IRAQI Academic Scientific Journals

University of Kirkuk Journal For Administrative and Economic Science (2025) 15 (2): 427-441

UKJAES

University of Kirkuk Journal For Administrative and Economic Science

ISSN:2222-2995 E-ISSN:3079-3521
University of Kirkuk Journal For Administrative and Economic Science

Muhammad Gashaw Qadir & Hussein Mohammad Mahmood Faqe. The Effect of Applying Fuzzy Logic on Improving Budget Estimates Based on Time-Driven Activities - Case Study. *University of Kirkuk Journal For Administrative and Economic Science* (2025) 15 (2):1-19.

Impact of Data Fuzzification Feedforward Neural Network Performance: A Comparative application

Gashaw Qadir Muhammad [1], Mohammad Mahmood Faqe Hussein[2]

[(1,2)] *University of Sulaimani, College of Administration and Economics, Sulaymaniyah, Iraq*

Gashaw.muhammadameen@univsul.edu.iq [1]
Mohammad.faqe@university.edu.iq [2]

Abstract: This paper studies the effectiveness of feed forward neural networks (FFNNs) in comprehending fuzzy data, specifically analyzing the Brent crude oil price dataset from January 1, 2018, to December 31, 2021. Crude oil prices exhibit intrinsic unpredictability, characterized by uncertainty and imprecision, rendering them suitable for fuzzy data analytics. FFNNs, recognized for their ability to model complex and nonlinear associations, are employed to predict and analyze price fluctuations in this ambiguous domain. The research evaluates the performance of FFNN against conventional statistical and machine learning models, incorporating essential assessment metrics such as prediction accuracy, mean squared error, and resilience to noisy or absent data. The results of this study indicate that the application of fuzzy pre-processing in combining with FFNNs decidedly improves model performance. Before employing fuzzy methodologies, the mean squared error (MSE) was recorded at 1.2959. Nonetheless, the incorporation of fuzzy pre-processing into the FFNN framework reduced the MSE to 1.0983, illustrating the enhanced ability of the combined strategy to address uncertainty and imprecision in the data. This augmentation illustrates the efficacy of fuzzy-enhanced neural networks in generating more precise forecasts for complex, volatile variables such as Brent crude oil prices.

Keywords: Fuzzy Data Analytics, FFNN, Fuzzy Preprocessing, R2, MSE.

تأثير ضبابية البيانات على أداء الشبكات العصبية ذات التغذية الامامية: دراسة مقارنة

الباحثة: كشاو قادر محمد[١]، أ.م.د. محمد محمود فقى حسين[٢]

[(٢،١)] *جامعة السليمانية، كلية الادارة والاقتصاد، السليمانية، العراق*

المستخلص: يتناول هذا البحث فعالية الشبكات العصبية ذات التغذية الامامية (FFNNs) في فهم البيانات الضبابية، وتحديدًا تحليل مجموعة بيانات أسعار خام برنت للفترة من ١ يناير ٢٠١٨ إلى ٣١ ديسمبر ٢٠٢١. تتميز أسعار النفط الخام بتقلبات جوهرية، تتسم بعدم اليقين وعدم الدقة، مما يجعلها مناسبة لتحليل البيانات الضبابية. تُستخدم الشبكات العصبية ذات التغذية الامامية (FFNNs)، المعروفة بقدرتها على نمذجة العلاقات المعقدة وغير الخطية، للتنبؤ بتقلبات الأسعار وتحليلها في هذا المجال الغامض. يُقيّم البحث أداء الشبكات العصبية ذات التغذية الامامية (FFNNs) مقارنةً بالنماذج الإحصائية التقليدية ونماذج التعلم الآلي، مُدمجًا مقاييس تقييم أساسية مثل دقة التنبؤ،

ومتوسط مربع الخطأ، والمرونة في مواجهة البيانات المشوشة أو الغائبة. تشير نتائج هذه الدراسة إلى أن تطبيق المعالجة المسبقة الضبابية بالتزامن مع الشبكات العصبية ذات التغذية الامامية (FFNNs) يُحسّن أداء النموذج بشكل ملحوظ. قبل استخدام المنهجيات الضبابية، سُجِّل متوسط الخطأ التربيعي (MSE) عند ١,٢٩٥٩. ومع ذلك، أدى دمج المعالجة المسبقة الضبابية في إطار عمل FFNN إلى خفض متوسط الخطأ التربيعي إلى ١,٠٩٨٣، مما يُظهر القدرة المُحسَّنة للاستراتيجية المُدمجة على معالجة عدم اليقين وعدم الدقة في البيانات. يُوضح هذا التحسين فعالية الشبكات العصبية المُحسَّنة بالضبابية في توليد تنبؤات أكثر دقة للمتغيرات المُعقَّدة والمتقلبة، مثل أسعار نفط برنت الخام.

الكلمات المفتاحية: تحليلات البيانات الضبابية، FFNN، المعالجة المسبقة الضبابية، المعالجة المسبقة الضبابية، معامل التحديد، متوسط مربعات الخطأ.

Corresponding Author: E-mail: Gashaw.muhammadameen@univsul.edu.iq

Introduction

Crude oil is a fundamental element of both national and international economies. Key factors contributing to heightened price volatility in the crude oil market include political events, extreme weather conditions, and speculative activities in financial markets, among others. The economy and communities are significantly affected by the fluctuations in oil prices, as they impact several commodities and services. An Artificial Neural Network (ANN) is a black-box model that acquires the nonlinear link between system inputs and outputs, emulating human brain function instead of relying on intricate mathematical models. Both fuzzy logic and FFNN address the intrinsic non-linearity of systems such as the crude oil market. Moreover, fuzzy logic provides tools like membership functions to represent ambiguity but FFNNs are adept at extracting complex patterns from data. A major issue in fields like energy economics financial forecasting and uncertainty-based decision-making is analyzing volatile and uncertain data. One prominent example is the crude oil market where intricate and frequently unanticipated factors like supply and demand fluctuations natural disasters and geopolitical instability affect price swings. Fuzzy data analysis has emerged as a reliable technique to lessen these limitations. This approach bridges the gap between strict numerical analysis and real-world ambiguity by creating fuzzy sets to represent uncertainty allowing for more nuanced modeling of imprecise data [1]. A time series is a group of observations or data points that have been documented chronologically. It is frequently used to analyze dynamics patterns and trends in temporally organized data from a range of disciplines such as economics environmental studies and finance.

Fuzzy data analysis is enhanced by FNNs, a form of artificial neural network, which have the capacity to acquire intricate, non-linear relationships. FNNs are composed of interconnected layers of nodes that transform input data through weighted connections and activation functions, thereby facilitating the extraction of patterns from highly complex datasets. They are particularly effective for tasks such as prediction, classification, and regression due to their adaptability and generalization capabilities.

1st: Research Problems

Fuzzy preprocessing combined with FFNN provides a novel and beneficial approach to dealing with uncertain datasets, such crude oil prices. Before neural network data processing, fuzzy pre-processing converts traditional numeric input into fuzzy sets so that the model can handle imprecision. This all-encompassing approach effectively addresses unstable markets by leveraging the exceptional learning capabilities of FFNNs and the benefits of fuzzy logic in managing uncertainty.

2nd: Goal of the Research:

This study focuses on the Brent crude oil price dataset from January 1, 2018, to December 31, 2021, in order to evaluate the effectiveness of fuzzy pre-processing-enhanced FFNNs. The paper shows how this approach improves predicted accuracy and noise resistance by comparing it to traditional statistical and machine learning models. This work adds to the body of knowledge on advanced

analytical techniques by highlighting their value in improving forecasting and decision-making in intricate, uncertain situations.

3rd: Methodology

1- Fuzzy logic

In order to identify outputs, fuzzy logic uses membership functions, which resemble human intuition. It operates with ranges, transforming precise inputs into varying degrees of membership. Optimizing membership functions remains a central research focus in fuzzy logic systems.

2- The fuzzy and Crisp sets

A fuzzy set is a set whose basics have varying degrees of membership, characterized by a membership function that assigns each element a value between 0 and 1. A crisp set is defined as a collection of elements with distinct, binary membership: each element is either a member of the set or it is not. There is no allowance for partial membership; each element is assigned a value of 1 if it belongs to the set and 0 if it does not.

3- Membership Functions

A membership function (MF) maps each point in an input space to a membership degree, ranging from 0 to 1. The membership degree quantifies the degree of membership in a fuzzy set, with values between 0 and 1 indicating partial membership. The membership function is crucial in designing fuzzy controllers, as it must reside within the [0,1] interval [6].

4- Features of Membership Function

A- Support: The support of fuzzy set B is the set of all point $x \in X$ such that $\mu_B(x) > 0$. Mathematically, we can present Support(B) = $\{(x, \mu_B(x))/\mu_B(x)) > 0)\}$.
B- Core: The core of a fuzzy set B is the set of all $x \in X$ such that $\mu_B(x)$=1. Mathematically, we can show core (B) = $\{(x, \mu_B(x))/\mu_B(x)) = 1)\}$
C- $\boldsymbol{\alpha}$-Cut: set of elements with degree $\geq \alpha$. Mathematically, we can indicate $\alpha{-}cut\ B_\alpha = \{x \epsilon X | \mu_B(x) \geq \alpha\}$
D-Strong $\boldsymbol{\alpha}$-Cut: $B_\alpha = \{x \epsilon X | \mu_B(x) \geq \alpha\}$, in this case 'B' is defined as Crisp set.
E-Triangular Membership Function – The subsequent text presents a collection of standard mathematical functions that govern the configuration of membership functions (MFs):
A triangular MF is characterized by the following equation:

$$\mu_B(x) = \begin{cases} 0 & x \leq \alpha \\ (x-\alpha)/(b-\alpha) & \alpha < x \leq b \\ (x-c)/(b-c) & b < x < c \\ 0 & x \geq c \end{cases} \quad (1)$$

here the choice "α to c" shows the support of the fuzzy set and "ɒ" is a unique point in the range and has the highest membership function value.

5- Fuzzification and De-fuzzification

Fuzzification is the transformation of precise quantities into imprecise states, acknowledging that many perceived deterministic quantities are actually characterized by significant uncertainty, often imprecise and represented by a membership function. However, de-fuzzification is the conversion of fuzzy set results from a fuzzy inference system into a precise numerical value, essential for real-world control or decision-making tasks. and a common method is centroid [4].

6- A. Categories of Neural Network Architecture:

(1) Feedforward Neural Network (FFNN): Consists of layers of neurons with input from the layer above and output from the layer below, devoid of feedback loops.

(2) Feedback Neural Network (FBN): is sent output from one layer to the next, facilitate bidirectional signal transmission through loops. They are powerful and complex, allowing synaptic connections among neurons, used in optimization for optimal configuration [9].

B. The main layers of a typical artificial neural network [10]

In order to increase numerical precision in network mathematical operations, the input layer of an artificial neural network (ANN) accepts external data, signals, characteristics, or measurements and normalizes them within activation functions. Most of the internal processing in a network is also carried out by the hidden or unseen layers and the neurons in these layers, which extract patterns associated with the system or process under analysis. In addition, neurons make up the output layer, which generates and displays the final network outputs from the processing done in earlier layers.

C. Training Procedures and Learning Neural Network Characteristics

The training process of a neural network involves tuning synaptic weights and thresholds to generalize solutions, using a learning algorithm to extract discriminant features from samples, enhancing the network's performance.

(1) Supervised Learning: Contains a training sample that includes input signals and the intended outputs; input/output data must be in a table. The learning algorithm continuously modifies synaptic weights and thresholds as the neural structures generate hypotheses based on this data.

(2) Unsupervised Learning: enables learning of clusters within a network, identifying similarities between elements without requiring prior knowledge, thereby enhancing the network's self-organization and accuracy.

(3) Reinforcement Learning: algorithms adjust neural parameters based on interaction with the network environment, evaluating performance through trial and error, using stochastic methods to modify probabilities for improved network performance.

7- Feed-Forward Neural Network (FFNN)

A FFNN is a fundamental model in machine learning. It has four main mechanisms: the input layer, one or more hidden layers, the output layer, and the weights that interconnect the neurons. A Feedforward Neural Network has an input layer that signifies dataset properties and transmits this information to hidden layers, which analyze the data via weighted neuron connections to facilitate intricate transformations. The output layer generates the model's predictions. The weights that determine the strength of these connections are essential to the model's performance, and the main objective of a Feedforward Neural Network is to optimize these weights for precise input-to-output mapping and efficient target variable prediction.

8- Types of Active Function

A. Hyperbolic Tangent Function (Tanh)

The tanh is an alternative activation function used in network processing, with many versions utilized in deep learning applications. The tanh function, is a smoother zero-centred function whose range lies between -1 to 1, thus the output of the tanh function is given by the equation

$$Y = \frac{Exp^{x} - Exp^{-x}}{Exp^{x} + Exp^{-x}} \qquad (2)$$

B. Linear Active function

The linear function calculates the neuron s output by simply returning the value passed to it.

$$Y = x \qquad (3)$$

9- Derivation of the Levenberg–Marquardt Algorithm

The Levenberg-Marquardt algorithm is an adaptive approach for solving nonlinear least squares problems, which is widely used in curve fitting and parameter estimation. It dynamically adjusts a damping factor to combine steepest descent's stability with the Gauss-Newton method's quick

convergence. To update the model parameters, the approach first linearizes the goal function and the solves a damped least squares problem. Its versatility makes it a reliable and effective tool for improving complicated, nonlinear models.

Table (1): Specifications of different algorithms

Algorithms	Update rule	convergence
EBP algorithm	$T_{k+1} = T_k - \beta g_k$	Stable, Slow
Newton algorithm	$T_{k+1} = T_k - H_k^{-1} g_k$	Unstable, Fast
Gauss-Newton algorithm	$T_{k+1} = T_k - (Z_k^t Z_k)^{-1} Z_k e_k$	Unstable, Fast
Levenberg-Marquardt algorithm	$T_{k+1} = T_k - (Z_k^t Z_k + mI)^{-1} Z_k e_k$	Stable, Fast

4th: Result and Discussions:

A. Data Description:

We extracted Brent crude oil production data daily spanning 1461 days, from 1 January 2018 to 31 of December 2022. The fuzzy time series model and FNN have been implemented to analyse the data set. The data was sourced from the markets insider [15] used MATLAB and python software's for analysing the data [15].

B. The Process of Implementing FNN

The application for building an FFNN for time series prediction is included in this section. For the data mentioned above, FFNN has been applied using the MATLAB software. The following procedures were used in this paper to applying the FNN for time series prediction:

(1) First Step

In this instance, we utilize the MATLAB software to identify data, as we require a specific type of data for the (FFNN) algorithm. The input data is denoted as $(X_t: t = 1,2, \dots, 1461)$, while the objective data is denoted as $(T; t = 1,2,3, \dots, 1461)$. The Augmented Dickey-Fuller (ADF) test was implemented to assess the stationary of the Brent petroleum oil price time series. The test results offer critical insights into whether the data necessitates transformation, such as differencing, to attain stationary for on-going time series analysis and model development.

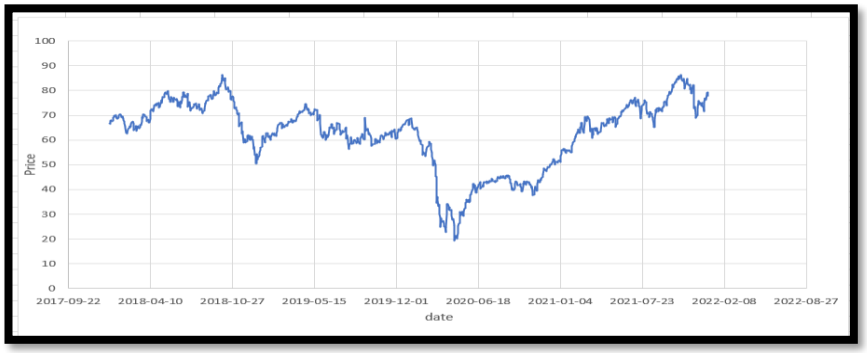

Fig (1): Daily price of Brent Oil Crude

Hypothesis Test:

H_0:: The time series is non-stationary and has a unit root

H_1: The time series lacks a stationary unit root.

Table (2): Represent the ADF Test of Brent Oil Crude price

Test	Test Statistic	P-Value	Critical Value
Augmented Dickey-Fuller test	-1.427248	0.5702	1% :-3.434621
			5% : -2.863313
			10%:-2.567763

Since the test statistic (-1.427248) is greater than the critical values at all significance levels (1%, 5%, and 10%) and the p-value (0.5702) is greater than the conventional cutoff of 0.05, we are unable to reject the null hypothesis of the Augmented Dickey-Fuller test. Given that there is little evidence to support the notion that the time series is stationary, it is likely to have a unit root and may need to undergo some sort of transformation, such as a difference, to become stationary.

Table (3): Represent the ADF Test of Brent Oil Crude Price After First Difference.

Test	Test Statistic	P-Value	Critical Value
Augmented Dickey-Fuller test	-38.38734	0.0000	1% :-3.434624
			5% : -2.863315
			10%:-2.567763

The low p-value (0.0000) and the extremely negative test statistic of -38.38734, which is far more negative than the critical values at all significance levels, lead us to strongly reject the null hypothesis. This implies that the time series is stationary as it does not have a unit root and its statistical properties do not change over time. There is no need for any transformation, including differencing, because the series is inherently stationary.

(2) Step two: Data normalization

This step's goal is to normalize the data by limiting it to either the (0, 1) or (-1, 1) ranges. This coding depends on the features of the particular ANN or FFNN type used, especially when the dataset has complex patterns. In these situations, using the first type of normalization described above might be more beneficial.

(3) Step three: Partitioning the data

This research involved partitioning the dataset into training, testing, and validation subsets through three distinct methodologies to ascertain the most effective allocation for the development and evaluation of FFNN. The following proportions were used: 70% for training, 15% for testing, and 15% for validation; 60% for training, 20% for testing, and 20% for validation; and 80% for training, 10% for testing, and 10% for validation. Every partitioning technique was analyzed to determine how it affected the predicted accuracy and generalization of the model. The main objective was to improve the allocation ratio to balance model training, mitigate over fitting, and guarantee adequate data availability for impartial testing and validation. The goal of the study is to compare the outcomes in various proportions in order to determine the best partitioning strategy for controlling the volatility and uncertainty in the Brent crude oil price dataset. The FFNN models' robustness and reliability for forecasting and decision-making in complex situations are ensured by this methodology.

(4) Step four: Develop the network architecture

Here, the FNN is composed of three layers: the input layer, hidden layer, and output layer. We select the optimal approach for the network's performance at this stage by evaluating two critical metrics: The minimum MSE and RMSE values. The selection of the optimal FFNN is contingent

upon the minimum MSE and RMSE values. In this study the best FFNN chosen to analyse data, one hidden layer network is used, which is explained in table (4) and figure (6) below.

Table (4) Represent the best architecture of (FFNN)

Layers	Nodes	Activation function
Input layer	1	----------
Hidden layers	16	Tansig (hyperbolic)
Output layer	1	Purelin

The table you gave appears to depict the architecture of a neural network, detailing the network's layers, node count, and activation functions to be precise.

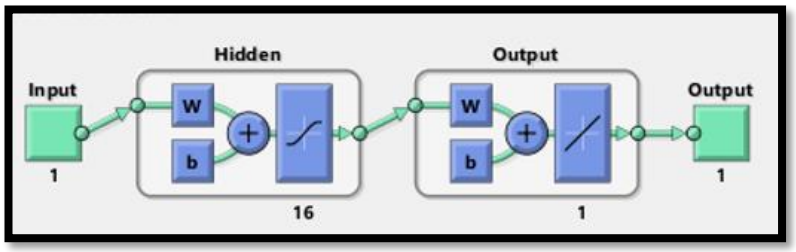

Fig (2) Represent the best architecture FFNN model (1-16-1)

The optimal neural network configuration is [1-16-1], determined by achieving minimum MSE and RMSE values. By using the TANSIG activation function, the number of nodes of hidden layers were adjusted iteratively. Partitioning of the dataset into training, testing, and validation was varied to identify the best performance setup and we get the best model is [Tansig _ Pureline output].

Table (5): The optimal neural network configuration dataset

MSE (all)	RMSE	MSE (training)	MSE (validate)	MSE (testing)	Rsq (training)	Rsq (validation)	Rsq (testing)	Rsq (all)
1.2959	1.1384	1.3867	0.94405	0.92129	0.99283	0.9944	0.99465	0.99314

Table (6): Performance Metrics of Neural Network Models with Varying Architectures and Data Split

(training,testing, validation)	Active function (hidden layer)	MODEL	MSE	RMSE	trainRsq	valRsq	testRsq	Rsq(all)	trainMSE	valMSE	testMSE	iteration
%80, %10 , %10	tansig	1_9_1	1.3135	1.1461	0.99309	0.99355	0.9921	0.99304	1.3411	1.0403	1.3663	18
%80, %10 , %10	tansig	1-14-1	1.311	1.145	0.99301	0.99366	0.99288	0.99306	1.3486	1.0571	1.2636	19
%80, %10 , %10	tansig	1-16-1	1.2959	1.1384	0.99283	0.9944	0.99465	0.99314	1.3867	0.94405	0.92129	25
%80, %10 , %10	tansig	1-17-1	1.3	1.1402	0.99266	0.99525	0.99514	0.99312	1.4203	0.8218	0.81497	28
%80, %10 , %10	tansig	1-18-1	1.3037	1.1418	0.99289	0.9942	0.99378	0.9931	1.3714	1.0101	1.0558	17
%80, %10 , %10	tansig	1-19-1	1.2992	1.1398	0.99384	0.99569	0.98369	0.99312	1.1867	0.77608	2.72	34
%80, %10 , %10	tansig	1-20-1	1.32	1.1489	0.99277	0.99508	0.99297	0.99301	1.3917	0.90049	1.1655	12
%70 , %15 ,%15	tansig	1_4_1	1.346	1.1602	0.99388	0.99278	0.98809	0.99287	1.188	1.182	2.2478	16
%70 , %15 ,%15	tansig	1_5_1	1.3223	1.1499	0.99289	0.9917	0.99471	0.993	1.3889	1.3841	0.94981	13
%70 , %15 ,%15	tansig	1_7_1	1.328	1.1524	0.99287	0.9926	0.99382	0.99297	1.4141	1.2005	1.0541	14

%70 , %15 ,%15	tansig	1_11_1	1.3177	1.1479	0.99427	0.98748	0.99171	0.99302	1.1315	2.1165	1.3875	16
%70 , %15 ,%15	tansig	1_12_1	1.3095	1.1444	0.9925	0.9951	0.99406	0.99306	1.4741	0.82776	1.0234	25
%70 , %15 ,%15	tansig	1-16-1	1.3006	1.1404	0.99274	0.99427	0.99399	0.99311	1.4326	0.9489	1.0367	27
%70 , %15 ,%15	tansig	1-17-1	1.3129	1.1458	0.99284	0.99217	0.995	0.99305	1.4127	1.2986	0.86117	34
%70 , %15 ,%15	tansig	1-18-1	1.321	1.1493	0.99294	0.9921	0.99416	0.993	1.3919	1.2948	1.0163	13
%70 , %15 ,%15	tansig	1-20-1	1.3612	1.1667	0.99233	0.9944	0.99361	0.99279	1.502	0.96022	1.1052	8
%60 , %20,%20	tansig	1_5_1	1.3206	1.1492	0.99283	0.9928	0.99377	0.99301	1.4439	1.1919	1.0795	15
%60 , %20,%20	tansig	1_15_1	1.3092	1.1442	0.99444	0.99032	0.99117	0.99307	1.0966	1.7655	1.4907	33
%60 , %20,%20	tansig	1_6_1	1.3255	1.1513	0.99389	0.9917	0.99098	0.99298	1.2292	1.3573	1.5827	27
%60 , %20,%20	tansig	1-19-1	1.3201	1.149	0.99428	0.99183	0.98983	0.99301	1.1238	1.4638	1.7653	21

In conclusion, this model appears to be doing rather well; it has low MSE/RMSE values and high R^2 values across training, validation, and testing datasets, indicating that it generalizes to unknown data with ease.

The model with the lowest MSE and highest R^2 values, 1-16-1 (80%, 10%, 10%), performs the best. Although it may result in overfitting, increasing the number of hidden neurons (e.g., 19, 20) can improve accuracy. Because it provides more training data, the (80%, 10%, 10%) split performs better than 60%, 20%, and 20%. The dataset split and ideal hidden layers are essential for striking a balance between generalization and accuracy.

(5): step five: Training FFNN

Figure3 illustrates the training process of the feedforward neural network (FNN) employing the Levenberg-Marquardt optimization algorithm, displaying three essential metrics across 25 epochs. The initial figure monitors the gradient, which diminishes consistently, signifying effective optimization and convergence of the model. The second plot illustrates the parameter "mu," which dynamically adjusts to equilibrate the gradient descent and Gauss-Newton methods, diminishing as the model nears the best solution. The third plot illustrates validation checks, emphasizing validation performance across epochs, culminating in a final validation failure count of six. These measures indicate successful training and imply the model's capacity to generalize effectively to novel data while reducing overfitting.

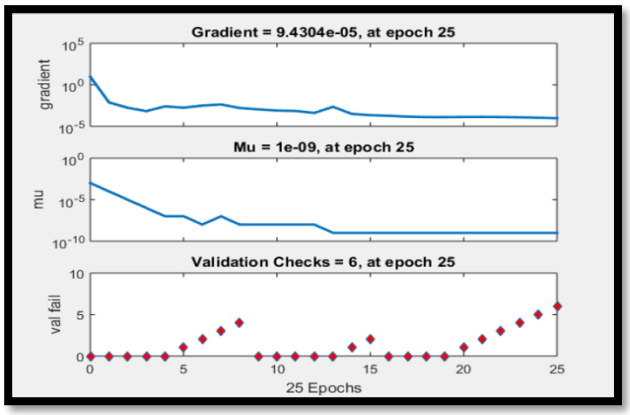

Fig (3): Training dataset Brent Crude Oil price

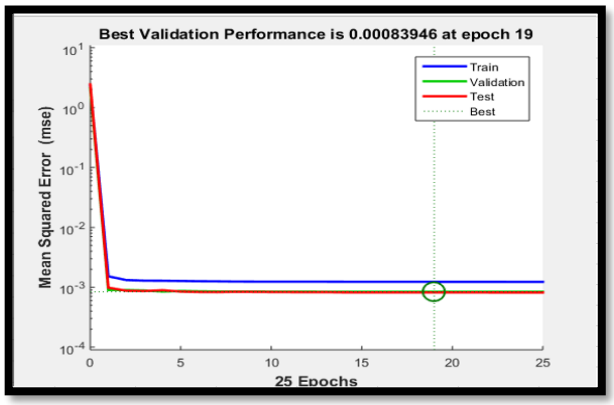

Figure (4): The training performance

Figure 4 depicts the training, validation, and testing performance of the FFNN across 25 epochs, utilizing mean squared error (MSE) as the performance metric. The graph shows a rapid decrease in MSE in the early epochs, indicating that the model learned well. The green circle represents epoch 19, which has the best validation performance with an MSE of 0.00083946. The validation error stabilizes after this point, suggesting that the model has effectively converged without over fitting. The robustness and generalization capabilities of the model for predicting patterns in the dataset are supported by the tight correspondence of training, validation, and testing errors. For all models, this model's performance is equivalent to an MSE of 0.2959. Table (5) shows the best of the best phases of the architectural model in (FFNN) for the

(6): Regression Plot:

Regression plots show that the model's mistakes are small and fairly evenly distributed, which is an essential indication of robustness and reliability. Moreover, the weights throughout all levels of the network have been meticulously adjusted to discern the fundamental patterns and correlations within the data. The results affirm that the proposed FFNN architecture is adept at handling the dataset's complexity and volatility, substantiating its predictive accuracy and overall efficacy in this investigation.

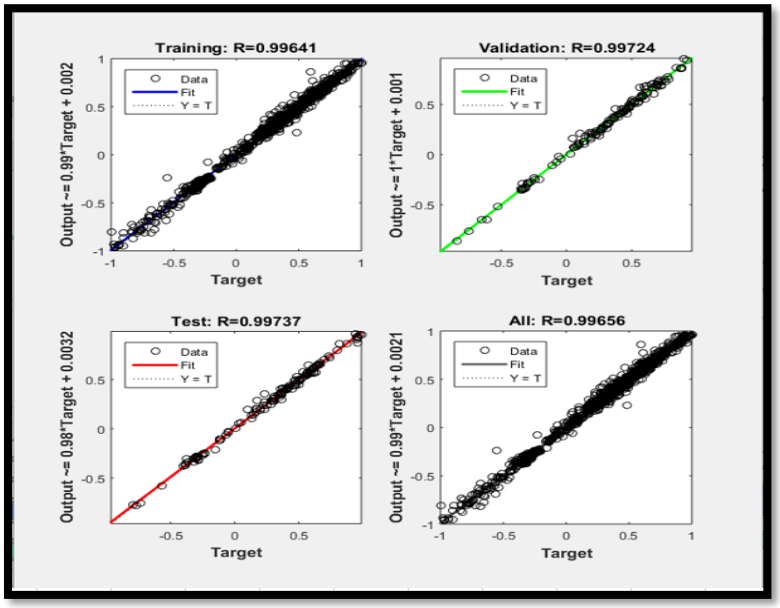

Fig (5): Regression plot of training, testing, validation and all data

Figure 5 presents regression charts that thoroughly assess the FFNN model with the architecture (1-16-1), encompassing training, validation, testing, and the aggregate data. The elevated correlation coefficients (R-values) for each dataset—R=0.99641 (training), R=0.99724 (validation), R=0.99737 (testing), and R=0.99656 (all data)—demonstrate exceptional concordance between the predicted and actual values, affirming the model's capacity to generalize proficiently across all datasets. This chart illustrates the superior performance of the proposed FFNN architecture, chosen following comprehensive experiments to identify the optimal configuration.

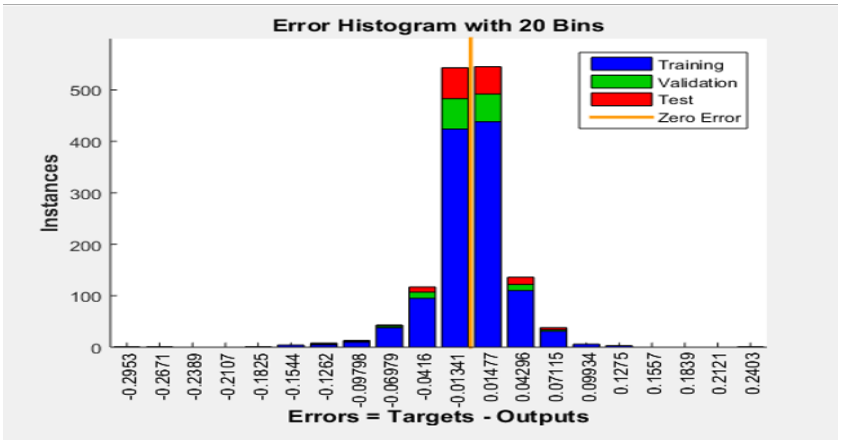

Fig (6) Error Histogram

The error histogram illustrates the distribution of errors (discrepancies between targets and outputs) for training, validation, and testing datasets subsequent to the implementation of the FFNN. The majority of errors are concentrated around zero, signifying elevated prediction accuracy across all subsets. The symmetrical and constrained error distribution validates the model's efficacy in identifying patterns and reducing discrepancies within the dataset.

C. Fuzzy Pre-processing with FFNN

Figure 7 compares the de-fuzzified data (red line), which were produced by Python using the centroid de-fuzzification technique, with the initial crude oil prices (blue line). In fuzzy logic, the centroid technique is frequently used to identify the fuzzy set's center of gravity and provide a single, distinct output. Through noise reduction and the preservation of key patterns and trends, this technique ensures that the de-fuzzified values faithfully represent the original fuzzy data. Using the centroid method, the fuzzy logic process, which was implemented in Python, converted the raw data into fuzzy sets and then de-fuzzed it, resulting in an exact match between the de-fuzzed values and the initial pricing. The effectiveness of fuzzy logic in data pre-processing to improve FFNN's performance for modeling and forecasting intricate, erratic datasets like crude oil prices is demonstrated here.

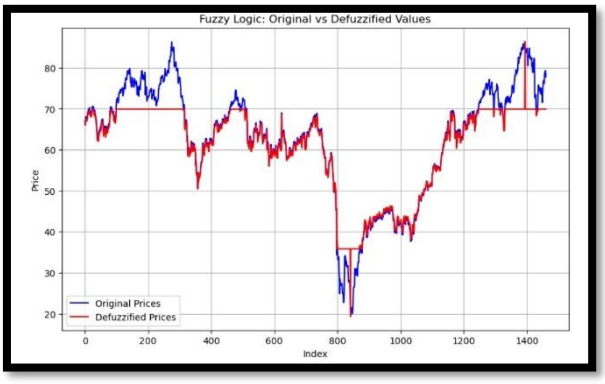

Fig (7) Applying Fuzzy Logic by Centroid Method

Table (7): Pre-processing fuzzy logic of data sets

Number	Original	De-fuzzified
1	66.87	66.273799
2	66.57	65.975268
3	67.84	67.24258
4	68.07	67.473086
5	67.62	67.022382
6	67.62	67.022382
.	.	.
.	.	.
.	.	.
1459	79.23	69.91368
1460	79.32	69.91368
1461	77.78	69.91368

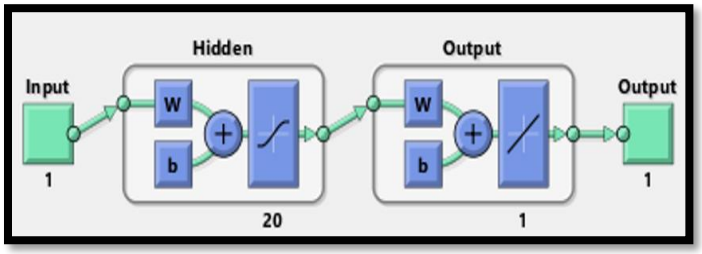

Fig (8): Show the top-performing FFNN architecture (1-20-1).

The figure depicts an FFNN with a single input node, a hidden layer with twenty neurons, and a single output neuron.Weighted sums and biases undergo activation functions to represent intricate relationships, whereas the output layer employs a linear function for regression applications. This architecture was optimized to adeptly manage the volatility and uncertainty inherent in the fuzzy-processed crude oil price dataset.

Table (8): Represent the best architecture of (FFNN)

(training,testing, validation)	Active function (hidden layer)	MODEL	MSE	RMSE	trainRsq	valRsq	testRsq	Rsq(all)	trainMSE	valMSE	testMSE	iteration
%80 , %10 ,%10	tansig	1_4_1	1.111	1.0541	0.99133	0.98701	0.98613	0.9904	1.0114	1.3735	1.6453	432
%80 , %10 ,%10	tansig	1_5_1	1.1079	1.0526	0.98943	0.99321	0.99588	0.99043	1.2346	0.72195	0.47956	14
%80 , %10 ,%10	tansig	1_6_1	1.109	1.0531	0.99386	0.96468	0.98974	0.99042	0.70546	4.2395	1.2066	31
%80 , %10 ,%10	tansig	1_7_1	1.209	1.0995	0.98866	0.99242	0.99443	0.98956	1.3341	0.80229	0.61497	11
%80 , %10 ,%10	tansig	1_11_1	1.1026	1.05	0.98997	0.99119	0.99414	0.99048	1.1802	0.94895	0.63584	19
%80 , %10 ,%10	tansig	1-14-1	1.1043	1.0509	0.99122	0.99457	0.98048	0.99046	1.024	0.59114	2.26	32
%80 , %10 ,%10	tansig	1-18-1	1.105	1.0512	0.98954	0.99429	0.99444	0.99045	1.227	0.63289	0.60177	26
%80 , %10 ,%10	tansig	1-20-1	1.0983	1.048	0.98964	0.99526	0.9931	0.99051	1.2145	0.54471	0.72286	22
%70 , %15 ,%15	tansig	1_3_1	1.1446	1.0698	0.99085	0.99309	0.98416	0.99011	1.0793	0.7009	1.8927	49
%70 , %15 ,%15	tansig	1_5_1	1.1092	1.0532	0.99111	0.98078	0.99601	0.99042	1.0454	2.0549	0.46143	129
%70 , %15 ,%15	tansig	1_8_1	1.103	1.0502	0.98989	0.99382	0.99027	0.99047	1.2087	0.62544	1.087	31
%70 , %15 ,%15	tansig	1_11_1	1.104	1.0507	0.99037	0.98933	0.99199	0.99046	1.1467	1.1385	0.86982	54
%70 , %15 ,%15	tansig	1-14-1	1.1047	1.051	0.99112	0.9924	0.98553	0.99046	1.0499	0.79026	1.6746	20
%70 , %15 ,%15	tansig	1-18-1	1.1188	1.0577	0.98917	0.9919	0.99481	0.99034	1.2997	0.82178	0.57196	9

%70 , %15 ,%15	tansig	1-19-1	1.1051	1.0513	0.99032	0.9954	0.98644	0.99045	1.1534	0.48204	1.5032	39
%70 , %15 ,%15	tansig	1-20-1	1.1262	1.0612	0.98905	0.99348	0.99343	0.99027	1.3092	0.67977	0.71877	8
%60 , %20,%20	tansig	1_3_1	1.1242	1.0603	0.99061	0.99316	0.98656	0.99029	1.1314	0.70197	1.5246	78
%60 , %20,%20	tansig	1_5_1	1.1093	1.0532	0.99326	0.97668	0.99462	0.99042	0.80342	2.537	0.59937	72
%60 , %20,%20	tansig	1_8_1	1.1035	1.0505	0.98949	0.99401	0.99028	0.99047	1.2822	0.61864	1.0522	88
%60 , %20,%20	tansig	1_12_1	1.1193	1.058	0.98831	0.99248	0.99487	0.99033	1.4165	0.78241	0.5648	20
%60 , %20,%20	tansig	1-18-1	1.1458	1.0704	0.99063	0.98416	0.99406	0.9901	1.1283	1.6931	0.65111	8
%60 , %20,%20	tansig	1-19-1	1.2286	1.1084	0.98913	0.99083	0.98876	0.98939	1.3121	0.95421	1.2524	8
%60 , %20,%20	tansig	1-20-1	1.1301	1.063	0.99055	0.98541	0.99388	0.99024	1.1344	1.5752	0.67196	8

All training, validation, and testing datasets show strong R^2 values, suggesting that the model is operating at a high level. The model's predictions, particularly on the validation and testing datasets, are in close agreement with the actual values, as evidenced by the comparatively low MSE and RMSE. Overfitting is not evident, and the model appears to generalize well to fresh, untested data.

Table (9): The optimal neural network configuration after fuzzy logic pre-processing dataset

Layers	Nodes	Activation function
Input layer	1	----------
Hidden layers	20	Tansig (hyperbolic)
Output layer	1	purelin

Table (10): Performance Metrics of NN with Varying Architectures and Data Split

MSE (all)	RMSE	MSE (training)	MSE (validation)	MSE (testing)	Rsq (training)	Rsq (validation)	Rsq (testing)	Rsq (all)
1.0983	1.048	1.2145	0.54471	0.72286	0.98964	0.99526	0.9931	0.99051

This table illustrates how model performance is affected by several neural network configurations, including variations in the number of hidden neurons and training-validation-testing splits. Simpler models may function well and converge more quickly, saving training time, but more complicated models (with more hidden neurons) typically score higher on R-squared values and have lower error metrics (MSE, RMSE). These models perform better in terms of generalization when the data split is 80/10/10, but models trained with 70/15/15 or 60/20/20 splits also do well.

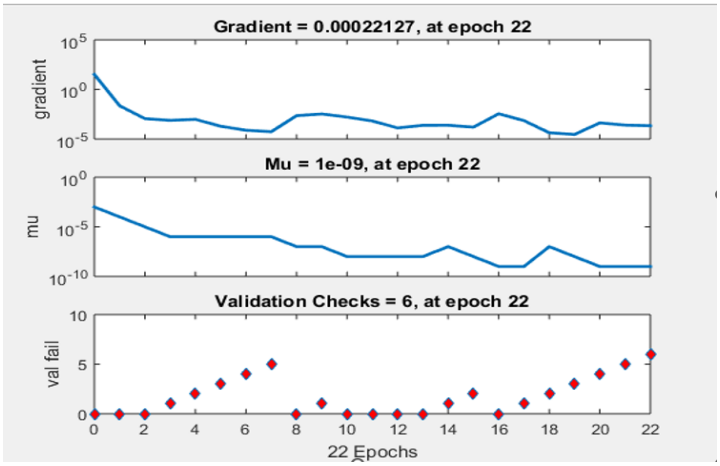

Fig (9): Training dataset after applying fuzzy pre-processing

Figure 9 illustrates the training process of the FNN following fuzzy pre-processing, with the gradient diminishing to 0.00022127 by epoch 22, signifying effective convergence. The "mu" value dynamically reduces to 1e-09, optimizing the balance between stability and speed. The validation checks remain at six, indicating that the model generalizes effectively without over fitting following the fuzzy pre-processing phase.

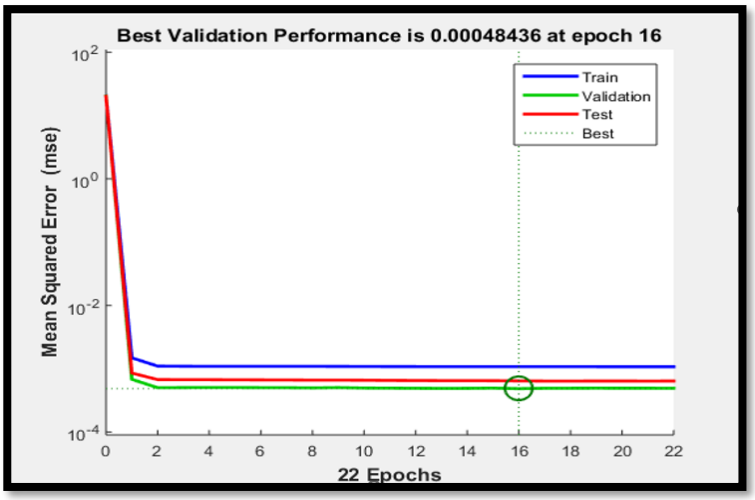

Fig (10): The training performance after applying fuzzy

Figure 10 illustrates the mean squared error performance of the FFNN subsequent to the implementation of fuzzy pre-processing. The optimal validation performance, evidenced by an MSE of 0.00048436, occurs at epoch 16, as denoted by the green circle. The tight correspondence of training, validation, and testing curves validates the efficacy of fuzzy pre-processing in enhancing the model's generalization and precision.

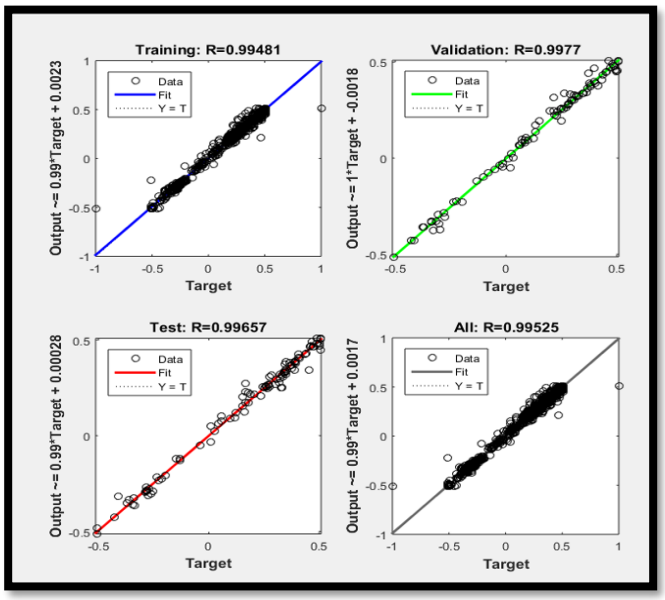

Fig (11): Regression plot of (training, testing, validation, and all data)

The regression plots illustrate the efficacy of the FFNN subsequent to fuzzy pre-processing, exhibiting high correlation coefficients: R=0.99481 (training), R=0.9977 (validation), R=0.99657 (testing), and R=0.99525 (overall). These figures affirm a robust correlation between expected and actual outputs, signifying the model's precision. The fuzzy preprocessing improved the network's capacity to generalize and accurately identify intricate patterns in the data.

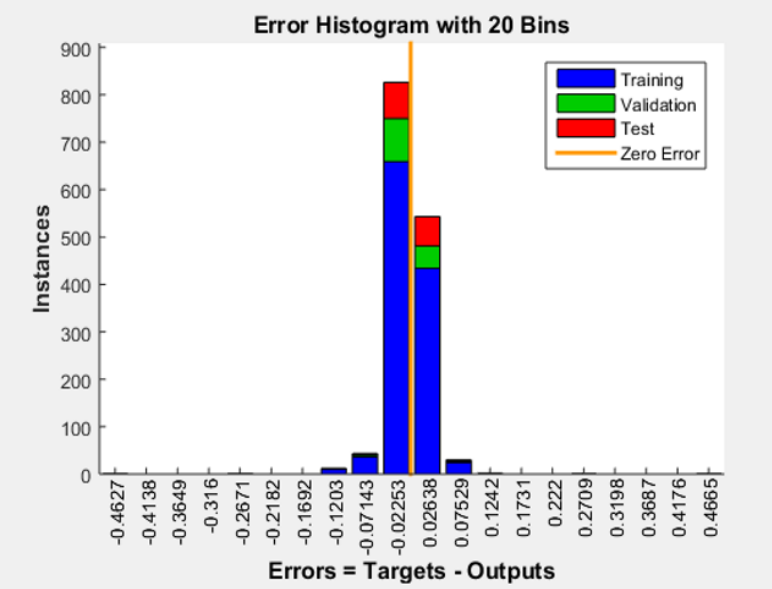

Fig (12): The error histogram

The error histogram depicts the distribution of errors (discrepancies between targets and outputs) for training, validation, and testing datasets subsequent to the implementation of fuzzy pre-processing and the FFNN. The majority of mistakes are clustered around zero, signifying elevated prediction accuracy and negligible variance across all datasets. The limited distribution further illustrates the efficacy of fuzzy preprocessing in enhancing the precision and robustness of the FFNN.

Table (11): Comparison of FFNN Performance Before and After Applying Fuzzy Logic

Metrics	FFNN before using fuzzy	FFNN after using fuzzy
Mean Square Error	1.2959	1.0983
Root Mean Square Error	1.1384	1.048
Trainin (MSE)	1.3867	1.2145
Validation (MSE)	0.94405	0.54471
Testing (MSE)	0.92129	0.72286
R-Squared (all)	0.99314	0.99051
Trainin Rsq	0.99283	0.98964
Validation Rsq	0.9944	0.99526
Testing Rsq	0.99465	0.9931
Iteration	25	22

For both training and testing data, fuzzy logic lowers prediction errors by improving the model's error metrics (MSE, RMSE). Additionally, as seen by lower validation and testing MSE, it improves generalization. The model continues to fit the data well even when R-squared values marginally decline. Additionally, because it takes fewer iterations to converge, fuzzy logic improves training efficiency.

5th: Conclusion and Recommendation

1- Conclusion

This study demonstrates the efficacy of FFNNs in conjunction with fuzzy pre-processing for the analysis of volatile data, including the price of Brent crude oil. The fuzzy-enhanced FFNNs' improved capacity to handle uncertainty is seen by the mean squared error (MSE) dropping from 1.2959 to 1.0983. When fuzzy approaches are combined with neural networks, the accuracy of data processing is greatly increased.

2- Recommendation's

For the analysis of complicated datasets with uncertainty, this methodology is advised. Hybrid models should be examined in future studies to improve prediction accuracy even more. Further performance gains could result from investigating different fuzzy and deep learning techniques.

6th: References

1- Alam G, Ihsanullah I, Naushad M, Sillanpää M (2022) Applications of artificial intelligence in water treatment for optimization and automation of adsorption processes: Recent advances and prospects. Chemical Engineering Journal 427:130011.
2- Ali OAM, Ali AY, Sumait BS (2015) Comparison between the effects of different types of membership functions on fuzzy logic controller performance. International Journal 76:76–83.
3- Aziz NR, Hussein MMF (2025) A COMPARATIVE ANALYSIS OF DE-FUZZIFICATION TECHNIQUES FOR SURVIVAL TIME DATA IN WEIBULL DISTRIBUTION. Journal For Administrative and Economic Science 15:201–214.
4- Baghdadi A, Babovic N, Kloft H (2023) Fuzzy Logic, Neural Network, and Adaptive Neuro-Fuzzy Inference System in Delegation of Standard Concrete Beam Calculations. Buildings 14:15.
5- Bede B (2013) Mathematics of Fuzzy Sets and Fuzzy Logic. https://doi.org/10.1007/978-3-642-35221-8.
6- Chakraverty S, Mall S (2017) Artificial neural networks for engineers and scientists: solving ordinary differential equations. CRC Press.
7- Chen Z, Ma M, Li T, Wang H, Li C (2023) Long sequence time-series forecasting with deep learning: A survey. Information Fusion 97:101819.
8- Da Silva IN, Hernane Spatti D, Andrade Flauzino R, Liboni LHB, Dos Reis Alves SF (2017) Artificial Neural Network Architectures and Training Processes. In: Artificial Neural Networks. Springer International Publishing, Cham, pp 21–28.
9- Hamad APDAS, Faqe APDMM, Mohamad ALSH (2023) Forecasting Life-Expectancy in Iraq During the Period (2022-2035) Using Fuzzy Markov Chain. University of Fallujah, Journal of Business Economics for Applied Research 5:347–372.
10- Hastie T, Tibshirani R, Friedman J (2009) Overview of Supervised Learning. In: The Elements of Statistical Learning. Springer New York, New York, NY, pp 9–41.
11- Hussein MMF, Saeed AA, Mohamad SH (2023) Comparison Markov Chain and Neural network models for forecasting population growth data in Iraq. University of Kirkuk Journal For Administrative and Economic Science 13.
12- Islam M, Chen G, Jin S (2019) An overview of neural network. American Journal of Neural Networks and Applications 5:7–11.
13- Markets Insider. In: markets.businessinsider.com. https://markets.businessinsider.com/commodities/oil-price. Accessed 21 Jan 2025.
14- Yu H, Wilamowski BM (2018) Levenberg–marquardt training. In: Intelligent systems. CRC Press, pp 12–1.
15- Zadeh LA (2015) Fuzzy logic—a personal perspective. Fuzzy sets and systems 281:4–20.