying themselves according to this information [Naseer, M., & Bin Tariq, D. (2015)]. The efficient market ensures investors they access similar information. The efficient market is based on the assumption that no system can't beat the market because if this system become general, everybody will use it [Guerrien, B., & Gun, O. (2011)]. Thus, it negates its potential profitability. Neural networks are used for prediction of stock prices because they are able to recognize the linear relationships between inputs and outputs [de Oliveira, F. A., et al (2013)]. Many researchers such as economists and financial experts have acknowledged the chaos in stock market and other complex systems [Ruis Estrada et al, (2016)]. With the ability of the neural networks to learn nonlinear relationships, there is a chance to overcomes traditional analysis and other computational methods [Peter G. Zhang, (2020)]. In addition to the stock market prediction, neural networks are used for other financial tasks. Experimental and trading systems are available to track commodity and futures markets, Forex trading, financial planning, and corporate stability and bankruptcy forecasts [ÓS Jóhannsson, (2020)]. Banks use neural network to investigate a loan application and estimate the probability of bankruptcy. While, financial managers can use neural networks for planning and making profitable portfolios at the right time. As investment and transaction levels developed, people looked for tools and methods that maximum their profitability and minimum their risk. Some prediction models such as Statistical methods, technical analysis, fundamental analysis and linear regression are methods which used to prediction. None of these methods has been proven to be able to predict the market consistently correctly but they are different in the level and quality of the prediction like the predictability and the accuracy and the complex ability of computation.

Artificial neural networks are one of the main tools used in machine learning. ANNs are excellent tools for finding patterns which are far too complex or numerous for a human programmer to extract [P Bhowmik, (2019)]. Machine Learning and deep learning have become new and effective strategies commonly used by quantitative hedge funds to maximize their profits [C Krauss et al, (2017)]. Finance is highly nonlinear and sometimes stock price data can even seem completely random [Caginalp, G., & DeSantis, M. (2011)]. Traditional time series methods such as ARIMA and GARCH models are effective only when the series is stationary [Siami-Namini, S., & Namin, A. S. (2018)] which is a restricting assumption that requires the series to be preprocessed by taking log returns (or other transforms). However, the main issue arises in implementing these models in a live trading system, as there is no guarantee of stationarity as new data is added. Using neural networks can help to solve this problem.

In this paper, the Genetic Algorithm (GA) is used to select the best and most related input variables i.e., as a feature Selection. Three meta-heuristic algorithms such as Harmony Search (HS), Particle Swarm Optimization (PSO) and Moth Flame Optimization algorithms are used to improve the prediction power of the Artificial Neural Network (ANN) and minimize the network error, obtaining optimized weights and the best number of hidden layer simultaneously. In order to compare the proposed models performance and to choose the best model according to the size of error, we used eight estimation criteria for error assessment. In experimental results show that a hybrid ANN-MFO algorithm has the best performance.

- One of the contribution of this paper is that the codes used are optimal. For example, we have used Improved HS codes i.e. improve tuning. The other is that, HS algorithm usually is used as feature selection but in this article, we have used HS as a method for training the network. The last point is that, we have tried to use a relatively new and novel algorithm for comparison such as MFO and WOA.

The rest of the paper is organized as follows; Section 2 is dedicated to a review of the literature. Section 3 describes the proposed algorithm. Section 4 examines the experimental process and the results. Finally, Section 5 is conclusion.

# 2. Literature review

Stock market is a public market which the stocks of companies traded in it [Pretti and Santi, (2015)]. This market provides opportunities for brokers and companies to investment and is one of the main indicators of the economic situation in the country. Companies who need financial resources can publish shares for their investors and they can participate in gains and losses of company. Investors can earn profit in two manners; 1) increasing the stock price 2) Cash dividend distribution among shareholders [Wang. T., et al (2015)]. Stock market is characterized by features such as non-linearity, discontinuity, and volatile

multifaceted elements because it is related to many factors such as political events, general economic condition, and broker's expectations [Hadavandi, Shavandi and Ghanbar, (2010)]. Also, the quick process of this data by using high-tech technology and communication systems causes the stock price fluctuates very fast. Therefore, many banks, financial institutions, big investors and brokers have to trade the stock within the shortest possible time [Khaled et al, (2016)]. Obtaining maximum profit is the ultimate goal of the investors. As a result, many researchers are looking for market forecasting capabilities in a variety of ways [Prasanna and Ezhilmaran, (2013)].

ANN dedicated the best and most validate method in the prediction of stock price [Adebayo, Saiang and Nordlund, (2015)]. The three most popular ANNs for stock prediction are the Radial Basis Function (RBF) [Dash, R., & Dash, P. K. (2015)], the Recurrent Neural Network (RNN) [Gao, T., & Chai, Y. (2018)], and Multilayer Perceptron (MLP). There are many methods for training the ANN and some of them are better than the others in finding the linear and non-linear relationship. ANN uses two thresholds for exploring of linear and non-linear qualifications. The number of input layer and hidden layer is very important in predictability. If we use too many layers, the ANN couldn't find the fittest choice and the structure will be complicated. In addition, too few layers mean that the ANN is unable to find the global solution and non-linear relationships [Sheela, K. G., & Deepa, S. N, (2013)]. The researchers have tried to discover some methods which have high speed with high accuracy and lower the error. For this reason, the metaheuristic algorithms are used. These methods are used to optimize the network and find the best number of input and hidden layers. The ANN models in forecasting stock price, stock return, exchange rate, inflation and imports works better than traditional statistical models [Lean Yu et al, (2007)].

Gocken et al (2016), by using technical indicators and hybrid ANN Based on GA and HS predicted the price index in the Turkish stock market. The results showed that the error of hybrid meta-heuristic algorithms is less than ANN. They compared the hybrid ANN-HS and ANN-GA model and found that the error of ANN-HS is less than ANN-GA. Hassanin et al. (2016) used the GWO to provide good initial solutions to the ANN. The results showed that GWO based ANN outperforms both GA based ANN and PSO based ANN. Faris et al. (2016) presented that MVO shows very competitive optimization results of the set of weights and biases for multi-layer perceptron networks. In addition, GA, PSO, DE, FFLY and Cuckoo Search are used to compare the performance of the proposed method. Rather et al. (2017) observed that the field of hybrid forecasting has received lots of attention from researchers to form a robust model. At this point, ANN provided successful results in stock market forecasting for different stocks. Chong et al. (2017) predicted future market trend of south Korea by examining the effect of three unsupervised feature extraction methods (PCA, auto encoder, and restricted Boltzmann machine (RBM)) on the deep learning network with three loss functions such as NMSE, RMSE and MSE. Sezer et al. (2017) proposed a stock trading system based on deep neural network for buy–sell–hold predictions. GA was used to optimize the technical analysis parameters and create the buy–sell point of the system.

Ahmed et al. (2019) used Ant Colony Optimization (ACO) in forecasting stock price of Nigerian stock exchange. They compared ACO with three other algorithms including Price Momentum Oscillator, Stochastic and Moving Average. They concluded that ACO has more accuracy and lower error than other methods. Ghanbari and Arian (2019) used support vector regression (SVR) and butterfly optimization algorithm (BOA) in forecasting stock market. They presented a new BOA-SVR model based on BOA and compared it with results of 11 meta-heuristic algorithms on NASDAQ data. The result indicated that the considered model can improve the results and optimizing the SVR Parameters. On the other hand, this model has worked very well with higher performance accuracy and lower time consumption compared to other models. Zhang et al. (2018) propose a stock price trend prediction system that can predict both stock price movement and its interval of growth (or decline) rate within predefined prediction durations. They trained a random forest model from historical data from the Shenzhen Growth Enterprise Market (China) to classify multiple clips of stocks into four main classes (up, down, flat, and unknown) according to the shapes of their close prices. Their evaluation shows that the proposed system is robust to the market volatility and outperforms some existing predictions methods in terms of accuracy and return per trade. Chandana and et al (2019) used a novel approach based on LSSVR and Machine Learning to constructing a stock price forecasting expert system with the aim of improving forecasting accuracy. Their system made the prediction of stock market values simpler, involving fewer computations, that using the other method. [Rajesh and et al (2019)] used Ensemble learning techniques for stock trend prediction which based on the percentage change in the stock price data. They predicted S&P500 and its future trend with ensemble learning. To achieve this, they considered two prediction methods: Heat Map and Ensemble Learning. Observations shows that Random Forest, SVM and K-neighbors classifiers show the most prominent results of all other possible

combinations. The accuracy of the prediction model is more than 51% whereas in comparison with prediction models with a single classifier labelling with 30% accuracy the model has increased the accuracy by 23%.

Concentrated on analyzing the time series and modeling of finance time series based on many guidelines taken from the investors. This prediction revealed a disadvantages and was that this technique is only for the non-linear time series and it will not work for the traditional models.

Sengupta, S., et al (2019) studied a comprehensive survey about different applications, historical and recent developments of PSO with hybridization perspectives. They concluded hybrid techniques can lead to better result with superiority.

Li et al. (2019) synthetically evaluated various ML algorithms and observed the daily trading performance of stocks under transaction cost and no transaction cost. They utilized 424 S&P 500 index component stocks (SPICS) and 185 CSI 300 Index Component Stocks (CSICS) between 2010 and 2017 and compared traditional machine learning algorithms with advanced deep neural network (DNN) models. The traditional machine learning algorithms are SVM, Random Forest, Logistic Regression, naïve Bayes, Classification and Regression Tree (CART), and extreme Gradient Boosting while the DNN architectures include Multilayer Perceptron (MLP), Deep Belief Network (DBN), Stacked Auto encoders (SAE), RNN, LSTM, and GRU. Their results show that traditional machine learning algorithms have a better performance in most of the directional evaluation indicators without considering the transaction cost, however, DNN models show better performance considering transaction cost.

Gharehchopogh, F. S., & Gholizadeh, H. (2019) provided an overview and summarized the review of WOA applications. They concluded that WOA can use in engineering, clustering, classification, Robot path and etc. so, WOA has the most application in solving optimization problems.

Kumar et al. (2020) reviewed and organized the published papers on stock market prediction using computational intelligence. The related papers are organized according to related datasets, input variables used, pre-processing methods, techniques used for future selection, forecasting methods and performance metrics to evaluate the models.

Shehab, M., et al (2020) presented a comprehensive review of the MFO algorithm and analyzed its main characteristics. They concluded that MFO is one of the promising and successful metaheuristic algorithm in various optimization problems in a wide range of fields such as power and energy systems, economic dispatch, engineering design and etc. Their Conclusions focused on the current work on MFO, highlight its weaknesses, and suggested possible future research directions.

Dubey, M., et al (2021) studied the applicability of HAS in different problem domains and investigated the future research directions of HSA too. They concluded that by changing HSA parameters and appropriate tuning, the predictability of HSA can increase.

According to the above reviewed papers, it can be inferred that study on stock market prediction is steel being raised among researchers. Also, it seems that hybrid methods are the permanent approach used in different researches. Considering the acceptance of ANN-based methods, the focus is to enhance the performance of ANN through some metaheuristics. Limitations of the previous methods provided in tabular form (Table1):

**Table1. limitations of the previous methods**

| No | Methods | Purpose | Limitations |
|---|---|---|---|
| 1 | ARIMA (Autoregressive integrated moving average model) | Forecasting and clustering | -doesn't work well for non-linear time series<br>-requires more data<br>- Takes a long time processing for a large dataset |
| 2 | BPNN (Back propagation neural network) | Forecasting | - Sensitive to noise<br>- Actual performance based on initial values<br>- Slow convergent speed<br>- Easily converging to a local minimum |
| 3 | CART (Classification and Regression Trees) | Classification and forecasting | - Unstable even when the training data are small changed |
| 4 | GP (Gaussian Process) | Classification and forecasting | -Generates" black box" models which are difficult to interpret<br>-Can be computationally expensive |
| 5 | GRNN (Generalized Regression neural network) | Classification and forecasting | - Requires more memory space to store the model<br>- Can be computationally expensive because of its huge size |

| No | Methods | Purpose | Limitations |
|---|---|---|---|
| 6 | Hierarchical clustering | Clustering | -The length of each time series is the same because of the Euclidean distance<br>- Useful only for small datasets because of its quadratic computational complexity |
| 7 | HMM (Hidden Markov Model) | Clustering, classification and clustering | - Requires parameters to be set and is based on user assumptions that may be false with the result that clusters would be inaccurate<br>- Takes a long time processing for a large dataset |
| 8 | K-Mean | clustering | - The number of clusters must be specified in advance<br>- Sensitive to noise<br>- Only spherical shapes can be determined as clusters<br>- Unable to handle long time series effectively because of poor scalability |
| 9 | KNN (K Nearest Neighbor | Classification and forecasting | - The number of nearest neighbor's must first be determined<br>- Can be computationally expensive<br>- Memory limitation<br>- Sensitive to the local structure of the data |
| 10 | LR (Logistic Regression) | Classification and forecasting | - Sensitive to outliers<br>- Strong assumptions |
| 11 | LSTM (Long Short Term Memory) | Classification and forecasting | - Lacks a mechanism to index the memory while writing and reading the data The number of memory cells is linked to the size of the recurrent weight matrices |
| 12 | MLP (Multi-Layer Perceptron) | Classification and forecasting | -Convergence is quite slow<br>- Local minima can affect the training process<br>- Hard to scale |
| 13 | PSO (Particle Swarm Optimization) | Forecasting | - Lacks a solid mathematical foundation for analyzing future development of relevant theories |
| 14 | RBF (Radial Basis Function Neural Network) | Classification and forecasting | -Classification process is slower than MLP |
| 15 | RF (Random Forest) | Classification and forecasting | - Requires more computational power and resources because it creates a lot of trees<br>- Requires more time to train than decision trees |
| 16 | RNN (Recurrent Neural Network) | Classification and forecasting | -Difficult to train |
| 17 | SOM (Self Optimizing Maps) | Clustering and classification | - Does not work well for time series of unequal length because of the difficulty involved in determining the scale of weight vectors<br>- Sensitive to outliers |
| 18 | SVM (Support Vector Machine) | Classification and forecasting | -Sensitive to outliers<br>- Sensitive to parameter selection |
| 19 | SVR (Support Vector Regression) | Forecasting | - Sensitive to users' defined free parameters |
| 20 | ANN (Artificial Neural Network) | Classification and forecasting | - Over fitting<br>- Sensitive to parameter selection - ANNs just give predicted target values for some unknown data without any variance information to assess the prediction |

As it is clear, Researchers have tried to combine methods for better results and have been successful.

# 3. Methodology

## 3.1. Sampling

The statistical population of the study is Tehran Stock Exchange. Given that the prediction of the price index is the subject of the present research, so, the statistical sample includes the entire target community within 2013-2018.

We use 42 technical indicators as input variables for daily value of Tehran Stock Exchange Index (TEPIX) prediction. Different data such as open price, close price, lowest price, highest price, last price, and the volume of the transaction comes from official site of Tehran Stock Exchange and TSE Client software.

## 3.2. Technical indicators

Technical indicators are effective tools to characterize the real market situation. Using technical indicators can be more informative than using pure prices [Nikfarjam, Emadzadeh, and Muthaiyah, (2010)] and it is very practical way for stock analysts and fund managers to analyze stock market. there are many technical indicators and selection of the useful indicators accurately is the key issue to make a profit for those stock market investors [Wei and Cheng, (2012)]. For this reason, we use GA for selecting fittest input variables. GA used for determining the indicators that has the most significant effect on the forecasting performance. By using GA, we can evaluate the usefulness of indicators or eliminate irrelevant ones to simplify the proposed model. In table 2, there are all considered technical indicators.

**Table 2. Important and most common technical indicators as input variables**

| **Technical Indicators and the method of calculation** | |
|---|---|
| $Diff=Close_{Today}-Close_{Yesterday}$ | $M_{Open}=Open_{Today}-Open_{Yesterday}$ |
| Close | $M_{High}=High_{Today}-High_{Yesterday}$ |
| High | $M_{Low}=Low_{Today}-Low_{Yesterday}$ |
| Low | $M_{Close}=Close_{Today}-Close_{Yesterday}$ |
| Open | $Acc_{Open}=M\ Open_{Today}-M\ Open_{Yesterday}$ |
| $SMA(5)=\frac{(Close1+Close2+\cdots+Close5)}{5}$ | $Acc_{Close}=M\ Close_{Today}-M\ Close_{Yesterday}$ |
| $SMA(6)=\frac{(Close1+Close2+\cdots+Close6)}{6}$ | $Acc_{High}=M\ High_{Today}-M\ High_{Yesterday}$ |
| $SMA(10)=\frac{(Close1+Close2+\cdots+Close10)}{10}$ | $Acc_{Low}=M\ Low_{Today}-M\ Low_{Yesterday}$ |
| $SMA(20)=\frac{(Close1+Close2+\cdots+Close20)}{20}$ | $\%K=\left[\frac{(Close-Low)}{(High-Low)}\right]*100$ |
| $EMA(5)_{Today}=\frac{Close\ Today*k+EMA(5)Yestarday*(1-k)}{5}$<br>$K=\frac{2}{5+1}.EMA(5)0=SMA(5)$ | %D=3-day SMA of %K |
| $EMA(6)_{Today}=\frac{Close\ Today*k+EMA(6)Yestarday*(1-k)}{6}$<br>$K=\frac{2}{6+1}.EMA(5)0=SMA(6)$ | Slow%K==Fast%D |
| $EMA(10)_{Today}=\frac{Close\ Today*k+EMA(10)Yestarday*(1-k)}{10}$<br>$K=\frac{2}{10+1}.EMA(10)0=SMA(10)$ | Slow%D=3-day SMA of %D |
| $EMA(20)_{Today}=\frac{Close\ Today*k+EMA(20)Yestarday*(1-k)}{20}$<br>$K=\frac{2}{20+1}.EMA(20)0=SMA(20)$ | $William's\%R=\frac{(Highest\ high-Close)}{(Highest\ high-Lowset\ low)}$ |
| $TMA(5)=\frac{(SMA(1)+SMA(2)+\cdots+SMA(5))}{5}$ | $RSI=100-\frac{100}{1+RS}.RS=\frac{Average\ Gain}{Average\ Loss}$ |
| $TMA(6)=\frac{(SMA(1)+SMA(2)+\cdots+SMA(6))}{6}$ | Middle Band=SMA(20) |
| $TMA(10)=\frac{(SMA(1)+SMA(2)+\cdots+SMA(10))}{10}$ | Upper Band= MA(TP,n)+m∗σ[TP,n] |
| $TMA(20)=\frac{(SMA(1)+SMA(2)+\cdots+SMA(20))}{20}$ | Lower Band= MA(TP,n)−m∗σ[TP,n] |
| $AccDist=AccDist_{Yesterday}+Volume*CLV$<br>$CLV=\frac{[(Close-Low)-(High-Close)]}{High-Low}$ | $MP=\frac{(High+Low)}{2}$ |
| MACD=EMA(12)-EMA(26) | $ROC=\frac{(Close\ today-Close\ N\ previous\ day)}{Close\ N\ previous\ day}$ |
| $Signal_{MACD}=EMA(MACD,9)$<br>$=MACD_{Today}*0.2+(Signal_{MACD\ Yesterday}*(0.8))$ | $Typical\ Price=\frac{(High+Low+Close+Open)}{4}$ |
| $Weighted\ Close=\frac{((Close*2)+High+Low)}{4}$ | Volume |
| $OBV=OBV_{prev}+\begin{cases}Volume.\ if\ Close>Close_{Prev}\\0.\ if\ Close=Close_{prev}\\-Volume.\ if\ Close<Close_{Prev}\end{cases}$ | |

In table 2, stochastic indicators means %K and %D have two types: Fast and Slow, Which Low and High are minimum and maximum price of the n period ago respectively. About RSI indicator, Average Gain and Average Loss is as the following:

Average Gain= [(previous Average Gain) x 13 + current Gain] / 14

Average Loss= [(previous Average Loss) x 13 + current Loss] / 14

According to Bollinger Band indicator, MA stand for Moving Average, TP means Typical Price, n is Equal to number of period which Is usually 20 and finally m which is standard deviation and often is 20.

The last one means σ[TP,n] is equal to standard deviation during n period of TP.

More details about OBV indicator is as the following:

OBV= Current On Balance Volume Level

$OBV_{prev}$ = Previous On Balance Volume Level

Volume = Latest Trading Volume Amount

In order to understand these indicators, it's important to define each one and most common one separately which are defined by most of the experts:

- Close price: Final price at which a security is traded on a given trading day.
- Open price: The price at which a first bid is proposed on a given trading day.
- Bollinger Bands: Bollinger bands are made by three lines which help investment analyzers to find the trend of stock price.
- Momentum: It indicates the amount by which stock prices are altered over a period of time.
- Oscillator: This is a technical analysis tool that is made from a trend indicator for discovering short-term overbought or oversold conditions.
- Stochastic: It is used to determine the signals of over-purchasing, overselling, or deviation.
- Moving Average Convergence/Divergence (MACD): This is a well-known indicator which shows the correlation between two price moving averages.
- Relative strength index: It compares the magnitude of recent gains with recent losses. It is an attempt to determine overbought and oversold conditions of an asset.

This study includes two main parts. The first one includes calculating technical indicators and selecting the most optimal indicator by using GA. Second one includes forecasting stock price index by using different hybrid ANN models and comparing their prediction error. So, we divide Tehran Stock Price Index data from 2013-2018 into two parts: training and testing. Then, it is analyzed with artificial intelligence algorithms and forecasting the next day stock price index. Like [Gocken et al (2016)], we used 70% and 30% of data for training and testing respectively. We compare models with 8 criteria for prediction error. For training ANN, different algorithms are used which the most popular exist in table 3. These algorithms exist in MATLAB toolbox.

**Table 3. Common algorithms for training ANN**

| Algorithm Name | Abbreviation Name |
|---|---|
| Levenberg-Marquardt backpropagation | Trainlm |
| Gradient descent backpropagation | Traingd |
| Gradient descent with momentum and adaptive lrarning rate backpropagation | Traingdx |
| Gradient descent with adaptive learning rate backpropagation | Traingda |
| Gradient descent with momentum backpropagation | Traingdm |
| Batch training with weight and bias learning rules | Trainb |
| BFGS quasi-Newton backpropagation | Trainbfg |
| Bayesian regulation backpropagation | Trainbr |
| One-step secant backpropagation | Trainoss |
| Resilient backpropagation | Trainrp |

In this research, we used 42 technical indicators as input variables. Because these variables to be usable and able to use them as input variables, they should be normalized between 0 and 1. So, the largest number will be 1 and the smallest number will be 0. We can do this with Matlab software.

$$\widetilde{S_i} = \frac{(S_i - S_{min})}{S_{max} - S_{min}} . i = 1 \ldots . N$$

In the above equation, numerator i is the number of data. Fig1. represents the research methodology.

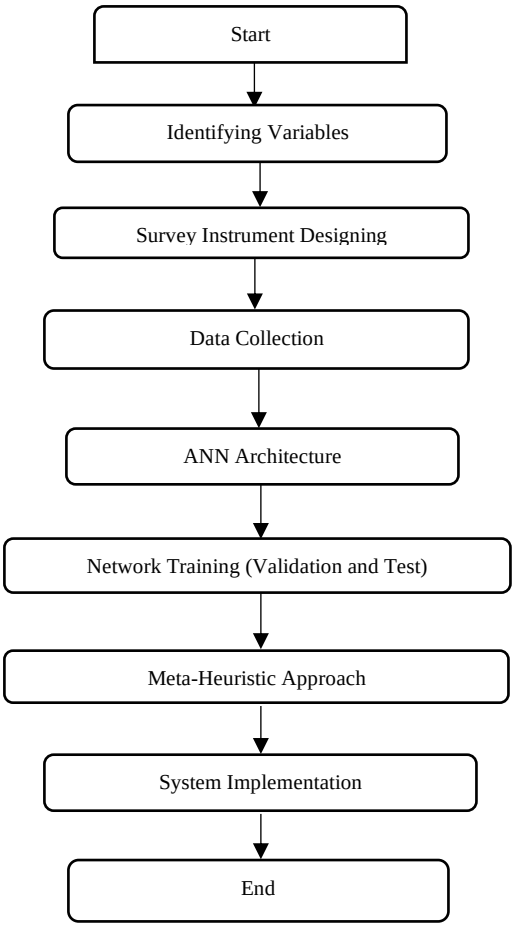


**Fig 1. research methodology**

## 3.3. Artificial Neural Network forecasting model

We have used ANN as a prediction method for several reasons: it can handle large amount of data sets; it has the ability to implicitly detect complex nonlinear relationships between dependent and independent variables; it has ability to detect all possible interactions between predictor variables; etc.

First, we use normal ANN without using any algorithms and in continue we get right into hybrid ANN for selecting input variables and determining the number of input and hidden layers. In this study, like [Gocken et al (2016)], used multi-layer perceptron (MLP) with three layers (two layers for input and output variables and one layer for hidden layer). Input layer, includes 42 input variables i.e. there are 42 neurons in input variable. Because output layer has one variable, it has one neuron. The number of hidden layers increases the ability of the model to detect complexity.

In this study, the hidden layer neurons of the normal neural network model are obtained through trial and error. So, we examine 1-32 neurons in hidden layer and choosing the fittest number of neurons which have the most accuracy as ANN model. For training ANN, we use error-back propagation. The minimization algorithm in learning the model is Levenberg-Marquardt algorithm which used for finding the minimum error point [de Rubio, J. J. (2020)]. The number of training epochs is 1000 and for the first time training rate is 0.01 and we decrease this rate to 0.001 in order to obtain more accurate results. ANN has two threshold functions. One of them is for recognizing the linear qualification and the other is for recognizing non-linear qualification of the model. The output function of hidden layer is sigmoid function and threshold function of output layer is pure line function. Figure 2 represents the architecture of the proposed neural network (Gocken et al, 2016).

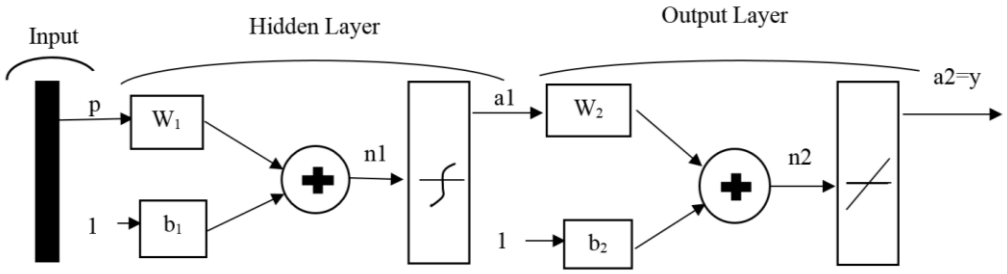


**Fig 2. Architecture of the proposed neural network**

In figure 2, P is the input pattern, $b_1$ is the vector of bias weights on the hidden neurons, and $W_1$ is the weight matrix between 0th (i.e. input) layer and 1th (i.e. hidden) layer. $a_1$ is the vector containing the outputs from the hidden neurons, and $n_1$ is the vector containing net-inputs going into the hidden neurons, $a_2$ is the column-vector coming from the second output layer, and $n_2$ is the column-vector containing the net inputs going into the output layer. $W_2$ is the synaptic weight matrix between the 1st (i.e. hidden) layer and the 2nd (i.e. output) layer and $b_2$ is the column-vector containing the bias inputs of the output neurons. Each row of $W_2$ matrix contains the synaptic weights for the corresponding output neuron [Ahmad, Jafri, Ahmad and Khan, (2007)]. Firstly, the neuron receives information from the environment and then this information multiplied by the corresponding weights is added together and used as a parameter within an activation (transfer) function. [Haider and Hanif, (2009)]. The transfer functions are used to prevent outputs from reaching very large values that can 'paralyze' ANN structure. For hidden layer, suitable transfer function is particularly needed to introduce non-linearity into the network because it gives the power to capture non-linear relationship between input and output [Ravichandran et al, (2005)].

## 3.4. GA-ANN forecasting model

GA has been used as feature selection for different reasons: 1. GA usually performs better than traditional feature selection techniques. 2. Genetic algorithms can manage data sets with many features. 3. They don't need specific knowledge about the problem under study. 4. These algorithms can be easily parallelized in computer clusters.

In this model, GA is used to input variable selection and used ANN as fitness function. GA finds the approximate optimal solution after the process of coding, decoding, and constant operation (reproduction, crossover, and mutation). Cross over and mutation is considered as the main operators of GA. GA use different coding instead of variables. In this study, the considered coding is binary coding. The usable chromosome contains 47 bits which 42 bits presents the existence or nonexistence of input (technical indicator) variables. If bit be "0" means don't exist variable and if be "1" means that exist variable and forming neuron in input layer. 5 other bits are equal to 1-32 ($2^5$=32) which shows the number of neurons in hidden layer. The population size of GA is 20 [davalou, (2017)]. The first population select randomly. The fitness function is ANN and its input variable is technical indicators and the number of hidden layer and its output is the amount of MSE. The smallest MSE in this series is the fittest choice for the next forecasting period. For increasing the training speed algorithms, the epochs are 100. We use 70 percent of data for training ANN. Other 30 percent used for validation and testing. At first, training (learning) rate is 0.01 which will be decrease with repeating training in order to obtain more exact result. If we want to obtain more accurate result, we can increase the epochs to 1000.

The considered parameters in the genetic algorithm are as follows:

**Table 4. GA parameters**

| Output Error | Output Activation Function | Input Activation Function | Mutation Rate | Crossover Rate | Number of Generation | Population Size |
|---|---|---|---|---|---|---|
| SSE | Logistic | Logistic | 0.1 | 0.9 | 50 | 50 |

Below figure represents the related flow chart of GA-ANN.

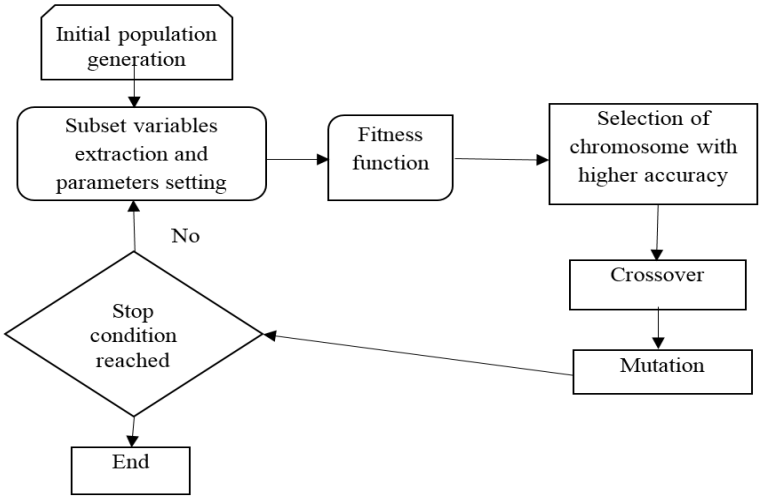


**Fig 3. Considered GA flow chart for training ANN**

Like [Gocken et al (2016)], we used roulette wheel for selecting parents and crossover percentage is 80. We use one-point crossover in crossover and doing fitness function for all of them. We use binary mutation and mutation percentage is 20. Among 20 parents and 20 children, we select 20 best individual as new generations. New generation continues with repeating above method until reaching termination condition. One of the termination conditions is repeating the best individual to 100 generations. If this condition doesn't hold, we check maximum generation condition. The maximum number of producing generation is equal to 2000. In the figure.4. you can see the mutation and crossover operator:

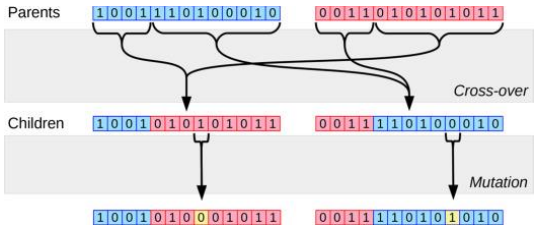


**Fig 4. Cross-over and mutation operator**

The crossover of two parent strings produces offspring (new solutions) by swapping parts or genes of the chromosomes. Crossover has a higher probability, typically 0.8-0.95. On the other hand, mutation is carried out by flipping some digits of a string, which generates new solutions.

Finally, the pseudo-code for GA-ANN is as the following:

**Table 5. GA-ANN Pseudo-Code**

| |
|---|
| function GENETIC-ALGPRITHM (population, FITNESS-FN) returns an individual |
| inputs : population, a set of individuals |
| FITNESS-FN, a function that measure the fitness of an individual |
| repeat |
| new_population <-- empty set |
| for i= = 1 to SIZE(population) do |
| x <-- RANDOM-SELECTION (population, FITNESS-FN) |
| y <-- RANDOM-SELECTION (population, FITNESS-FN) |
| Child <-- REPRODUCE (x ,y) |
| if (small random probability) then child <-- MUTATE(child) |
| add child to new_population |
| Population <-- new_population |
| until some individual is fit enough , or enough time has elapsed |
| return the best individual in population, according to FITNESS-FN |

## 3.5. PSO-ANN forecasting model

The PSO based on the behavior of birds was been proposed by [Elberhart and Kennedy (1995)]. In spite of some heuristic optimization algorithms such as GA and heuristic programming which is based on competition and survival of more powerful individuals, this algorithm is based on sharing information between population [Ismael et al, (2013)]. In this article, PSO has been used for multiple reasons: PSO is based on the intelligence. It can be applied into both scientific research and engineering use. Then PSO have no overlapping and mutation calculation. The search can be carried out by the speed of the particle. It including simple concept, easy implementation, robustness to control parameters, and computational efficiency when compared with mathematical algorithm and other heuristic

optimization techniques. This algorithm shows very good performance in many contexts such as Neural Network (NN) and fuzzy system control. It begins with initial population and in sequential iterations moving toward optimization answer. In each iteration, two answers calculate ($X^{Gbest}$ and $X^{i\text{-}pbest}$) which represent the best acquired location for each particle and best location in current location respectively. The foundation of PSO is that in each moment each particle set his/her location in searching space with best location which currently existed and the best location which existed in his/her neighborhood [Eberhart and Kennedy, (1995)]. Due to above facts, speed movement and the next particle location obtain from two next equations:

$$V_{j+1}^{i} = W_j V_j^i + c_1 r_1 \left(X_j^{i.pbest} - X_j^i\right) + c_2 r_2 \left(X_j^{Gbest} - X_j^i\right) \quad (2)$$

$$X_{j+1}^{i} = X_j^i + V_{j+1}^i \quad (3)$$

$c_1$and $c_2$ are importance of personal best and importance of neighborhood best respectively. Usually$c_1 + c_2 = 4$. No good reason other than empiricism. Accelerate constant $c_1$ and $c_2$ represent the particle stochastic acceleration weight toward the personal best (pbest) and the global best (gbest). Small accelerate constant may induce the particle wandering away in goal area; however, large accelerate constant may induce that the particle moves quickly to the goal area, even fly away from it. $r_1$and $r_2$are random numbers between 0,1. $V_j^i$ and $X_j^i$ are speed and location of particle jth and in iteration ith. Table 6, shows the parameters of PSO:

**Table 6. PSO parameters**

| Parameters | Size |
|---|---|
| Upper Bound | 1.5 |
| Lower Bound | -1.5 |
| $C_1$ | 1.5 |
| $C_2$ | 2.5 |
| Max Iteration | 2000 |

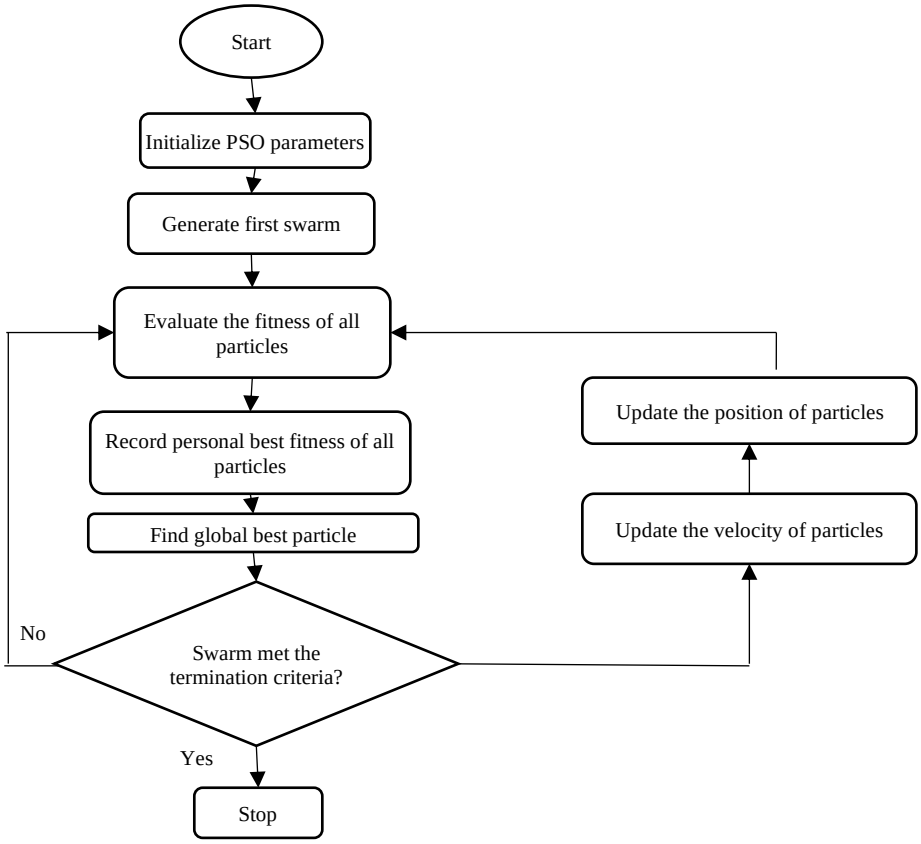


**Fig 5. Considered PSO flow chart for training ANN**

**Table 7. PSO-ANN Pseudo-Code**

| |
|---|
| For each particle |
| Initialize particle |
| END |
| Do |
| For each particle |
| Calculate fitness value |
| If the fitness value is better than the best fitness value (pBest) in history |
| set current value as the new pBest |
| End |
| Choose the particle with the best fitness value of all the particles as the gBest |
| For each particle |
| Calculate particle velocity according equation (a) |
| Update particle position according equation (b) |
| End |

## 3.6. HS-ANN forecasting model

Harmony Search (HS) is a meta-heuristic algorithm developed by [Geem et al (2001)] which is inspired by the natural musical performance process that occurs when a musician searches for a better state of harmony. The HSA achieves an optimal solution by finding the best "harmony" among the system components involved in a process. Musicians improve their experience based on aesthetic standards, whereas design variables can be improved based on an objective function. In comparison to other meta-heuristic in the literature, the HS algorithm imposes fewer mathematical requirements and can be easily adapted for solving various kinds of engineering optimization problems. Furthermore, numerical comparisons demonstrated that the evolution in the HS algorithm was faster than genetic algorithms. Therefore, the HS algorithm has attracted much attention and has been successfully applied to solve a wide range of practical optimization problems. In this study we used HSA for training ANN and finding the fittest number of input and hidden layer. HS has different parts and different steps. In HSA each solution is called as "harmony" and represented by an n-dimension real vector. An initial population of harmony vectors are randomly generated and stored in a harmony memory (HM). Then, a new candidate harmony is generated from all of the solutions in the HM by using a memory consideration rule, a pitch adjustment rule and a random re-initialization. Finally, the HM is updated by comparing the new candidate harmony and the worst harmony vector in the HM. The worst harmony vector is replaced by the new candidate vector if it is better than the worst harmony vector in the HM. The above process is repeated until a certain termination criterion is met. The HSA consists of three basic phases namely, initialization, improvisation of a harmony vector and updating the HM, which are described below respectively. In addition, other parameters of HS should be determined. These parameters are Harmony Memory Size (HMS) which is equal to 100, Harmony Memory Considering Rate (HMCR) which is equal to 0.95 and Pitch Adjusting Rate (PAR) which is 0.3 and bandwidth (bw) which is 0.2. We can show the HM with HMS*(N+1) like a bottom matrix which in this study N is 41.

Above phases will be explained in more details and different steps.

**Step 1.** Parameter initialization

Consider an optimal problem that is described by

$$\text{Minimize } F(x) \qquad x_i \in X_i, \quad i=1,2,\ldots N \qquad (4)$$

Where F(X) is the objective function, x is the set of design variables, $X_i$is the range set of the possible values for each design variable. The following HS algorithm parameters are also specified.

**Step 2.** Harmony Memory Initialization

The Harmony Memory (HM) matrix shown in (5) is filled with randomly generated solution vectors for HMS and sorted by the values of objective function f(x).

$$HM=\begin{pmatrix} x_1^1 & \cdots & \ldots & x_n^1 & f(x^1) \\ \vdots & & \ddots & & \vdots \\ x_1^{HMS} & \cdots & \ldots & x_n^{HMS} & f(x^{hms}) \end{pmatrix} \qquad (5)$$

A new harmony vector $x^1 = (x_1^1, x_2^1, \ldots, x_n^1)$ is generated based on three criteria: memory consideration, pitch adjustment and random selection. Generating a new harmony is called improvisation. The HMCR which varies between 0 and 1 is the rate of choosing one value from the historical values stored in the HM, while (1-HMCR) is the rate of randomly selecting one value from the possible range of values as shown in (6).

If (rand ( ) < HMCR)
$x_i' \in \{x_i^1 . x_i^2 . \ldots . x_I^{HMS}\}$
Else
$x_i' \in X_i$

Where rand () is a uniformly distributed random number between 0, 1 and is the set of the possible range of values for each decision variable. Every component obtained with memory consideration is examined to determine if pitch is to be adjusted. This operation uses the rate of pitch adjustment as a parameter as shown in the following:

If (rand ( ) < PAR)
$x_i' = x_i' \pm \text{rand}( ) * \text{bw}$
Else
$x_i' = x_i'$
end

Where bw is an arbitrary distance bandwidth for the continuous design variable and rand ( ) is uniform distribution between -1, 1.

**Step 4.** Update harmony memory

If the new harmony vector x′= (x1′, x2′…,xN′) has better fitness function than the worst harmony in the HM, the new harmony is included in the HM and the existing worst harmony is excluded from the HM.

**Step 5.** Checking the stopping criterion

If the stopping criterion which is based on the maximum number of improvisations is satisfied, the computation is terminated. Otherwise, step 3 and 4 are repeated. The HS parameters is as table 8:

In table 8. FW is the other name of BW (Band Width) which is one of the deciding factors for the time complexity and the performance of the algorithm and it's a random number between [0,1]. The BW needs to have both explorative and exploitative characteristics. The ideology is to use a large BW to search in the full domain and to adjust the BW dynamically closer to the optimal solution. FW-Damp is an influence within or upon an oscillatory system that has the effect of reducing, restricting or preventing its oscillations.

fig 6, shows the process of training ANN with HS.

**Table 8. HS parameters**

| Parameters | Size |
|---|---|
| Lower Bound | 11- |
| Upper Bound | 11 |
| HMS | 11 |
| NHMS | 100 |
| Max Iteration | 1000 |
| HMCR | 0.75 |
| PAR | 0.05 |
| Fret Width(FW) | 0.1 |
| FW-Damp | 0.95 |

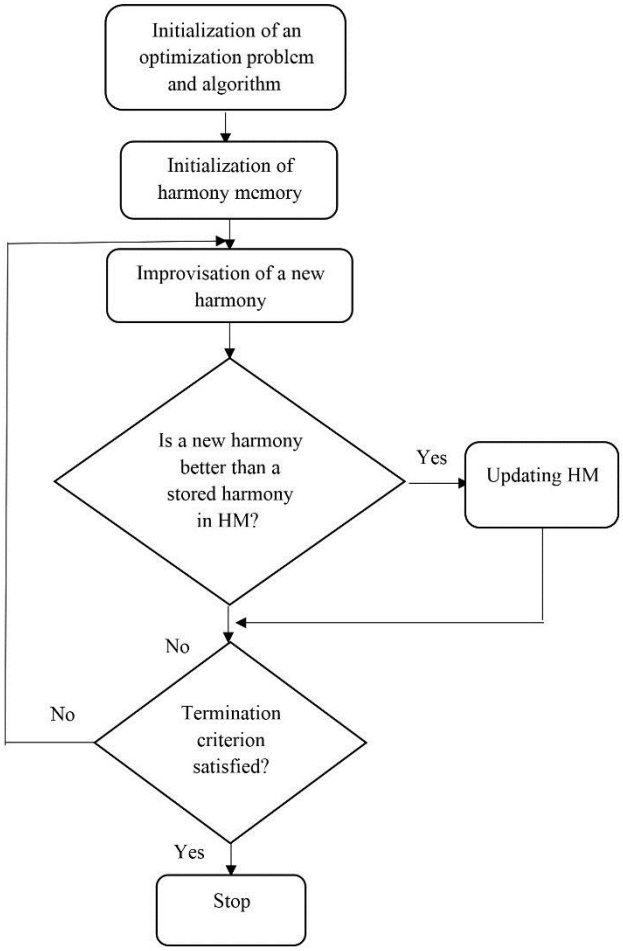


**Fig 6. Considered HS flow chart for training ANN**

Finally, the pseudo-code for HS-ANN is as the following:

**Table 9. HS-ANN Pseudo-Code**

```
for (j=1 to n) do
if (r1< HMCR) then
Xnew(j) = Xa(j) where a ∈(1,2,…,HMS)
if (r2< PAR) then
Xnew(j) = Xnew(j) ± r3 × BW where r1,r2,r3∈ (0,1)
endif
else
Xnew(j) = LBj + r × (UBj -LBj), where r ∈ (0,1)
endif
endfor
```

## 3.7. WOA

WOA is used as an optimization method because of showing significant advantages in handling a variety of modeling problems such as the exponential model, power model, delayed s-shaped model, and modified sigmoid model. This part is adapted from Seyed Ali Mirjalili and Lewis, A., (2016). The whale optimization algorithm (WOA) is a novel meta-heuristics algorithm proposed by Mirjalili at. el (2016). WOA is a population based method and simulates bubble-net attacking method of the humpback whales when they hunting their preys. The whales are living in groups and they are able to develop their own dialect. There are 7 types of whales and the humpback whale is one of these types. It has a special hunting mechanism which is called bubble-net feeding method. Humpback whales know the location of prey and encircle them. They consider the current best candidate solution is best obtained solution and near the optimal

solution. After assigning the best candidate solution, the other agents try to update their positions towards the best search agent as shown in the following equations;

$$D = |C.X^*(t) - X(t)| \quad (8)$$

$$X(t+1) = X^*(t) - A.D \quad (9)$$

Where (t) is the current iteration, A and C are coefficient vectors, $X^*$ is the position vector of the best solution, and X indicates the position vector of a solution, | | is the absolute value. The vectors A and C are calculated as follows:

$$A = 2a.r.a \quad (10)$$

$$C = 2.r \quad (11)$$

Where components of a are linearly decreased from 2 to 0 over the course of iterations and r is random vector in [0; 1]. The humpback whales attack the prey with the bubble-net mechanism. This mechanism is mathematical formulated as follow:

- ✓ Shrinking encircling mechanism:

In this mechanism, the value of A is a random value in interval [-a, a] and the value of a is decreased from 2 to 0 over the course of iterations as shown in Eq. 10.

- ✓ Spiral updating position mechanism

In this mechanism, the distance between the whale location and the prey location is calculated then the helix-shaped movement of humpback is created as shown in the following equation;

$$X(t+1) = D'.e^{bl}.\cos(2\pi l) + X^*(t) \quad (12)$$

Where $D' = | X^*(t) - X(t) |$ is the distance between the prey (best solution) and the $i^{th}$ whale, b is a constant, l is a random number in [-1; 1].

The humpback whales used the mentioned two mechanisms when they swim around the prey. We set the mathematical model of these two mechanisms, we assume that there is a probability of 50% to choose between these two mechanisms to update the position of whales as follow:

$$X(t+1) = \begin{cases} X^* - A.D & \text{if } p < 0 \cdot 5 \\ D'.e^{bl}.\cos(2\pi l) + X^*(t) & \text{if } p \geq 0 \cdot 5 \end{cases} \quad (13)$$

Where p is a random number in [0; 1].

In the exploration phase, the humpback whales (search agents) search for prey (best solution) randomly and change their positions according the position of other whales. In order to force the search agent to move far away from reference whale, we use the A with values > 1 or < 1 The mathematical model of the exploration phase is as follows:

$$D = |C.X_{rand} - X| \quad (14)$$

$$X(t+1) = X_{rand} - A.D \quad (15)$$

Where $X_{rand}$ is a random position vector chosen from the current population

The following steps is needed for operation of WOA:

Step 1. The standard whale optimization algorithm starts by setting the initial values of the population size n, the parameter a, coefficients A and C and the maximum number of iterations Max_itr.

Step 2. Initialize the iteration counter t.

Step 3. The initial population n is generated randomly and each search agent $X_i$ in the population is evaluated by calculating its fitness function f(Xi).

Step 4. Assign the best search agent X.

Step 5. The following steps are repeated until the termination criterion Satisfied.

Step 5.1. The iteration counter is increasing t = t + 1.

Step 5.2. All the parameters a, A, C, l and P are updated.

Step 5.3. The exploration and exploitations are applied according to the values of p and | A |

Step 6. The best search agent X is updated.

Step 7. The overall process is repeated until termination criteria satisfied.

Step8. Produce the best found search agent (solution) so far X.

You can see the WOA pseudo-code in the following:

**Table 10. WOA pseudo-code**

```
Randomly initialize the whale population
Evaluate the fitness values of whales and findout the best search agent X*.
While t < t_max
  Calculate the value of a
  For each search agent
    If h < 0.5
      If |A| < 1 then X(t+1) = X*(t) − A · D
      Else if |A| ≥ 1 then X(t+1) = X_rand(t) − A · D"
    End if
  Else if h ≥ 0 · 5 then
    D'. e^bl. cos(2πl) + X*(t)
  End if
  End for
  Evaluate the fitness of X(t + 1) and update X*.
End while
```

The WOA flowchart is presented below: (Fig.7. is adapted from Rana, N., et al, 2020).

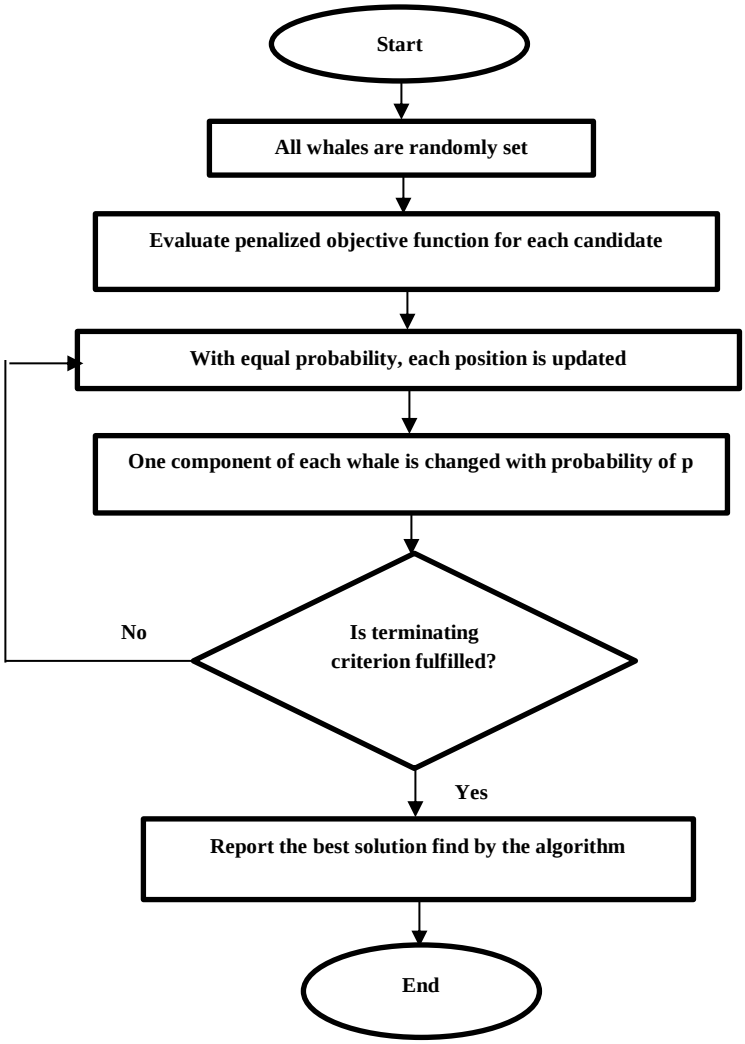


**Fig 7. WOA flowchart**

## 3.8. MFO Algorithm

MFO has been used as an optimization method because of having few setting parameters, being easy to understand and implement, and having fast convergence. Eventually, many scholars have put forward examples about the application of the algorithm in different fields to solve practical problems.

This part is adapted from Seyed Ali Mirjalili, (2015). MFO is a novel optimization algorithm which was proposed in 2016 by Mirjalili. It has simulated the navigation of moths in computer and has been widely used in science and industry. Moths are similar to the family of butterfly. They are two kinds: larvae and adult. They used to fly at night using moon light. They use a mechanism which is called transverse orientation for navigation. They set their angles with respect to the moon which can help them fly a long distance in a straight path. When a moths see the artificial light, they try to follow the same mechanism. In fact, moths are tricked due to the inefficiency of the transverse orientation.

In the MFO algorithm, moths are candidate solutions and the position of moths in the space are the problem's variables. So, moths can fly in different dimensional space and they change their position vector. Here, moths and flames are both solutions. In each iteration, there is difference between them in

updating and the way we treat with them. The moths are actual search agents that move around the search space, whereas flames are the best position of moths that obtains so far. As the main update mechanism of moths, a logarithmic spiral has been chosen. However, any types of spiral can be utilized here subject to the following conditions: Spiral's initial point should start from the moth Spiral's final point should be the position of the flame Fluctuation of the range of spiral should not exceed from the search space Considering these points, we define a logarithmic spiral for the MFO algorithm as follows:

$$S(M_i.F_i) = D_i e^{bt}\cos(2\pi t) + F_j \qquad (16)$$

Where $D_i$ indicates the distance of the $i-th$ moth for the $j-th$ flame, b is a constant for defining the shape of the logarithmic spiral, and t is a random number in [-1,1]. D is calculated as follows:

$$D_i = |F_j - M_i| \qquad (17)$$

Where $M_i$ indicate the $i-th$ moth, $F_j$ indicates the $j-th$ flame, and $D_i$ indicates the distance of the $i-th$ moth for the $j-th$ flame.

Equation16. is where the spiral flying path of moths is simulated. As may be seen in this equation, the next position of a moth is defined with respect to a flame. The t parameter in the spiral equation defines how much the next position of the moth should be close to the flame (t = -1 is the closest position to the flame, while t = 1 shows the farthest). Therefore, a hyper ellipse can be assumed around the flame in all directions and the next position of the moth would be within this space. Spiral movement is the main component of the proposed method because it dictates how the moths update their positions around flames. The spiral equation allows a moth to fly "around" a flame and not necessarily in the space between them. Therefore, the exploration and exploitation of the search space can be guaranteed. The exploitation for finding new solutions is important and proposed updating procedure can guarantee it. The best solutions obtained as the flame. So, the matrix F in the above equation always includes n recent best solutions obtained so far. The moths need to update their position. In order to further emphasize exploitation, we assume that t is a random number in [r,1] where r is linearly decreased from -1 to -2 over the course of iteration. Note that we name r as the convergence constant. With this method, moths tend to exploit their corresponding flames more accurately proportional to the number of iterations. In order to prevent local optima trap, each moth is obliged to update its position using only one of the flames. It each iteration and after updating the list of flames, the flames are sorted based on their fitness values. The moths then update their positions with respect to their corresponding flames. The first moth always updates its position with respect to the best flame, whereas the last moth updates its position with respect to the worst flame in the list. If all of the moths get attracted to a single flame, all of them converge to a point in the search spaces because they can only fly towards a flame and not outwards. Requiring them to move around different flames, however, causes higher exploration of the search space and lower probability of local optima stagnation. When you increasing the search space, you may decrease the chance of finding the best solution. To resolve this concern, an adaptive mechanism is proposed for the number of flames.

$$\text{FlameNumber} = \text{round}(N - l \times \frac{N-1}{T}) \qquad (18)$$

Where l is the current number of iteration, N is the maximum number of flames and T indicates the maximum number of iterations. There is N number of flames in the initial steps of iterations. However, the moths update their positions only with respect to the best flame in the final steps of iterations. The gradual decrement in number of flames balances exploration and exploitation of the search space. After all, the general steps of the P function are as follows.

The pseudo code of the MFO algorithm is presented below:

Table 11. Pseudo-Code of MFO Algorithm

```
Update the number of flames (FlameNumber)
Initialise the population of moths
Calculate the objective values
for all moths
    for all paramters
        update r and t
        Calculate D with respect to the corresponding moth
        Update the matrix M with respect to the
corresponding moth
    end
    calculate the objective values
    Update flames
end
```

there is MFO algorithm flowchart in the following:
(Fig.8. is adapted from M. A., Abdullah et al, 2019).

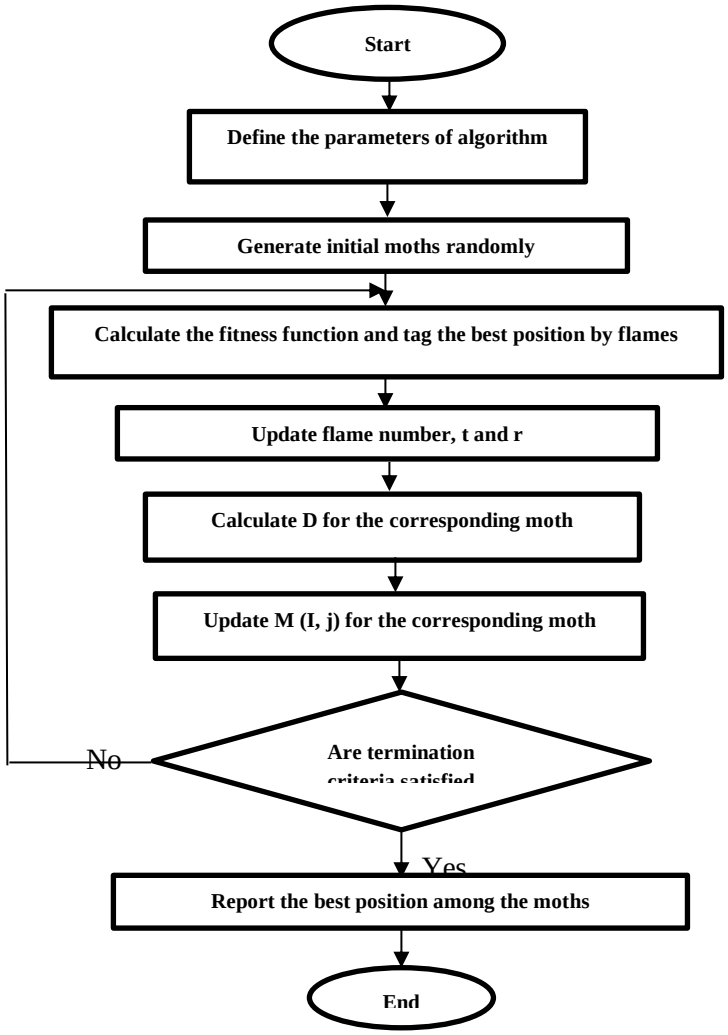


**Fig 8. MFO flowchart**

## 3.9. Loss functions

In this study, we use some loss functions which exist in MATLAB for determining the best performance model which has the highest (maximum) accuracy and the lowest (minimum) error. Table 12, shows the most common loss functions, but we will use some of them which are available in MATLAB.

Finally, due to calculated quantities for each loss function we compare their accuracy.

**Table 12. Most common loss functions**

| Error criterion formula | Error criterion |
|---|---|
| $MAE=\frac{1}{n}\sum_{i=1}^{n}\|ei\|$ | Mean Absolute Error |
| $MSE=\frac{1}{n}\sum_{i=1}^{n}ei^2$ | Mean Squared Error |
| $RMSE=\sqrt{\frac{1}{n}\sum_{i=1}^{n}ei^2}$ | Root Mean Squared Error |
| $MARE=\frac{1}{n}\sum_{i=1}^{n}\left\|\frac{ei}{ai}\right\|$ | Mean Absolute Relative Error |
| $MSRE=\frac{1}{n}\sum_{i=1}^{n}\left\|\frac{ei}{ai}\right\|$ ^2 | Mean Squared Root Error |
| $RMSRE=\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left\|\frac{ei}{ai}\right\|}$ ^2 | Root Mean Squared Relative Error |
| $MAPE=\frac{100}{n}\sum_{i=1}^{n}\left\|\frac{ei}{ai}\right\|$ | mean absolute percentage error |
| $MSPE=\frac{100}{n}\sum_{i=1}^{n}\left\|\frac{ei}{ai}\right\|$ ^2 | mean squared prediction error |

# 4. Findings and Results

The main purpose of study is forecasting daily value of Tehran Stock Exchange Index (TEPIX) with ANN and heuristic algorithms (GA as input variable selection), and meta-heuristic (HS, PSO, MFO and WOA for training ANN and selecting the number of hidden layers) from 2013 to 2018. In this study we used normal ANN model and hybrid ANN-HS, ANN-PSO, MFO and WOA. In ANN model, we used all the variables considered. However, in ANN-HS, ANN-PSO, MFO and WOA models, we reduced the input variables set to an optimal subset. First of all, as I mentioned earlier, we should normalize data in order to be usable. Because by doing so, the data is within the target range which means 0 and 1. In most cases, when you normalize data you eliminate the units of measurement for data, enabling you to more easily compare data from different places. In financial time series data, most of the time, there are some eliminated or unusable data which could affect in our research. So, we should consider some arrangements to cover them. We can solve this problem with a process which is called preprocessing. This process is different in accordance to situation. Maybe we have some outbound data which is different with other data and can affect and bias in our research. So, the data should be exact and correct. For preprocessing data, we used Alyuda NeuroIntelligence software which is user friendly and you can make ready and normalize your data without any difficulty. Also, you can use other software such as MATLAB and etc.

## 4.1. Data statistics results

Basic statistics are a very valuable information when designing a model, since they might alert to the presence of spurious data. It is a must to check for the correctness of the most important statistical measures of every single variable.

The data set contains the information for creating the predictive model. It comprises a data matrix in which columns represent variables and rows represent instances.

Variables in a data set can be of three types: The inputs will be the independent variables; the targets will be the dependent variables; the unused variables will neither be used as inputs nor as targets.

Additionally, instances can be:

Training instances, which are used to construct the model; selection instances, which are used for selecting the optimal order; testing instances, which are used to validate the functioning of the model; unused instances, which are not used at all.

### 4.1.1. Data preview table

The next table shows a preview of the data set obtained from the file ../Desktop/data/Normalized.csv. Here, the number of variables is 42, and the number of instances is 913.

### 4.1.2. Variables table

The following table depicts the names, units, descriptions and uses of all the variables in the data set. The numbers of inputs, targets and unused variables here are 42, 1, and 0, respectively.

**Table 13. Data preview table**

| | OPEN | HIGH | LOW | CLOSE | VOLUME | SMA-20 | SMA5 | SMA6 | ... | %K |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | -0.905987 | -0.908234 | -0.913041 | -0.914386 | -0.230618 | -0.958156 | -0.918144 | -0.922732 | ... | 0.782889 |
| 2 | -0.913749 | -0.913585 | -0.914315 | -0.912828 | 0.006807 | -0.955132 | -0.91153 | -0.917155 | ... | 0.796792 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 913 | 1 | 1 | 1 | 1 | 0.046246 | 1 | 1 | 1 | ... | 0.958139 |

**Table 14. Variables table**

| Row | Variables | Use |
|---|---|---|
| 1 | OPEN | Input |
| 2 | HIGH | Input |
| 3 | LOW | Input |
| 4 | CLOSE | Target |
| 5 | VOLUME | Input |
| 6 | $DIFF_{Close}$ | Input |
| 7 | SMA(5) | Input |
| 8 | SMA(6) | Input |
| 9 | SMA(10) | Input |
| 10 | SMA(20) | Input |
| 11 | TMA(5) | Input |
| 12 | TMA(6) | Input |
| 13 | TMA(10) | Input |
| 14 | TMA(20) | Input |
| 15 | EMA(5) | Input |
| 16 | EMA(6) | Input |
| 17 | EMA(10) | Input |
| 18 | EMA(20) | Input |
| 19 | $ACC_{Dist}$ | Input |
| 20 | MACD | Input |
| 21 | $Signal_{MACD}$ | Input |
| 22 | $M_{Open}$ | Input |
| 23 | $M_{High}$ | Input |
| 24 | $M_{Low}$ | Input |
| 25 | $M_{Close}$ | Input |
| 26 | $ACC_{Open}$ | Input |
| 27 | $ACC_{High}$ | Input |
| 28 | $ACC_{Low}$ | Input |
| 29 | $ACC_{Close}$ | Input |
| 30 | K% | Input |
| 31 | D% | Input |
| 32 | $SLOW_{K\%}$ | Input |
| 33 | $SLOW_{D\%}$ | Input |
| 34 | RSI | Input |

| Row | Variables | Use |
|---|---|---|
| 35 | R% | Input |
| 36 | Upper Band | Input |
| 37 | Middle Band | Input |
| 38 | Lower Band | Input |
| 39 | MP | Input |
| 40 | ROC | Input |
| 41 | Typical Price | Input |
| 42 | OBV | Input |

### 4.1.3. Instances pie chart

The following pie chart details the uses of all the instances in the data set. The total number of instances is 913. The number of training instances is 549 (60.1%), the number of selection instances is 182 (19.9%), the number of testing instances is 182 (19.9%), and the number of unused instances is 0 (0%).

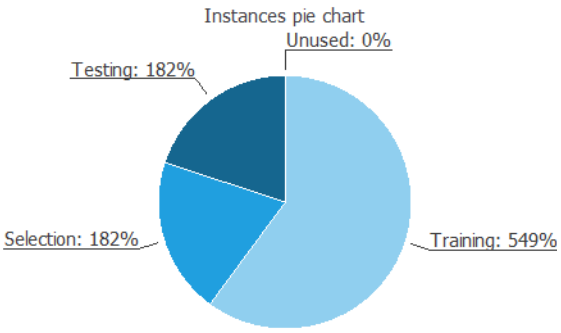


**Fig 9. Instances pie chart**

4.2. input variable selection algorithm

Model selection is applied to find a neural network with a topology that optimizes the error on new data. There are two different types of algorithms for model selection: Order selection algorithms and input selection algorithms. Order selection algorithms are used to find the optimal number of hidden neurons in the network. Inputs selection algorithms are responsible for finding the optimal subset of input variables.

One of the most important things in neural network is input variables. The number of input variables can affect the network. Too much input variables can make the network and calculations complex with a lot of time duration. On the other hand, a few input variables can cause the network stick in local solutions with low power of explanation. The genetic algorithm is used here for selecting the input variables.

**Table 15. some information and statistics about input selection algorithm**

| | Description | Value |
|---|---|---|
| Trials number | Number of trials for each neural network. | 1 |
| Tolerance | Tolerance for the selection error in the trainings of the algorithm. | 0.01 |
| Population size | Size of the population of each generation. | 10 |
| Initialization method | Initialization method used in the algorithm. | Random |
| Fitness assignment method | Fitness assignment method used in the algorithm. | RankBased |
| Crossover method | Crossover method used in the algorithm. | Uniform |
| Elitism size | Number of individuals which will always be selected for recombination. | 2 |
| Crossover first point | First point used in the OnePoint and TwoPoint crossover method. If it is 0 the algorithm selects a random point for each pair of offsprings. | 0 |
| Crossover second point | Second point used in the TwoPoint crossover method. If it is 0 the algorithm selects a random point for each pair of offsprings. | 0 |
| Selective pressure | Second point used in the TwoPoint crossover method. If it is 0 the algorithm selects a random point for each pair of offsprings. | 1.5 |
| Mutation rate | This is a parameter of the mutation operator. | 0.05 |
| Selection loss goal | Goal value for the selection error. | 0 |
| Maximum Generations number | Maximum number of generations to perform the algorithm. | 100 |
| Maximum time | Maximum time for the inputs selection algorithm. | 3600 |
| Plot training error history | Plot a graph with the optimum training error of each generation. | True |
| Plot selection error history | Plot a graph with the optimum selection error of each generation. | True |
| Plot generation mean history | Plot a graph with the mean of the selection error of each generation. | True |
| Plot generation standard deviation history | Plot a graph with the standard deviation of the selection error of each generation. | False |

This is a stochastic method based on the mechanics of natural genetics and biological evolution. We implement a quite general genetic algorithm with fitness assignment, selection, recombination and mutation operators.

The order selection algorithm chosen for this application is incremental order. This method starts with the minimum order and adds a given number of perceptron in each iteration.

### 4.2.1. selection of related input variables using GA

Some data sets have inputs that are redundant and it affects the error of the neural network. The inputs selection is used to find the optimal subset of inputs for the best error of the model. The genetic algorithm is used here as inputs selection algorithm in the model selection.

#### 4.2.1.1. Genetic Algorithm error plot

The next chart shows the error history for the different subsets during the genetic algorithm inputs selection process. The blue line represents the training error, its initial value is 0.0426695, and the final value after 100 generations is 0.0426695. The orange line symbolizes the selection error, its initial value is 0.0423943, and the final value after 100 generations is 0.0423943.

#### 4.2.1.2. Genetic Algorithm generation mean plot

The next chart shows the history of the mean of the selection error in each generation during the genetic algorithm inputs selection process. The initial value is 1.98419, and the final value after 100 generations is 1.37.

**Table 16. Some information and statistics about order selection algorithm**

| | Description | Value |
|---|---|---|
| Minimum order | Number of minimum hidden perceptrons to be evaluated. | 1 |
| Maximum order | Number of maximum hidden perceptrons to be evaluated. | 10 |
| Step | Number of hidden perceptrons added in each iteration. | 1 |
| Trials number | Number of trials for each neural network. | 3 |
| Tolerance | Tolerance for the selection error in the trainings of the algorithm. | 0.01 |
| Selection loss goal | Goal value for the selection error. | 0 |
| Maximum selection failures | Maximum number of iterations at which the selection error increases. | 5 |
| Maximum iterations number | Maximum number of iterations to perform the algorithm. | 1000 |
| Maximum time | Maximum time for the order selection algorithm. | 3600 |
| Plot training error history | Plot a graph with the training error of each iteration. | true |
| Plot selection error history | Plot a graph with the selection error of each iteration. | true |

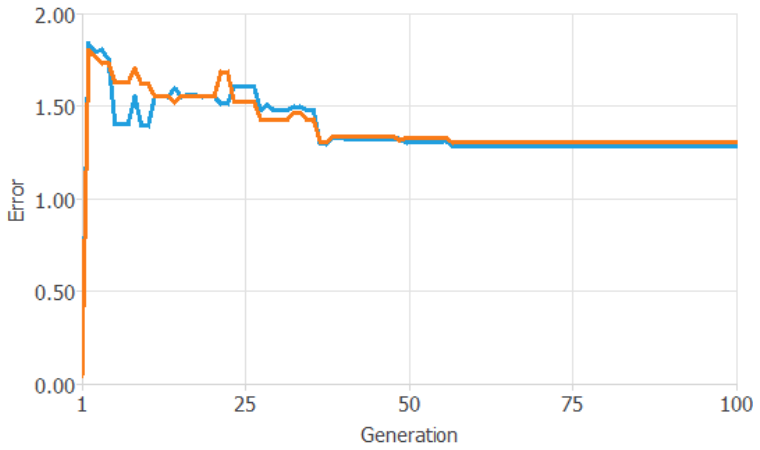


**Fig 10. Genetic Algorithm error plot**

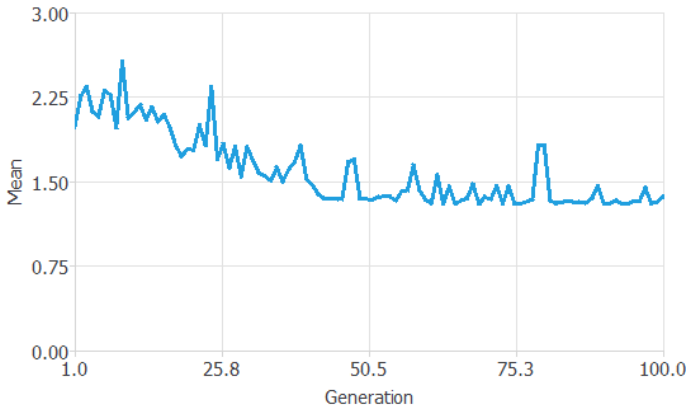


**Fig 11. Genetic Algorithm generation mean plot**

### 4.2.1.3. Genetic Algorithm results

The next table shows the inputs selection results by the genetic algorithm. They include some final states from the neural network, the error functional and the inputs selection algorithm.

**Table 17. Genetic Algorithm results**

| | Value |
|---|---|
| Optimal number of inputs | 11 |
| Optimum training error | 0.0426695 |
| Optimum selection error | 0.0423943 |
| Generations number | 100 |
| Elapsed time | 07:25 |

### 4.2.1.4. Final architecture

A graphical representation of the resulted deep architecture is depicted next. It contains a scaling layer, a neural network and an un-scaling layer. The yellow circles represent scaling neurons, the blue circles perceptron neurons and the red circles un-scaling neurons. The number of inputs is 11, and the number of outputs is 1. The complexity, represented by the numbers of hidden neurons, is 3.

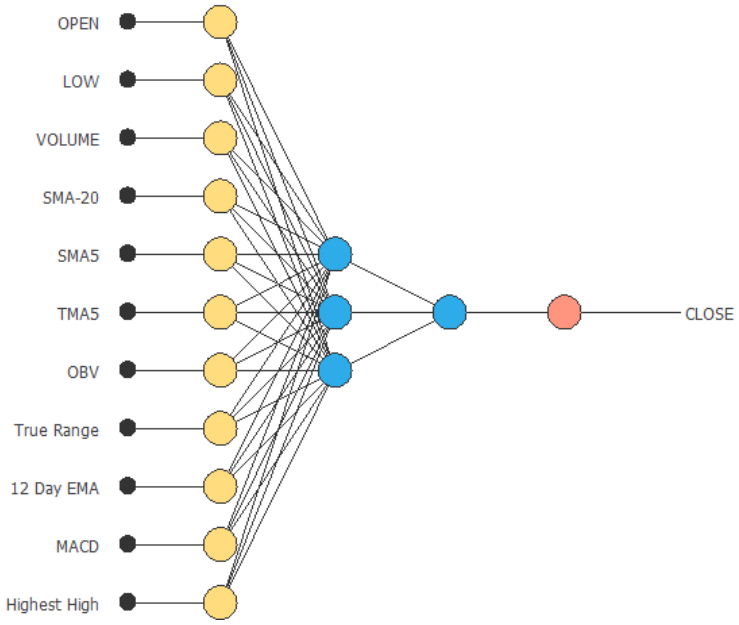


**Fig 12. Final architecture**

## 4.3. Artificial Neural Network (ANN)

After selection of best and most related input variables, now, it's time to get right into training network with ANN. One of the most important part in ANN is find the fittest number of hidden layers. There are different methods for this purpose. One of the methods which is common and usual is trial & error which the network tests the different layers with considered training and validating error. Finally, due to high power of explanation and lower error, the fittest network design and architecture will be find. In accordance to this information, we select 11-13-1 which means that 11 input layers, 13 hidden layers and 1 output layer with below information.

As you see, the best network contains highest $R^2$ and lowest AIC and error. On the other hand, we should consider overestimation which usually happen in ANN and training it which we don't have this problem.

You can see comprehensive results below:

For training network, we used Levenberg-Marquardt algorithm and you can see more details in below figures 14:

**Table 18. Architecture search**

| ID | Architecture | # of Weights | Fitness | Train Error | Validation Error | Test Error | AIC | Correlation | R-Squared |
|---|---|---|---|---|---|---|---|---|---|
| 1 | [11-2-1] | 27 | 0.985835 | 0.040755 | 0.041024 | 0.040079 | -5714.86779 | 0.994548 | 0.985835 |
| 2 | [11-28-1] | 365 | 0.999548 | 0.007977 | 0.009396 | 0.009024 | -6019.092448 | 0.999777 | 0.999548 |
| 3 | [11-18-1] | 235 | 0.999575 | 0.007216 | 0.008482 | 0.007753 | -6339.373773 | 0.999788 | 0.999575 |
| 4 | [11-11-1] | 144 | 0.99951 | 0.007706 | 0.008094 | 0.007157 | -6481.839496 | 0.999761 | 0.99951 |
| 5 | [11-24-1] | 313 | 0.999535 | 0.007712 | 0.008211 | 0.008168 | -6143.399969 | 0.999769 | 0.999535 |
| 6 | [11-15-1] | 196 | 0.999586 | 0.007395 | 0.008379 | 0.007625 | -6402.626605 | 0.999797 | 0.999586 |
| **7** | **[11-13-1]** | **170** | **0.999673** | **0.00638** | **0.006715** | **0.006939** | **-6543.365491** | **0.999843** | **0.999673** |
| 8 | [11-14-1] | 183 | 0.99954 | 0.007729 | 0.008113 | 0.008109 | -6402.055492 | 0.999772 | 0.99954 |
| 9 | [11-12-1] | 157 | 0.99952 | 0.008137 | 0.008191 | 0.008426 | -6423.132125 | 0.999763 | 0.99952 |

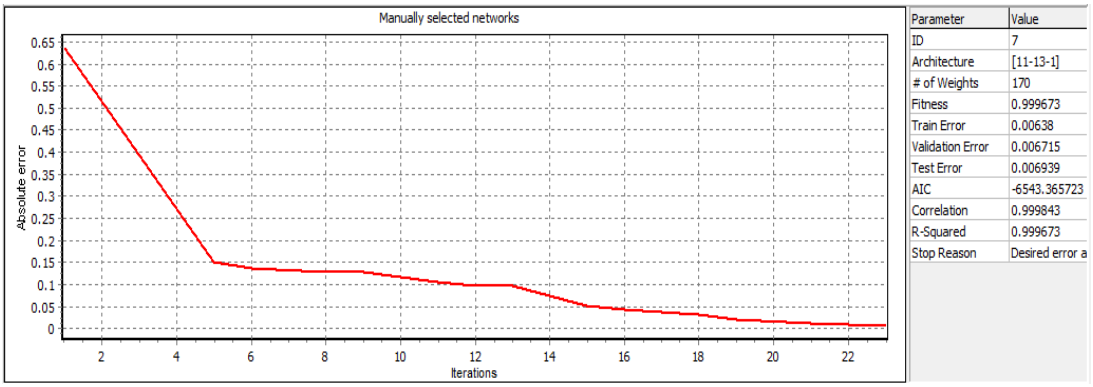


**Fig 13. manually selected networks**

**Table 19. Network properties**

[11-13-1] architecture selected for training
Hidden layer activation function: Logistic
Output parameters:
Close
Error function: sum-of-squares
Activation function: Logistic
Search parameters:
Search method: Heuristic search
Fitness criteria: R-squared
Number of retrains: 1
Hidden unit range:
Layer 1: from 2 to 28, search accuracy 2
Network architectures verified
[11-13-1] architecture had the best fitness
Verified architectures:
[11-2-1] fitness: 0.986445
[11-28-1] fitness: 0.999537

[11-18-1] fitness: 0.999551
[11-11-1] fitness: 0.989877
[11-24-1] fitness: 0.999522
[11-15-1] fitness: 0.999546
[11-21-1] fitness: 0.999531
[11-19-1] fitness: 0.99957
[11-20-1] fitness: 0.999524

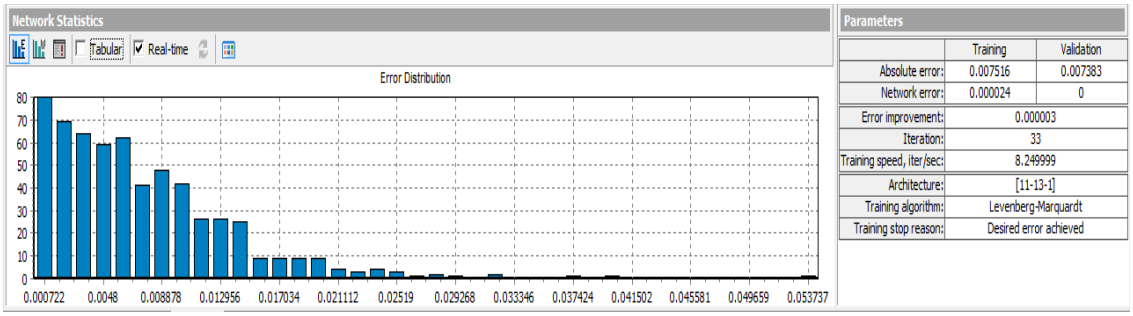


**Fig 14. Training network**

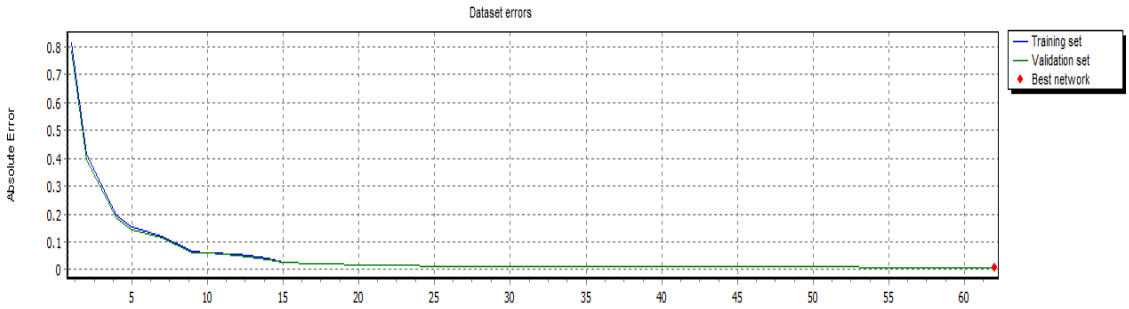


**Fig 15. Dataset errors**

**Table 20. Errors table**

| | Training | Selection | Testing |
|---|---|---|---|
| Sum squared error | 3.25717 | 1.26383 | 0.965757 |
| Mean squared error | 0.00593291 | 0.00694411 | 0.00530636 |
| Root mean squared error | 0.0770254 | 0.0833313 | 0.0728447 |
| Normalized squared error | 0.0284121 | 0.0329085 | 0.0236746 |
| Minkowski error | 10.2996 | 3.76554 | 3.21373 |

The procedure used to carry out the learning process is called training (or learning) strategy. The training strategy is applied to the neural network in order to obtain the best possible loss. The type of training is determined by the way in which the adjustment of the parameters in the neural network takes place.
The Levenberg-Marquardt algorithm is used here for training, which was designed to approach second-order training speed without having to compute the Hessian matrix. The Levenberg-Marquardt algorithm can only be applied when the loss functional has the form of a sum of squares (as the sum squared error, the mean squared error or the normalized squared error).
The next table measures all the errors of the model. it takes in account every used instance and evaluate the model for each use. It shows all the errors of the data for each use of them.

Below table, shows the actual versus output. You should continue training until the differences between actual and output become very little, so that they match.

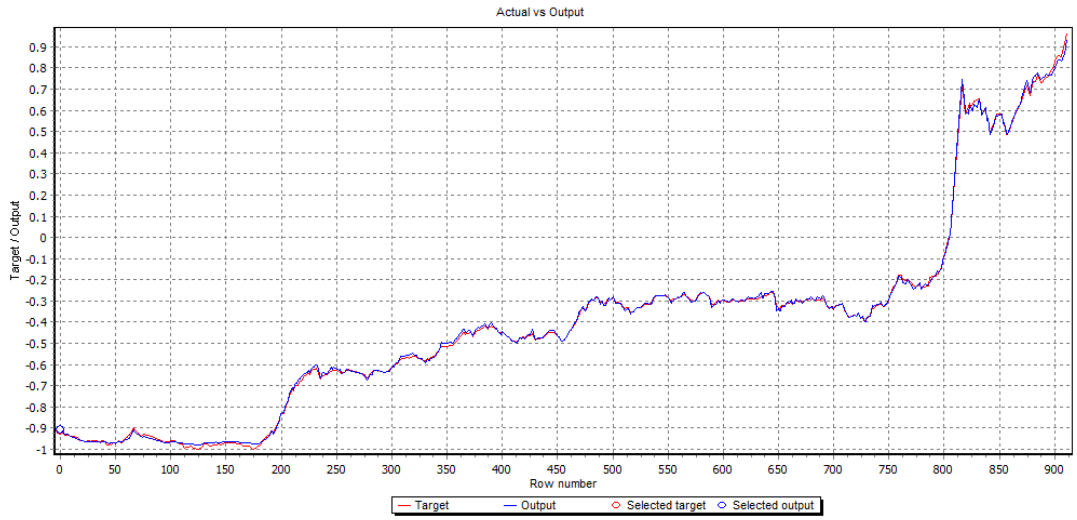


**Fig 16. Actual vs output**

## 4.4. Hybrid ANN with Particle Swarm Optimization (PSO) algorithm

After determining of network architecture and network structure, we can train network with different algorithm. One of the most common and applicable algorithm is Particle Swarm Optimization (PSO). Like Mahamad Nabab Alam in his article " Training Artificial Neural Network using Particle Swarm Optimization", seven steps have been used to train ANN using PSO:

1) Collect data
2) Create the network
3) Configure the network
4) Initialize the weight and biases
5) Train the network using PSO
6) Validate the network
7) Use the network

As we want to prediction stock price which means closing price, first of all, we should create fitness function. I do this as a M file in MATLAB with setting considered parameters as I mentioned in part (4.5).

Because we want to predict closing price, So, we have input and output argument. As I mentioned and calculated earlier, we have 11 input variables which are the most important and related variables and because we want to predict closing price, our target is closing price too. Firstly, we initialize the PSO which contains population and velocity with initial pbest and initial gbest. We select a number for $C_1$ and $C_2$ considered iteration. For instance, we have been used 1000 max iteration. Continuously, we should update parameters in order to meeting the goal, which is minimum error lost function which mean MSE. In this research, after 1000 iterations, our fval which means final value and obfuva which is objective function has been became 0.0175 with 0.0175 error function. You can see the regression with $R^2$ in below table:

In table 21, there are some information about training, validation and testing for hybrid ANN-PSO.

## 4.5. Hybrid ANN with Harmony Search (HS) algorithm

Like other algorithms such as GA, PSO … we used several steps for training network and solving problem. The network structure is Feed Forward ANN (FFANN). First, the number of iteration is 1000 and in order to achieving better results, we increase it to 5000.Finally, the result after 5000 iterations is as table 22:

After training network and calculating parameters, we calculated $x_{best}$ and $f_{best}$ which are equal to 2.4941.

## 4.6. WOA

Like MFO algorithm, first of all, let's setting the parameters such as the following table:

Below figures show the fitness function and convergence during iterations:

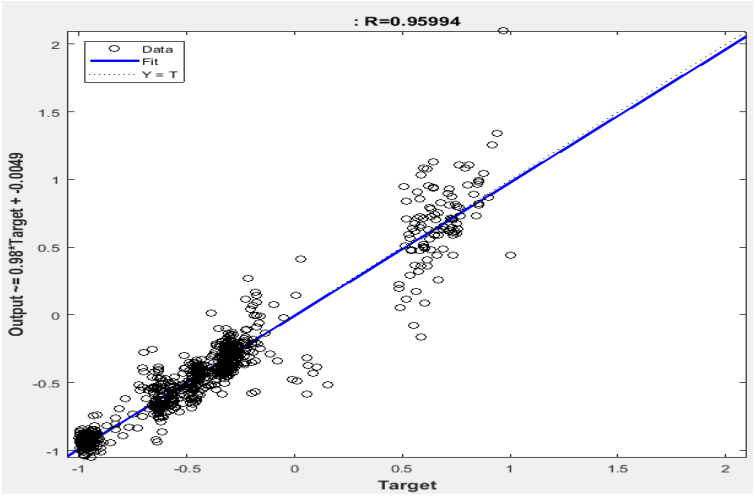


**Fig 17. Regression plot of the trained ANN**

**Table 21. Hybrid ANN-PSO**

| Symbol | MSE | RMSE | MAE | MAPE | MSRE | MARE | RMSRE | RMSPE | $R^2$ | Best Particle |
|---|---|---|---|---|---|---|---|---|---|---|
| TEPIX | 3.06e-05 | 0.0005 | 0.005 | 0.0003 | 0.0018 | 3.17e-06 | 1.4e-05 | 0.001 | 0.989 | 8 |

**Table 22. Hybrid ANN-HS**

| Symbol | MSE | RMSE | MAE | MAPE | MSRE | MARE | RMSRE | RMSPE | $R^2$ |
|---|---|---|---|---|---|---|---|---|---|
| TEPIX | 5.51e-07 | 0.0002 | 5.49e-15 | 6.88e-11 | 8.98e-07 | 6.86e-12 | 0.00002 | 0.002 | 0.9981 |

**Table 23. WOA parameters**

| | |
|---|---|
| Search agents number | 30 |
| Maximum number of iterations | 500 |
| Upper bound | 100 |
| Lower bound | -100 |
| Best score | 1.6828e-78 |
| dim | 12 |

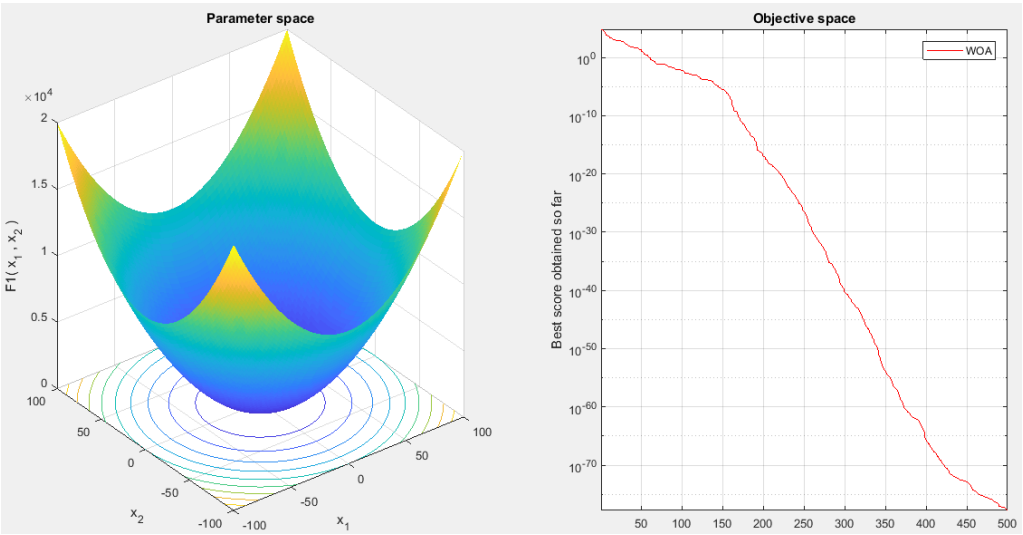


**Fig 18. Test Function and Convergence Curve**

## 4.7. MFO

First of all, let's setting the parameters such as the following table: Below figures show the fitness function and convergence during iterations:
You can see a clear decrease in each iteration until the best score has been obtained means 8.0081e-32.

## 4.8. Comparing results

In this section, in order to showing our contribution, different studies have been compared to each other and the results shows that our suggested model had better performance along with higher predictability.
According table 25, the best results are obtained in the whale optimization algorithm (WOA) and almost in MFO algorithms with the lowest loss functions and the highest R-squared.

**Table 24. MFO parameters**

| | |
|---|---|
| Search agents number | 30 |
| Maximum number of iterations | 1000 |
| Upper bound | 100 |
| Lower bound | -100 |
| Best score | 8.0081e-32 |
| dim | 12 |

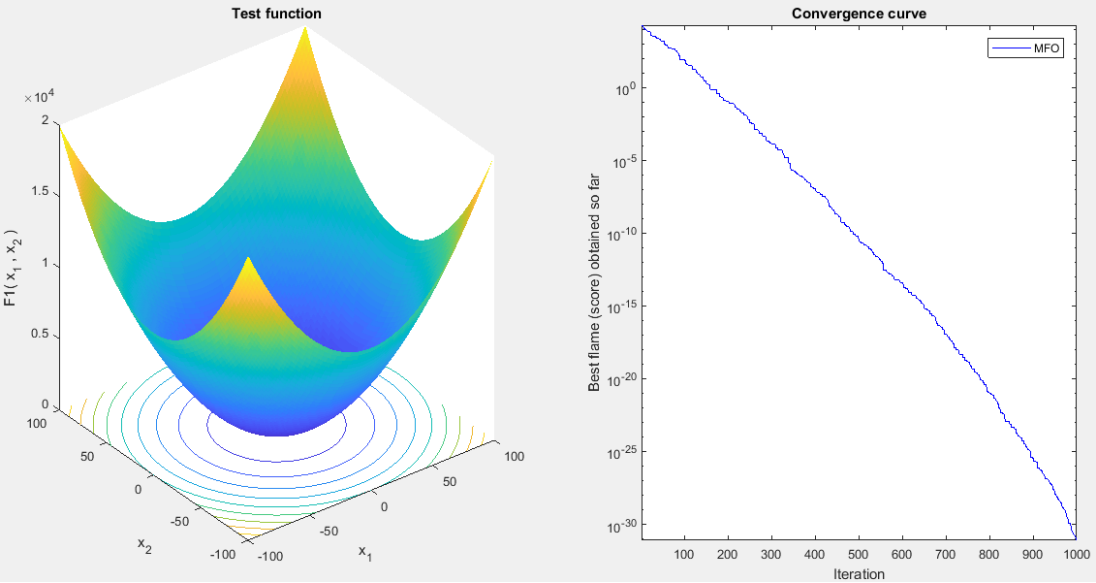


**Fig 19. Test Function and Convergence Curve**

**Table25. Comparative Study**

| Author & Date | Proposed Approaches | Data Type | MSE | MAE | $R^2$ |
|---|---|---|---|---|---|
| Ghasemiyeh et al. 2017 | GA-ANN | Train | 0.0074 | 0.0584 | 0.9866 |
| | | Test | 0.0079 | 0.0585 | |
| | PSO-ANN | Train | 0.0013 | 0.0253 | 0.9895 |
| | | Test | 0.0014 | 0.0260 | |
| | ICS-ANN | Train | 0.0076 | 0.0720 | 0.9972 |
| | | Test | 0.0068 | 0.0694 | |
| Mojtaba Sedighi et al. 2019 | ARIMA-SVM | Final Outcome | 1.0042 | 0.0142 | 0.9969 |
| | SVM-RF | Final Outcome | 0.000295 | 0.0245 | 0.9966 |

| Author & Date | Proposed Approaches | Data Type | MSE | MAE | $R^2$ |
|---|---|---|---|---|---|
| | ANFIS-SVM | Final Outcome | 3.5849 | 0.0117 | 0.9995 |
| | FA-MSVR | Final Outcome | 0.0014 | 0.0130 | 0.9986 |
| Safa, M., & Panahian, H. 2019 | HS-ANN | Final Outcome | 0.02776 | 0.05177 | 0.9641 |
| Emamverdi et al. 2016 | ANN | Final Outcome | 0.00030 | 0.0174 | 0.9893 |
| | ARIMA | Final Outcome | 0.00042 | 0.0162 | 0.9795 |
| Zheng, T. et al. 2013 | Wavelet neural networks | Final Outcome | 0.00510 | 6.742e-04 | 0.9877 |
| Dong, G. et al. 2013 | one-step ahead and multi-step ahead predictions | Final Outcome | 0.0043 | 0.1043 | 0.9012 |
| L. Wang, et al. 2016 | Delayed Neural Network (DNN) | Final Outcome | 1.60e-03 | 1.00e-07 | 0.9955 |
| Sin, E., & Wang, L. 2017 | Ensembles of Neural Network | Final Outcome | 2.05e-05 | 2.045e-09 | 0.9963 |
| Current research | ANN | Train | 0.005932 | 0.036408 | 0.9996 |
| | | Test | 0.005306 | 0.00621 | |
| | GA-ANN | Train | 0.04266 | 0.0130 | 0.9984 |
| | | Test | 0.00045 | 0.000532 | |
| | PSO-ANN | Train | 0.0000306 | 0.005 | 0.989 |
| | | Test | 0.0175 | 0.00216 | |
| | HS-ANN | Train | 5.51E-07 | 549E-15 | 0.9981 |
| | | Test | 0.000061402 | 0.00042 | |
| | WOA | Train | 1.6828E-78 | 3.52E-80 | 0.9995 |
| | | Test | 2.23E-65 | 1.75E-67 | |
| | MFO | Train | 8.0081E-32 | 7.45E-35 | 0.9996 |
| | | Test | 9.33E-28 | 8.53E-31 | |

## 5. CONCLUSION

In this article, we used neural network as a prediction method for prediction of TEPIX. Then, we used important technical indicators such as SMA, EMA, TMA and etc., as input variables. So, we selected the most important one with using GA. after that, we trained the network by using four meta-heuristic algorithms such as HS, PSO, WOA and MFO. After obtaining optimized indicators and weights by GA, we compute different loss functions for each algorithms. Among algorithms, WOA has the lowest training and testing error. I should note that ANN has the highest forecasting error. For evaluating performance of the model, we should test the model with new data which it is called testing performance and is the right indicator for forecasting performance. As you can see, GA has good condition in terms of training and testing error. But we used GA as feature selection and our focus is on other algorithms. The main advantages of using MFO and WOA is as follow:

- Speed up calculations
- Reduce model complexity

- Increase the network accuracy
- Ease of using models

Finally, WOA, MFO and HS-ANN have the lowest error respectively.

Although, hybrid models show good results, but there are some limitations:

First, hidden layer in network is 10 and is fix due to the default settings of MATLAB software. The performance of the model can be better by changing the number of hidden layers.

Second, fix activation function because this can affect the network performance.

Our suggestion for future research is focus on different parameters such as the number of hidden layer, activation function and the other models of HS such as HIS and etc. in addition, using different parameters of GA such as crossover and mutation rate can be interesting. One of the other offers is training network with other new metaheuristic algorithm such as bat algorithm and etc.

## Declarations

Funding: this research paper is supported by Khatam University.

Conflict of interest / Competing interests: The authors have no conflicts of interest to declare that are relevant to the content of this article.

- Funding: This study was funded by Khatam University
- Employment: Any organization or employment won't gain or loss financially through publication of this manuscript.
- Financial interests: The authors declare they have no financial interests

Ethics approval: Not applicable

Consent to participate: Not applicable

Consent for publication: Not applicable

Availability of data and material: The data was obtained through two sites which are called TSETMC and CODAL sites respectively Which are at the following address:

- http://tsetmc.ir/
- https://www.codal.ir/

On the other hand, there is a financial data software which is called TSECLIENT 2.0 and you can download data easily according to symbol, date, different variables and etc.

Code availability: All of the necessary pseudo-codes are mentioned in the manuscript.

Author's contribution: All of the authors have the same contribution.

## References

1) Abdullah, M. A., Ab Rashid, M. F. F., Ghazali, Z., & Rose, A. N. M. (2019). A case study of energy efficient assembly sequence planning problem. In IOP Conference Series: Materials Science and Engineering (Vol. 469, No. 1, p. 012013). IOP Publishing.
2) Ahmed, J., Jafri, M., Ahmad, J., & Khan, M. I. (2005). Design and implementation of a neural network for real-time object tracking. Paper presented at the Proceedings of machine vision and pattern recognition in 4th world enformatika conference, Istanbul.
3) Ahmed, M. K., Wajiga, G. M., Blamah, N. V., & Modi, B. (2019). Stock Market Forecasting Using ant Colony Optimization Based Algorithm. American Journal of Mathematical and Computer Modelling, 4(3), 52-57.
4) Bhowmik, P. (2019). Research Study on basic Understanding of Artificial Neural Networks. Global Journal of Computer Science and Technology.
5) Caginalp, G., & DeSantis, M. (2011). Nonlinearity in the dynamics of financial markets. Nonlinear Analysis: Real World Applications, 12(2), 1140-1151.
6) Chandana, P. H. (2019). A Survey on Soft Computing Techniques and Applications.
7) Chong, E., Han, C., & Park, F. C. (2017). Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. Expert Systems with Applications, 83, 187-205.
8) Chou, J.-S., & Nguyen, T.-K. (2018). Forward Forecast of Stock Price Using Sliding-Window Metaheuristic-Optimized Machine-Learning Regression. IEEE Transactions on Industrial Informatics, 14(7), 3132-3142.
9) Dash, R., & Dash, P. K. (2015, October). A comparative study of radial basis function network with different basis functions for stock trend prediction. In 2015 IEEE Power, Communication and Information Technology Conference (PCITC) (pp. 430-435). IEEE.
10) Davallou, M., & Azizi, N. (2017). The Investigation of Information Risk Pricing; Evidence from Adjusted Probability of Informed Trading Measure. Financial Research Journal, 19(3), 415-438.
11) de Oliveira, F. A., Nobre, C. N., & Zárate, L. E. (2013). Applying Artificial Neural Networks to prediction of stock price and improvement of the directional prediction index–Case study of PETR4, Petrobras, Brazil. Expert Systems with Applications, 40(18), 7596-7606.

12) de Rubio, J. J. (2020). Stability analysis of the modified Levenberg-Marquardt algorithm for the artificial neural network training. IEEE Transactions on Neural Networks and Learning Systems.
13) Dong, G., Fataliyev, K., & Wang, L. (2013, December). One-step and multi-step ahead stock prediction using backpropagation neural networks. In 2013 9th International Conference on Information, Communications & Signal Processing (pp. 1-5). IEEE.
14) Dubey, M., Kumar, V., Kaur, M., & Dao, T. P. (2021). A systematic review on harmony search algorithm: theory, literature, and applications. Mathematical Problems in Engineering, 2021.
15) Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. Paper presented at the MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science.
16) Efron, B., & Tibshirani, R. J. (1994). An introduction to the bootstrap: CRC press.
17) Emamverdi, G., Karimi, M. S., Khakie, S., & Karimi, M. (2016). Forecasting The Total Index of Tehran Stock Exchange. Financial Studies, 20(1).
18) Faris, H., Aljarah, I., & Mirjalili, S. (2016). Training feedforward neural networks using multi-verse optimizer for binary classification problems. Applied Intelligence, 45(2), 322-332.
19) Ftiti, Z., Guesmi, K., & Abid, I. (2016). Oil price and stock market co-movement: What can we learn from time-scale approaches? International review of financial analysis, 46, 266-280.
20) Gao, T., & Chai, Y. (2018). Improving stock closing price prediction using recurrent neural network and technical indicators. Neural computation, 30(10), 2833-2854.
21) Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search. simulation, 76(2), 60-68.
22) Ghanbari, M., & Arian, H. (2019). Forecasting Stock Market with Support Vector Regression and Butterfly Optimization Algorithm. arXiv preprint arXiv:1905.11462.
23) Gharehchopogh, F. S., & Gholizadeh, H. (2019). A comprehensive survey: Whale Optimization Algorithm and its applications. Swarm and Evolutionary Computation, 48, 1-24.
24) Ghasemiyeh, R., Moghdani, R., & Sana, S. S. (2017). A hybrid artificial neural network with metaheuristic algorithms for predicting stock price. Cybernetics and Systems, 48(4), 365-392.
25) Göçken, M., Özçalıcı, M., Boru, A., & Dosdoğru, A. T. (2016). Integrating metaheuristics and artificial neural networks for improved stock price prediction. Expert Systems with Applications, 44, 320-331.
26) Guerrien, B., & Gun, O. (2011). Efficient Market Hypothesis: What are we talking about? real-world economics review, 56(11), 19-30.
27) Hadavandi, E., Ghanbari, A., & Abbasian-Naghneh, S. (2010). Developing an evolutionary neural network model for stock index forecasting. Paper presented at the International Conference on Intelligent Computing.
28) Haider, A., & Hanif, M. N. (2009). Inflation forecasting in Pakistan using artificial neural networks. Pakistan economic and social review, 123-138.
29) Hassanin, M. F., Shoeb, A. M., & Hassanien, A. E. (2016). Grey wolf optimizer-based back-propagation neural network algorithm. Paper presented at the 2016 12th International Computer Engineering Conference (ICENCO).
30) Idris, M. A., Saiang, D., & Nordlund, E. (2015). Stochastic assessment of pillar stability at Laisvall mine using Artificial Neural Network. Tunnelling and Underground Space Technology, 49, 307-319.
31) Ismael, M., Heikal, M., & Baharom, M. (2013). Characteristics of compressed natural gas jet and jet-wall impingement using the Schlieren imaging technique. Paper presented at the IOP Conference Series: Earth and Environmental Science.
32) Joe, H. Y., Ruiz Estrada, M. A., & Yap, S. F. (2016). The Evolution of Complex Systems Theory and the Advancement of Econophysics Methods in the Study of Stock Markets Crashes. Labuan Bulletin of International Business & Finance (LBIBf), 14(1).
33) Jóhannsson, Ó. S. (2020). Forecasting the Icelandic stock market using a neural network (Doctoral dissertation).
34) Krauss, C., Do, X. A., & Huck, N. (2017). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. European Journal of Operational Research, 259(2), 689-702.
35) Lv, D., Yuan, S., Li, M., & Xiang, Y. (2019). An Empirical Study of Machine Learning Algorithms for Stock Daily Trading Strategy. Mathematical Problems in Engineering, 2019.
36) L. Wang, F. F. Chan, Y. Wang and Q. Chang, "Predicting public housing prices using delayed neural networks," 2016 IEEE Region 10 Conference (TENCON), 2016, pp. 3589-3592, doi: 10.1109/TENCON.2016.7848726.
37) Mirjalili, S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowledge-based systems, 89, 228-249.

38) Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. Advances in engineering software, 95, 51-67.
39) Naseer, M., & Bin Tariq, D. (2015). The efficient market hypothesis: A critical review of the literature. The IUP Journal of Financial Risk Management, 12(4), 48-63.
40) Panahian, H. (2018). P/E Modeling and Prediction of Firms Listed on the Tehran Stock Exchange; a New Approach to Harmony Search Algorithm and Neural Network Hybridization. Iranian Journal of Management Studies, 11(4), 765-786.
41) Prasanna, S., & Ezhilmaran, D. (2013). An analysis on stock market prediction using data mining techniques. International Journal of Computer Science & Engineering Technology (IJCSET), 4(3), 49-51.
42) Preethi, G., & Santhi, B. (2012). Stock market forecasting techniques: A survey. Journal of Theoretical and Applied Information Technology, 46, 24-30.
43) Rajesh and et al (2019). Stock trend prediction using Ensemble learning techniques in python International Journal of Innovative Technology and Exploring Engineering (IJITEE), 8(5).
44) Rana, N., Latiff, M. S. A., Abdulhamid, S. I. M., & Chiroma, H. (2020). Whale optimization algorithm: a systematic review of contemporary applications, modifications and developments. Neural Computing and Applications, 1-33.
45) Rather, A. M., Agarwal, A., & Sastry, V. (2015). Recurrent neural network and a hybrid model for prediction of stock returns. Expert Systems with Applications, 42(6), 3234-3241.
46) Rather, A. M., Sastry, V., & Agarwal, A. (2017). Stock market prediction and Portfolio selection models: a survey. Opsearch, 54(3), 558-579.
47) Ravichandran, K., Thirunavukarasu, P., Nallaswamy, R., & Babu, R. (2005). Estimation of return on investment in share market through ANN. Journal of Theoretical and Applied Information Technology, 3.
48) Safa, M., & Panahian, H. (2019). Ranking P/E Predictor Factors In Tehran Stock Exchange With Using The Harmony Search Meta Heuristic Algorithm.
49) Samadisoufi, R., & Noraei, M. The eminent effect of the business intelligence on the inner business processes and the role of the culture of the analytic decision making related (the subject of the study: Asia insurance of Zanjan province). Cumhuriyet Üniversitesi Fen-Edebiyat Fakültesi Fen Bilimleri Dergisi, 36(3), 2971-2981.
50) Sedighi, M., Jahangirnia, H., Gharakhani, M., & Farahani Fard, S. (2019). A Novel Hybrid Model for Stock Price Forecasting Based on Metaheuristics and Support Vector Machine. Data, 4(2), 75.
51) Sengupta, S., Basak, S., & Peters, R. A. (2019). Particle Swarm Optimization: A survey of historical and recent developments with hybridization perspectives. Machine Learning and Knowledge Extraction, 1(1), 157-191.
52) Sezer, O. B., Ozbayoglu, M., & Dogdu, E. (2017). A Deep neural-network based stock trading system based on evolutionary optimized technical analysis parameters. Procedia computer science, 114, 473-480.
53) Sheela, K. G., & Deepa, S. N. (2013). Review on methods to fix number of hidden neurons in neural networks. Mathematical Problems in Engineering, 2013.
54) Shehab, M., Abualigah, L., Al Hamad, H., Alabool, H., Alshinwan, M., & Khasawneh, A. M. (2020). Moth–flame optimization algorithm: variants and applications. Neural Computing and Applications, 32(14), 9859-9884.
55) Siami-Namini, S., & Namin, A. S. (2018). Forecasting economics and financial time series: ARIMA vs. LSTM. arXiv preprint arXiv:1803.06386.
56) Sin, E., & Wang, L. (2017, July). Bitcoin price prediction using ensembles of neural networks. In 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD) (pp. 666-671). IEEE.
57) Wang, T., Gao, H., & Qiu, J. (2015). A combined adaptive neural network and nonlinear model predictive control for multirate networked industrial process control. IEEE Transactions on Neural Networks and Learning Systems, 27(2), 416-425.
58) Wei, L.-Y., & Cheng, C.-H. (2012). A hybrid recurrent neural networks model based on synthesis features to forecast the Taiwan stock market. Int. J. Innov. Comput. Inf. Control, 8(8), 5559-5571.
59) Yang, J.-S., Nam, H.-J., Seo, M., Han, S. K., Choi, Y., Nam, H. G., . . . Kim, S. (2011). OASIS: online application for the survival analysis of lifespan assays performed in aging research. PloS one, 6(8), e23525.
60) Zhang, G. P., Patuwo, B. E., & Hu, M. Y. (2001). A simulation study of artificial neural networks for nonlinear time-series forecasting. Computers & Operations Research, 28(4), 381-396.
61) Zhang, J., Cui, S., Xu, Y., Li, Q., & Li, T. (2018). A novel data-driven stock price trend prediction system. Expert Systems with Applications, 97, 60-69.
62) Zheng, T., Fataliyev, K., & Wang, L. (2013, May). Wavelet neural networks for stock trading. In

Independent Component Analyses, Compressive Sampling, Wavelets, Neural Net, Biosystems, and Nanoengineering XI (Vol. 8750, p. 87500A). International Society for Optics and Photonics.