Int. J. Nonlinear Anal. Appl. 13 (2022) 2, 2121–2131
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2022.26974.3505

Time series forecast modeling for the Windows operating system performance using Box-Jenkins and LSTM models

Murtadha J. Assi*, Assmaa A. Fahad, Basad Al-Sarray

Department of Computer Science, University of Baghdad, Baghdad, Iraq

(Communicated by Mouquan Shen)

Abstract

Performance issues such as system resources leaking, application hang, and Software Aging (SA) can affect the system's reliability and minimize user experiences. Therefore, these issues need to be analyzed and forecasted to prevent incoming issues. Finding the root cause and analyzing the internal behaviors become troublesome due to the complexity of modern systems such as the Microsoft Windows Operating System OS. Microsoft builds multiple tools and platforms such as the Performance Monitor (PerfMon.exe) tool and Performance Counter for Windows (PCW) platform to monitor the activities inside Windows OS. This paper aims to use Windows OS tools for simulating performance issues in an experiment, data collection, and log format converting. In contrast to other works, the deep learning Long Short-Term Memory (LSTM) method and the Auto-regressive Integrated Moving Average (ARIMA) model were generated and compared. The best model that provides the lowest error rate of the prediction simulated performance issue was selected. The results declare the preference of using the ARIMA model with order (2,1,1) that provides the observed lowest error rate for both MAE and RMSE compared with other values in previous lags. And the observed LSTM has an error rate of 4.796, whereas the ARIMA model has an error rate of 0.0119. From those results, we can confirm of using the ARIMA model with its selected parameters can predict the small jump fluctuations behavior observed from the memory metric.

Keywords: Log, Memory leak, PerfMon, Forecast, Time series analysis
2020 MSC: 37M10

1 Introduction

Microsoft Windows is one of the most widely used computer operating systems in the world, it could face multiple performance issues that can make its services go down and increases the opportunity of system failure. Increasing the complexity of the recent software development and their internal operations, cause more challenges in the system reliability and performance need. Furthermore it makes challenges in detecting and forecasting the performance issues in fine-grained fluctuations which leads to significant performance affect overtime. Multiple Windows tools can be used to get an idea from the Windows system. In that concern, it can be a reliable way to forecast the consumption of system resources to prevent a system failure that could lead to loss of data. A memory leak is one of the performance issues, it deviates a service from its path and makes resource leaking which could happen related to an aging effect. Any leaky applications that consume memory due to aging affect can slowly shift the system from normal healthy

*Corresponding author
Email address: mortadha.j15@gmail.com; murtadha.aasi1201@sc.uobaghdad.edu.iq (Murtadha J. Assi)

Received: May 2022 *Accepted:* July 2022

state into abnormal state which become prone to failure. The granularity of the applications activities inside system resources, for example, memory metric that can change in small seconds and milliseconds and hence its behavior is mostly nonseasonal. The taxonomy for the aging-related faults stated [11], and it has been observed for a long time continues software systems [12]. Capturing mechanisms in modern systems, such as Windows systems provides rich log files that contain important system contents of system performance counters and processes activities, etc. [14].

Anomaly detection is an essential task to build reliable computer systems. It becomes challenging as systems and applications get increasingly more complex than ever before [15]. The purpose of predicting a performance issue is worthy to prevent incoming issues and reset the state of an application or system. Time series forecasting models that allow forecasting data depending on previous data, can be applied in multiple areas such as finance engineering, science, economics, management, medicine, etc. The term **time-series analysis** refers to the statistical approach to time series of analyzing trends and seasonality. A time series is a set of observed values that have been recorded, related to a specific variable over a period. It has multiple characteristics, such as trend, seasonality, and irregular or cyclic components. Whereas, one of the time-series properties is the Stationary, which means the series does not change its distribution over time. When a time series is stationary, it means that it has no trend (mean is zero) or seasonal variability [4].

Deep learning and ARIMA models are frequently compared in related works for time series data prediction [10], [3], and [7].Many research papers focused on performance analysis are problem-oriented and highlight a specific problem, e.g. specializing in system bottlenecks and resource usage [1], software optimization [5], and performance aging issues [9] [16], etc.

K.Gencer and F.Basciftci in [10] predicted the vulnerabilities using ARIMA model with the order (4,1,4) as the best model for prediction with error measure 18.449, whereas the error measure of the LSTM model is 26.830. I. Umesh et al. in [16] used time series forecasting for the aging related bugs performance issue. They used Moving Average by creating series of averages of different subsets, both cpu usage and memory availability has been used as performance indicators to predict the loading in the system.

In this article, we predict the performance issue at a specific time that represent the critical section of the loading. The design of the experiment is done to simulate the performance issue using couple of Windows OS tools for processes loading and data collecting. The memory usage that represented the most pron performance metric has been selected to collect its data during the loading. Collecting the series data of the system and processes inside memory has been done using PerfMon.exe tool.

The collected log data set is challenging because the the series data in seconds granularity behaviors inside memory, the time series analysis concepts has been used to understand the series data. Building the ARIMA model with the best correlated number of lags to check its prediction with a set of seconds observation. Furthermore, using the LSTM model to check if it work well with the seconds changes behavior inside memory, compared with ARIMA model.

The next sections of this paper are sorted as follows. The design of the experiment is presented in section 2. In section 3 Statistical Time series formulas conducted; The LSTM model discussed in section 4. The results of ARIMA and LSTM models is presented in Section 5. Section 6 is the Discussion. And Lastly, the Conclusions.

2 Design of the Experiment and Data Collection

To implement the designed system of this thesis, a disk machine with the following configuration is used: - Intel Core i5 8300H 4 cores of CPU - 8GB of RAM Ddr4 Crucial Inc. - Windows 10 Home 20H2 OS

2.1 Process memory in Windows operating systems OS

Both Applications and OS have three crucial ways to run out of memory, which is: committed memory, virtual address space, and physical memory (RAM), the most familiar type of memory leak is one or multiple memory that affects the system-wide commit memory if the system works out of committed memory. So, it can result in the system and/or application hanging. The processes are the most common source that charge the system committed memory, a process has both private and shareable committed memory. The shareable ones such as DLLs and EXEs which does not affect the system commit charge. Whereas the private one is loaded against the system commit charge [13]. Private bytes is a process counter metric that used by process for execution, it is a small amount of memory because it a only the process' data as local variables. The increase in the private bytes of total processes is charge against the system commit in a cumulative increase due to aging effect. It makes the memory metric prone to leaking, and commonly the leaking memory blame due to one or more application processes, as shows in Figure (1).

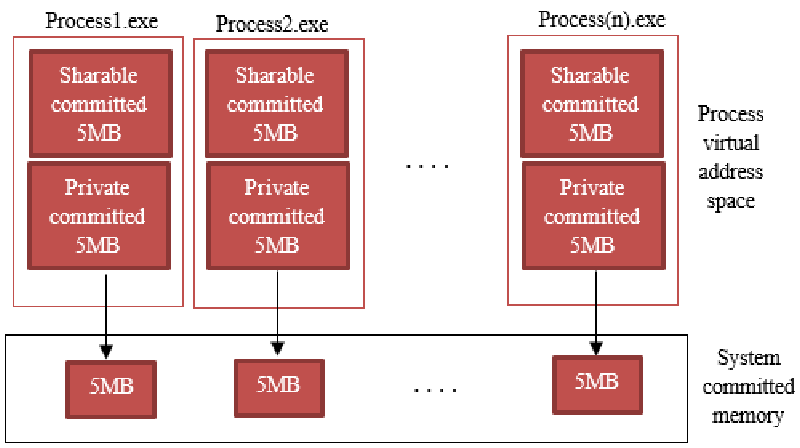

Figure 1: Only private bytes memory charge against system committed memory

To load the system commit metric, there are a couple of bench marking tools such as testlimit.exe command-line tool of Sysinternals [13]. Figure (2). presents the testlimit.exe command with specific parameters to load the private bytes metric. Further more, there are multiple applications they were running at the same time of running testlimit.exe such as Chrome browser, MS office, and other applications.

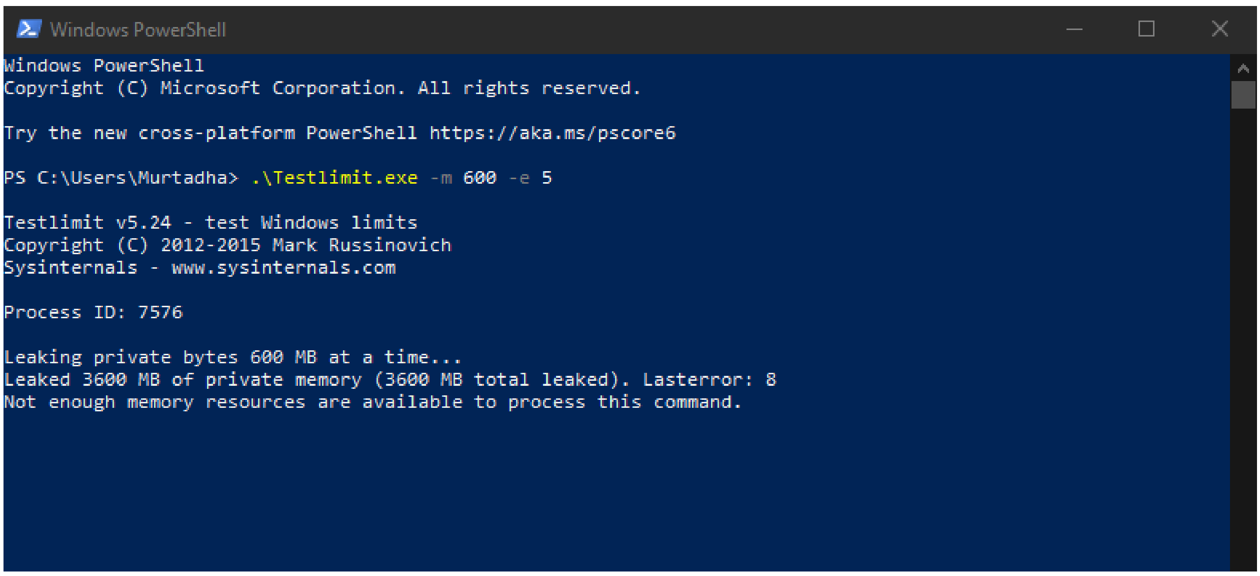

Figure 2: Testlimit.exe leaking private bytes

2.2 Data Collecting

Performance monitor (Perfmon.exe) is one of the most popular built-in Windows systems tools, it is used to monitor and collect the system metrics data. The Private bytes and working set metrics of the total processes are added to the experiment. The experiment takes around 3 hours till hit the system commit and leaked the memory, the observations has been collected every 5 seconds as a frequency.

Figure (3) shows the trend of the experiment state. The output of the PerfMon.exe tool is (.blg) binary log file, and it has been converted to (.csv) file format to deal with it in Python programming language for the time-series analysis. The Relog.exe is a command-line utility that has been used to convert the (.blg) format file into (.csv) format file.

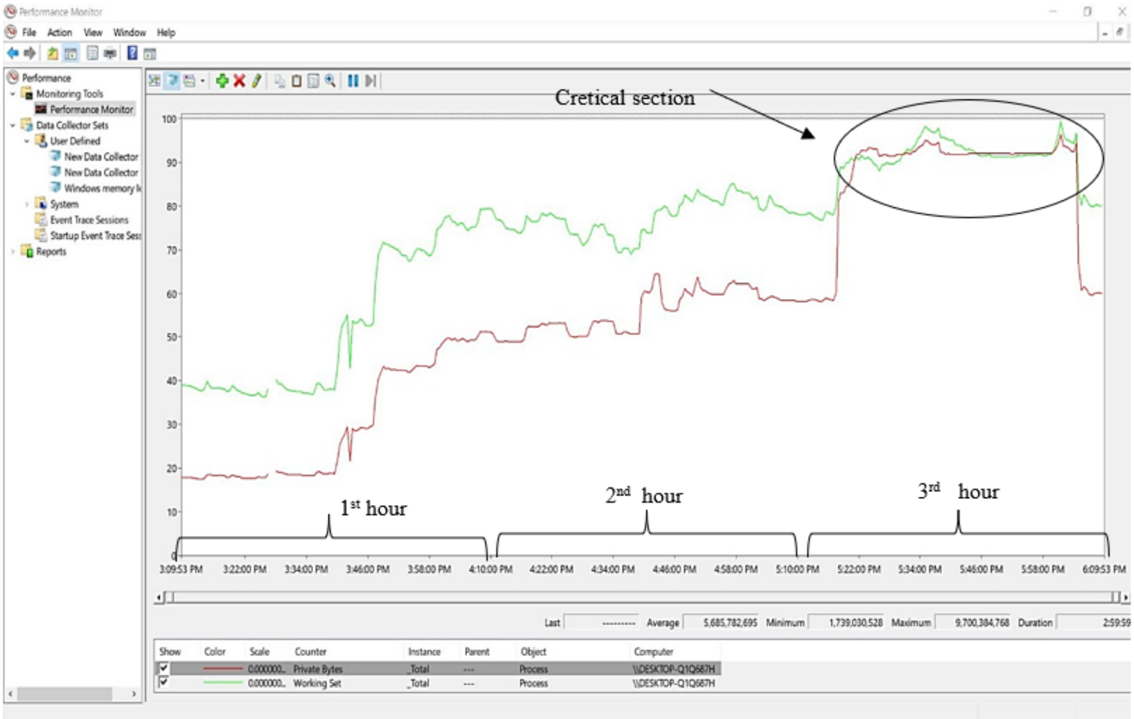

Figure 3: The Private bytes and working set trends over around 3 hours

3 Methodology

Generally, there are two types of time series analysis: univariate and multivariate. The univariate is dealing with one variable, (e.g. the private bytes metric), which is conducted in the study. Whereas the multivariate one deals with looking at the correlation between more than one variable. The following sections explain the statistical ARIMA model and deep learning LSTM model.

3.1 Auto-regressive Integrated Moving Average (ARIMA)

There are many methods to make predictions with the help of time series models, the most one is the ARIMA model, which known as the Box-Jenkins method [10].
The Box Jenkins methodology consists of:

The auto-regressive (AR) model regresses the variable on its own lagged values. the relationship between current and previous observed data. In the case that the time-series is stationary:

$$x_t = c + \sum_{i=1}^{q} \phi_i x_{t-i} + \epsilon_t$$

Where x_t is the Stationary variable, c is a constant, ϕ_i is the model parameter, ϵ_t is the error distributed with the mean 0 and variance σ_ϵ^2, and $\epsilon_t \sim N(0, \sigma_\epsilon^2)$.

The Moving Average (MA) model:

$$x_t = \mu + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$$

Where μ is the expected value of x_t, θ_i is the coefficient(what part the error last period is relevant in explaining the current value), $\epsilon_t \sim N(0, \sigma_\epsilon^2)$ is a noise.

Auto-regressive Moving Average (ARMA) model: The AR(p) and MA(q) can be combined into single model named ARMA (p,q) model, there are two

$$x_t = c + \sum_{i=1}^{q} \phi_i x_{t-i} + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$$

Where ϕ_i is the model parameter(the auto-correlation coefficient of delays), $\phi \neq 0, \theta \neq 0$, and $\sigma_\epsilon^2 > 0$ can be used. Both AR and MA model, using higher number of parameters make its data complex and reduce the activity. With ARMA model can use only lower number of parameters (p,q) [10].

Most of Time series are non-stationary, due to existing some characteristics such as the non-constant mean, and the seasonal. As in the experiment chart of process metrics in figure (3). It has a non-stabilized mean which comes from loading the private commit metric of the process (Testlimit.exe) and the total live processes, which makes the data be non-stationary time series.

Differencing is a process that used for stabilizing the mean of the non-stationary data. Since the data is non-stationary (didn't have stabilized mean), there are either using differencing to stabilize the mean or through the ARIMA model that already has the subtraction parameter degree (d) to do differencing by itself automatically.

The Box-Jenkins or ARIMA (p,d,q) is non stationary model using subtraction (d), it has the goal of using lower number of parameters and relies on parsimony principle. The parameters p and q represents the degrees of the Auto-regressive (AR) model and the Moving Average (MA) model, respectively. And d is the degree of subtraction to integrate the time-series to ensure Stationary. To test the existence of the stationary in the data set the Adfuller test is used which is developed by [8].

The model that used better coefficient lags (delays) should be strong and good for prediction while choosing the number of coefficients is a challenge, this study relies on the Log likelihood ratio test (LLT), Akaike Information criteria (AIC), and lowest Mean Absolute Error (MAE) when comparing two models that give the same variation, then the one with higher LLT and the AIC scores is lower that makes sense to pick the better-fit model. The Mean Absolute Error (MAE) is defined:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|P_i - A_i|$$

Where N, P_i and A_i denoted the total number of prediction intervals, the predicted values, and observed values, respectively.

3.2 Auto-correlation Function (ACF) and Partial Auto-correlation Function (PACF)

- ACF

The ACF used to measure the correlation between the current value and the value at the previous time spot [6]. Typically, internal patterns exist in the univariate time series data which measures how correlated a given sequence is with the lagged version of itself. ACF consider both direct and indirect affects, and it can only take values between (-1, +1). The auto-correlation function at lag k is estimated as the following [2]:

$$\hat{\rho_k} = \frac{\sum_{t=1}^{n-d}(x_t - \bar{x})(x_{t-k} - \bar{x})}{\sum_{t=1}^{n}(x_t - \bar{x})^2}$$

$$\bar{x} = \frac{\sum_{t-1}^{n} x_t}{n}$$

where is n is the sample size , $\bar{x}$ is the mean, and the x_t is the value of the variable.

- PACF

It measure the degree of the relationship between t and x_{t+m} and stable others values x_{t+1}, x_{t+2}, ... x_{t+m-1}. It represent only the direct affect on the lags [2].

$$\gamma_{kk} = \frac{\sum_{i=1}^{k-1}\gamma_{k-1,i}.\gamma_{k-i}}{1 - \sum_{i=1}^{k-1}\gamma_{k-1,i}.\gamma i} \qquad k = 1,2,3 \ ...$$

4 Long Short-Term Memory (LSTM)

LSTM is a deep learning method,it is used for many years and constituted the state-of-the-art for variety applications and problems such as time series data analysis, translation, voice recognition and more[4]. It can maintain the long-term layer information in memory. The limitation in the Recurrent Neural Network RNN of keeping the data in memory for long period of time, was the motivation to build this method. The time series classification communities realized the LSTM model due to their advantage over other models. In terms of classification accuracy, outperform several traditional time series classification models, while requiring minimal pre-processing of the data. The internal architecture of the LSTM presented in Figure(4)

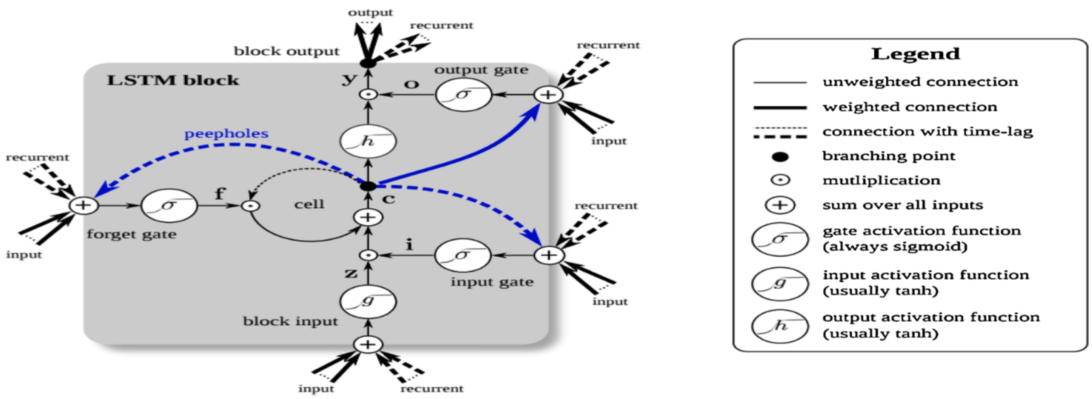

Figure 4: LSTM internal Architecture

The basics contents of LSTM networks is memory blocks, which was invented to tackle vanishing gradients by remembering the parameters of the networks for long time. Gates in LSTM helps in data processing with the help of activation function (sigmoid) and the output is in between 0 or 1. The activation function needed to forward the positive values only. The positive values are needed to move for the next gates to get a clear output.The LSTM model and the equation of the gates are given by [10]. The principle of LSTM works depending on the generator that get number of input and one predicted output. As declared previously, the series data in this article has been observed randomly each 5 seconds, each 12 observations represent 1 minute. So, the series input that has been used in this study is as $[1, 2, 3, ..., 24] > [25] and [25, 26, 27, ..., 48] > [49]$, 24 input and one output. The LSTM model can deserve the earlier inputs in it memory, we will see if the earlier inputs affect the prediction or not.

5 Forecasting Time Series data (Applied practical)

This section aims to discussing the results using both the ARIMA and LSTM models to forecast the memory degradation issue in Windows System to prevent memory leak issue. The Adfuller test applied in this study to check the stationary of the series with the p-value that observed to be larger than (0.05), we can confirm the non-stabilize mean in figure (6) that shows the collected series data of the variable (Private Bytes) in memory. The mean can be stabilized either by the referencing method, or through the Box-Jenkins (ARIMA) model.We tested the hypothesis if the result test greater than 5 percent of the Adfuller table then $H_0 : \theta >= 1$ is Non-stationary versus $H_0 : \theta < 1$ then it is stationary. The result shows that the test statistic result of our raw data is -1.61 which is greater than -2.86 the 5 percent in the Adfuller table, as shown in the figure(5)

```
The result :              (-1.611147502985147,
P value :                  0.47737326585264805,
                           2,
No. of Observations :      2139,
Augmented dicky            {'1%': -3.4334108531807006,
fuller Test Table :         '5%': -2.862892168387536,
                            '10%': -2.5674898285322496},
                           83263.39023764747)
```

Figure 5: The Adfuller test statistic result

The normalization step can be done by $x = \frac{x-\mu}{\sigma}$. Both ACF and PACF functions were checked as an identification process, to determine the appropriate number of lags that need to incorporate into auto-regressive model. Figure

Figure 6: Visualize the Total private bytes series data

(6) visualize the ACF on the data with k = 30. The blue area around the X-axis represents Significance. All of the coefficient values are positive and fall between (0.9 and 1) which is outside the blue area, and are significantly different from zero. That suggests the existence of the auto-correlation for that lags. Since ACF capture both direct and indirect effect on the current state (x_t). The PACF is deal with only direct affect version of lags without considering the indirect affect to get most efficient model, which is examined in Figure (7).

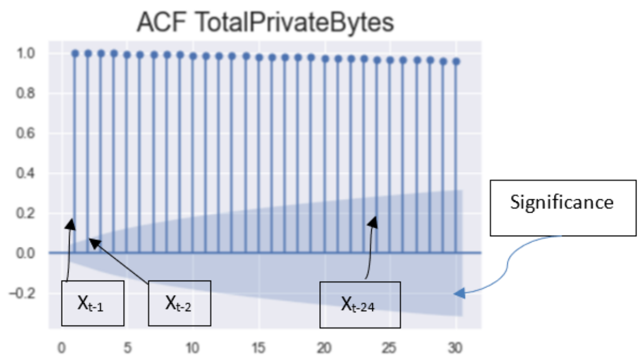

Figure 7: The ACF of the Private bytes series data

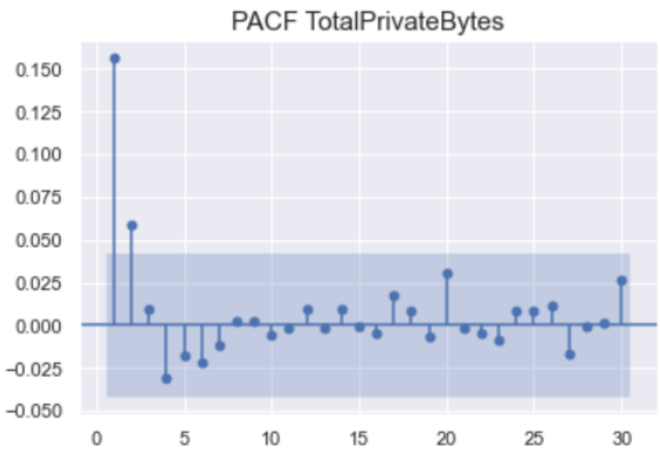

Figure 8: The PACF of the Private bytes series data

To evaluate several models we have been used AIC and LLT tests. Multiple lags of the ARIMA model have been conducted depending on ACF and PACF. Selecting the most appropriate order is the ARIMA model compared with other ones depending on the results of their LLT and AIC as shown in Table (1). We can see that the best model between the different ARIMA models is the ARIMA (2,1,1) model parameters. that we took it through the series data for prediction.

5.1 Applying Box Jenkins Model

Considering the Box-Jenkins parsimony principle. The ARIMA (2,1,1) was chosen as the most suitable model for the series data as a result of many comparisons. To evaluate the forecast between the actual and predicted data, MAE and RMSE are compared with multiple ARIMA model. As shown in Table 2.

Table 1: Different ARIMA models with their LLR and AIC

Model with order	LLR Test	AIC Test
ARIMA(0,1,0)	-42158.0257	84320.052
ARIMA(1,1,0)	-42131.4761	84278.356
ARIMA(0,1,1)	-42134.6236	84282.705
ARIMA(2,1,0)	-42127.7759	84275.091
ARIMA(1,1,1)	-42128.2053	84274.945
ARIMA(3,1,0)	-42127.6761	84275.900
ARIMA(2,1,1)	-42127.7287	84274.619
ARIMA(3,1,1)	-42127.3418	84277.829

Table 2: Statistical metrics used to determine the model

Fit Statistics	ARIMA(2,1,0)	ARIMA(2,1,2)	ARIMA(2,1,1)
MAE	0.00875	0.00876	0.00874
RMSE	0.01189	0.0119	0.01188

We carry out the ARIMA (2,1,1) forecasting inside the series data to predict the behavior at the critical section of memory usage 17:30 to 17:34 as shown in figure(9) which is challenging behavior due to fine-grained fluctuations, it shows that the predicted values closer to the observed one. Furthermore the prediction with observed data presented in figure 10 during the period of experiment from 16:10 to 17:35, the prediction shows that the data will keep increasing on the memory. We can confirm that ARIMA with only one previous lag coefficient of Moving Average MA(1) can give positive affect on the current state x_t, and whenever go back with previous lags coefficient it will gives negative affect on the current and coming data and hence in forecasting.

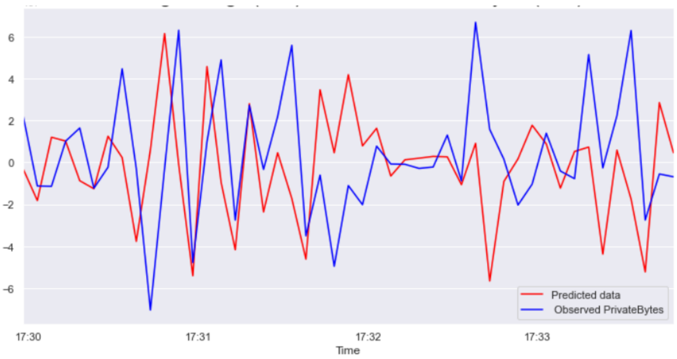

Figure 9: Memory usage prediction by ARIMA (2,1,1)

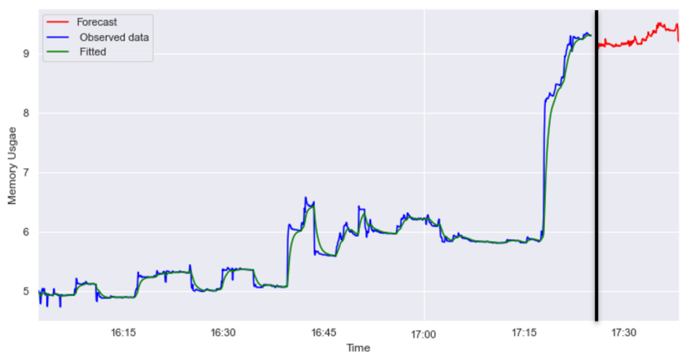

Figure 10: Memory leak usage period forecasted

The observed Memory usage metrics has been tabulated in Table 3 compared with the predicted data using ARIMA model. Different time slot of data at the critical section were captured are inserted in the table. As mentioned above the allocation on the memory by the system and user processes varies in different time slots.

Lastly, the Exponential Weighted Moving Average (EWMA) has been used to smooth the data and improve the variance, the moving average has been calculated for each 12 observation slot (each one minute), figure(11) shows the result. It put more weight at the recent data and less on the long previous data.

5.2 Applying The Deep learning LSTM Model:

Scaled Data is essential in the preprocessing steps for data modeling, to make the data be in a clear range from 0 to 1. MinMaxScaler of the sklearn package in python has been used to calculate the details of the series data to be

Table 3: Observed and Predicted data using ARIMA (2,1,1) model

Time	Observed Data	Predicted Data
16:00:53	4.82	4.84
16:00:58	4.91	4.82
16:01:03	4.91	4.92
16:01:08	4.89	4.91
...	...	...
17:35:33	9.48	9.47
17:35:38	9.47	9.47
17:35:43	9.46	9.47
17:35:43	9.45	9.46

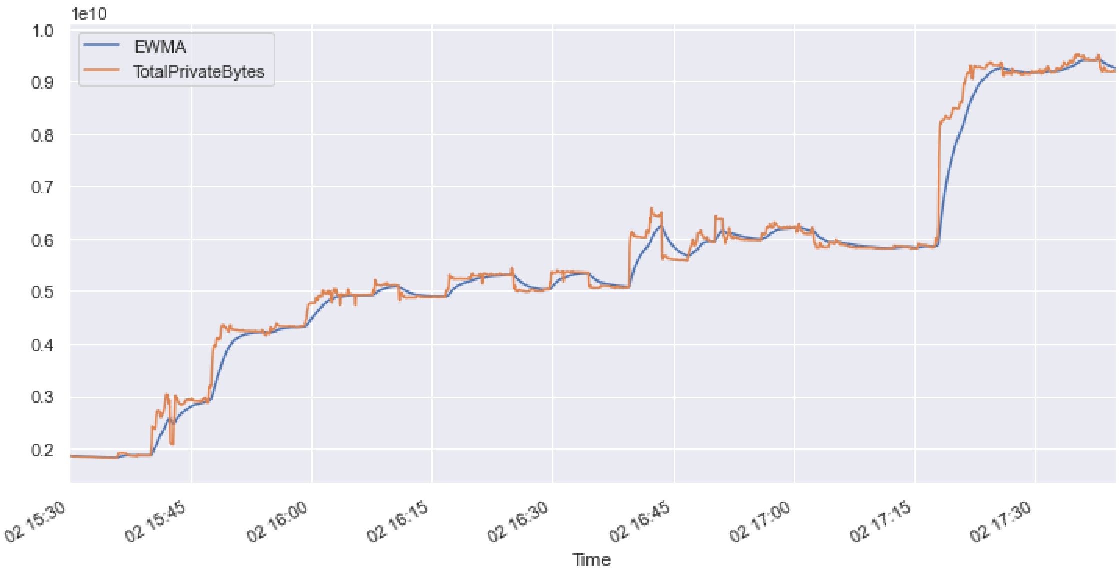

Figure 11: Smoothing the data using EWMA

scaled. The scaling makes the data understood by the model and easy to train. Furthermore, the LSTM model do not need to know if the the data stationary or not as the ARIMA model because it can deal with complex trends.

Figures (12) and (13) show the fitted and predicted data using of LSTM model respectively. The forecasting future memory usage at the period of the critical section in Figure (12) shows that dropping in prediction while looping of the output. The MSE for the LSTM model has been calculated 4.796 which is smaller than the error measure in ARIMA model. Furthermore, This slopping in prediction with LSTM model meaning that the early previous observed coefficient lags has negative affect on the current state X_t and on the future predictions, as shown in the following Figures.

Table 4 shows the slopping in the predicted data using LSTM model compared with the prediction one of the ARIMA model. It confidence that LSTM which can deserve the earlier input lags for long time can effect negatively on the coming data, hence can not lead to be suitable for predicting the processes behavior inside memory that comes with small changes.

6 Discussion

In time series analysis, for using the classic Box-Jenkins model, checking whether the data is stationary using the Augmented Dicky-Fuller test is important to go in any direction for prediction. The transformation of time series data to stationary is done by applying the degree of subtraction. The LSTM model do not need for stationary checking, for applying the deep learning LSTM method, 24 observations number set as input and has been given to the generator to

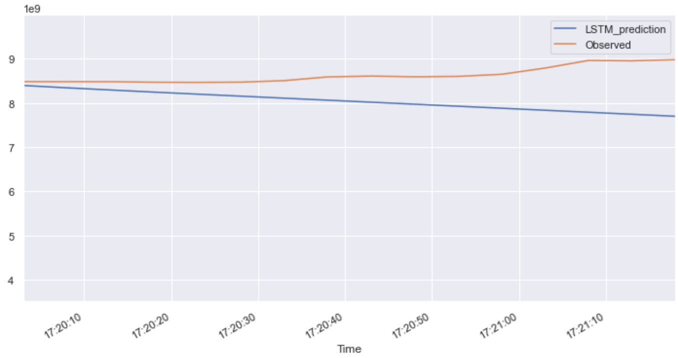

Figure 12: The fitted and osberved values using LSTM

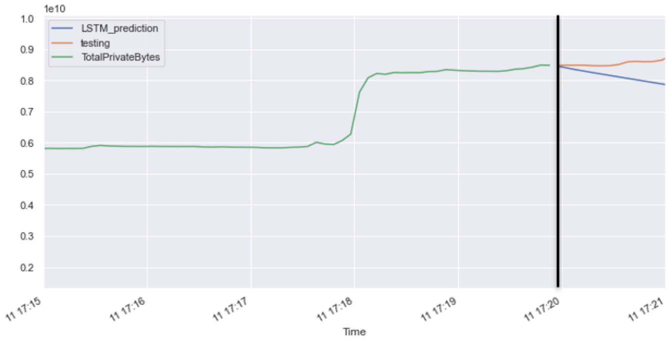

Figure 13: The predicted and osberved values using LSTM

Table 4: Observed and Predicted data using ARIMA (2,1,1) model

Time	Observed Data	Predicted Data
17:35:33	9.48	4.70
17:35:38	9.47	4.68
17:35:43	9.46	4.67
17:35:43	9.45	4.66
...	...	...
17:50:48	9.18	3.96
17:50:58	9.18	3.96
17:51:33	9.19	3.95
17:51:48	9.18	3.95

get predict 1 minute in future. The ARIMA model with coefficients lags orders (2,1,1) is used as a best-fitted model depending on the number of statistical metrics LLR and AIC tests. And the evaluation was processed using MAE and RMSE which are included in Table 2.

Conclusion

The main contribution of this study is to predict the performance issue in Windows system resources. We design an experimental study that simulated a memory commit performance issue due to aging affect. The series data was randomly collected from memory usage, its trend towards instability and it has been confirmed by the Adfuller test. The loading increased the private bytes in the Windows 10 system and makes abnormal behavior. The time series forecasting has been conducted using both ARIMA and LSTM models fitted to forecast whether the private bytes type memory leak charge system committed memory. This study shows that ARIMA model with the order (2,1,1) gives best prediction, and its MAE error rate is 0.00874, which preferred results to other ARIMA model numbers of order (2,1,2) with 0.00876 MAE. And the fitted LSTM model predict with MAE error measurement 4.796 . Predicted and observed values of both ARIMA and LSTM are compared. So, based on the given results in tables (3) and (4), the traditional ARIMA model can gives closed prediction for the anomaly fine-grained fluctuations on the Windows processes behavior inside Memory metric and predicting the anomaly load at the critical section.

References

[1] A. Agelastos, B. Allan, J.Brandt, P. Cassella, J. Enos, J. Fullop, A. Gentile, S. Monk, N. Naksinehaboon, J. Ogden, M. Rajan, M. Showerman, J. Stevenson, N. Taerat and T. Tucker, *The lightweight distributed metric service: A scalable infrastructure for continuous monitoring of large scale computing systems and applications*, Int. Conf. High Perform. Comput. Networking, Storage Anal. SC 2015, pp. 154–165.

[2] S.H.J. Alsaedi, *Forecasting the numbers of cardiac diseases patients by using Box-Jenkins model in time series analysis*, Int. J. Nonlinear Anal. Appl. **13** (2022), no. 1, 1673–1681.

[3] N.H. Albin Zehe, A.H. Andr´e Bauer, M. Zufle and S. Kounev, *Time series forecasting for self-aware system*, Schloss Dagstuhl **5** (2020).

[4] B. Auffarth, *Machine Learning for Time-Series with Python*, Packt, 2021.

[5] C. Bezemer, E. Milon, A. Zaidman and J. Pouwelse, *Detecting and analyzing I/O performance regressions*, J. Softw. Evol. Process **26** (2014), no. 12, 1193–1212.

[6] G.E.P. Box, G.M. Jenkins, G.C. Reinsel and G. M. Ljung, *Time series analysis: forecasting and control*, John Wiley & Sons, 2015.

[7] P.P. Deb and I. Chatterjee, *Time-series forecasting using lstm*, Schloss Dagstuhl **5** (2022).

[8] D.A. Dickey and W.A. Fuller, *Distribution of the Estimators for Autoregressive Time Series With a Unit Root*, J. Amer. Statist. Assoc. **74** (1979), no. 366, 427–431 (1986).

[9] M. Ficco, R.Pietrantuono and S. Russo, *Aging-related performance anomalies in the apache storm stream processing system*, Futur. Gener. Comput. Syst. **86** (2018), 975–994.

[10] K. Gencer and F. Başçiftçi, *Time series forecast modeling of vulnerabilities in the android operating system using ARIMA and deep learning methods*, Sustain. Comput. Informatics Syst. **30** (2021).

[11] M. Grottke, R. Matias, and K.S. Trivedi, *The fundamentals of software aging*, IEEE Int. Conf. Softw. Reliab. Eng. Work. ISSRE Wksp, 2008.

[12] Y. Huang, C. Kintala, N. Kolettis and N. Dudley Fulton, *Software rejuvenation: Analysis, module and applications*, Proc. Annu. Int. Conf. Fault-Tolerant Comput., 1995, pp. 381–390.

[13] C. Huffman, *Windows Performance Analysis Field Guide*, Elsevier, 2014.

[14] M. Kubacki and J. Sosnowski, *Exploring operational profiles and anomalies in computer performance logs*, Microproc. Microsyst. **69** (2019), 1–15.

[15] Guineng Zheng Vivek Srikumar Min Du, F. Li, *Deeplog: Anomaly detection and diagnosis from system logs through deep learning*, Proc. 2017 ACM SIGSAC Conf.Comput. Commun. Security, 2017. no. Oct 2017.

[16] I.M. Umesh, G.N. Srinivasan and M.Torquato, *Software aging forecasting using time series model*, Indones. J. Electr. Eng. Comput. Sci. **8** (2017), no. 3, 589–596.