Partial Granger causality-based feature selection algorithm for workload prediction in cloud systems

Changhoon Lee
Artificial Intelligence Lab
Okestro
Seoul, South Korea
ch.lee@okestro.com

Eunsoo Ko
Artificial Intelligence Lab
Okestro
Seoul, South Korea
es.ko@okestro.com

Minjae Song
Artificial Intelligence Lab
Okestro
Seoul, South Korea
mj.song@okestro.com

Hoyeong Yun
Artificial Intelligence Lab
Okestro
Seoul, South Korea
hy.yun@okestro.com

Wooju Kim
Dept. of Industrial Engineering
Yonsei University
Seoul, South Korea
wkim@yonsei.ac.kr

***Abstract*—The development of the cloud technology led to the rising interest in multi/hybrid cloud and emergence of artificial intelligence for IT operations(AIOps). The core elements of AIOps involve predicting how each metric of the data center will change in the future. Compared to general multivariate time-series prediction problems, causal relationships between each variable have a significant impact on cloud metric prediction. This paper focuses on the causality between variables, which partially arises when a specific event occurs in a cloud data center to achieve good predictive performance. The proposed model detects partial causality and sums it up again to extract key variables that explain the target variable well. Through this, variables with higher predictive performance than existing methods were found. We also propose a structure that improves the performance of the prediction model and minimizes inference time through a variable selection technique based on partial causality. By applying this to the actual operating cloud environment, it was proved that it can be effectively applied to the real world.**

Keywords—AIOps, Feature Selection, Granger Causality, Prediction

I. INTRODUCTION

In the on-premise(physical server based) development environment, the introduction of cloud including virtualization technology has led to many changes in the SW development environment. In particular, from the perspective of infrastructure managers, many companies and institutions began to adopt them because they were fascinated by the apparent benefits. The primary benefits of cloud technology include efficient network load response, quick recovery during failures, and flexible server capacity calculation. [1] Despite such exceptional features, infrastructure administrators have been required to understand and respond to not only simple servers but also the complex environment of the cloud system. Due to the limitations in responding to such situations based on individual administrator judgment or rule-based processes, the use of concept of artificial intelligence for IT operations (AIOps) has risen considerably. [2] This facilitates the integration of AI in IT operations with the overall aim of optimizing data centers efficiently, increasing customer satisfaction, and improving development productivity. From a research perspective, algorithms that can appropriately calculate resource capacity and preemptively respond to failures are considered key elements.

To maximize the performance of such algorithms, it is important to accurately analyze all sorts of data (metric, log, trace) collected from data centers and predict the values that they would show in the future. By predicting resource usage (metric), an appropriate capacity of resources can be calculated in accordance with the usage patterns for several months ahead. Furthermore, by predicting resource usage and potential log patterns, failure-related patterns can be preemptively detected and root causes can be analyzed, and this further shortens resolution times. However, because cloud technology has led to an increase in the complexity of environments, data pipelines have also become sophisticated and complex, requiring feature engineering that takes this into account.

For feature engineering, like other multivariate data, the correlation between numeric variables has significant implications. [3] Various studies have been conducted, including the use of correlation between the server temperature and disk-related variables and failures, [4] or diagnosing the cause of failures using the correlation between alarms generated based on network-related variables. Altogether, this helps analyze multiple events occurring at the same time from various perspectives.

Researchers have also focused on maximizing the performance of the target algorithm by analyzing the Granger causality between individual variables to leverage the characteristics of time-series data. [5] Patel et al. solved a multivariate time-series prediction problem involving CPU usage, memory usage, etc. using a Granger causality-based model.

Based on a detailed analysis of various studies, it was observed that when performing feature engineering using correlation, there are certain limitations in problems that require predicting future points, not just the current point in time. When Granger causality was used as an alternative, although causality could be effectively detected in some variables, there were many key variables wherein casualty could not be detected. In particular, while it is difficult to find clear correlations or causality with the target variable for most times, when a specific event occurs in the target variable, the variables that can play a significant role in triggering these events have been identified.

Therefore, we inferred that partial Granger causality, which occurs in the case of a meaningful event in the target variable, could play an important role. We propose a partial Granger causality-based feature selection method at the time of event occurrence, which maximizes the prediction performance of various numerical data occurring in cloud data centers, and can also be applied to several other derivative functions where causality is important.

Section 2 explains several key concepts associated with this research. Section 3 explains how we collected and preprocessed the data and implemented the proposed method. Section 4 presents the proposed technique and several other techniques into various prediction models, along with a detailed comparison of their performance. Finally, Section 5 presents conclusions pertaining to limitations and implications that can be derived from this research, along with the directions for subsequent research.

II. Related Work

A. Multivariate Prediction

a) Vector Autoregression

The Vector Autoregression(VAR) is an algorithm that examines the dynamic relationship between multivariate variables over time. [6] The VAR(p) formula between K variables is as follows.

$$Y_t = c + A_1 Y_{t-1} + A_2 Y_{t-2} + \cdots + A_p Y_{t-p} + e_t \quad (1)$$

In Equation (1), Y_t is a k-dimensional vector, c is a constant vector, and $A_1 \ldots A_p$ is a coefficient matrix of k by k dimensions. Here, e_t represents an error term vector. [7] Bussmann et al. proposed a method to identify non-linear relationships in time series data, and their method was referred to as neural additive vector autoregression. This model uses a deep neural network to nonlinearly extract Granger causal influences from multivariate time series. [8] The VAR has also been combined with recurrent neural networks for multivariate time-series data prediction.

b) KNeighbors Regressor

The KNeighbors Regressor(KNN regressor) is an algorithm that derives the value to be predicted by calculating the average of the nearest neighbors. Given the input training dataset $(X_1, y_1), (X_2, y_2), \ldots, (X_n, y_n)$ and the new input data point X_{query}, the output is the predicted value y_{query} for X_{query}.

It measures the distance $d(X_i, X_j)$between data points X_i and X_j, and it further selects K nearest neighbors. At this point, after selecting the K nearest neighbors to X_{query}, the predicted value y_{query} is calculated as the average of the dependent variable values of the K neighbors.

$$y_{query} = \left(\frac{1}{K}\right) * \sum_{i=1}^{K} y_i$$

[9] Farahnakian et al. proposed a prediction method, referred to as the KNN-UP(KNN-based Utilization Prediction), which is a dynamic consolidation algorithm that turns off hosts with low usage and leaves only the minimum hosts based on current and future resource usage. They used the KNN algorithm based on past resource usage data to predict future resource demand. This is a representative example of using KNN regression to predict hosts with excessive or insufficient load. [10] Ban et al. used KNN regressor in multivariate time series data regression.

c) Long Short Term Memory

[11] The Long Short Term Memory(LSTM) is an algorithm aimed at solving the vanishing gradient problem of Recurrent Neural Networks(RNNs). It has the characteristic that a single unit is composed of a cell, input gate, forget gate, and output gate, making it possible to control the retention of long and short-term memory.

The formula for the input gate is as follows.

$$i_t = \sigma(W_{xi}{}^T x_t + W_{hi}{}^T h_{t-1} + b_i)$$

The formula for the forget gate is as follows.

$$f_t = \sigma(W_{xf}{}^T x_t + W_{hf}{}^T h_{t-1} + b_f)$$

The update of the cell state is as follows.

$$g_t = \tanh(W_{xg}{}^T x_t + W_{hg}{}^T h_{t-1} + b_g)$$

The cell state is as follows.

$$C_t = f_t * C_{t-1} + i_t * g_t$$

The output gate is represented as follows.

$$o_t = \sigma(W_{xo}{}^T x_t + W_{ho}{}^T h_{t-1} + b_o)$$

The computation for the hidden state is as follows.

$$h_t = o_t * \tanh(C_t)$$

where,

i_t: Output of the input gate

f_t: Output of the forget gate

g_t Output of the update cell state

C_t: Current cell state

o_t: Output of the output gate

h_t: Output of the hidden state

x_t: Input vector at the current time step,

h_{t-1}: Hidden state at the previous time step

W_{xi}, W_{xf}, W_{xo}, W_{xg}: the weight matrices of each of the four layers for their connection to the input vector x_t.

W_{hi}, W_{hf}, W_{ho}, W_{hg}: the weight matrices of each of the four layers for their connection to the previous short-term state h_{t-1}.

b_i, b_f, b_g, b_o: the bias terms for each of the four layers

*: Matrix multiplication

[a, b]: Concatenation of vectors a and b

[12] Dang et al. proposed an efficient multivariate automatic scaling framework using Bi-LSTM in a cloud computing environment. They used the Bi-LSTM technique to predict future resource workloads. [13] Patel and Kushwaha attempted to predict the CPU utilization of cloud servers at continuous time steps using a prediction model method combining 1-dimensional Convolutional Neural Networks (1D CNN) and LSTM, referred to as the pCNN-LSTM(parallel CNN-LSTM).

B. Feature Selection

a) Recursive Feature Elimination

Recursive feature elimination (RFE) is an algorithm that sequentially removes the least important variables using feature coefficients after training on all features. [14] Nkiama et al. used a decision tree-based classifier as a recursive feature elimination technique to eliminate unrelated features with network traffic information and use only the related features, in the purpose of detecting intrusion. [15] Yan and Zhang proposed the SVM-RFE+CBR(support vector machine-recursive feature elimination and correlation bias reduction) method, which adds linear and nonlinear characteristics to improve SVM-RFE under specific conditions where it becomes biased.

b) RandomForest regressor - feature importance

The Random Forest regressor is an algorithm based on the ensemble method of Random Forest, which combines multiple decision trees to make predictions and averages the results to obtain the final predicted value. In Random Forest, the feature importance is an indicator of the contribution of each feature to the prediction. It is calculated by averaging the impurity decreases across all decision trees. This decrease is a decrease in impurity before and after the node split which was based on a particular feature in each decision tree. [16] Alduailij et al. adopted a method for selecting features using Mutual Information and Random Forest Feature Importance methods for DDoS Attack Detection.

c) Granger Causality Test

[17]The Granger causality test is a statistical algorithm that evaluates the causal relationship between two variables in time series data.

$$Y_t = \alpha + \sum(\beta_i * Y_{t-i}) + \varepsilon_t \quad (1)$$

$$Y_t = \alpha + \sum(\beta_i * Y_{t-i}) + \sum(\gamma_i * X_{t-i}) + \varepsilon_t \quad (2)$$

In these two equations, if it is shown through the p-value that (2) is a better regression model than (1) by performing a test such as F test, it is stated that x "Granger causes" y. [18] Sun et al. proposed a method for feature selection in multivariate numerical time series problems based on the Granger causality method. They stated that the Granger causality method is significant because it selects suitable time windows at the same time it chooses the appropriate relevant features. [19] Hmamouche et al. developed an algorithm which considers the hidden relationship that may occur between variables when using Granger causality for feature selection. [20] Dong and Kluger applied the feature selection method using Granger causality to high-dimensional biological data.

III. Methodology

A. Data Collection

To validate the feasibility of the proposed method, we prepared a virtualization environment based on OpenStack, an open-source platform. This environment consists a total of 34 resources, including 5 physical machines (PM) used as Hypervisors and 29 virtual machines (VM) created on those Hypervisors. This environment, where various services are being developed, was used for various purposes such as Web Server, DB(Databases), Kubernetes Cluster, etc. To collect metric data from these resources, we used ELK Stack, which is an agent-based server data collection standard pipelines. Through this, we collected data related to CPU, Memory, Disk, and Network every 5 minutes, accumulating data for five months from March 11 to July 11, 2022.

The collected data consists of a total of 89 variables. Among these, (A) CPU Utilization, (B) Memory Utilization, (C) Disk read bytes, (D) Disk write bytes, (E) Network in bytes, and (F) Network out bytes were selected as the target variables to be predicted. We seek to analyze the performance of the proposed algorithm based on a comparison with the performance of the multivariate time series prediction model that includes the remaining 83 data according to each variable selection technique.

Table 1 Hierarchical Structure of Virtual Resources

Physical Machine (PM)	Virtual Machine (VM)	# of instances
Contrabass #1	platform-haproxy-vmware	5 (include PM)
	platform-mkdocs	
	platform-nexus	
	platform-ptlmail	
Contrabass #2	bigdata-elasticsearch-data-3	28
	bigdata-elasticsearch-master-1	
	bigdata-kafka-1	
	bigdata-kafka-2	
	bigdata-kafka-3	
	bigdata-logstash-1	
	bigdata-logstash-2	
	cicd-harbor	
	platform-consul-1	
	platform-consul-2	
	platform-consul-3	
	platform-dns	
	platform-etcd0	
	platform-haproxy-lb	
	platform-k8s-master0	
	platform-k8s-worker0	
	platform-k8s-worker1	
	platform-k8s-worker2	
	platform-k8s-worker3	
	platform-k8s-worker4	
	platform-k8s-worker5	
	platform-keycloak-1	
	platform-keycloak-2	
	platform-nfs	
	platform-ptldb-1	
	bigdata-elasticsearch-data-3	
	bigdata-elasticsearch-master-1	
Contrabass #3	-	1
Contrabass #4	gu-dev-mariadb	2 (incude PM)
Contrabass #5	platform-dev-mariadb	3 (include PM)
	platform-haproxy	
Total		39

B. Preprocessing

In the case of VM units, we initially removed resources that were created midway and possessed a short data accumulation period. Additionally, we conducted further filtering based on the presence of minimal activity by checking the CPU usage and the number of processes running on the resource.

For the metric units, we initially removed variables that had negligible variance and were close to Unique or Categorical variables. Furthermore, we eliminated variables that conveyed nearly the same meaning as the target variables and had a correlation coefficient close to one.

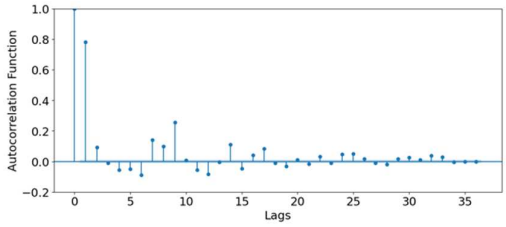

Figure 1 ACF(Auto Correlation Function) Plot for setting lag values

When selecting the lag value for the model, we utilized the average of the univariate standard ACF values of the overall target variables. We found that if the lag value was too large, the marginal increase in average accuracy according to the number of variables was minimal, and this resulted in a decrease in the feasibility of practical application in complex cloud environments where large volumes of data are collected in real time. Therefore, when applying the prediction model, we standardized the lag value to observe up to three steps back (15 minutes).

C. Model Architecture

The proposed algorithm is implemented in three major steps. The process consists of selecting the window to be analyzed through the Event Detector, checking the partial causality of the dependent variable through the Causality Detector, and deriving the final feature list by summing the partial causality found per window.

Algorithm 1: Function `feature_selection_with_partial_granger` for feature selection using partial Granger causality

```
function feature_selection_with_partial_granger
  (vm_df, window_size = 288, var_th_p = 75, top_k = 13);
Input  : DataFrame vm_df
Output: list feature_list containing selected features for each target
        feature
// Step 1: Retrieve important windows based on variance
window_imp ← empty list;
foreach window in windows of vm_df do
    if var(window) > var_th_p of vars(window) then
        window_imp.append(window);
    end
end
// Step 2: Perform Granger causality tests for windows
for window in window_imp do
    apply Granger causality test between target_feature and each
     candidate feature, and store the result in partial_granger_matrix;
end
// Step 3: Select features based on Granger causality
   results
feature_list ← empty list;
Count the number of values in partial_granger_matrix that are less than
  0.05, sort the values, and select the top-k elements, then store them in
  feature_list;
return feature_list;
```

Figure 2 Partial granger causality based feature selection algorithm

1) Event Detector: From the entire time series, a window that is as large as the actual window size is extracted and the occurrence of events is checked based on basic statistics such as the variance of the target variable in that section. The window is then slid by the stride value to explore the next window.

2) Causality Detector: By using the window list obtained through the Event Detector, we check for causality between the target variable and the remaining variables per section, based on Granger causality, and we store the p-values.

3) Partial Causality Summation: By using the Granger p-value matrix obtained through the Causality Detector, we sum each section and extract the variables that most frequently demonstrated causality.

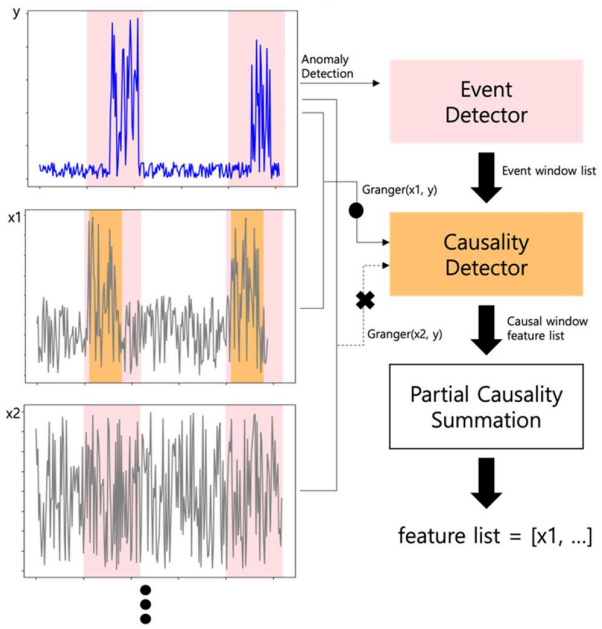

Figure 3 Granger causality performed In dependent variable y at the point where the event occurred (red), confirming that x1 has Granger causality (yellow) and that x2 does not

The three main feature selection techniques are the filter method, wrapper method, and embedded method. The filter method uses statistical measurements to discover correlations between features and then selects the features with a high correlation coefficient (influence).

The wrapper method is another approach to feature selection, where subsets of features are made iteratively to find the optimal feature combination.

The embedded method performs feature selection during the learning process of a specific machine learning algorithm.

We sought to test and compare various methods using the correlation coefficient method in the filter method, the RFE method with linear regression as the estimator in the Wrapper method, and the feature significances method which is embedded in the random forest regressor package in the embedded method.

We used a total of three prediction models: KNN regressor, LSTM, and VAR. The KNN regressor was chosen because it is a relatively lightweight algorithm that generally offers good performance with default parameters, and it also

works well with non-linear problems. We chose the LSTM model due to its high compatibility with time series data and its ability to process sequences of diverse features. The VAR model was used because it is most suitable for autoregressive models that depend not only on independent variables but also on dependent variables.

IV. Evaluation

To validate the performance of our algorithm and the comparison algorithms, we standardized the number of variables created by the feature selection results. We also used several prediction models to facilitate an objective comparison. For the predictive performance, we used two metrics: RMSE(Root Mean Squared Error) and MAE(Mean Absolute Error).

The formula for RMSE is as follows.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

The formula for MAE is as follows.

$$MAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n}$$

The main parameters for algorithm training are the basic size of the window wherein partial Granger causality is checked and the stride, which determines the unit to slide the window. Considering both the simple prediction performance of the model and the training time for the algorithm, we set the stride equal to the window size and set the window size to 288 (one day).

Table 2 Performance of Feature Selection Algorithms by Error Index by Prediction Model

Error Index	Prediction Model	RFE Based	RF Based	Correlation	Ours
RMSE	KNN	0.26360	0.27854	0.25583	0.22700
MAE	KNN	0.23394	0.24437	0.22323	0.19697
RMSE	LSTM	0.18197	0.17499	0.18066	0.15010
MAE	LSTM	0.32057	0.31441	0.31886	0.28593
RMSE	VAR	0.09989	0.12229	0.15955	0.05254
MAE	VAR	0.25013	0.25931	0.29000	0.15005

According to the performance test of the feature selection techniques for each indicator and prediction model, our algorithm exhibited superior performance in most indicators. In each prediction model, both RMSE and MAE showed the same trend in performance ranking in the feature selection model. Also, VAR prediction model consistently exhibited the lowest RMSE values across all feature selection models than other prediction models. Especially, in VAR model, our model showed the most dramatic performance improvement. In VAR, our model showed a 67% performance improvement over the correlation-based model, which showed the worst RMSE. In KNN, our model showed 18% performance improvement and in LSTM it was 17%.

In addition, the prediction performance of individual target feature in each prediction model was compared for each variable selection algorithm. In this performance comparison, the Error Index was set to RMSE and the prediction model was set to KNN Regressor for efficient experiment.

Table 3 Prediction Performance of Feature Selection Algorithms by y Variable

Target Feature	RFE Based	RF Based	Correlation	Ours
(A)CPU Utilization	0.03971	0.04503	0.03641	0.03541
(B)Memory Utilization	0.14830	0.15589	0.14962	0.08580
(C)Disk Read Bytes	0.44087	0.43683	0.38216	0.38857
(D)Disk Write bytes	0.33974	0.35384	0.35715	0.30134
(E)Network In Bytes	0.31887	0.35764	0.31981	0.28382
(F)Network Out Bytes	0.29410	0.32202	0.28984	0.26704

From the comparison of the predictive performance for individual variables, it was observed while other model showed better performance for some variable, our experimental results generally showed superior performance. For example, although for Disk Read bytes variable correlation-based feature selection model showed better performance than ours by 0.00641, our model showed the best performance in all other variables. Specifically, for CPU utilization, our model showed 21% better performance than the model that showed worst performance, and 45% for memory utilization, 15% for Disk Write bytes, 20% for Network In Bytes, and 16% for Network Out Bytes.

When our algorithm performed better, variables related to cpu steal or cpu iowait were selected as key variables in CPU Utilization. In addition, in terms of memory utilization, variables related to filesystem usage and disk write were selected as key variables. In terms of cpu utilization prediction performance, increasing values such as cpu steal or cpu iowait makes it difficult to perform other processes in the same physical machine or distribute cpu resources to multiple virtual machines, affecting the overall use of cpu. In terms of memory utilization prediction performance, as many of the vm's analyzed are related to data pipelines, if values such as filesystem usage or disk write increase, requests to save or query big data-related DB are processed and affect memory usage. In this way, it has been confirmed that variables selected through our algorithm can have important meaning even when using domain knowledge.

V. Conclusion

We successfully minimized the load on the cloud system by minimizing the number of variables that must be collected through an effective feature selection technique. A simple reduction in the large volume of collected data to about a dozen produces many benefits in terms of network load and storage space in the data collection environment. In addition to maximizing the prediction model's performance, we also improved the performance of techniques for cloud resource

optimization and anomaly pattern detection, which are connected to the predictive model.

However, a considerable volume of the data generated in cloud data centers, including unstructured data like logs and traces, often contain information related to causality. Therefore, using these data is essential to create functions that optimize the data center or respond effectively to failures. Accordingly, to create a root cause analysis model that detects the root causes of failures in cloud systems, our follow-up research will analyze causality including not only metric data but also logs and traces. We plan to apply this analysis to various models such as prediction and anomaly detection.

ACKNOWLEDGMENT

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. 2021-0-00256, Development of Heterogeneous Virtualization$^{(VM+Container)}$ Integrated Operation Technology for Intelligent Management of Cloud Resources)

REFERENCES

[1] Y. Li, Z.M.Jiang, H.Li, A.E.Hassan, C.He, R.Huang, Z.Zeng, M.Wang, and P.Chen., "Predicting Node Failures in an Ultra-Large-Scale Cloud Computing Platform: An AIOps Solution," *ACM Trans. Softw. Eng. Methodol.* vol. 29, pp. 1-24, Apr. 2020

[2] Y. Dang, Q. Lin and P. Huang, "AIOps: Real-World Challenges and Research Innovations," in 2019 *IEEE/ACM 41st Int. Conf. on Software Engineering*, Montreal, QC, Canada, pp. 4-5

[3] K.V.Vishwanath, N.Nagappan, "Characterizing cloud computing hardware reliability," in 2010 *1st ACM symposium on Cloud computing*. New York, USA, pp. 193–204.

[4] A. T. Bouloutas, S. Calo and A. Finkel, "Alarm correlation and fault identification in communication networks," *IEEE Transactions on Communications*, vol. 42, pp.523-533, Feb-Apr 1994

[5] Y.S.Patel, R,Jaiswal and R.Misra, "Deep learning-based multivariate resource utilization prediction for hotspots and coldspots mitigation in green cloud data centers," *The Journal of Supercomputing*, vol. 78, pp 5806–5855, May. 2022

[6] B.Xu and B. Lin, "Assessing CO2 emissions in China's iron and steel industry: A dynamic vector autoregression model," *Applied Energy*., vol 161, pp375-386, 2016

[7] B. Bussmann, J. Nys, S. Latré, "Neural Additive Vector Autoregression Models for Causal Discovery in Time Series," in *24th International Conference Discovery Science*, Halifax, NS, Canada, October 11-13, 2021, pp.446-460

[8] B. Agung, S. Isti, "Hybrid vector autoregression-recurrent neural networks to forecast multivariate time series jet fuel transaction price," in *2020 IOP Conference Series Materials Science and Engineering*, vol. 909, pp. 10, doi: 10.1088/1757-899X/909/1/012079

[9] F. Farahnakian, T. Pahikkala, P. Liljeberg and J, Plosila, "Energy Aware Consolidation Algorithm Based on K-Nearest Neighbor Regression for Cloud Data Centers," in *2013 IEEE/ACM 6th Int. Conf. on Utility and Cloud Computing*, pp. 256-259

[10] T. Ban, R. Zhang, S. Pang, A. Sarrafzadeh, D. Inoue "Referential kNN Regression for Financial Time Series Forecasting," in *2013 Int. Conf. on Neural Information Processing,* pp. 601-608

[11] A. Geron, "Forecasting Multivariate Time Series", in *Hands-on Machine Learning with Scikit-Learn, Keras TensorFlow* 3rd. Oreilly, 2022, pp.620

[12] Q. Dang, M. Nhat and M. Yoo, "An Efficient Multivariate Autoscaling Framework Using Bi-LSTM for Cloud Computing," *Applied Sciences*, vol. 12, pp. 3523-2543, Mar. 2022

[13] E. Patel, D.S. Kushwaha, "A hybrid CNN-LSTM model for predicting server load in cloud computing," *The Journal of Supercomputing*, vol. 78, pp. 1-30, May. 2022.

[14] H. Nkiama, S.Z.M. Said and M. Saidu, "A Subset Feature Elimination Mechanism for Intrusion Detection System," *International Journal of Advanced Computer Science and Applications*, vol. 7, 2016 .

[15] K. Yan and D. Zhang, "Feature Selection and Analysis on Correlated Gas Sensor Data with Recursive Feature Elimination," *Sensors and Actuators B: Chemical*, vol. 212, pp.353-363, June. 2015

[16] M. Alduailij, Q.W.Khan, M. Tahir, M. Sardaraz, M. Alduailij, F. Malik, "Machine-Learning-Based DDoS Attack Detection Using Mutual Information and Random Forest Feature Importance Method," *Symmetry*, vol. 14, 2022

[17] C.W.J.Granger, "Investigating Causal Relations by Econometric Models and Cross-spectral Methods," *Econometrica*, vol. 37, pp. 424-438, Aug. 1969

[18] Y. Sun, J. Li, J. Liu, C. Chow, B. Sun, and R. Wang, "Using causal discovery for feature selection in multivariate numerical time series," *Machine Learning*, vol. 101, pp. 377-395, Oct. 2015

[19] Y.Hmamouche, A.Casali, L.Lakhal, "A Causality Based Feature Selection Approach for Multivariate Time Series Forecasting," in *The Ninth International Conference on Advances in Databases, Knowledge, and Data Applications*, Barcelone, Spain, May. 2017

[20] M. Dong, Y. Kluger, "GEASS: Neural causal feature selection for high-dimensional biological data," in *The Eleventh Int. Conf. on Learning Representations*, 2022