A Predictive Model for Software Cost Estimation Using ARIMA Algorithm

Moatasem M. Draz*[1], Osama Emam[2], Safaa M. Azzam[3]
Department of Software Engineering-Faculty of Computers and Information, Kafrelsheikh University, Kafrelsheikh, Egypt[1]
Department of Information Systems-Faculty of Computers and Artificial Intelligence, Helwan University, Helwan, Egypt[1, 2, 3]

***Abstract*—Technology is a differentiator in business today. It plays a different and decisive role by providing programs that contribute to this. To build this software while avoiding risks during the implementation and construction process, it is necessary to estimate the cost. The cost estimation process is the process of estimating the effort, time, and resources needed to build a software project. It is a crucial process as it provides good planning during the construction and implementation process and reduces the risks you may be exposed to. Therefore, previous studies sought to build models and methods to estimate this, but they were not accurate enough to complete the process. Therefore, this study seeks to build a model using the Autoregressive integrated moving average (ARIMA) algorithm. Five datasets the COCOMO81, COCOMONasaV1, COCOMONasaV2, Desharnais, and China were used. The dataset was processed to remove noise and missing values, visualized to understand it, and linked using a time series to predict the future values of the data. It will then be trained on the ARIMA algorithm. To ensure the effectiveness and efficiency of the model for use, four famous evaluation criteria were used: mean magnitude of relative error (MMRE), root mean square error (RMSE), mean magnitude of relative error (MdMRE), and prediction accuracy (PRED). This experiment showed impressive software cost estimation results, with MMRE, RMSE, MdMRE, and PRED results being 0.07613, 0.04999, 0.03813, and 95% for the COCOMO81 dataset, respectively. The results were high for the COCOMONasaV1 dataset, reaching 0.02227, 0.02899, 0.01113, and 97.1%. The COCOMONasaV2 results were 0.01035, 0.00650, 0.00517, and 99.35%, respectively. The China dataset showed good prediction results of 0.00001, 0.00430, 0.00008, and 99.57%, respectively. The results were impressive and promising for the Desharnais dataset, showing 0.00004, 0.0039, 0.00002, and 99.6%. The results of this study are promising and distinctive compared to recent studies, and they also contribute to good business planning and risk reduction.**

Keywords*—Software cost estimation; software effort estimation; promise repository; SCE; ARIMA

I. Introduction

Today, the software industry represents a differentiating element in all fields, as business owners depend on technology to conduct their business which is a strong pillar in business speed. As a result, the pressure on software houses has become very great [1]. This led to the production of software that was expensive and had little or even poor efficiency at times. To control this and produce highly efficient and optimal software, it was necessary to estimate the software cost.

Estimating the cost of software is crucial and necessary to ensure the efficiency of the project. It is also a differentiating element for companies to calculate their advantages and estimate their resources, as well as the effort expended to build the project in addition to the time required for it [2]. It also enables stakeholders to know what is needed to implement their project as well. All of this contributes directly to customer satisfaction.

The importance of estimating the cost of software lies in good planning and effective management of the project. It also gives a time estimate for delivery time, as well as estimating the resources needed for this, which contributes to reducing damage to the implemented projects, as well as reducing the technical costs necessary for this, which earns the company a good reputation [3].

The process of estimating the cost of software is carried out through several inputs, which are the project requirements and cost factors so that the process is completed and its output is the time, effort, and resources required for this.

Many researchers have presented numerous studies over the past years, some of which were based on their work on mathematical equations and are called algorithmic methods, the most famous of which are the Constructive Cost Model (COCOMO) [4] and the Software Life Cycle Model (SLIM) [5]. Others also presented methods that depend in their work on the experiences of employees of software houses and are called non-algorithmic methods, such as expert judgment [6].

In recent years, researchers have turned to using learning algorithms such as machine learning [7-10], and some have relied on deep learning techniques [11,12]. Despite the large number of studies that have been conducted, these models are not effective, and the prediction accuracy is not good enough to use these models in the forecasting process. Software houses face difficulty and complexity in the process of forecasting and estimating the cost of software [13].

The map of this study is clear and multiple as it uses software cost estimation, which is an important branch of software construction that falls under the umbrella of software engineering. Autoregressive integrated moving average (ARIMA) [14] algorithm is also used, which is one of the optimization algorithms within the umbrella of machine learning within artificial intelligence and data science techniques.

This study seeks to present a model based on machine learning techniques to predict software cost estimates. Five datasets, namely COCOMO81, COCOMONasaV1, COCOMONasaV2, Desharnais, and China, were collected

from the promise repository [15]. The data was processed to remove noise and missing values and represented to understand them, as well as linking them to apply time series technology to predict the future values of the data. ARIMA algorithm is used to be trained on the datasets used. ARIMA algorithm is used due to the efficiency and accuracy of its results. To evaluate the proposed model, four famous evaluation criteria are used for prediction: mean magnitude of relative error (MMRE) [16], root mean square error (RMSE) [17], mean magnitude of relative error (MdMRE) [18], and prediction (PRED) [19].

This paper makes a significant contribution to the software industry by:

- Processing to remove noise and missing values from data, in addition to representing and analyzing to understand the data, as well as linking it using time series to predict future values.
- Using five data sets of medium and large sizes to train and test the proposed model under the same conditions according to the evaluation criteria used in previous studies.
- Applying the ARIMA algorithm to datasets after linking them to time series to produce the highest possible efficiency and accuracy to reduce error rates resulting in the forecasting process.
- The cost estimation prediction results are very promising compared to previous studies.

This study represents a distinct model in the software estimation process as the proposed model combines distinct sets of results that prove the effectiveness of the model and its efficiency in future studies.

The rest of this article is organized as follows: Section II presents the literature review. Section III describes the proposed methodology. Section IV elaborates on the evaluation and di. Section V summarizes our findings and suggests future research directions.

II. Related Work

The software cost estimation process has been a puzzle for researchers over recent years. Many researchers sought to invent techniques that contributed to predicting this process, some of which relied on mathematical equations in their work and called them algorithmic methods. Some also relied on elements of experience from developers and project managers within programming houses, which are called non-algorithmic methods. However, during the last two decades, many researchers have relied on learning techniques, which are considered a lifeline in this industry, as many have relied on machine learning and deep learning techniques to estimate this process.

Shukla et al. [20] presented a model called ANFIS which is an intelligent model using AI to improve software cost estimation forecasting and was trained and tested using the Desharnais dataset collected from the PROMISE repository. Model performance was evaluated by MAE and RMSE metrics. It was compared to the regression model, where the RMSE value was 780.97 compared to 3007.05 for the regression model.

Posbiezny et al. [21] built a model using neural networks, support vector machines with cross-validation, and generalized linear models in which the described set of algorithms was averaged using the ISBSG datasets. The effectiveness of the model was verified using MAE, MMRE, mean square error (MSE), RMSE, MMER, balanced mean relative error (MBRE), and PRED. The model effectively predicted the program effort estimate during the evaluation process according to a fixed period.

Vijayvargiya et al. [22] presented several algorithms to calculate the resources needed to build a Bermuda project and the time needed. Linear regression, support vector regression, artificial neural networks, decision trees, and bagging algorithms were used. These algorithms were trained on the ISBSG and Desharnais datasets. To compare them, three evaluation criteria were used: the mean absolute error (MAE), the mean square error (MSE), and the R square error. The evaluation result demonstrated the superiority of the decision tree and the random forest algorithm over other algorithms, as these algorithms enhanced the cost-benefit analysis of performance.

Kumar et al. [23] compared several algorithms for predicting effort estimation using Stochastic Gradient, K-Nearest Neighbor (KNN), Decision Tree, Bagging, Random Forest, AdaBoost, and Gradient Neighbor Boosting. The COCOMO'81 and China datasets were used to train the algorithms, and three criteria were used to evaluate the proposed algorithms: mean square error (MSE), root mean square error (RMSE), and R2. The comparison results showed the superiority of the gradient boosting regression algorithm compared to other algorithms in predicting software cost estimates.

Rahman et al. [24] compared decision tree, support vector regression (SVR), and K-nearest neighbor (KNN) algorithms for software cost estimation. They used Edusoft Consulted LTD datasets. The data was processed and analyzed, and the proposed algorithms were trained. The criteria of mean absolute error (MAE), mean square error (MSE), and R-square were used to test the proposed model. The results showed that the decision tree algorithm was superior in prediction to other algorithms.

Sharma et al. [25] compared algorithms for cost estimation forecasting where they compared Local Neighborhood Information-based Neural Network (LNI-NN), Fuzzy-based Neural Network (NFL), GA-based Adaptive Neural Network (AGANN), and GEHO-based NFN. To complete the comparison, the COCOMO81, COCOMONasaV1, COCOMONasaV2, China, and Desharnais datasets were used. The effectiveness of the algorithms was tested using four criteria: mean relative error (MMRE), root mean square error (RMSE), mean magnitude relative error (MdMRE), and prediction accuracy (PRED).

Zhang et al. [26] used the XGBoost algorithm to predict software cost estimation using machine autoencoders on COCOMO81 and Albrecht and Desharnais datasets. They

analyzed the data used to remove outliers and used regression trees to fill in the missing features. To evaluate the proposed model, three famous criteria were used: MMRE, MdMRE, and PRED. The prediction results for the model were 0.21, 0.16, and 0.71, respectively.

Many of the challenges faced by software houses lie in forecasting and estimating the cost of software. From the examination of previous studies, there are several challenges, as the forecasting accuracy of software cost estimation was not sufficient and effective enough to make an accurate forecast. Also, the studies used a very small number of data sets to train and test the proposed models. Therefore, this study seeks to build a model to predict cost estimation through the ARIMA algorithm using five datasets COCOMO81, COCOMONasaV1, COCOMONasaV2, China, and Desharnais. Data sets were collected from the PROMISE repository to be displayed and analyzed, and the correlation between them was found to predict future values using time series, and then the proposed algorithm was applied to them. The proposed model was evaluated using four evaluation criteria: mean relative error (MMRE), root mean square error (RMSE), and mean magnitude were used. Relative error (MdMRE), and prediction accuracy (PRED). The study showed promising results that avoided the challenges faced by previous studies.

III. PROPOSED MODEL

The process of software cost estimation prediction is crucial in the software industry, so many studies have sought to predict it, but they have not been sufficient and effective in completing this process. Therefore, this study seeks, through the use of artificial intelligence algorithms, to build a model that can predict cost estimates. The process is done by collecting data from the Promise warehouse, displaying it, visualizing it, and analyzing it to understand it. Then link them together through time series to predict future values. The data is divided in fixed proportions into two groups to conduct the training process for the ARIMA algorithm. Followed by a scaling process to make all values at one close level to avoid the model ignoring values during the training process. The algorithm is trained on data sets, followed by a testing process to ensure the effectiveness and accuracy of the proposed model. This process is done using four criteria, as shown in Fig. 1.

The study faced several challenges during the implementation process. The quality of the data was not sufficient to complete the process and represented the biggest challenge during the implementation process, as noise and missing values were removed and the data was processed to understand it. There were also values in the data that were higher than the rest of the values, which represented another challenge and were addressed using data scaling to keep all the data at one level so that the model would not ignore them during the training process.

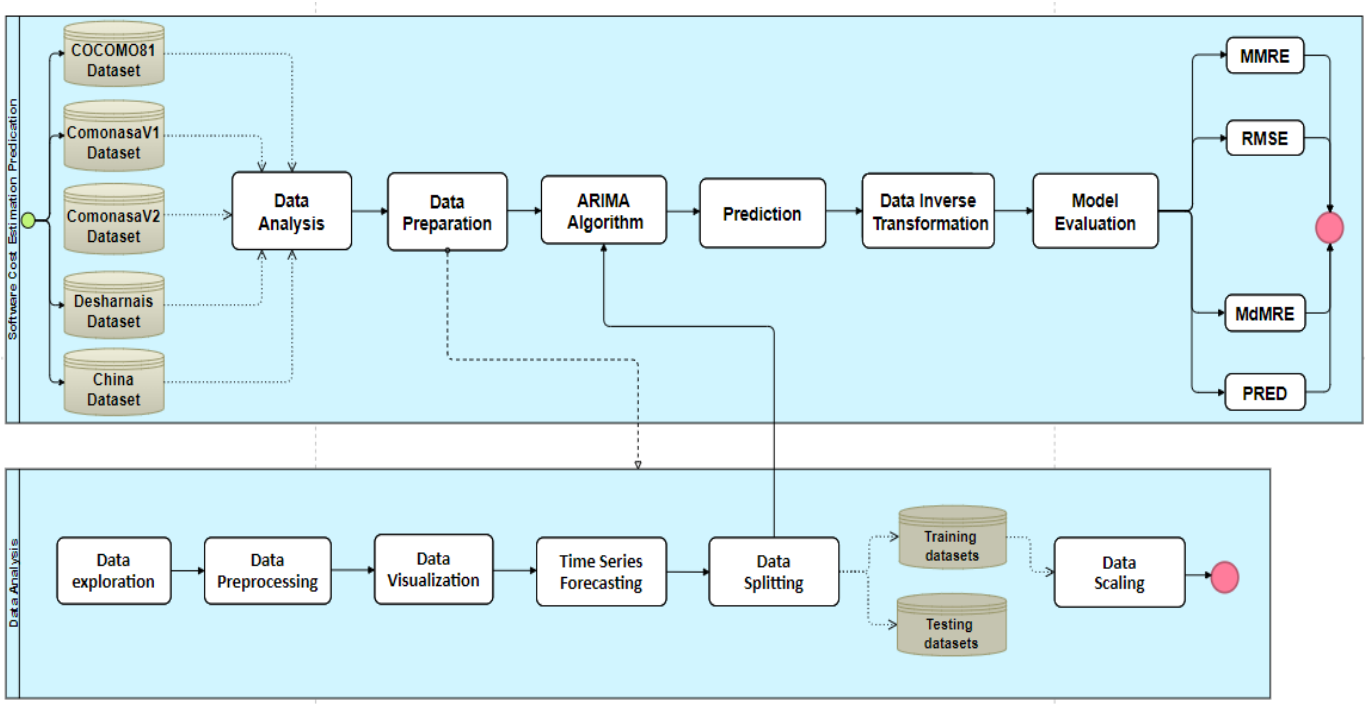

Fig. 1. The proposed model for software cost estimation predication process.

A. Datasets

The experiment was conducted using very popular and freely available datasets. They have been used previously in numerous studies to predict cost estimation. Collected from the Promise repository are the COCOMO81, COCOMONasaV1, COCOMONasaV2, Desharnais, and China. Their sizes range from 60 to 499. While the number of its features ranges between 10 and 24. The effort of these groups is measured in units of person-hour or person-month as shown in Table I.

TABLE I. STATISTICS OF THE DATASETS

Datasets	Source Repository	No. of projects	No. of Features	No. of missing values	Output Attribute-Effort (Unit)
COCOMO81	PROMISE	63	17	0	Person-month
COCOMONasaV1	PROMISE	60	17	0	Person-month
COCOMONasaV2	PROMISE	93	24	0	Person-month
Desharnais	PROMISE	81	10	4	Person- hours
China	PROMISE	499	15	0	Person- hours

B. Data Analysis

The data analysis [27] process is an important element, as data quality represents a major challenge in the training process, so the study focused on every step of exploring the data, as well as processing it to remove noise from it, and then visualizing it. It also used time series technology to connect them predict future values, and then divide the data into training and test sets to train the ARIMA algorithm.

1) Data exploration: It is a very important process to explore data, as it consists of understanding the data as it represents a statistical distribution. This process was done by uploading the data adding it to Google Drive and loaded to Google Colab [28]. To be explored through the info() function. The function gives the number of lines and columns for each data set and also checks whether it contains null values. It also indicates whether the values are numeric or textual. To maintain the state of the data in that form, the data is copied using the copy() function, and the original data is preserved.

2) Data preprocessing: The data processing process is an important step used to remove noisy data and missing values to prepare it for training from the raw data. First, noisy or erroneous data is identified and removed or corrected to ensure the quality of the data. Missing, anomalous, or extreme values negatively affecting the model's operation are discovered, treated, or removed [29]. To convert data into numeric values, text and non-numeric values are converted to numeric values. The index value was also determined to be the basic feature on which the prediction process depends.

3) Data visualization: Data representation plays an important role in the data analysis process. Through graphics such as graphs, animations, charts, and visual representations, what the data presents can be understood more clearly and confirm the structure and format of the data. Visual displays of information convey complex data relationships and data-based insights in an easy-to-understand manner. The Corr() function is also used to confirm the format of the data and discover the extent of correlation between features to produce values that represent the extent of the correlation. If the result is 1, this means that the correlation between the features is very high and ideal, but if the value is zero, it means that there is no correlation between them. If the value is negative, this means that the relationship is inverse between the two properties. All of this contributes to obtaining a deep understanding of the data, making distinctive engineering decisions, and building a highly efficient predictive model.

4) Time series forecasting: Time series is a basic technique for data learning and is one of the most popular data science techniques in the world of statistics and machine learning. It aims to provide an analytical approach by examining observations of past data to predict future values. The idea of time series is based on taking advantage of the time dimension as an essential factor for linking data points. The time column is converted to a historical and chronological format, where the data is indexed while maintaining the original time order. This structured format allows time series models to capture the temporal dynamics and inherent autocorrelation between the data. It is used in various fields such as finance, economics, and engineering. It involves a comprehensive analysis of sequential data points to identify underlying patterns, trends, and dependencies [30]. Forecasting is done through the time dimension as a basic factor for linking data in the form of time series by converting the time column into a historical and temporal format, where the data is indexed while maintaining the temporal order established as an indicator of the data sequence.

5) Data splitting and scaling: The datasets are split 80-20% and are used for training and testing respectively. The largest percentage is used in the training process to allow the model to learn the basic patterns and relationships between the data, ensuring its ability to make reliable predictions on new samples that have not been seen before. While the rest of the percentage is used in the testing process to ensure the accuracy of the proposed model [31]. In addition, the data is scaled to place it in a specific range or scale to ensure that all features have equal importance in the analysis to avoid the dominance of some features during the analysis process due to their high values, to avoid overfitting and the model.

C. The Proposed ARIMA Algorithm

The Auto-Regressive Integrated Moving Average (ARIMA) is a powerful statistical tool utilized in the field of time series analysis and forecasting. It introduced by Box and Jenkins in their seminal work, captures various temporal structures by integrating three primary components: Auto regression (AR), Differencing (I), and Moving Average (MA). It is particularly powerful due to its flexibility in modeling a wide range of time series behaviors, from simple trends to complex seasonal patterns [32].

The Autoregressive (AR) component of an ARIMA specifies that the current value of the time series is a linear function of its previous values. The term "autoregressive" indicates that the model regresses the variable on its prior values. The order of the AR component, denoted by p, signifies the number of lagged observations included in the model. [33] The general form of the AR(p) model is given by Eq. (1):

$$X_t = \emptyset_1 X_{t-1} + \emptyset_2 X_{t-2} + \cdots + \emptyset_p X_{t-p} + \in_t \quad (1)$$

Where X_t represents the value of the time series at time t, $\emptyset_1, \emptyset_2, \ldots, \emptyset_p$ are the coefficients of the model, and $\in_t$ is a white noise error term, which is assumed to have zero mean and constant variance.

The coefficients $\emptyset_1, \emptyset_2, \ldots, \emptyset_p$ determine the influence of past values on the current value. For example, in an AR(1) model (p=1), the current value X_t is directly proportional to the immediately preceding value X_{t-1} plus a stochastic error term $\in_t$.

The Integrated (I) component addresses the non-stationarity in the time series by differencing the data. Stationarity is a key property in time series analysis, implying that the statistical properties of the series do not change over time [34]. Non-stationary data can exhibit trends, seasonal patterns, or other structures that make them unsuitable for traditional time series models without transformation. Differencing is a technique to remove these non-stationary components. The order of differencing required to achieve stationarity is denoted by d. The first differenced series is defined through Eq. (2):

$$\Delta X_t = X_t - X_{t-1} \quad (2)$$

For higher-order differencing, the operation is applied recursively. For example, second-order differencing (d=2) is given by Eq. (3):

$$\Delta^2 X_t = \Delta(\Delta X_t) = (X_t - X_{t-1}) - (X_{t-1} - X_{t-2}) = X_t - 2X_{t-1} - X_{t-2} \quad (3)$$

Differencing transforms a non-stationary series into a stationary one, making it suitable for modeling with AR and MA components.

The Moving Average (MA) component models the dependency between an observation and a residual error from a moving average model applied to lag observations. The order q of the MA model indicates the number of lagged forecast errors included in the model. The general form of the MA(q) model is expressed as shown in Eq. (4):

$$X_t = \in_t + \theta_1 \in_{t-1} + \theta_2 \in_{t-2} + \cdots + \theta_q \in_{t-q} \quad (4)$$

Where $\theta_1, \theta_2, \ldots, \theta_q$ are the parameters of MA, and $\in_t$ is a white noise term.

Unlike the AR model, which uses past values of the series, the MA model uses past forecast errors. These errors capture the unexpected movements in the time series, and the MA component accounts for these by adjusting the model based on past error terms.

Combining these three components, the ARIMA model is denoted as ARIMA(p,d,q), where p is the number of lag observations (autoregressive terms), d is the number of times the raw observations are differenced, q is the size of the moving average window.

The general form of the ARIMA(p,d,q) model is as Eq. (5):

$$\Delta^d X_t = \emptyset_1 \Delta^d X_{t-1} + \emptyset_2 \Delta^d X_{t-2} + \cdots + \emptyset_p \Delta^d X_{t-p} + \in_t + \theta_1 \in_{t-1} + \theta_2 \in_{t-2} + \cdots + \theta_q \in_{t-q} \quad (5)$$

Where $\Delta^d X_t$ represents the d-th differenced value of X_t.

Using an ARIMA model for time series forecasting involves several critical steps. These steps ensure that the model is appropriate for the data and that the predictions are reliable. The process includes model identification, parameter estimation, and model diagnostic checking.

1) Model identification:

a) Stationarity Testing: A key assumption of the ARIMA model is that the time series data should be stationary. Stationarity implies that the statistical properties of the series (mean, variance) do not change over time.

- Visual Inspection: Plot the time series data to visually inspect for trends or seasonality.
- Statistical Testing: Apply the Augmented Dickey-Fuller (ADF) test to statistically verify stationarity.

b) Selecting p and q: Use the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots to identify potential values for p (AR terms) and q (MA terms).

- ACF Plot: Indicates the correlation between the time series with its own lagged values.
- PACF Plot: Indicates the partial correlation of the time series with its own lagged values, controlling for the values of the time series at all shorter lags.

Significant spikes in the ACF and PACF plots suggest the values for q and p, respectively. In this study the potential values of p, q are 1, and also d is 1.

2) Parameter estimation: Estimate the parameters ϕ (AR coefficients), θ (MA coefficients), and other model coefficients using methods such as Maximum Likelihood Estimation (MLE).

3) Model diagnostic checking: Analyze the residuals of the fitted model to ensure they resemble white noise (i.e., they have a constant mean, constant variance, and no autocorrelation).

- Ljung-Box Test: Test for autocorrelation in residuals.
- Residual Plots: Plot the residuals to check for patterns.

D. Data Inverse Transformation

When analyzing the data at the beginning of the experiment, the data are scaled such that large values are rounded to the same range so that the proposed model does not ignore some values or overfitting occurs [35]. The scale transformation is reversed to return the model output to the original data scale. This process is important in the real world, as the measured data may not be interpretable in the original context. Therefore, the data is carefully measured, the proposed

algorithm is trained, and then the data is returned to its original form. This process is done using the inverseTransform() function. This process provides good, actionable insights into their original field.

IV. Evaluation and Discussion

After building the model and training it on the data sets that were divided by 20-80%, the testing process is carried out to ensure the effectiveness of the proposed model, where a computer is used with precise specifications that are explained in the experiment preparation section, by also evaluating it using four famous standards, which are MMRE, RMSE, MdMRE, and PRED are explained in the evaluation criteria section. A section was also added explaining the results of the experiment, as well as a section to discuss the results and comparison with previous studies under the same conditions on the same datasets.

A. Experimental Setup

The experiment was conducted on a laptop PC with an Intel Core i7 CPU, 64GB of RAM, and an NVIDIA GTX 1050i GPU. The datasets were also divided by 20-80%, with the largest percentage being used in the model training process, while the rest of the data was used in the model testing and evaluation process. The experiment was conducted through several tools. Google Drive was used to upload data sets for the experiment to it and then uploaded to Google Colab to conduct the experiment. This study used the Python language to present, describe, represent, and analyze the data used, train the algorithm, and then test it.

B. Evaluation Criteria

After completing training the model on the proposed algorithm. The model testing process is a crucial step to ensure the accuracy and effectiveness of the model. The estimation process is done using four famous criteria, which are mean magnitude of relative error (MMRE) [16], root mean square error (RMSE) [17], mean magnitude of relative error (MdMRE) [18], and prediction (PRED) [19].

1) Mean Magnitude Relative Error (MMRE): It is one of the most popular forecasting benchmarks and is used in software engineering forecasting to calculate the average relative difference between actual and predicted values. It is represented by Eq. (6) and Eq. (7):

$$\mathrm{MRE} = \frac{|\text{Actual effort} - \text{Estimated effort}|}{\text{Actual effort}} \times 100 \quad (6)$$

$$\mathrm{MMRE} = \frac{1}{\mathrm{M}} \sum_{1}^{\mathrm{M}} \mathrm{MRE} \quad (7)$$

Where m is the total data points and $\sum$ denotes the sum of values in the entire dataset [16].

2) Root Mean Square Error (RMSE): It is widely used in forecasting operations, as it represents the average size of the difference between the actual and expected values, and it needs the actual expected and corresponding values to calculate it, and this is done through Eq. (8).

$$RMSE = \sqrt{\frac{\sum (P_i - Oi)^2}{n}} \quad (8)$$

Where n is the number of data points, P is the expected value, O is the actual value, and ^2 denotes the squared difference [17].

3) Mean Magnitude of Relative Error (MdMRE): It is a statistical measure of prediction and is similar to the average size, except that it calculates the absolute average and is measured by determining the relative error for each prediction and finding the absolute difference between the actual and expected values. Then the relative error is arranged in ascending order through the following equation, which is used to calculate the error = |(P - A)| /A [18].

4) PRED: It is one of the most famous and widespread metrics as it indicates the accuracy of the model and its value increases as the accuracy of the model improves. It is expressed as a percentage in projects where the percentage of expected values matches the actual values and can be measured through Eq. (9)

$$PRED = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{Estimation\ Effort - Actual\ Effort}{Actual\ Effort} \right| K\% \quad (9)$$

Where k% is the percentage of error between the actual estimate and the effort estimate [19].

C. Experimental Results

The four criteria described in the previous section were used to test the model to measure its effectiveness and accuracy in predicting software cost estimates, as the experiment showed promising results on the five datasets used as shown in Table II.

TABLE II. The Results of the Proposed Model on the Five Datasets

Method	Metrics	COCOMO81	COCOMONasaV1	COCOMONasaV2	China	Desharnais
The proposed model	MMRE	0.07613	0.02227	0.01035	0.00001	0.00004
	RMSE	0.04999	0.02899	0.00650	0.00430	0.00339
	MdMRE	0.03813	0.01113	0.00517	0.00008	0.00002
	PRED	95.0	97.1	99.35	99.57	99.6

Table II displays the software cost estimation prediction rates of the proposed model on the five datasets using the four evaluation criteria, where the results show very promising prediction and low value of error rates. The COCOMO81 and COCOMONasaV1 datasets show very good percentages in reducing error rates and also promising percentages in prediction accuracy, reaching 95% for the COCOMO81 data set and 97.1 for the COCOMONasaV1 data set. While the ratios were very unique and significantly distinct for the COCOMONasaV2, China, and Desharnais datasets. The error rates recorded the lowest possible rates, almost noticeable,

while the prediction accuracy recorded rates exceeding 99% for the three datasets.

D. Comparison and Discussion

To ensure the effectiveness of the proposed model, it is compared with other models under the same conditions described, such as using the same datasets as well as the evaluation criteria used. The model was compared with recent studies. It was compared to the Sharma [25] model, which was used by four algorithms in its study: local mutual information-based neural network (LNI-NN), fuzzy-based neural network (NFL), GA-based adaptive neural network (AGANN), and GEHO-based NFN (GEHO-NN) for software cost estimation. It was also compared with the model of Zhang et al. [26] who used the XGBoost algorithm under the same conditions. Table III shows a comparison between the proposed model and previous models.

TABLE III. COMPARISON BETWEEN THE PROPOSED MODEL AND THE STATE-OF-THE-ART

Method	Metrics	COCOMO81	COCOMONasaV1	COCOMONasaV2	China	Desharnais
LNI-based NN [25]	**MMRE**	0.224	0.243	0.225	0.240	0.32
	RMSE	0.261	0.183	0.383	0.148	0.312
	MdMRE	0.256	0.249	0.249	0.255	0.336
	PRED	28.51	50	50	44	22.22
Neuro-fuzzy logic [25]	**MMRE**	0.213	0.236	0.196	0.220	0.296
	RMSE	0.178	0.131	0.290	0.075	0.173
	MdMRE	0.256	0.215	0.215	0.240	0.223
	PRED	29.92	62	62	70	32
Adaptive GA-based NN [25]	**MMRE**	0.199	0.231	0.174	0.192	0.197
	RMSE	0.130	0.065	0.232	0.056	0.111
	MdMRE	0.235	0.172	0.172	0.218	0.181
	PRED	46.15	73.87	70	76	47.05
GEHO-based NFN [25]	**MMRE**	0.174	0.220	0.128	0.167	0.112
	RMSE	0.055	0.060	0.960	0.39	0.060
	MdMRE	0.223	0.130	0.130	0.168	0.100
	PRED	57.14	83.14	83.14	84	88.23
XGBoost [26]	**MMRE**	0.21	0.37	-	-	0.38
	RMSE	-	-	-	-	-
	MdMRE	0.16	0.36	-	-	0.37
	PRED	71	37	-	-	22
The proposed model	**MMRE**	**0.07613**	**0.02227**	**0.01035**	**0.00001**	**0.00004**
	RMSE	**0.04999**	**0.02899**	**0.00650**	**0.00430**	**0.00339**
	MdMRE	**0.03813**	**0.01113**	**0.00517**	**0.00008**	**0.00002**
	PRED	**95.0**	**97.1**	**99.35**	**99.57**	**99.6**

Table III highlights the comparison between the proposed model and other models from previous studies during 2023 and 2024. The comparison shows the superiority of the proposed model in predicting software cost estimation compared to previous models. The model excelled in reducing the error rates in the five datasets and increasing the prediction accuracy of the software estimate, which ranged from 95 to over 99%. The error rates also decreased on the MMRE, RMSE, and MdMRE criteria.

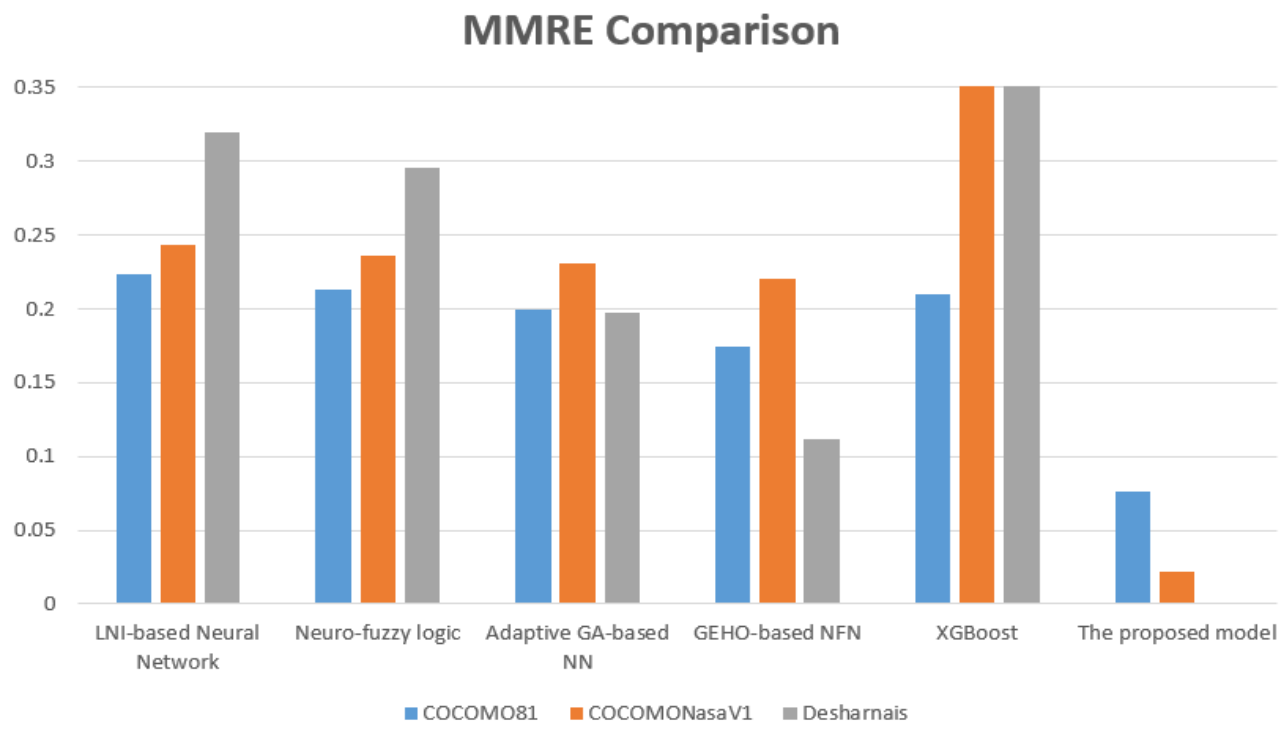

Fig. 2. Comparison between the proposed model and others on the MMRE measure.

Fig. 2 shows a comparison between the proposed model and previous studies according to the MMRE standard, where the results show significant superiority of the proposed algorithm. The COCOMONasaV2 and China datasets were excluded from the study since they were not used in Zhang's [26] study. However, there is a big difference in reducing error rates for the targeted study, as it showed a significant and distinct absence of error rates with the Desharnais data set, while the rates were very good and promising also for the COCOMO81 and COCOMONasaV1 datasets. The percentages were also clearly and prominently distinct according to the MdMRE standard, as the error rate decreased significantly and clearly for the three data sets. It decreased by a large and clear percentage for the COCOMONasaV1 and Desharnais datasets, and the decrease rates were also very good for the COCOMO81 dataset as shown in Fig. 3.

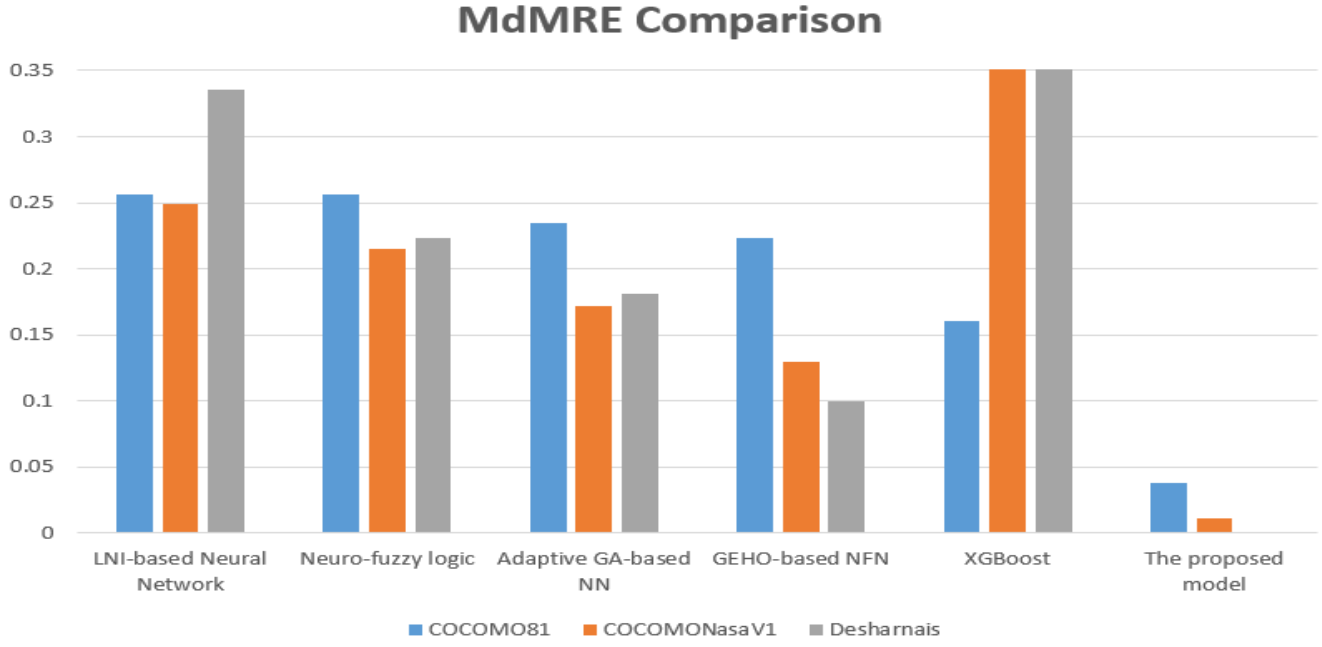

Fig. 3. Comparison between the proposed model and others on the MdMRE measure.

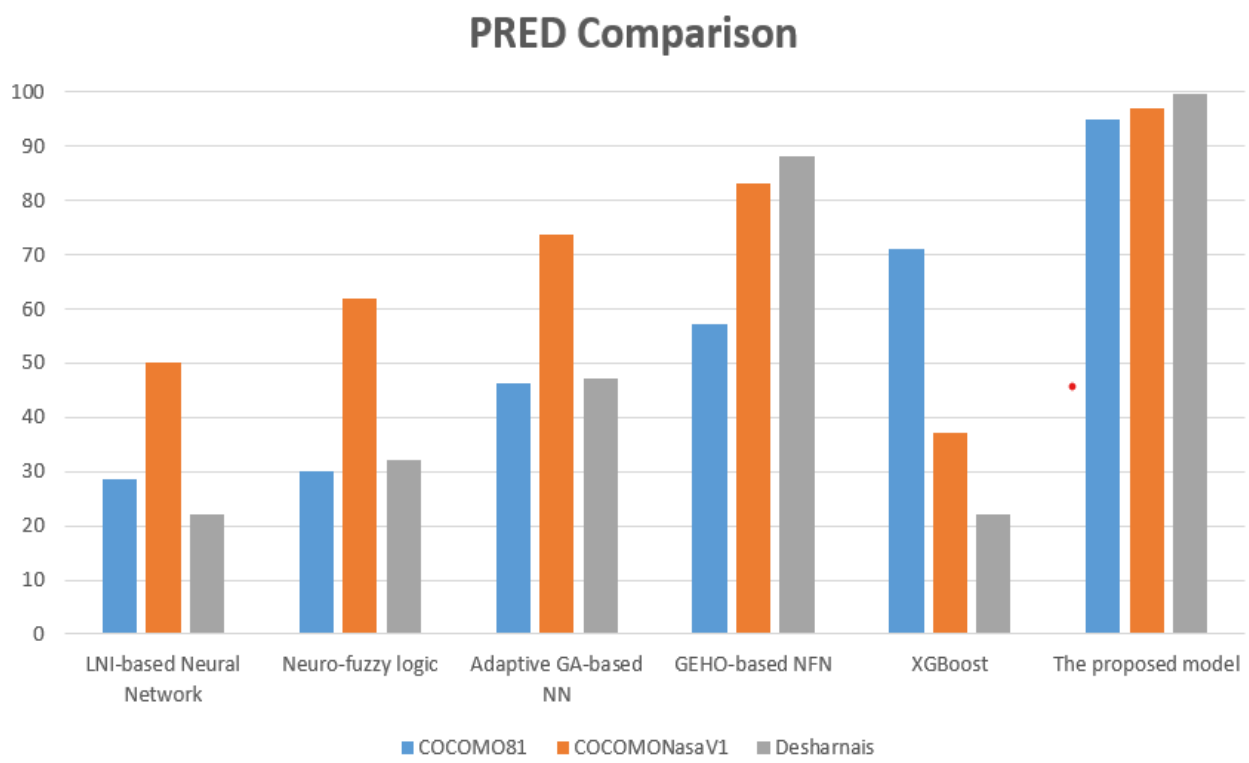

Fig. 4. Comparison between the proposed model and others on the PRED metric.

The most significant difference in explaining excellence was the PRED standard, which is more commonly used in detection and prediction processes. The results of the study showed a very prominent and clear distinction compared to previous studies, in which the prediction percentages ranged between 22 as the lowest prediction percentage and 88.23 as the highest percentage reached by the studies. However, the results of the proposed model were very promising, as the COCOMO81 data set recorded an accuracy rate for predicting the software cost estimate of 95%, while the COCOMONasaV1 data set recorded an accuracy rate of 97.1%, and the rate was very promising for the Desharnais data set, which recorded the highest percentage so far at 99.6% as shown in Fig. 4.

V. CONCLUSION

The software estimation process is one of the crucial steps today in the software industry, which plays the main role in the software production process. It represents the points of connection between the client's requirements and his budget, in addition to the indicator of controlling the workflow within the software houses on the desired projects. With the spread of the software industry over the past few decades, stakeholders have tended to accelerate the pace of their work to keep pace with the times, which has increased pressure on software houses to implement their work. Therefore, there was an urgent need for models that can estimate the cost of software perfectly. Researchers have created several models to evaluate this, some of which relied on traditional methods or mathematical equations and called them algorithmic methods. Some relied on experts' judgments and opinions and called them non-algorithmic methods. However, some relied in their work on learning techniques such as artificial intelligence, including machine learning and deep learning methods. However, previous studies have shown that prediction rates are not stable and sufficient to complete the process, so the need to create new models was very urgent. This study seeks to build and present a model that can predict software cost estimation using the ARIMA algorithm on five datasets, namely COCOMO81, COCOMONasaV1, COCOMONasaV2, China and the Desharnais dataset. The data was collected, presented, and processed to remove noise and missing values. It was also analyzed and visualized to identify and link them. The data is linked using time series technology to predict the future values of the data, and the process is very effective in increasing the model's performance. The data was split 80-20 for training and testing. The proposed model will be trained and tested on data sets. The model was evaluated using four popular prediction criteria, namely MMRE, RMSE, MdMRE, and PRED. The model shows a promising distinction in its results compared to other models, which contributes to reducing risk levels and contributes mainly to good project planning, which contributes effectively to the cost estimation forecasting process.

Although the model is distinguished in its work, some limitations must be addressed in the future, especially about data sets, as the model was trained on five data sets. However, we hope to train and test it on other data sets to ensure its effectiveness and accuracy. We also hope to apply it in real-time, which addresses Constant assumptions, computational overhead, and secondary evaluation problems that are used to enhance model response.

In future work, we seek to transform the model into a tool through which the project data can be entered, which are the

client's requirements in addition to the cost factors so that the user obtains an output estimating the effort and cost necessary to build the project. We also seek to train the model and test it on other data sets, as well as in real-time.

REFERENCES

[1] B. Khan, W. Khan, M. Arshad, and N. Jan, "Software Cost Estimation: Algorithmic and Non-Algorithmic Approaches", International Journal of Data Science and Advanced Analytics, vol. 2, no. 2, 2022.

[2] M. M. Draz, O. Emam, and Safaa M. Azzam, "Software cost estimation predication using a convolutional neural network and particle swarm optimization algorithm", Scientific Reports, vol. 14, no. 1, pp. 13129, 2024.

[3] W. Zhang, et al., "Dimensionality reduction and machine learning based model of software cost estimation", Frontiers in Physics, vol. 12, 2024.

[4] M. M. Draz, O. Emam, and S. Azzam, "Five Decades of Software Cost Estimation Models: A survey", FCI-H Informatics Bulletin, 2024.

[5] Ghafory, Hamayoon, and F. A. Sahnosh, "The review of software cost estimation model: SLIM", Int. J. Adv. Acad. Stud, vol. 2, pp. 511-515, 2020.

[6] R. Christopher and R. Roy, "Expert judgment in cost estimating: Modelling the reasoning process", Concurrent Engineering, vol.9, no. 4, pp. 271-284, 2001.

[7] L. Jooon-kil and Ki-Tae Kwon, "Software cost estimation using SVR based on immune algorithm", 2009 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking, and Parallel/Distributed Computing. IEEE, 2009.

[8] Corazza and Anna, "Using tabu search to configure support vector regression for effort estimation", Empirical Software Engineering, vol. 18, pp. 506-546, 2013.

[9] M. Isa, L. Ebrahimi, and F. Gharehchopogh, "A hybrid approach of firefly and genetic algorithms in software cost estimation", MAGNT Research Report, pp. 372-388, 2014.

[10] Ritu and Pankaj Bhambri, "Software effort estimation with machine learning–A systematic literature review", Agile software development: Trends, challenges and applications, pp. 291-308, 2023.

[11] Sreekanth, "Evaluation of estimation in software development using deep learning-modified neural network", Applied Nanoscience, vol. 13, no. 3, pp. 2405-2417, 2023.

[12] F. Nirodha, K. Dilshan, and H. Zhang, "An artificial neural network (ANN) approach for early cost estimation of concrete bridge systems in developing countries: the case of Sri Lanka", Journal of Financial Management of Property and Construction, vol. 29, no. 1, pp. 23-51, 2024.

[13] Ritu and P. Bhambri, "Software effort estimation with machine learning: A systematic literature review", agile software development: Trends, challenges and applications, pp. 291-308, 2019.

[14] Shumway and H. Robert, "ARIMA models: Time series analysis and its applications: with R examples", pp. 75-163, 2017.

[15] Promise Repository. [Online]. Available: http://promise.site.uottawa.ca/SERepository/datasets-page.html (accessed: April. 12 2024).

[16] J. Magne, T. Halkjelsvik, and K. Liestol, "When should we (not) use the mean magnitude of relative error (MMRE) as an error measure in software development effort estimation?", Information and Software Technology, vol.143, 106784, 2022.

[17] Hodson and O. Timothy, "Root mean square error (RMSE) or mean absolute error (MAE): When to use them or not.", Geoscientific Model Development Discussions, pp. 1-10, 2022.

[18] G. Somya and P. K. Bhatia, "A non-linear technique for effective software effort estimation using multi-layer perceptrons, 2019 International Conference on Machine Learning", Big Data, Cloud and Parallel Computing (COMITCon). IEEE, 2019.

[19] A. Idri, I. Abnane, and A. Abran, "Evaluating pred (p) and standardized accuracy criteria in software development effort estimation", Journal of Software: Evolution and Process, vol. 30, no. 4, 2018.

[20] S. V. Singh, L. U. BBDITM, H. K. Shukla, and R. B. Singh, "Cost Estimation of Software by ANFIS based Artificial Intelligence Approach", IJRDASE, vol. 21, no. 1, 2021.

[21] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, "An effective approach for software project effort and duration estimation with machine learning algorithms", J Syst Softw, vol. 137, pp.184–196, 2018.

[22] Vo., Van. et al. "Toward improving the efficiency of software development effort estimation via clustering analysis", IEEE Access, vol. 10, pp. 83249-83264, 2022.

[23] P. Kumar, H. Behera, J. Nayak, and B. Naik, "A pragmatic ensemble learning approach for effective software effort estimation", Innovations in Systems and Software Engineering, vol. 18, no. 2, pp. 283–299, 2022.

[24] M. Rahman et al., "Software effort estimation using machine learning technique", International Journal of Advanced Computer Science and Applications, vol. 14, 2023.

[25] S. Sharma and S. Vijayvargiya, "Modeling of software project effort estimation: a comparative performance evaluation of optimized soft computing-based methods", International Journal of Information Technology, vol 14, no. 5, 2487-2496, 2022.

[26] Zhang et al., "Dimensionality reduction and machine learning based model of software cost estimation", Frontiers in Physics, vol. 12, 2024.

[27] Hazari and Animesh, "Data Analysis: Descriptive and Analytical Statistics." Research Methodology for Allied Health Professionals: A comprehensive guide to Thesis & Dissertation", Singapore: Springer Nature Singapore, pp. 79-98., 2024.

[28] Google Colab. [Online]. Available: https://colab.google (accessed: April. 15 2024).

[29] A. Samer et al., "Artificial intelligence and machine learning overview in pathology & laboratory medicine: A general review of data preprocessing and basic supervised concepts", Seminars in Diagnostic Pathology, vol. 40, no. 2, 2023.

[30] J. Sanchez, "Time Series for Data Scientists: Data Management, Description, Modeling and Forecasting", Cambridge University Press, 2023.

[31] J. J. Faraway, "Does data splitting improve prediction?,", Statistics and Computing, vol. 26, pp. 49-60, 2016.

[32] I. V. Kontopoulou, et al., "A review of ARIMA vs. machine learning approaches for time series forecasting in data driven networks", Future Internet, vol. 15, no. 8 , pp. 255, 2023.

[33] C. Liu, S. C. Hoi, P. Zhao, and J. Sun, "Online arima algorithms for time series prediction", In Proceedings of the AAAI conference on artificial intelligence, vol. 30, no. 1, 2016.

[34] J. Fattah, L. Ezzine, Z. Aman, H. El Moussami and A. Lachhab, "Forecasting of demand using ARIMA model", International Journal of Engineering Business Management, vol. 10, 2018.

[35] Yan and Yanjun, "Inverse data transformation for change detection in wind turbine diagnostics", 2009 Canadian Conference on Electrical and Computer Engineering, IEEE, 2009.