

Performance assessment of time series forecasting models for simple network management protocol-based hypervisor data

Yuggo Afrianto^{1,2}, Rendy Munadi¹, Setyorini¹, Toto Widyanto³

¹School of Computing, Faculty of Informatic, Telkom University, Bandung, Indonesia

²Department of Informatics Engineering, Ibn Khaldun University, Bogor, Indonesia

³Informatics Engineering Study Program, Institut Sains and Technology Al Kamal, Jakarta, Indonesia

Article Info

Article history:

Received Feb 10, 2024

Revised Nov 15, 2024

Accepted Nov 24, 2024

Keywords:

Hypervisor

Machine learning

PyCaret

Simple network management protocol

Time series forecasting

ABSTRACT

Time series forecasting is vital for predicting trends based on historical data, enabling businesses to optimize decisions and operations. This paper evaluates forecasting models for predicting trends in simple network management protocol (SNMP)-based hypervisor data, essential for resource allocation in cloud data centers. Addressing non-stationary data and dynamic workloads, we use PyCaret to compare classical models like autoregressive integrated moving average (ARIMA) with advanced methods such as auto ARIMA. We assess 30 models on metrics including CPU utilization, memory usage, and disk reads, using synthetic and real-time datasets. Results show the naive forecaster model excels in CPU and disk read predictions, achieving low root mean squared errors (RMSE) of 0.71 and 869,403.35 for monthly and daily datasets. For memory usage predictions, gradient boosting with conditional deseasonalisation and detrending outperforms others, recording the lowest RMSE of 679,917.6 and mean absolute scaled error (MASE) of 4.46 on weekly datasets. Gradient boosting consistently improves accuracy across metrics and datasets, especially for complex patterns with seasonality and trends. These findings suggest integrating gradient boosting and naive forecaster models into cloud system architectures can enhance service quality and operational efficiency through improved predictive accuracy and resource management.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Rendy Munadi

School of Computing, Faculty of Informatic, Telkom University

40257 Buah Batu, Bandung, Indonesia

Email: rendymunadi@telkomuniversity.ac.id

1. INTRODUCTION

The field of cloud computing has experienced rapid growth, with numerous virtual machines now operating within physical server environments [1]. As demand for virtualization technology increases, effective time-series forecasting methods are essential, particularly for managing simple network management protocol (SNMP)-based hypervisor data [2]. Such data supports tracking and managing performance across virtual instances deployed in the cloud. However, time-series forecasting has certain weaknesses when dealing with 'non-stationary' data, as it often struggles to model dynamic and volatile workloads. This challenge limits the adaptability of forecasting methods when faced with unpredictable, fluctuating workloads, especially in SNMP-based hypervisor contexts, where these limitations pose significant obstacles to system administrators seeking to optimise resource management in cloud data centers [3].

Accurate workload prediction in cloud computing is crucial for optimising resource allocation and operational efficiency. Forecasting methods are generally divided into two main categories: those based on

fundamental time-series dynamics and those using machine learning and artificial neural networks [4]. A range of methods has been developed to address workload prediction challenges, including classical statistical approaches such as autoregressive (AR), moving average (MA), and the autoregressive integrated moving average (ARIMA) model [5], as well as more sophisticated models such as seasonal autoregressive integrated moving average (SARIMA) [6], [7]. Additionally, contemporary techniques like multiple linear regression (MLR), ridge regression (RR) [8], and adaptive neuro-fuzzy inference systems (ANFIS) [9] have also been applied. Recent research by Kumar and Singh [10] have indicated that auto ARIMA is a promising approach, offering high prediction accuracy for web server workloads and demonstrating its potential to proactively optimise resource allocation. However, traditional models such as auto ARIMA often struggle with the complexity and dynamics of cloud data centre environments, characterised by rapidly evolving data variability. This limitation is particularly evident over longer forecasting horizons and during periods of significant data fluctuation, underscoring the need for models that are more adaptive to changing conditions.

In recent years, automated machine learning (AutoML) frameworks, such as PyCaret, have emerged as powerful tools to simplify model selection, evaluation, and optimisation. PyCaret is a software tool that integrates a range of machine learning algorithms and time-series forecasting models within a Python wrapper, which includes assemblies of several machine learning frameworks, such as scikit-learn, XGBoost, LightGBM, CatBoost, spaCy, Optuna, Hyperopt, and Ray. Studies indicate that PyCaret's capabilities significantly enhance implementation, versatility, and customisation [11]. This tool has proven influential in time-series analysis and forecasting across multiple contexts [12]. Its applications range from forecasting trends in the COVID-19 pandemic [13] to data collection and formatting processes for various forecasting projects [14]. However, the potential for hypervisor management in the cloud computing domain with actual and synthetic SNMP-based hypervisor data still needs to be explored.

This research aims to evaluate the efficacy of 30 distinct time-series forecasting models using PyCaret on variety of authentic and synthetic datasets derived from SNMP-based hypervisor systems. The objective is to leverage the PyCaret toolkit to identify optimal forecasting methodologies for predicting hypervisor components accurately, specifically CPU utilization, memory utilization, and the number of disk reads. The findings of this study will support organisations in making informed decisions regarding resource allocation and overall operational efficiency in cloud environments. The primary contributions of this paper are as follows: first, the application of time-series forecasting models to SNMP hypervisor data through PyCaret, followed by analysis of the time series windows and forecasting models that yielded the best results; second, an evaluation of resource management approaches in light of the integrated forecasting measures.

2. METHOD

This study employs a machine learning approach to evaluate the efficacy of a time series forecasting model utilizing SNMP-based hypervisor data. The research was conducted using the stages illustrated in Figure 1. This section will discuss the dataset, experimental design, and model evaluation employed in this study. Furthermore, Figure 1 illustrates the essential steps for utilizing Pycaret to evaluate the model. The framework outlines procedures for acquiring load data from synthetic and real-time databases, providing an overview of the Pycaret interface for model evaluation. This systematic approach facilitates a comprehensive understanding of the workflow and tasks designed to ensure practical evaluation and comparison of multiple forecasting models in performance.

2.1. Dataset

This research employed a time series dataset, as detailed in Table 1 of the research paper. The dataset is divided into two principal categories: synthesis data, which is simulated data, and real-time data, which is obtained directly from the source. This information is crucial for accurately interpreting the data analyzed in the study and verifying the forecasting models [15].

- Synthesis: the scenario-based system generated a dataset containing CPU load, memory usage, and small computer system interface (SCSI) disk bytes as variables collected at different times.
- Real-time: a few of the datasets available include CPU utilization, memory usage, and SCSI disk bytes obtained for one month, one week, and one day, respectively.

Synthetic data is associated with a robust correlation to elevated prediction accuracy, as the generation of synthetic data adheres to a desired pattern. In contrast, data from real-time sources typically exhibit a weaker correlation with forecasting accuracy, as numerous external factors beyond an individual's control can influence their nature. The following characteristics are present in the dataset: The initial step is to evaluate CPU utilization for the monitored device by examining the CPU idle value. A low CPU idle value may indicate that the device operates under a high load. Secondly, to assess the memory load of the monitored device, a low memory-free value has the same significance, whereby a high load may be placed on

the device. Thirdly, through SCSI disk bytes, to assess the number of bytes written to the SCSI disk on the monitored device, a high byte count may suggest that the device is busy.

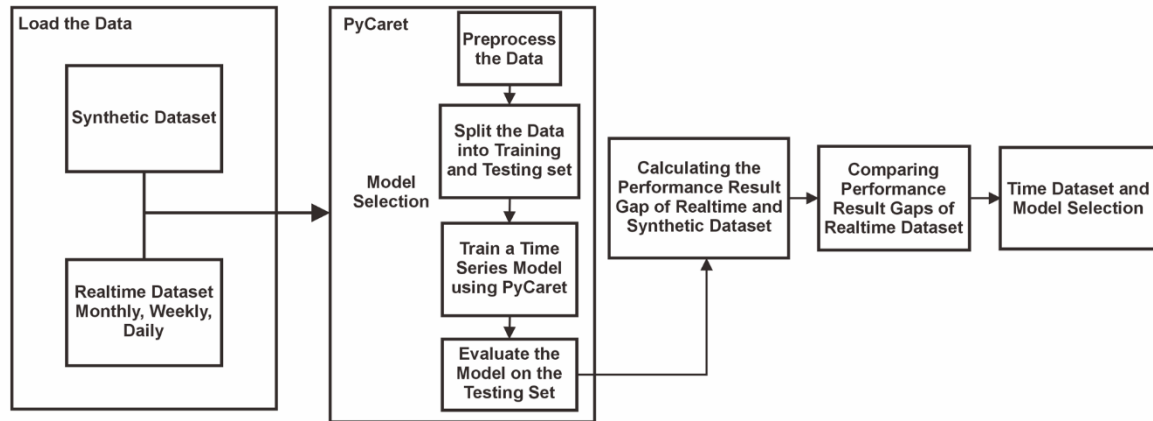


Figure 1. Framework assesment

Table 1. Time series dataset

Dataset	Metrics	Workload description
Synthetic and realtime (month, week, day)	CPU utilizations	Percentage of CPUs Idle
	Memory usage	Percentage of memory free
	Combined SCSI disk bytes	Bytes read

2.2. Experimental setup

2.2.1. Hypervisor workload forecasting

The dataset employed in this research is centred on data extracted from SNMP sources, obtained from network management servers and illustrative of the diverse workloads observed in cloud data centres. The dataset is employed to evaluate the forecasting models across a range of prediction windows, thereby facilitating the assessment of their accuracy. Figure 2 illustrates the general cloud system architecture incorporating a forecasting module. In this context, the resource manager receives input workload data from the hypervisor data centre and analyses historical data from real-time or synthetic workloads to forecast cloud resource management effectively.

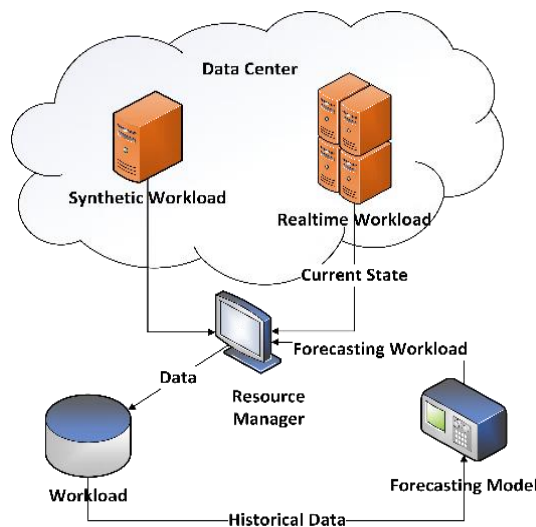


Figure 2. Load forecasting architecture hypervisor [10]

Equation for time-series analysis of data in SNMP is described by the variable \hat{Y} , which is a collection of workload values over time t , where y_t is the workload at time t . Function f represents the previous workload analysis used to estimate outcome of future events. This equation can be expressed as (1):

$$\hat{Y}_{t+1} = f(y_t, y_{t-1}, \dots, y_1) \quad (1)$$

Where \hat{Y}_{t+1} is represents the predicted workload at the next time step ($t+1$), f is a function that uses the previous workload values to estimate the future workload1, and y_t, y_{t-1}, \dots, y_1 are the workload values at the current and previous time steps.

This approach is pertinent in the context of SNMP, where data collection is continuous. It is noteworthy for its capacity to illustrate the evolution of the network system and the monitored devices with greater clarity. However, it is important to recognise a potential limitation, namely that the predicted values (\hat{Y}_{t+1}) must be validated against the actual values. This can be achieved by measuring the forecast error (e_{t+1}).

2.2.2. Scenario of creating synthetic dataset

Synthetic datasets have utility in research as they are able to imitate scenarios that do not always occur in real practice. In the evaluation of time-series forecasting models for SNMP-based hypervisor data context, the synthetic dataset is employed as follows: i) model testing: the primary objective of synthetic datasets is to facilitate model testing under controlled and repeatable scenarios. As discussed in [15], [16], the use of synthetic data is a key aspect of optimising data-driven prognostic models and modelling time series forecasting. In this manner, researchers are able to replicate simulated workloads that are otherwise unfeasible in the real world for a variety of reasons; ii) performance analysis: the models are exercised on synthetic workloads for the purpose of evaluating the forecast model's performance against some known workloads. Different models are tested and best practices are recorded to ascertain which model would work best under a particular situation; and iii) model validation: research is also conducted to test the effectiveness of the synthetic techniques in forecasting models on workload that may exist in a real-world context. This increases the confidence level of the applicability of the models across various parameters.

A single physical machine and four virtual ones (vm1, vm2, vm3, and vm4) were employed in the execution of a series of test scenarios, the objective of which was the generation of a synthetic dataset. Each test scenario was composed of two phases: a stress test phase and a pause phase. During the stress test, the load code was used on the virtual machine to imitate extensive usage. During the pause phase, the virtual machine was idle to recuperate. The scenario for creating a synthetic dataset is illustrated in Table 2.

Table 2. Scenario of creating synthetic dataset

Load VM	Interval testing (minutes)									
	0	10	20	30	40	50	60	70	80	
vm1				40	10				40	
vm2	10			30	20				30	
vm3	20			20	30				20	
vm4	30			10	40				10	
Descriptions:										
	Stress test time									
	Pause time									
	Lag time between stress tests									

2.2.3. Scenario of collecting real-time dataset

The real-time dataset collection process is illustrated in Figure 3 for this research. Figure 3 depicts the system architecture and components involved in gathering real-time data for analysis. Real-time datasets play an important role in research to understand, analyze, and optimize system performance based on actual data obtained from the environment being monitored. Real-time datasets were collected using energy monitoring and information systems. Workload energy represents the collection of energy data through sensors. Workload CIT deals with monitoring components such as CPU, memory, and I/O units, with data sent via TCP/IP (UDP), SNMP, the MySQL database, and Python. Cacti is a monitoring system designed for network and system monitoring. It allows users to visualize and analyze performance metrics, track trends, and generate reports. Database monitoring workload focuses on database performance metrics, analyzing workloads, and accessing datasets for further evaluation. Both monitoring systems play crucial roles in managing IT infrastructure components, providing valuable insights to maintain optimal performance and troubleshoot issues.

This research utilises the same dataset employed by Kumar and Singh [10] to facilitate a comparison of the findings with the results of other literature. This comprises a real-world dataset of web server traces, namely the number of hyper text transfer protocol (HTTP) requests for the NASA server, Calgary server, and Saskatchewan server, which is used to predict web server workload. The experiments are performed with a 5, 10, 20, 30, and 60-minute duration period window (PWS).

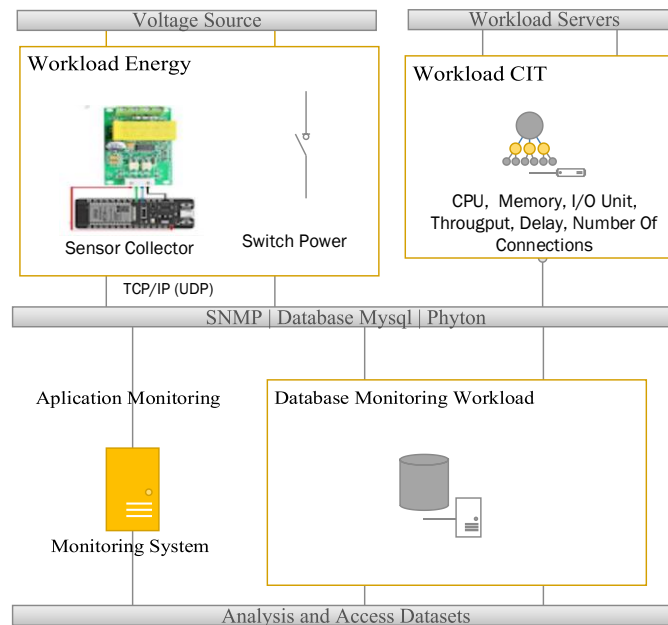


Figure 3. Collecting real-time dataset

2.3. Evaluation

2.3.1. Time series forecasting models

PyCaret is a machine learning framework that does not require the use of supplements and is straightforward to use and functional, thereby making it accessible to those new to the discipline [17]. The framework assists users at each stage of the machine learning process, from data preparation to model analysis and execution. The primary objective of analysing time-series data is to identify trends, which are captured by pertinent statistics and characteristics of the data being handled. There are various methods of forecasting time series, and this research utilised 30 models from the Pycaret library, as presented in Table 3.

Various studies have utilized PyCaret, a machine learning library, in various applications. These include diabetes classification and prediction [18], intrusion detection system performance analysis on the UNSW-NB15 dataset [19], hyperparameter tuning for image classification [20], and AutoML implementation on the PowerBI application [21]. PyCaret was also used in the evaluation of the predictive power of multiple regression models for groundwater contamination [22], intrusion detection using supervised and unsupervised learning methods on the Cicans 2017 dataset [23], machine learning-based network-based intrusion detection system (NIDS) analysis and modelling for IoT networks [24], and URL detection for phishing websites [25]. This research demonstrates the versatility and capabilities of PyCaret in various fields and contexts.

2.3.2. Performance measure indicators

It is challenging to comprehend the underlying causes of individual forecasting errors due to the complex and multifaceted nature of the process, which is not merely a consequence of a single numerical value. A number of methods have been proposed in the literature for the evaluation of prediction models and their relationships. Each of these methods has its own merits and demerits. For example, the root mean squared error (RMSE) is frequently declared to be influenced heavily by outlying values. In this instance, three measures were used for the evaluation of the model's predictive accuracy: the RMSE, the mean absolute error (MAE), and the mean absolute scaled error (MASE) [10]. The time required for processing each forecasting model and making predictions is TT.

Table 3. Time series forecasting models

Name	Reference
Naive Forecaster	sktime.forecasting.naive.NaiveForecaster
Grand Means Forecaster	sktime.forecasting.naive.NaiveForecaster
Seasonal Naive Forecaster	sktime.forecasting.naive.NaiveForecaster
Polynomial Trend Forecaster	sktime.forecasting.trend.PolynomialTrendForeca...
ARIMA	sktime.forecasting.arima.ARIMA
Auto ARIMA	sktime.forecasting.arima.AutoARIMA
Exponential Smoothing	sktime.forecasting.exp_smoothing.ExponentialSm...
ETS	sktime.forecasting.ets.Auto ETS
Theta Forecaster	sktime.forecasting.theta.ThetaForecaster
STLF	sktime.forecasting.trend.STLForecaster
Croston	sktime.forecasting.croston.Croston
BATS	sktime.forecasting.bats.BATS
TBATS	sktime.forecasting.tbats.TBATS
Linear w/ Cond. Deseasonalize and Detrending	pycaret.containers.models.time_series.BaseCdsD...
Elastic Net w/ Cond. Deseasonalize and Detrending	pycaret.containers.models.time_series.BaseCdsD...
Ridge w/ Cond. Deseasonalize and Detrending	pycaret.containers.models.time_series.BaseCdsD...
Lasso w/ Cond. Deseasonalize and Detrending	pycaret.containers.models.time_series.BaseCdsD...
Lasso Least Angular Regressor w/ Cond. Deseasonalize and Detrending	pycaret.containers.models.time_series.BaseCdsD...
Bayesian Ridge w/ Cond. Deseasonalize and Deseasonalize and Detrending	pycaret.containers.models.time_series.BaseCdsD...
Huber w/ Cond. Deseasonalize and Detrending	pycaret.containers.models.time_series.BaseCdsD...
Orthogonal Matching Pursuit w/ Cond. Deseasonalize and Detrending	pycaret.containers.models.time_series.BaseCdsD...
K Neighbors w/ Cond. Deseasonalize and Detrending	pycaret.containers.models.time_series.BaseCdsD...
Decision Tree w/ Cond. Deseasonalize and Detrending	pycaret.containers.models.time_series.BaseCdsD...
Random Forest w/ Cond. Deseasonalize and Detrending	pycaret.containers.models.time_series.BaseCdsD...
Extra Trees w/ Cond. Deseasonalize and Detrending	pycaret.containers.models.time_series.BaseCdsD...
Gradient Boosting w/ Cond. Deseasonalize and Detrending	pycaret.containers.models.time_series.BaseCdsD...
AdaBoost w/ Cond. Deseasonalize and Detrending	pycaret.containers.models.time_series.BaseCdsD...
Extreme Gradient Boosting w/ Cond. Deseasonali...	pycaret.containers.models.time_series.BaseCdsD...
Light Gradient Boosting w/ Cond. Deseasonalize...	pycaret.containers.models.time_series.BaseCdsD...
CatBoost Regressor w/ Cond. Deseasonalize and De...	pycaret.containers.models.time_series.BaseCdsD...

3. RESULTS AND DISCUSSION

The PyCaret time-series forecasting module is equipped with preprocessing features and offers a variety of algorithms, including over 30 statistical, time-series, and machine learning models. Beyond model training, it provides capabilities for automated hyperparameter tuning, ensembling, model analysis, and deployment. The typical workflow in PyCaret involves five steps: setup, computation, analysis, prediction, and model saving.

3.1. Analysis workload dataset

The results of all the workload dataset analysis were reproportioned with one of the weekly memory free flow dataset workloads and the results showed a changing trend. The data were visualized in a plot that displayed the flow size values on the y-axis and time on the x-axis, representing a type of time-series data. The analyzed dataset covered the period from October 25 to November 22, 2023. The term "memory free" is used to describe the amount of unutilised memory that is available for use within a system. In Figure 4 of the research article, the authors present a memory free datasets trend analysis, which illustrates trends in the amount of free memory over a specified time frame. The memory free flow trend analysis plot provides insights into the fluctuations in memory free flow over time and any changes in the regular deposit of memory. Memory trend analysis is a crucial aspect of system performance, resource and capacity planning in the field of information technology. It enables the formulation of well-informed decisions regarding memory management forms, performance enhancement and the deployment of resources in alignment with historical trends and patterns.

This case study on hypervisor data analyzes multiple time series of diverse work metrics. A statistical analysis is performed, as shown in Table 4, using the following attributes: i) test: name of the statistical test applied; ii) test name: brief description of the statistical test; iii) data: the type of data tested in the research; iv) property: properties of the data being tested; v) settings: settings used during the testing process; and vi) value: the value of the statistical test result.

The time series characteristics are used to determine the most appropriate estimation method for each time series. The prediction task focuses on several hypervisor performance metrics characteristic of a time series. This summary outlines the approach to evaluating and selecting the most appropriate forecasting models for time series analysis in various application contexts, using the following summary statistics tests [26]:

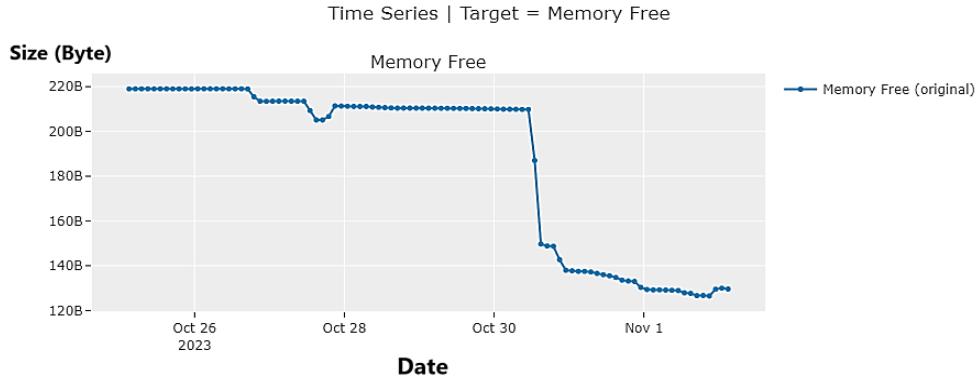


Figure 4. Trend analysis of memory free datasets

Table 4. Statistical analysis of memory free datasets

Test	Test Name	Data	Property	Setting	Value
Summary	Statistics	Transformed	Length	None	97
Summary	Statistics	Transformed	# Missing values	None	0
Summary	Statistics	Transformed	Mean	None	1.8765E+11
Summary	Statistics	Transformed	Median	None	2.10314E+11
Summary	Statistics	Transformed	Standard deviation	None	37482887512
Summary	Statistics	Transformed	Variance	None	1.40497E+21
Summary	Statistics	Transformed	Kurtosis	None	-1.33902
Summary	Statistics	Transformed	Skewness	None	-0.774785
Summary	Statistics	Transformed	# Distinct values	None	97
White Noise	Ljung-Box	Transformed	Test statistic	{'alpha': 0.05, 'K': 24}	982.632081
White Noise	Ljung-Box	Transformed	Test statistic	{'alpha': 0.05, 'K': 48}	1062.022633
White Noise	Ljung-Box	Transformed	p-value	{'alpha': 0.05, 'K': 24}	0
White Noise	Ljung-Box	Transformed	p-value	{'alpha': 0.05, 'K': 48}	0
White Noise	Ljung-Box	Transformed	White noise	{'alpha': 0.05, 'K': 24}	FALSE
White Noise	Ljung-Box	Transformed	White noise	{'alpha': 0.05, 'K': 48}	FALSE
Stationarity	ADF	Transformed	Stationarity	{'alpha': 0.05}	FALSE
Stationarity	ADF	Transformed	p-value	{'alpha': 0.05}	0.873311
Stationarity	ADF	Transformed	Test Statistic	{'alpha': 0.05}	-0.589619
Stationarity	ADF	Transformed	Critical value 1%	{'alpha': 0.05}	-3.502705
Stationarity	ADF	Transformed	Critical value 5%	{'alpha': 0.05}	-2.893158
Stationarity	ADF	Transformed	Critical value 10%	{'alpha': 0.05}	-2.583637
Stationarity	KPSS	Transformed	Trend stationarity	{'alpha': 0.05}	FALSE
Stationarity	KPSS	Transformed	p-value	{'alpha': 0.05}	0.01
Stationarity	KPSS	Transformed	Test statistic	{'alpha': 0.05}	0.304834
Stationarity	KPSS	Transformed	Critical value 10%	{'alpha': 0.05}	0.119
Stationarity	KPSS	Transformed	Critical value 5%	{'alpha': 0.05}	0.146
Stationarity	KPSS	Transformed	Critical value 2.5%	{'alpha': 0.05}	0.176
Stationarity	KPSS	Transformed	Critical value 1%	{'alpha': 0.05}	0.216
Normality	Shapiro	Transformed	Normality	{'alpha': 0.05}	FALSE
Normality	Shapiro	Transformed	p-value	{'alpha': 0.05}	0

3.1.1. White noise test

To evaluate the randomness of the time series data and to confirm the absence of significant autocorrelations, we performed a white noise test utilizing the Ljung-Box statistic. This test is essential for determining whether the data is suitable for time series modeling or if it represents purely random noise without any predictable patterns [27]. The following components were analyzed in this test:

- Ljung-box transformed test statistics: transformed Ljung-box test statistics.
- p-value: the p-value for the Ljung-box test.
- White noise: whether the data follows a white noise pattern.

3.1.2. Stationarity test

Determining the stationarity of time series data is essential before applying forecasting models, as many time series techniques assume stationarity to produce reliable results. Non-stationary data can lead to misleading inferences and poor predictive performance. To assess the stationarity of our dataset, we conducted the augmented dickey-fuller (ADF) test, which evaluates the presence of a unit root in the data. The following components were examined in this test:

- Azure data factory transformed: data transformed with Azure data factory transformation.
- Stationarity: conclusion whether the data is stationary.
- p-value: the p-value for the ADF test.
- ADF transformed test statistics: transformed ADF test statistics.
- Critical value: critical value for the ADF test at a certain significance level.
- Trend: whether there is a trend in the data.

3.1.3. Normality test

Evaluating the normality of the data distribution is essential in time series analysis, as many statistical methods assume that the underlying data is normally distributed. To determine if our dataset meets this assumption, we conducted the Shapiro-Wilk test, which is effective for detecting departures from normality in small to medium-sized samples. The following components were assessed in this test:

- Shapiro transformed: data transformed with Shapiro transformation.
- Normality: conclusion whether the data follows a normal distribution.
- p-value: the p-value for the Shapiro test.

The results of the statistical analysis are presented in Table 5. Reveals a consistent trend, indicating that the disk model frequently exhibits white noise and stationary characteristics across all periods. In contrast, normal distributions are infrequent, with only one instance observed for the memory model under synthetic dataset conditions. This suggests that temporal changes may significantly impact hypervisor performance metrics. The absence of normal distribution in real-time data suggests possible irregularities in data homogeneity and symmetry over extended durations. To address these irregularities, we performed further analysis and forecasting models to assess the performance of different forecasting models on the time series data [28].

Table 5. Results of statistical analysis

Dataset	Model	White noise	Stationer	Normal distributions
Monthly	CPU	False	False	False
	Memory	False	False	False
	Disk	True	True	False
Weekly	CPU	False	False	False
	Memory	False	False	False
	Disk	True	True	False
Daily	CPU	False	True	False
	Memory	False	False	False
	Disk	True	True	False
Syntetic	CPU	False	False	False
	Memory	False	False	True
	Disk	False	True	False

3.2. Analysis of forecasting performance

The performance of 30 forecasting models, which are time series in nature and are based on hypervisor data, was evaluated using real and synthetic datasets. The models were monitored via SNMP, and participants and/or caregivers, researchers, and models were assessed on key performance indices, including RMSE, MAE, and MASE, across a daily, weekly, and monthly period. Furthermore, the time required to process each model was evaluated in order to assess the efficiency of their computation.

3.2.1. Performance of idle CPU forecasting accuracy

A summary of the accuracy of the various models in predicting the CPU idle rate is provided in Table 6. Additionally, Table 6 presents other evaluation metrics, including RMSE, MASE, and MAE, which were employed to assess the performance of each model on the monthly, weekly, daily, and synthetic datasets. The forecasting accuracy of the monthly dataset the 'naive forecaster' model exhibited the lowest RMSE of 0.71, which was lower than that recorded by models 'auto ARIMA' and box-cox, ARMA error, trend, and seasonality ('BATS'), where RMSE=1.19 and 1.17, respectively. However, it had lower ranks of 11 and 12 out of the overall dataset. The weekly dataset 'exponential smoothing' and error, trend and sea-sonal ('ETS') models have identical performance with an RMSE of 0.65 and rank 7, while 'theta forecaster' has a slightly higher RMSE of 0.66 and rank 9. The daily dataset model 'grand means forecaster' shows the best performance with an RMSE of 0.68 and rank 4, followed by 'auto ARIMA' and 'ARIMA' with an RMSE of 0.73 and rank 5. The synthetic dataset was used to benchmark each real-time dataset model 'AdaBoost w/ cond. Deseasonalize and detrending' ranked first with the best performance,

followed by 'linear w/ cond. Deseasonalize and detrending' and 'ridge w/ cond. Deseasonalize and detrending' ranked 2nd and 3rd respectively.

From the results listed in Table 6, it can be seen that the grand means forecaster model on the daily dataset has the lowest RMSE and the 4th rank, indicating good prediction performance. On the monthly dataset, the naive forecaster model has the lowest RMSE but is ranked lower. On the synthetic dataset, the AdaBoost w/ cond. Deseasonalize and detrending ranked first with the best performance. This information is essential for selecting the best model for forecasting the availability of unused CPUs to help plan and manage system resources efficiently. Overall, the daily dataset and the grand means forecaster model with lower RMSE, MASE, and MAE and close to the synthetic ranking value show better CPU forecasting performance.

Table 6. Performance accuracy forecasting of CPU idle

Dataset	Model	RMSE	MASE	MAE	Ranking
Monthly	Naive forecaster	0.71	2.56	0.56	10
	Auto ARIMA	1.19	5.36	1.06	11
	BATS	1.17	5.41	1.07	12
Weekly	Exponential smoothing	0.65	2.03	0.54	7
	ETS	0.65	2.03	0.54	7
	Theta forecaster	0.66	2.06	0.55	9
Daily	Grand means forecaster	0.68	1.36	0.58	4
	Auto ARIMA	0.73	1.47	0.63	5
	ARIMA	0.73	1.47	0.63	5
Syntetic	AdaBoost w/ Cond. deseasonalize and detrending	0.59	0.93	0.51	1
	Linear w/ Cond. deseasonalize and detrending	0.61	0.93	0.51	2
	Ridge w/ Cond. deseasonalize and detrending	0.61	0.93	0.52	3

3.2.2. Performance of free memory forecasting accuracy

The memory free forecasting accuracy performance on the monthly dataset model 'Huber w/ cond. The deseasonalize and detrending overall, daily dataset, and grand means forecaster model with lower RMSE, MASE, and MAE and close to the synthetic ranking value show better CPU forecasting performance. model had a very high RMSE, indicating a less-than-optimal performance. The extra trees with cond. deseasonalize and detrending had the highest MASE, and the model rank ranged from 10 to 12 out of the entire dataset. The weekly theta forecaster model has a significantly lower error value than the BATS and TBATS models, which have identical error values but different rankings of eight for BATS and seven TBATS. Daily dataset gradient boosting w/ cond. Deseasonalize and detrending shows the best performance in all error metrics among the daily dataset models, while 'Bayesian ridge w/ cond. Deseasonalize and detrending' performed below it but still ranked 6th overall. Synthetic dataset the 'exponential smoothing' model performed the best with the lowest error in all metrics among the synthetic dataset models, while the 'ETS' model performed below it but still ranked 2nd overall. The results of the accuracy of the different models in predicting memory free as shown in Table 7.

From the results listed in Table 7, it can be seen that the Huber w/ cond. model. Deseasonalize and detrending on the monthly dataset has the highest RMSE and ranks 10th, indicating less than optimal prediction performance. On the weekly dataset, the theta forecaster model has a low MASE and ranks 7th, indicating better performance than the BATS model. The analysis from Table 7 provides insight into which models are most effective in predicting memory-free rates at various data frequencies. Overall, the daily dataset and gradient boosting are done using cond. Deseasonalization and detrending with lower RMSE, MASE, and MAE and close to the synthetic rank value show better forecasting memory performance.

Tabel 7. Performance accuracy forecasting of memory free

Dataset	Model	RMSE	MASE	MAE	Ranking
Monthly	Huber w/ Cond. Deseasonalize and Detrending	3271875524.78	6.61	2731561697.07	10
	Auto ARIMA	3767137446.90	7.56	3135176015.51	11
	Extra trees w/ Cond. Deseasonalize and Detrending	4421809533.71	8.88	3697388235.58	12
Weekly	Theta forecaster	1347869846.82	0.94	1164727393.11	7
	BATS	1710867790.86	1.14	1434401664.64	8
	TBATS	1710867790.86	1.14	1434401664.64	8
Daily	Gradient boosting w/ Cond. Deseasonalize and Detrending	679917621.37	4.46	562941495.97	4
	CatBoost regressor w/ Cond. Deseasonalize and Detrending	817036128.95	5.20	656224572.32	5
	Bayesian ridge w/ Cond. Deseasonalize and Detrending	845857410.17	5.37	676449659.13	6
Syntetic	ETS	531650395.06	0.69	477258423.00	2
	Exponential Smoothing	527319448.26	0.69	480880807.52	1
	Bayesian Ridge w/ Cond. Deseasonalize and Detrending	565535362.33	0.70	488370640.92	3

3.2.3. Performance accuracy forecasting of disk read

The forecasting accuracy performance of the disk read dataset on the monthly dataset; naive forecaster model has better forecasting performance with the lowest rank compared to the decision tree w/ cond model. Desesasonalize and detrending and BATS models. However, the monthly dataset was not superior to the weekly and daily datasets. The Weekly dataset in the naive forecaster model has better forecasting performance than the extra and decision trees with cond. Desesasonalize and detrending with. However, the monthly dataset was not superior to the weekly and daily datasets. The daily dataset and the naive forecaster model performed best by excelling in all monthly and weekly models and datasets. The results of the accuracy of the different models in predicting disk read as shown in Table 8.

Table 8. Performance accuracy forecasting of disk read

Dataset disk read	Model	RMSE	MASE	MAE	Ranking
Monthly	Decision Tree w/ Cond. Deseasonalize and Detrending	882218.20	6.23	384421.61	9
	Naive Forecaster	869403.35	6.27	386646.91	7
	BATS	873858.07	6.23	385887.01	8
Weekly	Naive Forecaster	36034.60	0.12	23248.43	4
	Extra Trees w/ Cond. Deseasonalize and Detrending	36923.95	0.16	29859.68	5
	Decision Tree w/ Cond. Deseasonalize and Detrending	36923.95	0.16	29859.68	5
Daily	Naive Forecaster	15855.61	0.52	13257.11	1
	Croston	16753.33	0.59	15243.52	2
	Grand Means Forecaster	20927.18	0.73	19268.03	3
Syntetic	Grand Means Forecaster	5216888.90	0.73	4039774.44	12
	TBATS	5001328.06	0.76	4189157.24	10
	BATS	5026264.73	0.77	4204561.59	11

Overall, the daily dataset and naive forecaster model were the best performing datasets and models. This model achieved the lowest rank among all tested models, indicating a smaller prediction error rate and more accurate performance. The analysis of this table provides a deeper understanding of which models are most effective in predicting disk read rates at various data frequencies. However, the synthetic dataset had the highest rank because it had a large prediction residual value compared with the other datasets, as shown in Figure 5.

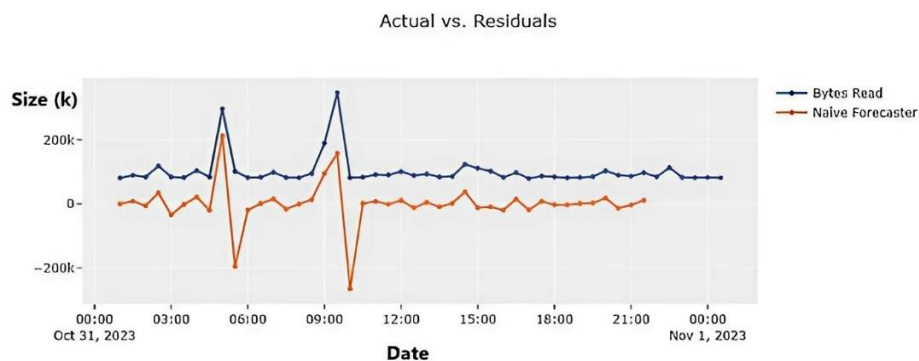


Figure 5. Residual forecast disk read syntetic data

3.2.4. Performance of forecasting process time efficiency

The time efficiency of various forecasting models is shown in Table 9, presenting the processing times for predicting CPU, memory, and disk read performance. Table 9 includes the most accurate forecasting models and datasets from the performance tests. The results indicate that the best dataset for forecasting hypervisor resource load is the daily dataset across multiple forecasting models. The naive forecaster model is the most efficient for forecasting CPU performance, with a runtime of 1.71 seconds, aligning with Yoo and Sim [7] findings that simple models often perform well in highly volatile scenarios. The naive forecaster makes predictions using straightforward strategies [29]. For memory datasets, gradient boosting w/ conditional deseasonalization and detrending demonstrates the best performance, with a processing time of 0.13 seconds.

Our results align with existing research on the effectiveness of framework for comparing accuracy of time-series forecasting methods. Similar to our findings, a study by Hyndman and Athanasopoulos [30] also identified gradient boosting had the highest accuracy in gradient boosting, which is often used in machine learning competitions, although it was lower than naive, a simple prediction method. This reinforces the notion that gradient boosting ability to handle complex relationships within data can be advantageous for resource load prediction.

The gradient boosting technique was proposed as a one-of-a-kind applied gradient boosting machine, particularly for regression and classification trees. The “boosting” concept is the root of gradient boosting, which merges the forecasting of weak learners with additive training methods to develop a strong learner [31]. Gradient boosting with conditional deseasonalization and detrending is a specific application of gradient boosting that incorporates the removal of seasonal and trend components from time-series data before model training [32].

The findings of this study further indicate that PyCaret, as an AutoML framework, simplifies the process of selecting optimal models and tuning hyperparameters, essential for practical use in cloud data centres. This contrasts with the manual approach in previous studies, which required model selection and testing to be conducted individually. With AutoML capabilities, processing time and computational resources can be optimised, enabling practitioners to efficiently select the best model without repeated adjustments to model parameters, as discussed in the research by Westergaard *et al.* [12]. This information is essential for selecting a model that is accurate in prediction and efficient in using computing resources.

Table 9. Performance of forecasting process time efficiency

Dataset	Model forecasting	TT (Sec)
CPU (Daily)	Naive Forecaster	1.71
Memory (Daily)	Gradient Boosting w/ Cond. Deseasonalize and Detrending	0.13
Disk read (Daily)	Naive Forecaster	1.90

3.2.5. Comparison of workload prediction models on web server trace datasets

The results of testing the HTTP request datasets from NASA, Calgary, and Saskatchewan servers are presented here to evaluate the performance of the proposed gradient boosting model compared with Auto ARIMA. Performance evaluation uses error metrics, including RMSE, MAE, and MASE. These tests provide a comprehensive overview of each model's accuracy and suitability in the context of the research conducted by Kumar and Singh [10].

The NASA server dataset was tested with Auto ARIMA and gradient boosting models using conditional deseasonalisation and detrending across four prediction windows, presented in Table 10 for periods of 5, 10, 30, and 60 minutes. As the prediction window lengthens, Auto ARIMA demonstrates relatively consistent performance, though errors (RMSE and MAE) increase. The lowest MASE value, 4.42, is achieved at a 10-minutes window, highlighting its medium-term accuracy. Gradient boosting with deseasonalisation and detrending generally outperforms Auto ARIMA across nearly all prediction windows, exhibiting notably lower RMSE and MAE, especially in short- to medium-term predictions (PWS 10 and 30).

Table 10. NASA server dataset

Model	PWS	RMSE	MAE	MASE
Auto Arima	5	249.99	218.28	5.16
	10	380.20	334.67	4.42
	30	1818.54	1664.29	8.36
	60	4478.16	4173.28	9.79
Gradient Boosting w/ Cond. Deseasonalize and Detrending	5	215.27	190.71	4.51
	10	251.42	220.73	2.91
	30	855.47	758.42	3.81
	60	2854.49	2544.58	5.97

Testing on the Calgary server dataset involved the same models and settings, as shown in Table 11. Auto ARIMA, RMSE, and MAE values increase with longer prediction windows, though it achieves a low MASE value of 1.51 at a 10 minutes window, indicating optimal medium-term fit. The gradient boosting model with deseasonalisation and detrending achieves lower RMSE and MAE at the shortest prediction window of 5, underscoring its high accuracy. However, its RMSE and MAE values slightly increase over longer windows, while remaining competitive.

Table 11. Calgary server dataset

Model	PWS	RMSE	MAE	MASE
Auto Arima	5	23.53	14.67	2.03
	10	27.19	18.83	1.51
	30	70.76	53.08	1.91
	60	109.16	81.24	1.74
Gradient Boosting w/ Cond. Deseasonalize and Detrending	5	11.75	9.22	1.27
	10	30.36	27.70	2.22
	30	76.60	69.97	2.52
	60	94.45	80.88	1.73

A comparative analysis using the Saskatchewan server dataset similarly assessed the efficacy of Auto ARIMA and gradient boosting across four prediction windows, as shown in Table 12. Auto ARIMA, RMSE, and MAE values rise with window length, though MASE remains stable around 1.9, indicating consistent performance. Gradient boosting with deseasonalisation and detrending outperforms Auto ARIMA across almost all prediction windows, achieving significant accuracy improvements in the 5 and 60 minutes windows.

Table 12. Saskatchewan server dataset

Model	PWS	RMSE	MAE	MASE
Auto Arima	5	44.57	32.25	1.76
	10	84.84	61.73	2.02
	30	180.71	132.64	1.91
	60	347.07	259.83	1.92
Gradient Boosting w/ Cond. Deseasonalize and Detrending	5	38.33	28.97	1.58
	10	71.30	57.61	1.88
	30	159.69	133.85	1.92
	60	296.16	248.67	1.84

In terms of overall accuracy and efficiency, gradient boosting consistently outperforms Auto ARIMA across all datasets and prediction windows, demonstrating a superior ability to capture complex patterns and manage variability in web server workloads. Although Auto ARIMA performs adequately in medium-term predictions, its effectiveness diminishes in longer timeframes, as indicated by rising RMSE and MAE values, revealing limitations in adapting to dynamic changes. Gradient boosting, on the other hand, demonstrated greater scalability and adaptability at varying time scales, maintaining lower error metrics for both short- and long-term predictions. Its ability to integrate deseasonalisation and detrending enhances predictive accuracy, making it well-suited for datasets with inherent seasonality and trends. These findings suggest that future research could focus on refining gradient boosting models further, exploring hybrid methods, and optimising model complexity to balance accuracy with computational efficiency.

4. CONCLUSION

This study presents a performance assessment of time-series forecasting models for SNMP-based hypervisor data using Pycaret. We evaluated the performance of our method on two types of datasets: real-time data collected from a physical server environment running multiple virtual machines and synthetic data created based on a specific workload scenario. We compared 30 different time-series forecasting models using three evaluation metrics, namely, RMSE, MASE, and MAE, and the computation time required. The evaluation results obtained the best dataset using daily data with several forecasting models, according to the resource load on the hypervisor. The best naive forecaster model for forecasting CPU performance with a runtime of 1.71 seconds and disk read runtime of 1.90 seconds. Meanwhile, gradient boosting w/ cond. deseasonalize and detrending the best for memory dataset with process times of 0.13. Based on the test results with NASA, Calgary, and Saskatchewan server datasets, from the comparison between Auto ARIMA and gradient boosting with deseasonalize & detrending models, while both models have their merits, gradient boosting with deseasonalize and detrending provides a more robust solution for forecasting in cloud data centre environments, adapting well to the complexities and dynamics of real-world server workloads. In the future, researchers can concentrate on enhancing and confirming the effectiveness of our approach by using more extensive and varied datasets. They can also integrate it with other resource management in decision support systems for use in real-time live migration settings in hypervisor clusters. Our research has the potential to improve the quality and efficiency of resource management maintenance in data centres, which can ultimately enhance the quality of service (quality of service) and service level agreements (SLA).

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the financial support from DRTPM Kemdikbudristek and Telkom University for this research under the Doctoral Dissertation Research Program scheme 2024, under Research Contract Number 106/E5/PG.02.00.PL/2024; 043/SP2H/RT-MONO/LL4/2024; 034/LIT07/PPM-LIT/2024.




REFERENCES

- [1] M. C. Martachinniceneait, "SNMP for cloud environment energy efficiency," *Research Square*, pp. 1-27, 2021, doi: 10.21203/rs.3.rs-720622/v1.
- [2] S. B. Shaw, C. Kumar, and A. K. Singh, "Use of time-series based forecasting technique for balancing load and reducing consumption of energy in a cloud data center," in *2017 International Conference on Intelligent Computing and Control (I2C2)*, IEEE, 2017, pp. 1-6, doi: 10.1109/I2C2.2017.8321782.
- [3] S. Steiner, "Harnessing data to make better-informed decisions," *Scientia*, 2022, doi: 10.33548/SCIENTIA768.
- [4] S. Jadon, A. Patankar, and J. K. Mileczek, "Challenges and approaches to time-series forecasting for traffic prediction at data centers," in *2021 International Conference on Smart Applications, Communications and Networking, SmartNets 2021*, IEEE, 2021, pp. 1-8, doi: 10.1109/SmartNets50376.2021.9555422.
- [5] H. Jdi and N. Falih, "Comparison of time series temperature prediction with auto-regressive integrated moving average and recurrent neural network," *International Journal of Electrical and Computer Engineering*, vol. 14, no. 2, pp. 1770-1778, Apr. 2024, doi: 10.11591/ijeecs.v14i2.pp1770-1778.
- [6] V. Savchenko *et al.*, "Network traffic forecasting based on the canonical expansion of a random process," *Eastern-European Journal of Enterprise Technologies*, vol. 3, no. 2-93, pp. 60-69, 2018, doi: 10.15587/1729-4061.2018.131471.
- [7] W. Yoo and A. Sim, "Time-series forecast modeling on high-bandwidth network measurements," *Journal of Grid Computing*, vol. 14, no. 3, pp. 463-476, 2016, doi: 10.1007/s10723-016-9368-9.
- [8] S. N. Wahyuni, E. Sediono, I. Sembiring, and N. N. Khanom, "Comparative analysis of time series prediction model for forecasting covid-19 trend," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 28, no. 1, pp. 600-610, Oct. 2022, doi: 10.11591/ijeecs.v28.i1.pp600-610.
- [9] P. Shastri, B. R. Dawadi, and S. R. Joshi, "Intelligent approach to switch replacement planning for internet service provider networks," *Sustainable Futures*, vol. 2, 2020, doi: 10.1016/j.sfr.2020.100036.
- [10] J. Kumar and A. K. Singh, "Performance assessment of time series forecasting models for cloud datacenter networks' workload prediction," *Wireless Personal Communications*, vol. 116, no. 3, pp. 1949-1969, 2021, doi: 10.1007/s11277-020-07773-6.
- [11] M. Ali, "PyCaret: an open source, low-code machine learning library in python," *ACS Omega*, vol. 6, p. 6791-6797, 2021.
- [12] G. Westergaard, U. Erden, O. A. Mateo, S. M. Lampo, T. C. Akinci, and O. Topsakal, "Time series forecasting utilizing automated machine learning (autoML): a comparative analysis study on diverse datasets," *Information*, vol. 15, no. 1, 2024, doi: 10.3390/info15010039.
- [13] S. Maurya and S. Singh, "Time series analysis of the covid-19 datasets," in *2020 IEEE International Conference for Innovation in Technology, INOCON 2020*, IEEE, 2020, pp. 1-6, doi: 10.1109/INOCON50539.2020.9298390.
- [14] J. Siebert, J. Groß, and C. Schroth, "A systematic review of packages for time series analysis †," *Engineering Proceedings*, vol. 5, no. 1, 2021, doi: 10.3390/engproc2021005022.
- [15] V. Cerqueira, L. Torgo, and I. Mozetič, "Evaluating time series forecasting models: an empirical study on performance estimation methods," *Machine Learning*, vol. 109, pp. 1997-2028, 2020, doi: 10.1007/s10994-020-05910-7.
- [16] T. Lindgren and O. Steinert, "Low dimensional synthetic data generation for improving data driven prognostic models," in *2022 IEEE International Conference on Prognostics and Health Management, ICPHM 2022*, IEEE, 2022, pp. 173-182, doi: 10.1109/ICPHM53196.2022.9815660.
- [17] Y. K. Phua, T. Fujigaya, and K. Kato, "Predicting the anion conductivities and alkaline stabilities of anion conducting membrane polymeric materials: development of explainable machine learning models," *Science and Technology of Advanced Materials*, vol. 24, no. 1, 2023, doi: 10.1080/14686996.2023.2261833.
- [18] P. Whig, K. Gupta, N. Jiwani, H. Jupalle, S. Kouser, and N. Alam, "A novel method for diabetes classification and prediction with pycaret," *Microsystem Technologies*, vol. 29, no. 10, pp. 1479-1487, 2023, doi: 10.1007/s00542-023-05473-2.
- [19] Abdullah, F. B. Iqbal, S. Biswas, and R. Urba, "Performance analysis of intrusion detection systems using the pycaret machine learning library on the unsw-nb15 dataset," *B.Sc. Thesis*, Department of Computer Science and Engineering, Brac University, Dhaka, Bangladesh, 2021.
- [20] K. Arai, J. Shimazoe, and M. Oda, "Method for hyperparameter tuning of image classification with pycaret," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 9, pp. 276-282, 2023, doi: 10.14569/IJACSA.2023.0140930.
- [21] D. J. C. Sihombing, J. U. Dexius, J. Manurung, M. Aritonang, and H. S. Adinata, "Design and analysis of automated machine learning (autoML) in powerbi application using pycaret," in *2022 International Conference of Science and Information Technology in Smart Administration, ICSINTESA 2022*, 2022, pp. 89-94, doi: 10.1109/ICSINTESA56431.2022.10041543.
- [22] T. Huynh, H. Mazumdar, H. Gohel, H. Emerson, and D. Kaplan, "Evaluating the predictive power of multiple regression models for groundwater contamination using pycaret," in *WM2023 Conference*, Arizona, USA: WM Symposia, Inc., 2023, pp. 1-15.
- [23] S. Krsteski, M. Tashkovska, B. Sazdov, L. Radojichikj, A. Cholakovska, and D. Efnusheva, "Intrusion detection with supervised and unsupervised learning using pycaret over cicids 2017 dataset," in *Artificial Intelligence Application in Networks and Systems*, 2023, pp. 125-132, doi: 10.1007/978-3-031-35314-7_12.
- [24] M. Karanfilovska, T. Kochovska, Z. Todorov, A. Cholakovska, G. Jakimovski, and D. Efnusheva, "Analysis and modelling of a ml-based nids for iot networks," in *Procedia Computer Science*, 2022, pp. 187-195, doi: 10.1016/j.procs.2022.08.023.
- [25] P. Rani, "PyCaret based url detection of phishing websites," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 11, no. 1, pp. 908-915, 2020, doi: 10.17762/turcomat.v11i1.13589.
- [26] J. E. Monogan, "Time series analysis," in *Political Analysis Using R*, Springer, Cham, 2015, pp. 157-186, doi: 10.1007/978-3-319-23446-5_9.
- [27] M. Mahan, C. Chorn, and A. P. Georgopoulos, "White noise test: detecting autocorrelation and nonstationarities in long time series after arima modeling," in *Proceedings of the 14th Python in Science Conference*, 2015, doi: 10.25080/Majora-7b98e3ed-00f.
- [28] S. N. Rao, G. Shobha, S. Prabhu, and N. Deepamala, "Time series forecasting methods suitable for prediction of cpu usage," in *CSITSS 2019 - 2019 4th International Conference on Computational Systems and Information Technology for Sustainable*




- Solution, Proceedings*, IEEE, Dec. 2019, pp. 1–5, doi: 10.1109/CSITSS47250.2019.9031015.
- [29] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts: Melbourne, 2021.
- [30] J. Sekitani and H. Murakami, “Framework for comparing accuracy of time-series forecasting methods,” in *2022 12th International Congress on Advanced Applied Informatics (IIAI-AAI)*, IEEE, 2022, pp. 669–672, doi: 10.1109/IIAIAI55812.2022.00136.
- [31] O. Alshboul, A. Shehadeh, G. Almasabha, and A. S. Almuflhi, “Extreme gradient boosting-based machine learning approach for green building cost prediction,” *Sustainability*, vol. 14, no. 11, May 2022, doi: 10.3390/su14116651.
- [32] M. Przybyla, “New time series with pycaret,” *Towards Data Science*. 2021. Accessed: Dec. 25, 2023. [Online]. Available: <https://towardsdatascience.com/new-time-series-with-pycaret-4e8ce347556a>

BIOGRAPHIES OF AUTHORS






Yuggo Afrianto    is currently a permanent lecturer at Ibn Khaldun University, Bogor-West Java, Informatics Engineering bachelor's Study Program. He holds a bachelor's degree in informatic engineering from the Ibn Khaldun University in Bogor, a master's degree in computer science from IPB University in Bogor. In 2017. Research in the fields of net centric computing he is a member of cyber physical system (CPS) Telkom University and a member of the Association for Informatics and Computer Higher Education. He can be contacted at email: yuggo@uika-bogor.ac.id.






Prof. Rendy Munadi    has been a lecturer at the Faculty of Electrical Engineering at Telkom University (formerly, STT Telkom) Bandung, Indonesia since 1993. The teaching materials taught include wireless sensor networks, data and protocol networks, broadband networks, internet of things in the master field of study while in the field of undergraduate studies include traffic engineering, future new network, and seminar proposals. The field of research that is being carried out is manufacture of a hemoglobin measuring device by utilizing the internet of things (IoT) based machine learning method, partner: Hasan Sadikin Hospital. He has serving experience in universities since 1998 is as Head of Academic Administration (formerly STT Telkom) and Head of Telecommunication Engineering Study Program in 2005-2006, served as Vice Chancellor for Academic Affairs in 2006-2010. Since 2018, he has served as an assessor for FTE lecturer certification. Currently, he is a senior lecturer in the network cyber management (NCM) expertise group. He can be contacted at email: rendymunadi@telkomuniversity.ac.id.



Dr. Setyorini    is currently a permanent lecturer at Telkom University, Bandung West Java, in an informatics master's study program. She received a B.S. degree in informatics from Telkom University, and received M.S. and doctoral degrees in computer engineering from Bandung Institute of Technology. Her research interest in parallel computing has also focused on distributed systems. She can be contacted at email: setyorini@telkomuniversity.ac.id.



Dr. Toto Widyanto    is a lecturer at Al Kamal Institute of Science and Technology (ISTA). The teaching materials taught include machine learning, mobile computing, artificial intelligence, telecommunications, and innovation. Besides education field, he is a consultant for more than two decades in green data centers and green buildings design and development. He is also a chairman of Indonesia Green Data Center Professional Association (IPUSTAH-ID). He holds professional certifications for greenhip professional (GP) and certified data center specialist (CDCS). He can be contacted at email: toto.widyanto@ista.ac.id.