sciendo

JAISCR, 2024, Vol. 14, No. 4, pp. 287 – 308
10.2478/jaiscr-2024-0016

OPTIMIZATION OF THE USE OF CLOUD COMPUTING RESOURCES USING EXPLORATORY DATA ANALYSIS AND MACHINE LEARNING

Piotr Nawrocki*, Mateusz Smendowski

Faculty of Computer Science, AGH University of Krakow, al. Mickiewicza 30, 30-059 Krakow, Poland

**E-mail: piotr.nawrocki@agh.edu.pl*

Submitted: 18th April 2024; Accepted: 4th June 2024

Abstract

Rapid growth in the popularity of cloud computing has been largely caused by increasing demand for scalable IT solutions, which could provide a cost-effective way to manage the software development process and meet business objectives. Optimization of cloud resource usage remains a key issue given its potential to significantly increase efficiency and flexibility, minimize costs, ensure security, and maintain high availability of services. This paper presents a novel concept of a *Cloud Computing Resource Prediction and Optimization System*, which is based on exploratory data analysis that acknowledges, among others, the information value of outliers and dynamic feature selection. The optimization of cloud resource usage relies on long-term forecasting, which is considered a dynamic and proactive optimization category. The analysis presented here focuses on the applicability of classical statistical models, XGBoost, neural networks and Transformer. Experimental results reveal that machine learning methods are highly effective in long-term forecasting. Particularly promising results – in the context of potential prediction-based dynamic resource reservations – have been yielded by prediction methods based on the BiGRU neural network and the Temporal Fusion Transformer.

Keywords: cloud computing, machine learning, exploratory data analysis, optimization, resource management

1 Introduction

The rapid development of cloud computing has been largely due to increasing demand for scalable, efficient, flexible and – most of all – cost-effective IT solutions, which are capable of managing the software development process and ensuring that business objectives are met. The relevance of clouds is embodied by the NIST (*National Institute of Standards and Technology*) reference architecture, which defines them as self-sufficient highly available environments, offering scalable resources in a flexible manner depending on user requirements, and enabling the use of solutions based on cost monitoring and control [1]. Operators of public computational clouds also provide a range of service provisioning models that can be flexibly tailored to meet the specific requirements of individual users [2].

However, despite the numerous benefits outlined, cloud computing solutions come with significant risks of inadvertent underprovisioning or overprovisioning of resources. Inadequate resource

management can result in the inability to fulfill user requests, leading to violations of Service Level Agreements (SLAs) [9]. Conversely, excessive reservations may incur additional costs for operators. Therefore, it is crucial to strike a balance between underprovisioning and overprovisioning of resources, maintaining equilibrium in terms of resource utilization and service levels, both quantitatively and qualitatively.

Users of computational clouds frequently employ a conservative strategy, which involves reserving resources with a wide safety margin, significantly exceeding resource usage under most circumstances – this is referred to as the static approach [9]. In many cases, additional resources are provisioned only when there are problems with serving incoming requests, and such resources are not subsequently released.

Given the significant increase in the use of cloud computing, techniques which enable resource usage optimization are of key importance in modern IT solutions. Optimizing the usage of cloud resources not only reduces end-user costs, but also limits wastage where resources are reserved but not actually utilized. The latter feature is particularly important for operators and providers of cloud platforms: it enables them to enroll new users while maximizing the ratio between resource usage and resource availability. In effect, such optimization has profound financial and business ramifications, and helps achieve the vision of Green Cloud Computing by reducing the consumption of energy [34].

The major contributions of this paper can be summarized as follows:

- designing a novel solution that provides resource usage predictions and optimization;
- using long-term prediction as an optimization category representing dynamic and proactive approaches;
- carrying out an evaluation using CPU usage data collected from a real-life cloud-based production system;
- conducting an exploratory analysis of the time series, which involves evaluating the significance of outlier data points;
- comparing the results obtained from different prediction techniques, including XGBoost, neural networks, and Temporal Fusion Transformer (*TFT*), with baseline statistical methods.

The rest of this paper is structured as follows: Section 2 contains a description of related work, Section 3 is concerned with presenting the model and architecture of the modular resource prediction and optimization system, Section 4 describes the exploratory data analysis and system evaluation, and Section 5 contains the conclusion and further work.

2 Related work

Improving the efficiency of cloud resource utilization encompasses various aspects, ranging from data center-level solutions to end-user-centric features, including predictive optimization methods. However, all data-driven actions taken in a cloud environment should undergo monitoring and verification. Monitoring systems enable the capture of historical resource usage data and provide timestamped values for specific performance metrics. This data can then be utilized to enable predictive modeling. Advanced monitoring platforms generate vast amounts of data, facilitating in-depth analysis and the training of machine learning models.

While reactive approaches have been popular in the past, the current trend is towards proactive optimization algorithms. Therefore, resource usage prediction is tied primarily to the problem of virtual machines taking a certain amount of time to boot up – typically between five and eight minutes [3]. This is why it is crucial to prepare a suitable pool of resources in advance, so that during a demand spike incoming requests may continue to be served without delay, maintaining a high quality of service without the need to statically reserve excess resources. As concerns time series forecasting, one of the most frequently applied statistical models is ARIMA (*Autoregressive Integrated Moving Average*) [4]. This model is recommended for data which exhibit trends as well as cyclical and seasonal variations; however, its usability is frequently restricted to cases where linear dependencies exist in the dataset. In [5], the authors report on the EEMD-ARIMA (*Ensemble Empir-*

ical Mode Decomposition-ARIMA, which extends the base ARIMA model, improving short-term resource usage prediction performance (vs. the base model) – however, the system implemented and deployed by the authors does not facilitate effective long-term prediction.

Statistical learning algorithms interpret historical resource usage data as tightly correlated time series. In complex scenarios – that is, when the patterns present in the data are nonlinear, long-term and intricate – such algorithms typically do not ensure high prediction accuracy. This is why there has been a growing reliance on machine learning methods capable of detecting periodic variations in highly stochastic datasets. In [3], short-term CPU (*Central Processing Unit*) usage prediction based on a Recurrent Neural Network (*RNN*) along with evolutionary optimization algorithms significantly outperformed methods based on linear regression, moving averages or feed-forward neural networks. However, anomalies present in input data were seen as an obstacle, hindering practical use of the proposed long-term prediction method. Furthermore, the authors of [8] analyzed key problems encountered by recurrent neural networks in the training process, which ultimately enabled the use of an LSTM (*Long Short-Term Memory*) network for load prediction.

In the context of predictive modeling, it is imperative to consider the interdependencies among various resource utilization patterns. Among these, a critical aspect is the intricate relationship between CPU and memory usage, which is significantly influenced by the number of I/O operations. The FLNN (*Functional Link Neural Network*) proposed by [6] and trained using the PSO (Particle Swarm Optimization) genetic algorithm enabled such dependencies to be taken into account. This specific architecture was selected on the basis of its simple structure and the attendant lower computational complexity. Evidence of how prediction accuracy benefits from awareness of dependencies between the use of various resources is offered in [10]. The list of metrics considered by the authors included, among others, the number of concurrent tasks, CPU and memory usage, as well as demand for hard drive space. Multivariate prediction using an LSTM network proved significantly superior to the corresponding univariate prediction. In [11], the authors propose another network with limited long-term memory; on the other hand, better prediction accuracy was achieved with the SaDE solution (*Self-adaptive Differential Evolution*), which is both simpler and faster than the reference PSO algorithm. In [13], particular emphasis is placed on structural analysis of neural prediction models, where the use of sparse architectures with a lower number of parameters enables more frequent retraining. The ability to adapt to incoming data is necessary in order to accommodate the dynamics of the environment and to maintain high predictive power. In production systems, single-pass training of models usually proves insufficient. In consequence, the bidirectional LSTM network achieves better results than the reference ARIMA model; however, all of the above-mentioned publications focus on short-term prediction. In contrast, the authors of this paper employ neural networks for long-term prediction tasks, which have been less frequently studied and yet may enable more optimal use of resources than in the case of short-term forecasting.

In [7], a comprehensive system is presented, which integrates critical elements such as anomaly detection, long-term prediction, the generation of weekly resource reservation plans, and real-time evaluation of the optimization loop. By actively monitoring and controlling the quality of predictions, the authors have established specific criteria for retraining existing models. A noteworthy advantage of this solution is its incorporation of historical performance data, which plays a pivotal role in sustaining the desired QoS (*Quality of Service*). Automatic generation of resource reservation plans on the basis of predictions was also reported in [12], where testing involved various prediction models, such as neural networks, linear regression, RepTree and M5P. Nevertheless, this research does not incorporate exploratory data analysis to uncover trends, dependencies, and patterns that may hold significance in the prediction process.

In [17], the author underscores the inherent flexibility of cloud computing; however, given the time needed to provision resources, the use of cloud platforms may potentially lead to SLA violations. To address this challenge, an algorithm rooted in Holt-Winters exponential smoothing was proposed, taking into account multi-seasonal cycles [32]. Its ef-

ficacy was subsequently compared with DES (*Double Exponential Smoothing*) and TES (*Triple Exponential Smoothing*) methods. Upon further evaluation using the CloudSim simulator, MAPE (*Mean Absolute Percentage Error*) was found to be less than 29%, which represents a major improvement compared to the straightforward application of both DES and TES, where the corresponding metrics were 44% and 135%, respectively. Although the prediction algorithm proposed by the author can be applied to arbitrary periods, the empirical assessment primarily favored a short-term (15-minute) prediction horizon. Additionally, it is noteworthy that the evaluation was conducted solely in a simulated environment, lacking validation against real-world data. In contrast, our research, as presented in this paper, leverages validation within a real-life usage scenario and focuses on assessing the performance of long-term predictions.

In [18], the authors present a new policy for the adaptive scaling of a Kubernetes cluster. Applying a default scaling strategy, which is based on static thresholds, often constrains system management capabilities. Consequently, vertical scaling (via *VPA – Vertical Pod Autoscaler*) is not commonly implemented in production environments. Furthermore, this mechanism necessitates the termination of running PODs – abstract objects that represent one or more tightly-coupled containers – and the subsequent creation of new ones adapted to altered resource allocations. This directly impacts the availability of applications operating within a cloud cluster; hence, the authors extended the standard Kubernetes orchestrator architecture by incorporating a new module for resource usage prediction, which is based on an LSTM neural network. The proposed solution resulted in a better resource allocation and improved QoS parameters compared to the default autoscaling mechanism. While specific details regarding the prediction window were not provided by the authors, the utilization of an LSTM architecture with a single neuron in the hidden layer suggests a short-term predictive approach, primarily focused on one-step-ahead predictions.

The prediction of cloud resource usage can be distilled to the broader challenge of time series forecasting. As a result, all modeling techniques applicable to such data may be potentially used in the context of cloud optimization, even if they were not originally developed for this purpose. For example, in [14], the authors focus on long-term time series prediction of financial market data. Changes in stock market indices are challenging to model effectively due to rapid and random variations, small-scale oscillations as well as large spikes – all of which are also observed in the context of cloud resource usage. The authors relied on three indices in their experiments (*Standard's & Poor's 500 Index, China Securities Index 300, Shanghai Stock Exchange 180.*), covering all of the aforementioned phenomena in order to better approximate the properties of a production environment. The outcome of this study was an adaptive deep learning model called LSTM-BN (*LSTM – Batch Normalization*), which was tasked with predicting the overall trends in stock prices, i.e. informing the user whether the closing price on the following day would be higher or lower than the closing price on the current day. During the evaluation phase, the model trained using the provided configuration exhibited subpar performance when processing a test dataset. Consequently, the authors explored an alternative approach, which involved employing multiple LSTM-BN networks operating in parallel and making decisions through a majority voting mechanism. This led to satisfactory accuracy of predictions as well as decent efficiency – measured as the estimated return on investment – compared to a reference GRU (*Gated Recurrent Unit*) network. Nevertheless, the presented research focused on analyzing and improving the structure of the model without engaging in complex feature engineering.

In [15], the authors report on their experimental evaluation of short-term prediction models within the context of power line usage. Such models enable the optimization of pricing strategies, effective management of electrical loads and efficient delivery of power to end customers. Similar to computational clouds, power lines are characterized by fluctuating load conditions that depend on many external factors. In the presented context, these factors mainly include human activity, geographical location, and season, all of which make accurate forecasting a non-trivial task. The authors conducted their evaluation on dataset that represents the consumption of electrical energy by individual households and contains over 2 million samples collected over a period of 47 months. The authors decided to limit its size by restricting analysis to measure-

ments gathered at 15-minute intervals. This decision was based on exploratory data analysis which revealed low variability of data within each such interval. The primary focus of the authors' work was on application of LSTM and TCN (*Temporal Convolutional Network*) networks, which resulted in a prediction error on the order of 0.75 for the RMSE (*Root Mean Squared Error*) metric. In addition, the authors demonstrated that the use of GRU cells, which are simpler than LSTM, did not significantly impact the quality of results.

The authors of [28] draw attention to certain properties of real-world time series, such as nonstationarity and frequent changes in temporal distribution. Standard supervised learning approaches are based on the assumption that each sample in the training set is independently derived from a fixed distribution (sometimes referred to as IID – *Independently and Identically Distributed*). In practice, significant differences in prediction accuracy between the training and validation datasets, as well as between the validation and testing datasets can be observed. In response to this challenge, in order to improve prediction and mitigate the impact of nonstationarity, the authors proposed an architecture-independent method called SAF (*Self-Adaptive Forecasting*). In terms of results, when compared to the sequence-to-sequence model, the use of LSTM and TFT (*Temporal Fusion Transformer*) based on the self-attention model reduced the MSE (*Mean Squared Error*) by over 40% and 19%, respectively. The attention mechanism enables the model to focus on key features of the sequence depending on the context, thus facilitating better assimilation of information and enhancing sequence processing accuracy—particularly in NLP (*Natural Language Processing*). In this context, it enabled the authors to achieve promising results in the realm of time series prediction.

In [29], the authors present a performance assessment of Transformer-based approaches. The proposed Transformer employs multi-head attention or self-attention mechanisms, which are highly efficient at identifying semantic relationships among elements in long data sequences. The authors acknowledge that self-attention exhibits permutational invariance, which could result in the loss of temporal information carried by the samples. This raises questions about the suitability of the Transformer architecture in the presented context. Consequently, the authors propose a new model called LTSF-Linear (*Long-Term Time Series Forecasting*-Linear), which consists of multiple single-layer linear models. This model facilitates time series prediction by computing weighted sums based on historical data. The authors conducted an analysis using nine diverse reference datasets, including road traffic patterns, energy consumption, economic changes, weather phenomena, and disease progression. The results revealed that the LTSF-Linear model significantly outperforms the base FEDformer (*Frequency Enhanced Decomposed Transformer*) model in most cases, with average accuracy improvements ranging from 20% to 50%. While the authors cannot definitively assert that the quantity of training data is sufficient for the unrestricted application of the model, the findings of this study suggest that the size of the training dataset is not a limiting factor when considering the feasibility of Transformer-based approaches.

Our study of the literature suggests a strong trend towards the use of machine learning and deep learning algorithms for cloud resource usage optimization, with a clear preference for dynamic and proactive methods. In the context of cloud optimization, machine learning plays a particularly noteworthy role when dealing with nonstationary and nonlinear data, where attention mechanisms emerge as significant competitor to recurrence. Nevertheless, classical statistical methods are still being used as a reference, and – in some cases – an augmentation of novel optimization techniques. Simpler models, and even historical rule-based mechanisms, may still be preferred primarily due to the simplicity and interpretability of their results.

Resource usage and time series prediction may be treated as independent optimization mechanisms or as contributors to the overall process of improving the operation of computational clouds. Moreover, time series prediction methods may be successfully applied to the optimization of cloud resource usage. It should be noted that most research focuses on short-term prediction, since it generally yields better results (in terms of evaluation metrics) for chaotic and nonstationary time series. On the other hand, long-term forecasting is significantly more valuable from the point of view of rational

decision-making and resource reservation planning. Notably, many of the published studies focus on machine learning aspects at the expense of data exploration or feature engineering. For non-trivial datasets, particularly when the distribution of samples varies over time, appropriate preparation of data is crucial. Moreover, it appears that complex models do not always ensure the best solution for a given problem – some studies aim at simplifying mechanisms in order to improve the interpretability of results while reducing computational complexity or the time required to train or adapt the model to an evolving historical dataset. Despite the fact that exploratory data analysis and the verification of the informational value of outlier samples are pivotal, many studies still tend to overlook these critical stages, focusing solely on the application of more complex models as an alternative to simpler ones when the latter do not yield satisfactory results. However, dedicating due attention to exploratory data analysis appears to be an indispensable step in making data-driven decisions. It is essential to emphasize that a system aimed at forecasting resource utilization must also be designed in a manner that does not generate significant costs and overhead.

3 Cloud computing resource prediction and optimization system

The developed system model embodies a proactive and dynamic approach to optimizing cloud computing resource utilization, emphasizing long-term prediction as the cornerstone of its methodology. Long-term prediction is a considerably more complex task than short-term prediction in terms of accuracy, especially when modeling time series with chaotic, dynamic, and highly variable characteristics. Nevertheless, it is precisely long-term prediction that enables making more informed decisions regarding dynamic resource allocation. As a consequence, the *Cloud Computing Resource Prediction and Optimization System Model* is based on the dual-loop continuous optimization scheme, where the fundamental aspects involve the use of historical data to make cyclical forecasts for a specified time horizon, followed by the evaluation of prediction quality through newly acquired real-world data. The operation of the first loop is synchronized with the length of the prediction horizon. Initially, it includes the *Anomaly Detection Module* to detect and handle outlier samples. Next, the *Data Analysis Module* enables the preparation of data in a format acceptable to the appropriate long-term forecasting model, which is trained and utilized within the *Training Module* and *Prediction Module*, respectively. Once a set of predicted values is obtained, the system transitions to the second loop, with its cycle synchronized with temporal intervals between samples. The *Monitoring Module* carries out the decision-making process to determine the exit point from the second loop and, consequently, the start of another iteration of the system, by comparing actual and predicted resource utilization based on selected error metrics. Additionally, newly recorded actual resource consumption gradually expands the set of historical data.

The *Cloud Computing Resource Prediction and Optimization System* architecture consists of five modules (Figure 1). In the context of time series prediction, a critical aspect is the analysis of outliers whose proper interpretation and handling can significantly impact the accuracy of long-term forecasts [16]. The anomaly detection process makes it possible to determine whether the outlier samples introduce unwanted noise and disturbance into predictive models and whether they carry significant informational value. At the start of the system's operation, the *Anomaly Detection Module* segregates a training and validation set from the available historical data. To ensure the correctness of this division, it is essential to consider the temporal dependency among recorded metric values, as random sample placement in sets would disrupt the order determined by the timestamp. Moreover, the division of historical resource utilization metrics into two subsets allows both for the application of a classical validation set and the utilization of k-fold cross-validation techniques for time series [23]. Another stage carried out within this module involves fitting the anomaly detection model selected to the previously extracted training set, and subsequently detecting and labeling outlier samples in both the training and testing sets using an unsupervised learning model (*Isolation Forest* [25]).

Subsequently, *Data Analysis Module* is responsible for appropriately preparing the time series and adapting it to the required format for the applied

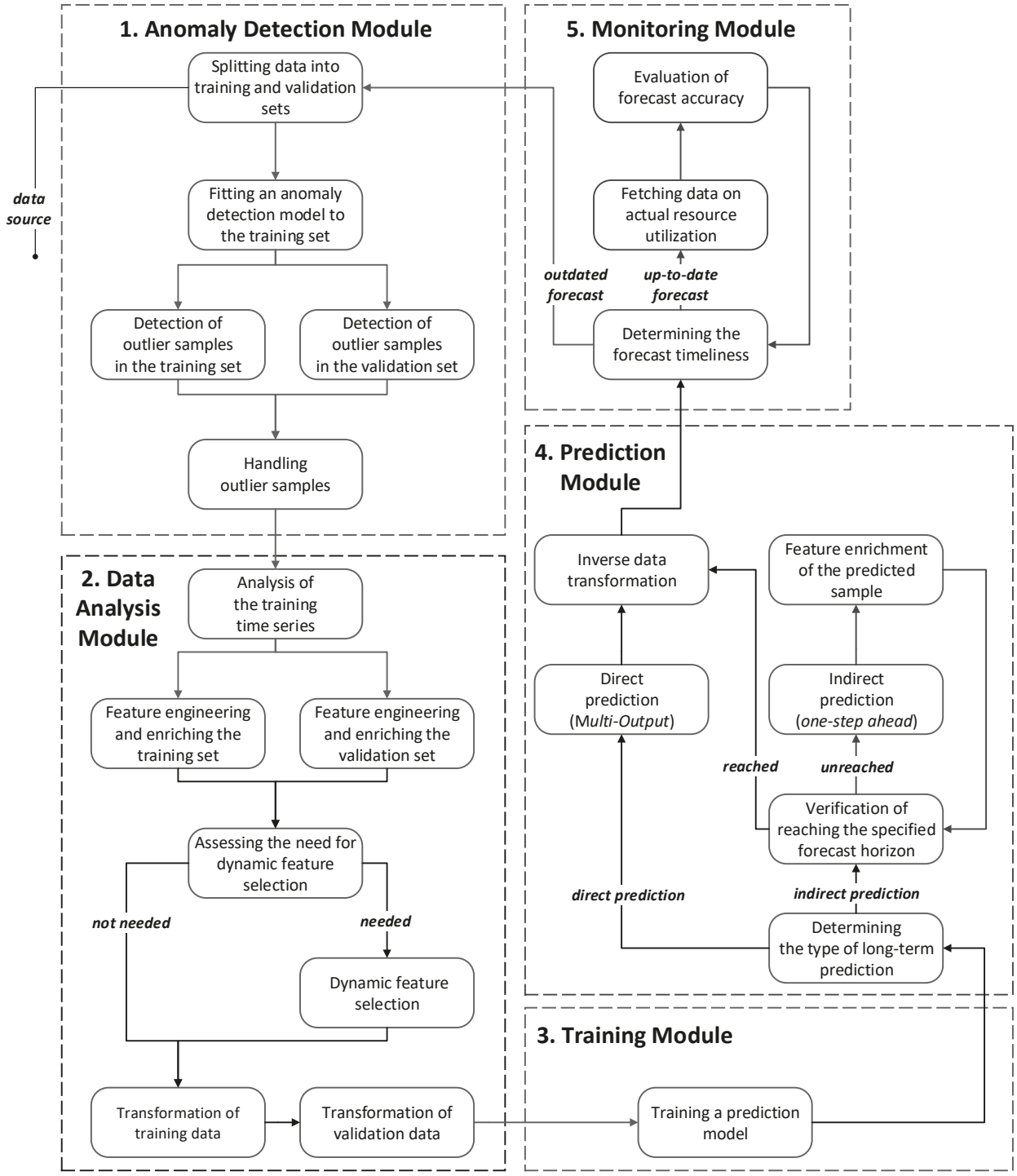

Figure 1. Cloud Computing Resource Prediction and Optimization System Architecture.

prediction model. Within this module, data enrichment and feature engineering are carried out. Data enrichment is performed separately for the training and validation sets to minimize the risk of information leakage. However, adding too many features can necessitate working with more complex models, which may have a negative impact when considering periodic fine-tuning with continuously acquired real-world data. Simultaneously, poorly correlated or inappropriate features can significantly deteriorate model results or lead to overfitting if proper regularization mechanisms are not applied. Therefore, maintaining a balance between feature enrichment and model complexity control is crucial. To achieve this, the concept of dynamic feature selection has been proposed, which is based on automatically selecting the most significant features depending on the characteristics of the training data. The dynamic feature selection mechanism consists of two components: first, creating a feature ranking (utilizing the built-in functionality of the XGBoost model), and second, aggregating information about features that will not be directly utilized (using Principal Component Analysis – *PCA* [30]). The use of dynamic feature selection accelerates the training and enables the use of smaller models. However, the use of dynamic feature selection is not always beneficial. When operating on a set of fully valuable features, it may turn out that all predictors obtained after enrichment are essential and should remain in the dataset. The final stage of this module's operation is data transformation (first on the training set and then on the validation set), encompassing all operations related to scaling values or removing certain components from the time series.

The third module of the *Cloud Computing Resource Prediction and Optimization System* is the *Training Module*, which is responsible for conducting the training of the selected time series prediction model. This module employs the early stopping mechanism [24] to monitor the model's ability to generalize on the validation set and effectively prevent overfitting. Following the completion of training, the model is saved and versioned.

The *Prediction Module's* task is to perform long-term forecasts of time series values representing the utilization of a selected cloud computing resource over an agreed-upon time horizon. Given the previously applied data transformations, it is important that the prediction results obtained are further processed through inverse transformations, such as returning to the original value scale. Within the *Prediction Module*, two alternative strategies have been delineated to achieve a long-term prediction horizon [21] – multi-step ahead forecasting. These distinct strategies have been labeled as separate paths within the architecture of the discussed module. The *indirect prediction* option reflects an approach where a one-step-ahead forecasting model is used. An alternative option is *direct prediction*, which allows obtaining values for the entire prediction horizon in a single step. The iterative approach is susceptible to error accumulation but is generally less computationally demanding. Training a model for direct prediction, although less sensitive to error accumulation, is characterized by a higher degree of complexity. The ability to use one of the two prediction options is particularly significant in the context of sample enrichment and feature engineering. Taking this important difference into account helps prevent unwanted information leakage.

The last module of the *Cloud Computing Resource Prediction and Optimization System*, operating in the second loop, is the *Monitoring Module*. It cyclically verifies whether the prediction previously made is up to date. If it is, the system acquires newly recorded resource consumption values and, based on them, assesses the quality of the forecast. Simultaneously, the resource consumption values recorded continuously extend the historical data set. In the system developed, the second loop operates at intervals converging with the frequency of recording resource consumption values. When the resource consumption prediction becomes outdated, meaning that a complete set of real-life resource utilization data over the past prediction horizon has been obtained, the system exits the second loop and transitions to the first loop, commencing another iteration. Exiting the second loop is equivalent to obtaining a comprehensive assessment of prediction quality.

In the *Monitoring Module*, five distinct error metrics have been employed to compare actual values with predictions. These metrics include RMSE, NRMSE (*Normalized RMSE*), MAE (*Mean Absolute Error*), MAPE, and MdAE (*Median Absolute Error*) [22].

The overall operation of the *Cloud Computing Resource Prediction and Optimization System* can be defined at a high level as periodic interactions between the first and second loops. The first loop, involving modules 1-4 of the system, operates at intervals consistent with the chosen prediction horizon. Each successive iteration of the system results in new metrics being recorded, thereby expanding the historical data. Consequently, anomaly detection mechanisms, prediction models and other solutions that operate strictly on the training set, such as data scaling mechanisms in the transformation process, can be gradually fine-tuned and updated on growing historical data. Optional fine-tuning models on new data helps mitigate the degradation of prediction performance in accordance with the dynamics of the cloud environment. The *Cloud Computing Resource Prediction and Optimization System* takes into account the presence of outlier samples in the data and conducts statistical and exploratory data analysis in conjunction with the process of feature enrichment and engineering. Additionally, the system can employ dynamic feature selection, which is particularly relevant in non-tree-based models.

4 Evaluation

As part of the evaluation of the developed system, exploratory time series analysis and assessments of long-term prediction methods were conducted in various experimental scenarios.

4.1 Exploratory Data Analysis

The fundamental objective of the EDA (*Exploratory Data Analysis*) was to learn the structure and semantics of the dataset, appropriately prepare the time series, handle missing and erroneous values, and identify challenges facing long-term prediction methods. An important aspect of the research was also feature engineering and dataset enrichment, including the application of the dynamic feature selection.

A significant portion of the research that leverages resource utilization forecasting for optimization purposes is based on publicly available datasets. One representative example are the cluster load metrics recorded in a Google data center in 2019, known as the *Google Cluster Workload Traces*. To evaluate the *Cloud Computing Resource Prediction and Optimization System*, data from a Microsoft Azure-based production environment were utilized. Over a period of twenty months, multiple web application migrations were conducted between two virtual machines. The metric recorded using monitoring systems was CPU utilization, available at three aggregation levels (one minute, ten minutes and one hour).

The first step in the time series preprocessing involved handling irregular readings, identified by a missing timestamp or a value that fell outside the interval from 0 to 100, beyond which percentage CPU utilization cannot be correctly interpreted. In both of these cases, a moving average with a window width of 24 samples was used. In the second stage, data with hourly aggregation were utilized; this decision was driven by two reasons. Firstly, this interval was aligned with the goal of effectively capturing trends using long-term prediction models rather than overemphasizing local fluctuations in CPU utilization characteristics. Secondly, considering a weekly prediction horizon, this distance between the samples balanced the dataset size with a reduction in the computational complexity of the problem. If minute-level or ten-minute-level intervals were adopted, weekly predictions would entail forecasting 10,080 and 1,008 samples, respectively. In order to achieve alignment with the defined prediction horizon, a slight reduction in the original dataset, consisting of 13,476 samples, was performed. Ultimately, a time series containing 13,440 measurements was obtained, representing CPU utilization over an 80-week period.

Furthermore, the data were split into training and test sets in an *81.25:18.75* ratio. Consequently, the last 2,520 samples, representing a 15-week period were for the objective evaluation of the generalization ability of the long-term prediction models employed in the *Cloud Computing Resource Prediction and Optimization System*. In the case of time series data, the data were divided into two disjoint sets according to the timestamp, without random sampling. This is particularly important because the characteristics of the training and test sets in time series data can differ significantly, going beyond the assumptions of standard supervised learning. Additionally, the test set was used for simulat-

ing the progressive acquisition of data during the operation of the prediction and optimization system. The further part of the exploratory data analysis was conducted on the training set, following best practices to prevent unwanted information leakage and to develop strategies that could be objectively verified on the test set. Furthermore, the preliminary analysis of the training time series allowed for the observation of irregularities and variability in the dataset, confirming the importance of conducting exploratory data analysis. However, such analysis did not reveal a clear seasonality or trend in the examined time series.

A crucial step in the exploratory analysis of a time series is determining its stationarity and defining an appropriate data transformation that maintains statistical parameters such as mean and standard deviation constant over time. The first transformation applied was normalization, where the mean value was subtracted from each sample, and the result was divided by the standard deviation calculated within individual local windows. The second transformation involved differencing operations, subtracting the value recorded at the previous timestamp from each sample.

To determine the stationarity of the training subset of the original time series and the series obtained after applying transformations, statistical tests, including the DF (*Dickey-Fuller*) and KPSS (*Kwiatkowski-Phillips-Schmidt-Shin*), were conducted, with a significance level set at the standard value of 0.05 [19] [20]. According to the null hypothesis of the first test and the results presented in Table 1, the training series was found to be non-stationary. Nevertheless, the test statistic value for the normalized time series and especially for the differenced time series, along with a p-value of zero, decisively rejects the null hypothesis in favor of the alternative hypothesis of data stationarity.

The second test conducted was the KPSS test, where the null hypothesis suggests the stationarity of the time series. For the original training time series, the test statistic value obtained of 15.324, at a significance level of 0.01, i.e. lower than 0.05, leads to the rejection of the null hypothesis in favor of the alternative hypothesis indicating nonstationarity of the series (Table 1). However, the results obtained for the transformed series, with p-values greater than 0.05, do not allow for the rejection of the null hypothesis, indicating that the series are stationary.

Table 1. Results of statistical tests determining the stationarity of training time series.

	Statistic		**p-value**	
Time Series	**DF**	**KPSS**	**DF**	**KPSS**
Original Time Series	-2.818	15.324	0.0558	0.01
Normalized Time Series	-19.02	0.131	0.0	0.1
Time Series after Differencing	-21.39	0.031	0.0	0.1

The determination of stationarity enables efficient data analysis, and becomes mandatory especially when using classical time series modeling methods. In summary, a single differencing operation enabled the transformation of the training time series into a stationary one.

In the subsequent stage of exploratory data analysis, autocorrelation function (*ACF*) and partial autocorrelation function (*PACF*) on the differenced stationary time series were employed to identify seasonality in the data. The primary aim of seasonal decomposition of the time series was to disentangle and separate components such as trend, seasonality, cyclical fluctuations, and noise. An additive model was used to determine the relationships among these components. The results of the prior autocorrelation and partial autocorrelation analysis made it possible to observe daily and three-day periodic patterns in the data, which were utilized when specifying the assumption of seasonality during decomposition. As an additional layer, weekly seasonality was examined to verify patterns within a one-week horizon. Figure 2 illustrates the outcome of the decomposition for a sample four-week period in the training dataset. The presence of seasonality in the data implies that the time series exhibits a periodic regularity in its structure, which has the potential to be captured by the model for forecasting future CPU utilization. Simultaneously, in Figure 2, it can be observed that the weekly seasonal component has a significantly more complex structure than the daily one. This confirms the hypotheses arising directly from the data, indicating that long-term prediction is more challenging than short-term modeling.

Furthermore, the next step was the analysis of outliers to verify their impact on prediction accuracy. In this stage, outliers were identified, and

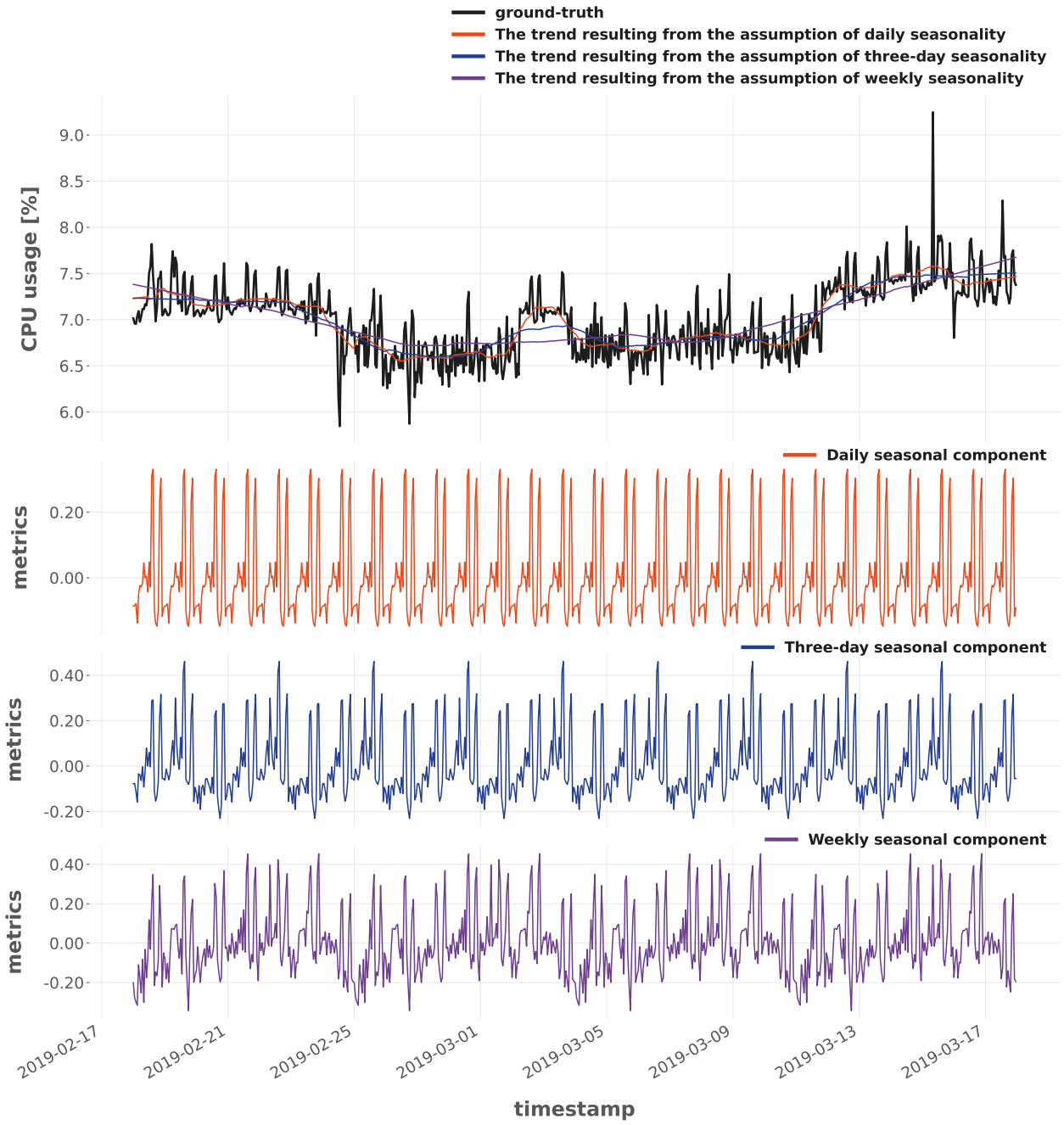

Figure 2. Seasonal decomposition applied with respect to the selected period of the training time series.

potential risks associated with their removal or replacement were analyzed. To detect them, an unsupervised model – the Isolation Forest – was used with a configuration of 90 estimators and a contamination rate set to 0.05, meaning that 5% of the analyzed range was assumed to be outliers. These outliers represented short-term, local irregularities manifested as spikes in resource utilization, which did not indicate a long-term trend change. In the context of defining experimental scenarios, the crucial question was to assess whether such points constituted noise that did not add value to predictive models or whether they represented significant information that should be retained in its original form. The outcomes of outlier analysis tend to exhibit idiosyncratic characteristics, as the desirability or undesirability of outliers cannot be universally asserted. Consequently, the pivotal emphasis lies on exploratory data analysis, affording the opportunity to scrutinize this aspect preemptively, well in advance of the implementation of predictive modeling procedures.

Subsequently, feature engineering aimed at constructing an enriched data representation. Firstly, timestamp-specific properties were harnessed to engineer features directly rooted in the calendar, encompassing variables to determine the hour, quarter, month, day number in the year, day number in the week, day number in the month, week number in the year, as well as the categorization of whether a sample was registered on a working day, weekend, or holiday. The variation in CPU utilization patterns across different times of the day represents a pivotal factor that can be attributed to the dynamic activity of users in cloud computing environments. Subsequently, cyclical features were introduced based on the previously constructed calendar features. This incorporation addressed cyclical patterns concerning hours, days of the week, months of the year, and days of the month. The primary purpose of introducing cyclical features, achieved through the use of the sine and cosine functions, was to illustrate that 5 AM and 7 AM are equally close as 11 PM and 1 AM. Similar patterns can be demonstrated in relationships between days of the week or months of the year. In the subsequent stage of feature engineering, a distinct analysis of time series in both the time and frequency domains was carried out. To achieve this, the Fast Fourier Transform (*FFT*) was used to obtain data representation in frequency domain. Furthermore, the preceding seasonal decomposition and correlation analysis informed the determination of sliding window sizes, within which statistical features were computed. Consequently, for both time-domain and frequency-domain time series representations, independent statistical features were constructed within sliding windows of 72 and 168 samples in length. These features encompassed moving averages, moving standard deviations, moving medians, as well as maximum and minimum values within the respective windows, along with their pairwise differences. Additionally, the following predictors were introduced, further enriching the sample representations within sliding windows: the count of samples above and below the mean or median, the count of peaks in the time window—defined as points where adjacent values have lower values. Ultimately, this comprehensive data enrichment culminated in the creation of 90 features.

Exploratory data analysis allows for a deeper understanding of the dataset, its structure and its peculiarities. This process facilitates domain knowledge acquisition and a better grasp of the problem context. Properly conducted exploratory data analysis unveils patterns, dependencies and trends within the data, which can be highly relevant for forecasting tasks. This translates into the ability to define more precise experimental scenarios. Simultaneously, exploratory data analysis helps identify missing data, assess the significance of outliers, and handle irregular data points. This stage is of paramount importance and can significantly impact the effectiveness and reliability of prediction models, especially through the feature engineering process. The significance of feature engineering is further underscored by its dedicated use within a separate *Data Analysis Module* in the *Cloud Computing Resource Prediction and Optimization System*.

4.2 Long-term prediction method evaluation

The *Cloud Computing Resource Prediction and Optimization System* enables the assessment of various time series forecasting models and provides flexibility in defining different experimental scenarios. In the *Prediction Module*, determining a baseline model was essential, for which the sta-

tistical Holt-Winters exponential smoothing model was recognized. Subsequently, the reasons for using XGBoost were primarily its flexibility, speed of operation, the ability to natively handle missing data, no need for data transformation or scaling, and the possibility of directly using both categorical and numerical features [27]. Another advantage of this model is the ability to control model complexity by specifying the maximum size of constructed trees, which is particularly useful when dealing with a large number of features. The XGBoost model was used in a one-step ahead prediction variant, which implies the choice of the *indirect prediction* path. Another model considered was the Recurrent Neural Network, which enables dependencies between samples that are distant in time to be captured and modeled. However, classical RNNs suffer from the vanishing or exploding gradient problem [14]. Consequently, alternative architectures were analyzed, using LSTM and GRU cells. Finally, the TFT model [31] was selected. The attention mechanism represents a significant advancement compared to recurrent networks [26]. Both in the case of neural networks and of the transformer-based model, the *direct prediction* approach was used.

In the research conducted, exponential smoothing with seasonality, specifically Holt-Winter's Exponential Smoothing with Seasonality, was used in three different configuration variants (*Methods 1, 2*, and *3*). A common feature of *Methods 1, 2* and *3* was the adoption of an additive model in terms of the relationships between time series components, such as trend, seasonality, and noise, and the introduction of a *damping factor*. This factor enabled the influence of past observations to be reduced. The first method assumes a seasonality with a period of 24 time units, which, in the case of hourly sample intervals, can be effectively interpreted as daily seasonality. On the other hand, *Methods 2* and *3*, respectively, assume a three-day and weekly period for the occurrence of periodic patterns.

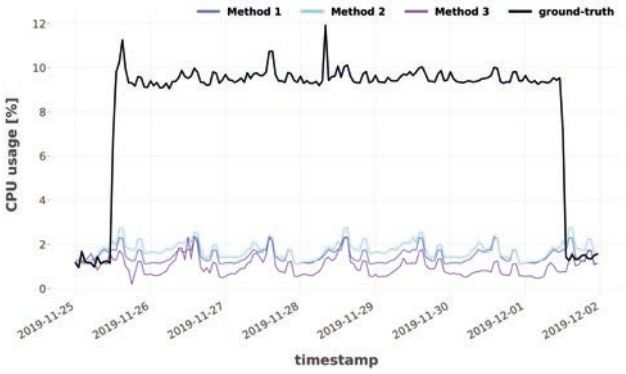

Figure 3. Results of long-term CPU utilization predictions obtained using baseline methods.

Each of the three baseline models was independently fitted to the entire training dataset without prior application of the outlier detection procedure. Additionally, each model underwent fine-tuning on gradually expanding training set after each completed prediction for a period of one week. Analyzing the error metrics obtained on the test dataset (Table 2), it was evident that *Method 3* turned out to be the least accurate, consistently yielding the highest error metric values. Focusing solely on error metrics, one might conclude that benchmark methods yield relatively good results, with *Method 2* achieving the best performance. It is essential to assess them in tandem with an analysis of the predicted characteristics juxtaposed with the ground-truth CPU utilization, as depicted in Figure 3. In general, baseline methods do not react to drastic changes in characteristics, which could potentially lead to both resource overprovisioning and underprovisioning if these methods were to be adopted within the *Cloud Computing Resource Prediction and Optimization System*.

Table 2. Error metric values for prediction results on the test dataset obtained using baseline methods.

	RMSE	NRMSE	MAE	MAPE	MdAE
Method 1	2.223	0.141	0.914	0.267	0.327
Method 2	2.168	0.137	0.936	0.276	0.374
Method 3	2.339	0.148	0.999	0.261	0.415

The first non-baseline machine learning model tested in the *Cloud Resource Utilization Prediction and Optimization System* was the XGBoost model. However, before proceeding with the evaluation of long-term prediction results on the test dataset, it was necessary to appropriately tune the model's hy-

perparameters. To assess its generalization ability, a validation set was first extracted, comprising the last 5 weeks of the training dataset, totaling 840 samples. Both Mean Squared Error and Mean Absolute Error cost functions were verified. However, considering the chaotic nature and rapidity of changes in time series, a single arbitrary validation set was abandoned in favor of k-fold time series cross-validation. As a result, the XGBoost model was trained with 240 estimators (equivalent to the maximum number of rounds of constructing successive trees to improve prediction errors – boosting rounds), a maximum tree depth of 7, and a learning rate of 0.28. Additionally, the early stopping mechanism was configured to terminate the learning process if there was no improvement in results on the currently considered validation set in k-fold time series cross-validation for six rounds. The XGBoost model was trained to predict values one time step ahead. Consequently, during the prediction phase, an iterative approach was used to expand the prediction to the adopted weekly time horizon. Ultimately, this model was employed to assess various experimental scenarios using the *Cloud Resource Utilization Prediction and Optimization System*.

Following the fine-tuning of the prediction model's hyperparameters, a similar process was conducted for the Isolation Forest. To minimize the search space, the focus was placed solely on the number of estimators and the contamination fraction, ultimately setting them at 90 and 0.05, respectively. Thanks to the aforementioned flexibility of the XGBoost model, it was unnecessary to apply data transformation or scaling features within the *Data Analysis Module*. Additionally, in the feature enrichment process, lagged features from the week preceding the prediction were introduced. This allowed the model to access values recorded over the time period from *t-167* to *t* during training and predict at time *t+1*. Since the model's complexity was controlled using the maximum tree size, only the most significant predictors were used, eliminating the need for dynamic feature selection. Furthermore, as feature enrichment included features derived from statistics within a time window, outliers were addressed by removing them.

Consequently, 11 testing scenarios were defined, representing 11 long-term prediction methods (*Methods 4-14*) based on the XGBoost model. Firstly, *Method 4* assumes no updates or re-adaptation of the model to the growing historical dataset and does not make use of the *Anomaly Detection Module*. *Method 5* encompasses all the assumptions of *Method 4* but incorporates a single use of the Anomaly Detection Module – against the entire training set. *Method 6* mirrors *Method 5*, with the *Anomaly Detection Module* applied differently. After each iteration of k-fold cross-validation, which separates the training and validation sets, the first set is divided into portions representing four-week periods. An independent anomaly detection model is then fitted to each portion, representing local data ranges, to identify and label outliers, which are subsequently removed. *Method 7* involves dividing the training datasets, acquired successively in k-fold cross-validation for time series, into portions representing weekly periods. Outlier samples are detected using independent models for each of the seven-day subsets obtained. *Method 8* incorporates the re-adaptation of the XGBoost model used for long-term prediction to follow the dynamics of changes in CPU utilization characteristics. *Method 9* implements the assumption of *Method 8*, addressing the active adaptation of the model to newly acquired CPU utilization data, primarily preventing performance degradation. The Anomaly Detection Module is utilized once. Outlier samples are filtered and labeled as such in the first iteration of the system using training data. Subsequently, in each system iteration, the XGBoost model becomes updated. Furthermore, *Method 10* combines the assumptions regarding periodic model updates implemented in *Method 8* with the local utilization of the *Anomaly Detection Module* for each of the extracted four-week historical data portions. *Method 11* combines the testing scenarios of *Method 7* and *Method 8*. Consequently, the *Training Module* is used not only for training the prediction model but also for its re-adaptation. Consequently, *Methods 12, 13, and 14* are direct extensions of *Methods 9, 10* and *11*, where an additional aspect of anomaly detection and handling on newly acquired CPU utilization data (from the test set) has been implemented. One of the fundamental assumptions of the testing scenarios defined was to examine the impact of samples classified as outliers on prediction accuracy, aiming to verify whether the *Anomaly Detection Module* should be utilized.

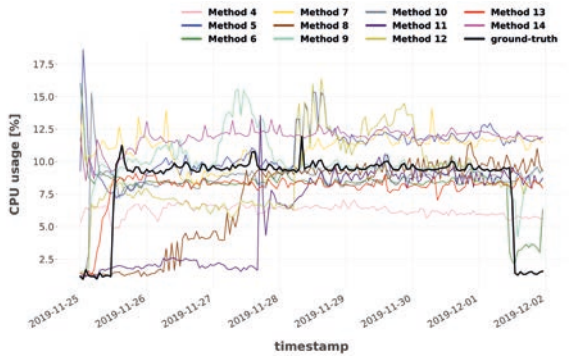

Figure 4. Results of long-term CPU utilization predictions obtained using XGBoost-based methods.

Analyzing the results (Figure 4), it's evident that predictions are closer to actual utilization compared to baseline methods. Notably, *Method 8* shows an adaptive response to changing characteristics, gradually adjusting forecasts to match actual utilization levels. These findings validate the importance of model retraining for evolving datasets and highlight the advantages of proactive approaches over baseline methods.

Table 3. Error metric values for prediction results on the test dataset obtained using XGBoost-based methods.

	RMSE	NRMSE	MAE	MAPE	MdAE
Method 4	2.477	0.157	1.355	0.502	0.407
Method 5	6.757	0.428	5.756	3.316	7.051
Method 6	14.762	0.936	9.352	5.121	1.882
Method 7	19.45	1.233	13.565	7.819	5.353
Method 8	1.814	0.115	0.909	0.335	0.335
Method 9	2.701	0.171	1.359	0.526	0.332
Method 10	2.573	0.163	1.185	0.58	0.321
Method 11	2.332	0.148	1.082	0.317	0.325
Method 12	2.627	0.167	1.195	0.546	0.321
Method 13	3.209	0.203	1.851	0.75	0.419
Method 14	3.209	0.192	1.399	0.646	0.334

Analyzing the error metrics obtained on the test dataset, as presented in Table 3, it can be observed that the methods utilizing anomaly detection performed significantly worse than the leading *Method 8*, which has an RMSE metric value of 1.184, which is 16% better than the best baseline method. Even in the case of model retraining, *Methods 9, 10* and *11* yielded worse results than the method without anomaly detection. Moreover, additional anomaly detection applied to newly acquired data through the *Monitoring Module*, as reflected in *Methods 12, 13* and *14*, did not contribute to performance improvement. This leads to the conclusion that a dual approach to anomaly handling is required. In the case of datasets where trend and seasonality are not easily discernible, removing anomalies or replacing them with other metrics can lead to valuable information being lost. Especially in the case of historical data, sudden spikes and drops can provide crucial insights, enabling better predictions. However, such values should remain within the proper CPU utilization range (0-100). On the other hand, the analysis of detected outliers may include those that fall outside this range and were interpreted as incorrect readings during exploratory data analysis, being replaced with a rolling window-based mean. In the case of changes in the dataset's characteristics, it is essential to develop a strategy for periodically adapting the model to new data. In the case of *Method 8*, retraining was synchronized with the progress of the first loop.

Further analysis of the results obtained has shown that the anomaly detection process strongly depends on the dataset utilized. If anomalies are removed from the dataset or the retraining procedure is omitted, the risk of degrading the model's performance is high. Furthermore, when operating on models predicting one step ahead, expanding predictions in an iterative manner may result in a rapid accumulation of errors.

As a result of analyzing the error metrics of the developed system based on the XGBoost model, a decrease in prediction accuracy was observed when the *Anomaly Detection Module* was applied. There was also a high risk of error accumulation associated with the iterative method of forecasting multiple steps ahead. Therefore, in the case of test scenarios for neural networks, the focus was on direct CPU usage prediction with a weekly prediction horizon, with weekly retraining of models in all defined test cases. Undoubtedly, regular adaptation to new data, especially in the face of potential changes in their characteristics, proved to be crucial in preventing system performance degradation. As a result, several alternative neural network architectures were considered, utilizing both LSTM and GRU cells. Time series k-fold cross-validation was also applied.

The following section presents an overview of the experimental scenarios (*Methods 15-24*) and provides a detailed description of architectural con-

figurations for each neural network model. *Method 15* leveraged a neural network architecture comprising two LSTM layers, each with sizes of 64 and 32 neurons. In *Method 16*, LSTM layers were substituted with more streamlined GRU layers, featuring sizes of 16 and 8. Given the high dynamics of cloud environments and the potential need for frequent model retraining, it is advisable to favor simpler models. Hence, an attempt to focus on input sequence lengths was made. *Method 17* involved the use of a shorter input sequence length – 24 samples. *Method 18* adopted a neural network architecture with two LSTM layers, sized 16 and 8, with the incorporation of Dropout layers following each LSTM layer to randomly deactivate 20% of neurons. *Method 19* employed an input sequence length spanning three days and introduced bidirectional LSTM layers with sizes of 8 and 4. *Method 20* featured an input sequence length equal to 72 and two bidirectional layers with GRU cells, sized 16 and 8, respectively. After each layer, a Dropout layer was applied to randomly deactivate 20% of neurons. *Method 21* employed a sequence length equal to the prediction horizon and Dropout layers that deactivate 30% of neurons. Furthermore, *Method 22* utilized bidirectional LSTM layers. On the other hand, *Method 23* employed GRU cells with the ELU (*Exponential Linear Unit*) activation function. *Method 24* investigated the merit of introducing additional lagged features from the past 168 samples relative to each sample in the input sequence. In the face of the risk of employing too many irrelevant predictors, a mechanism of dynamic feature selection was implemented in *Methods 17, 19, 21* and *24*. After applying this mechanism, the number of features was reduced to 37.

A common feature shared among all evaluated neural networks architecture was the dense output layer with a size of 168 neurons, enabling the implementation of a direct prediction approach for the entire time horizon. In the *Data Analysis Module*, data scaling was applied to each feature separately, ensuring a mean value of zero and a standard deviation of one. Furthermore, for each prediction method, the ADAM (*Adaptive Moment Estimation*) optimizer was employed [33]. It is worth noting that the use of neural models such as LSTM and GRU for supervised learning required the preparation of data sequences in an appropriate three-dimensional format: *number of sequences, input sequence length, number of features*, which responsibility was assigned to the *Data Analysis Module*. In particular, sequence lengths were tailored to the specific test scenarios. Generally, shorter time sequences allow the model to capture only short-term dependencies among the data, whereas longer sequences enable the consideration of correlations and patterns over extended time periods. Overly short sequences may lead to information loss and an underestimation of the problem's complexity.

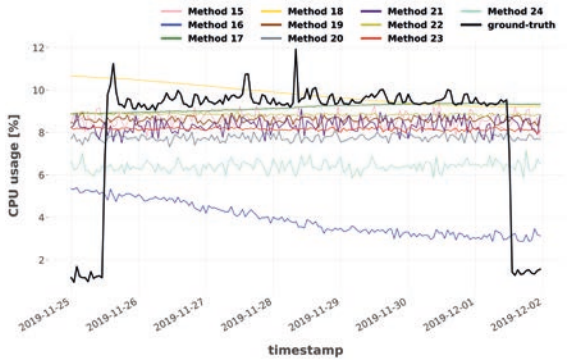

Figure 5. Results of long-term CPU utilization predictions obtained using neural network-based methods.

Compared to XGBoost-based methods, the proper selection of neural network parameters posed a significantly greater challenge here. This was because the learning process involved forecasting for the entire time horizon rather than predicting a single value one step ahead. In Figure 5, which presents prediction results and the actual CPU usage characteristics, it can be observed that almost all the methods applied consistently yield results close to the actual CPU usage. An exception is *Method 16*, for which the predicted characteristic remains below the actual usage. Similar to the analysis of results for prediction methods using the XGBoost model, it is challenging to evaluate prediction methods when the actual CPU usage was relatively stable throughout the entire duration, hence the focus on a selected period of the test set in which visualizations are consistently presented. Analyzing the error metric values obtained on the entire test dataset, as presented in Table 4, the results favor *Method 21*.

Table 4. Error metric values for prediction results on the test dataset obtained using neural network-based methods.

	RMSE	NRMSE	MAE	MAPE	MdAE
Method 15	4.262	0.290	2.883	1.613	1.285
Method 16	4.905	0.334	2.346	1.166	0.889
Method 17	3.385	0.230	2.289	1.190	1.208
Method 18	5.957	0.405	3.644	1.835	1.223
Method 19	6.133	0.417	4.014	1.935	1.689
Method 20	5.390	0.367	3.411	1.926	1.425
Method 21	1.79	0.122	1.404	0.663	1.207
Method 22	2.682	0.182	1.694	0.796	0.651
Method 23	3.234	0.220	2.686	1.405	2.627
Method 24	2.081	0.142	1.552	0.708	1.078

During hyperparameter tuning, it was observed that a larger batch size was preferred. In particular, changing this parameter from 32 to 64 in the case of *Method 15* led to significantly better results while maintaining the same architecture. With a small batch size, the gradients computed based on a small number of samples in the backpropagation process resulted in greater variability in their estimation and a more unstable learning process. In the context of *Method 17*, in order to maintain its flexibility under conditions of frequent or very frequent adaptation to new data, it was more advantageous to use models with a simpler architecture. The mechanism of dynamic feature selection also proved to be valuable. When defining test scenarios for methods based on neural networks, a decision was made to forgo the use of the *Anomaly Detection Module*. Nevertheless, this module remains a crucial element of the system, and its use is strongly dependent on the data set. Simultaneously, the Huber loss function was employed to reduce the impact of potential outliers on the learning process in the absence of their detection and handling through dedicated module.

The research conducted revealed that when aiming to forecast over a longer time horizon, longer input data sequences should be employed. The most effective prediction method turned out to be *Method 19*, utilizing the BiGRU (*Bidirectional GRU*) architecture, which confirmed the hypothesis that GRU cells can serve as a viable alternative to LSTMs, and the bidirectional nature of layers contributes to more accurate modeling of data sequences. Dropout layers also play an undeniable role, as without them neural models would be highly susceptible to overfitting. Although recurrent networks are designed to model temporal dependencies in data, the addition of lagged features marginally improved prediction results. In summary, one of the key elements that enable the *Cloud Computing Resource Prediction and Optimization System* to perform better was the utilization of the dynamic feature selection mechanism. It is worth noting that in the case of many proposed prediction methods, particularly *Method 21*, the predicted resource utilization remains above the actual value. This is a favorable phenomenon since this slight overestimation is far less problematic from the point of view of cloud resource users than underestimation. As concerns the RMSE error metric, the predicted CPU utilization on the test dataset was approximately 17.5% better than that achieved using the best baseline method. Therefore, the use of the dynamic feature selection mechanism alongside the selected neural network architectures and hyperparameters significantly contributed to improving the system's overall prediction accuracy and effectiveness in optimizing cloud resource utilization. Additionally, it is important to highlight that the flexibility of the system in adapting to varying data characteristics, including trends and seasonal patterns, makes it a valuable tool for resource management in dynamic cloud computing environments. This adaptability is particularly crucial in addressing the challenges posed by the increasing scale and complexity of cloud-based applications and services.

The third long-term prediction model evaluated in the *Training Module* and *Prediction Module* was Temporal Fusion Transformer. During the model training process using k-fold cross-validation for time series, 45 training epochs were employed alongside an early stopping mechanism determined by the patience parameter, set to 5. In subsequent iterations of the system, the *Training Module* fine-tuned the model on a progressively growing historical data set. This process occurred over a smaller number of epochs, specifically 5, with a correspondingly reduced patience parameter value of 2, aimed at minimizing the computational complexity of the process. Taking into account the insights gained from the tests of previous models, we decided against removing or replacing the outlier samples identified using the *Anomaly Detection Module*. Attempts to exclude these samples led to increased errors on validation sets. The feature selection mechanism, dynamically reducing the number of features from 90 to 37, played a significant role

in this model. Through this reduction, we were able to decrease the model's parameter count from 15.9 million to 7 million. Therefore, it can be observed that, particularly in the case of long-term prediction models characterized by complex architectures with numerous parameters, the meticulous selection of predictors in the data set is vital. The dynamic feature selection indirectly contributes to reducing potential costs associated with maintaining models in production environments.

The TFT model was utilized to define *Method 25*, incorporating four attention heads and a hidden layer consisting of 160 neurons for processing both continuous (hidden continuous size) and discrete (hidden size) variables. Dropout layers were employed to ensure regularization by randomly deactivating 10% of neurons. The output sequence size was specified as two-dimensional with dimensions *7, 168*. This configuration, coupled with the Quantile Loss cost function and a learning criterion minimizing prediction error for each quantile individually, facilitated the determination of confidence intervals for the obtained values. Additionally, a learning rate annealing strategy was implemented during training, initially reducing the learning rate from its original value of 0.001. Subsequently, if no improvement in validation set error was observed for four epochs, the learning rate was decreased tenfold. For more intricate models, the dynamic application of a learning rate scheduling strategy proved pivotal. Initially, a relatively high learning rate was advantageous in accelerating the learning process during the TFT model's early training stages when it had not yet fully adapted to the data. However, as training progressed, the learning rate was systematically diminished to mitigate overfitting and increase model stability, especially given the TFT model's prolonged training duration compared to the neural network.

Figure 6 presents the prediction results obtained for *Method 25* in comparison to the actual CPU usage. The characteristic proved to be highly accurate, and the extreme quantiles shown made it possible to visualize the confidence interval along with the 0.5 quantile (the median of the distribution). The error metric values obtained for *Method 25* based on the TFT model are presented in Table 5. These values turned out to be smaller than those of the system utilizing both the XGBoost model and neural networks, especially the BiGRU architecture, which yielded the best results among the previously evaluated methods. For *Method 25*, the 0.05 quantile was chosen as the representative reference point. However, it is worth noting that accurate long-term forecasting remains a significant challenge, even in the case of transformer architecture. Sudden drops and spikes causing long-term changes in CPU usage prove to be particularly challenging to predict. Nevertheless, it is possible to observe the significant advantage of the attention mechanism over recurrent networks, which emerged in a considerably simplified parameterization procedure. However, the high computational requirements of the TFT model necessitate a complex training procedure and may increase the costs related to model maintenance.

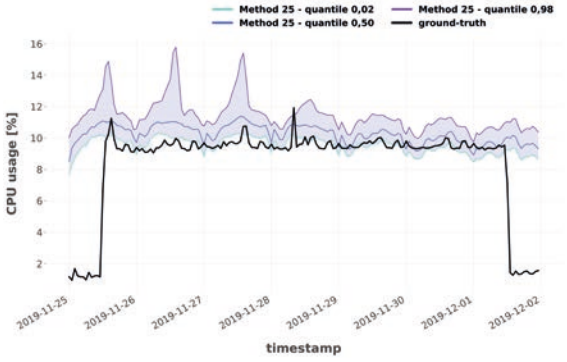

Figure 6. Results of long-term CPU utilization predictions obtained using the TFT-based method.

Table 5. Error metric values for prediction results on the test dataset obtained using the TFT-based method.

Method 25	RMSE	NRMSE	MAE	MAPE	MdAE
0.02 quantile	1.403	0.095	0.505	0.234	0.268
0.10 quantile	1.445	0.098	0.543	0.246	0.292
0.25 quantile	1.464	0.100	0.559	0.251	0.301
0.50 quantile	1.487	0.101	0.582	0.256	0.313
0.75 quantile	1.513	0.103	0.607	0.263	0.321
0.90 quantile	1.552	0.106	0.643	0.272	0.339
0.98 quantile	1.714	0.117	0.773	0.305	0.412

In the context of long-term time series forecasting, transformer-based models are capable of processing time series of varying lengths, making them highly flexible. Unlike prediction methods based on neural networks, they do not require a fixed length of the input sequence data, which was observed to affect the results significantly. Simultaneously, it is the attention mechanism that enables the modeling

of long-term patterns in sequences. However, the multitude of parameters in the transformer architecture may suggest a tendency to overfit, especially on smaller training datasets. Therefore, it is essential to consider mechanisms such as Dropout, early stopping, dynamic feature selection, and adaptive learning rate schedulers. Ultimately, *Method 25* (quantile 0.50) achieved a 31.4% better result than the best-performing baseline method.

5 Conclusion

The optimization of cloud resource usage is a multifaceted challenge that can be categorized into various domains. These domains range from low-level solutions that can be implemented by data center administrators to user-centric systems. In the past, most optimization systems relied on reactive mechanisms or on rigidly defined rules, which sometimes did not ensure satisfactory results in highly dynamic cloud environments. Consequently, more interest is now accorded to dynamic, proactive frameworks, such as load balancing algorithms, dynamic allocation of resources, or dynamic assignment of tasks to active virtual machines in a cluster. One of the most promising directions of research involves resource usage forecasting. In this scope, short-term forecasting algorithms are currently more prevalent; however, these do not have the same potential as long-term predictions, which can accurately guide decision-making processes.

Our proposed concept of a system for predicting and optimizing cloud resource usage fits in with the proactive and dynamic optimization paradigm. The approach applied enabled long-term forecasting (one week in advance) of CPU load. Notably, most time series prediction models are natively adapted to short-term forecasting, which is easier to achieve in practice.

In the research presented, we devote particular attention to the role of exploratory data analysis, which, in the case of long-term modeling, plays a key role as it not only enables better understanding of the time series, but also permits effective configuration of prediction method. Other important aspects of the research presented include analysis of the stationarity of the series, the potential for the use of statistical testing, or the applicability of seasonal decomposition which may reveal periodic patterns in input data. An important element of the developed concept is the use of feature engineering and feature enrichment, which may significantly improve the accuracy of predictions, although it also represents a certain risk. Improper application of feature engineering may cause unwanted flow of information between the training and test datasets. Consequently, a dynamic feature selection mechanism has been proposed, where only the most important features are selected, while others are aggregated to reduce the dimensionality of representation. This process enables time series prediction models to acknowledge patterns only in the scope of relevant predictors. The presented research on cloud resource usage optimization indicates that any proposed solution should, first and foremost, rely on the proper preparation of the input time series along with the identification of outliers.

Prediction results may be used to develop dynamic resource reservation plans, thus contributing to the optimization of resource usage. In this context, particularly promising results, as indicated by the experiments reported, have been achieved using methods based on the BiGRU network and the TFT. The former exhibited a distinct advantage in that its forecasts overestimated actual CPU usage, even though the TFT model showed lower error metric values. With regard to RMSE error metrics, the predicted CPU usage characteristics calculated for the test dataset using the BiGRU network represented an improvement by approximately 17.5% compared to the best reference method. The TFT model outperformed the best baseline method by 31.4%; however, in its case the predictions were not consistently higher than actual usage, which may be regarded as a risk in practical applications in dynamic resource reservation planning.

One of the key causes of cloud resource overprovisioning is the users' lack of awareness of actual resource usage by their applications. On the one hand, a flexible pay-as-you-go model provides users with a lot of freedom, but the longer such solutions reside in the cloud, the more costs they generate, especially if they are not managed properly. In order to avoid potential failures during peak load times, clients often overestimate the amount of resources they actually require. Resource usage prediction alleviates this issue and enables more optimal reservation of cloud resources. It is, how-

ever, worth remembering that modeling complex processes which exhibit stochastic-like behavior necessitates quality control. In the course of further development, it might therefore be useful to consider extending the features of the *Monitoring module*, adding the capability for induced retraining or recalculating predictions if the current forecast is found to significantly diverge from actual use within its assigned time frame. Further work may also involve other types of resources such as memory usage, disk I/O or network traffic. Simultaneous forecasting of CPU and memory usage could enable the development of an algorithm capable of generating dynamic plans for the reservation of resources (virtual machines) along with a cost estimation metric. Another interesting area of research is the use of ensemble learning methods to dynamically select models depending on the conditions observed.

Acknowledgements

The research presented in this paper was supported by funds from the Polish Ministry of Science and Higher Education allocated to the AGH University of Krakow.

References

[1] Herman, M., Iorga, M., Salim, A., Jackson, R., Hurst, M., Leo, R., Mishra, A., Landreville, N. and Wang, Y. NIST Cloud Computing Forensic Reference Architecture. (National Institute of Standards,2023)

[2] Osypanka, P. and Nawrocki, P. Resource Usage Cost Optimization in Cloud Computing Using Machine Learning. IEEE Transactions On Cloud Computing. pp. 1-1 (2020)

[3] Mason, K., Duggan, M., Barrett, E., Duggan, J. and Howley, E. Predicting host CPU utilization in the cloud using evolutionary neural networks. Future Generation Computer Systems. 86 pp. 162-173 (2018)

[4] Chen, J. and Wang, Y. A Resource Demand Prediction Method Based on EEMD in Cloud Computing. Procedia Computer Science. 131 pp. 116-123 (2018), Recent Advancement in Information and Communication Technology:

[5] Chen, H., Fu, X., Tang, Z. and Zhu, X. Resource Monitoring and Prediction in Cloud Computing Environments. 2015 3rd International Conference On Applied Computing And Information Technology/2nd International Conference On Computational Science And Intelligence. pp. 288-292 (2015).

[6] Nguyen, T., Tran, N., Nguyen, B. and Nguyen, G. A Resource Usage Prediction System Using Functional-Link and Genetic Algorithm Neural Network for Multivariate Cloud Metrics. 2018 IEEE 11th Conference On Service-Oriented Computing And Applications (SOCA). pp. 49-56 (2018).

[7] Nawrocki, P., Grzywacz, M. and Sniezynski, B. Adaptive resource planning for cloud-based services using machine learning. Journal Of Parallel And Distributed Computing. 152 pp. 88-97 (2021).

[8] Kumar, J., Goomer, R. and Singh, A. Long Short Term Memory Recurrent Neural Network (LSTM-RNN) Based Workload Forecasting Model For Cloud Datacenters. Procedia Computer Science. 125 pp. 676-682 (2018), The 6th International Conference on Smart Computing and Communications.

[9] Osypanka, P. and Nawrocki, P. QoS-aware Cloud Resource Prediction for Computing Services. IEEE Transactions On Services Computing. pp. 1-1 (2022).

[10] Gupta, S. and Dinesh, D. Online adaptation models for resource usage prediction in cloud network. 2017 Twenty-third National Conference On Communications (NCC). pp. 1-6 (2017).

[11] Kumar, J. and Singh, A. Workload prediction in cloud using artificial neural network and adaptive differential evolution. Future Generation Computer Systems. 81 pp. 41-52 (2018).

[12] Sniezynski, B., Nawrocki, P., Wilk, M., Jarzab, M. and Zielinski, K. VM Reservation Plan Adaptation Using Machine Learning in Cloud Computing. Journal Of Grid Computing. 17 pp. 797-812 (2019).

[13] Gupta, S., Dileep, A. and Gonsalves, T. Online Sparse BLSTM Models for Resource Usage Prediction in Cloud Datacentres. IEEE Transactions On Network And Service Management. 17, 2335-2349 (2020).

[14] Fang, Z., Ma, X., Pan, H., Yang, G. and Arce, G. Movement forecasting of financial time series based on adaptive LSTM-BN network. Expert Systems With Applications. 213 pp. 119207 (2023).

[15] Gasparin, A., Lukovic, S. and Alippi, C. Deep learning for time series forecasting: The electric load case. CAAI Transactions On Intelligence Technology. 7, 1-25 (2022).

[16] Nawrocki, P. and Sus, W. Anomaly detection in the context of long-term cloud resource usage planning. Knowledge And Information Systems. 64, 2689-2711 (2022,10).

[17] A., A. Using Multiple Seasonal Holt-Winters Exponential Smoothing to Predict Cloud Resource Provisioning. International Journal Of Advanced Computer Science And Applications. 7 (2016).

[18] Dixit, A., Gupta, R., Dubey, A. and Misra, R. Machine Learning Based Adaptive Auto-scaling Policy for Resource Orchestration in Kubernetes Clusters. Internet Of Things And Connected Technologies. pp. 1-16 (2022).

[19] Salles, R., Belloze, K., Porto, F., Gonzalez, P. and Ogasawara, E. Nonstationary time series transformation methods: An experimental review. Knowledge-Based Systems. 164 pp. 274-291 (2019).

[20] Musbah, H., Aly, H. and Little, T. A proposed novel adaptive DC technique for non-stationary data removal. Heliyon. 9 (2023).

[21] Ben Taieb, S., Bontempi, G., Atiya, A. and Sorjamaa, A. A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. Expert Systems With Applications. 39, 7067-7083 (2012).

[22] González-Sopeña, J., Pakrashi, V. and Ghosh, B. An overview of performance evaluation metrics for short-term statistical wind power forecasting. Renewable And Sustainable Energy Reviews. 138 pp. 110515 (2021).

[23] Bergmeir, C., Hyndman, R. and Koo, B. A note on the validity of cross-validation for evaluating autoregressive time series prediction. Computational Statistics and Data Analysis. 120 pp. 70-83 (2018).

[24] Li, X., Cao, J., Guo, J., Liu, C., Wang, W., Jia, Z. and Su, T. Multi-step forecasting of ocean wave height using gate recurrent unit networks with multivariate time series. Ocean Engineering. 248 pp. 110689 (2022).

[25] Carletti, M., Terzi, M. and Susto, G. Interpretable Anomaly Detection with DIFFI: Depth-based feature importance of Isolation Forest. Engineering Applications Of Artificial Intelligence. 119 pp. 105730 (2023).

[26] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, Ł. and Polosukhin, I. Attention is All you Need. Advances In Neural Information Processing Systems. 30 (2017).

[27] Giannakas, F., Troussas, C., Krouska, A., Sgouropoulou, C. and Voyiatzis, I. XGBoost and Deep Neural Network Comparison: The Case of Teams' Performance. Intelligent Tutoring Systems. pp. 343-349 (2021).

[28] Arik, S., Yoder, N. and Pfister, T. Self-Adaptive Forecasting for Improved Deep Learning on Non-Stationary Time-Series. CoRR. abs/2202.02403 (2022).

[29] Zeng, A., Chen, M., Zhang, L. and Xu, Q. Are Transformers Effective for Time Series Forecasting? (arXiv,2022).

[30] Gárate-Escamila, A., Hajjam El Hassani, A. and Andrès, E. Classification models for heart disease prediction using feature selection and PCA. Informatics In Medicine Unlocked. 19 pp. 100330 (2020).

[31] Lim, B., Arık, S., Loeff, N. and Pfister, T. Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. International Journal Of Forecasting. 37, 1748-1764 (2021).

[32] Canela, M., Alegre, I. and Ibarra, A. Holt-Winters Forecasting. Quantitative Methods For Management: A Practical Approach. pp. 121-128 (2019).

[33] Singarimbun, R., Nababan, E. and Sitompul, O. Adaptive Moment Estimation To Minimize Square Error In Backpropagation Algorithm. 2019 International Conference Of Computer Science And Information Technology (ICoSNIKOM). pp. 1-7 (2019).

[34] Jayalath, J., Chathumali, E., Kothalawala, K. and Kuruwitaarachchi, N. Green Cloud Computing: A Review on Adoption of Green-Computing attributes and Vendor Specific Implementations. 2019 International Research Conference On Smart Computing And Systems Engineering (SCSE). pp. 158-164 (2019).

Piotr Nawrocki is Professor in the Faculty of Computer Science at the AGH University of Krakow, Poland. His research interests include distributed systems, computer networks, mobile systems, machine learning, cloud computing and service-oriented architectures. He has participated in several EU research projects including MECCANO, 6WINIT, UniversAAL and national projects including IT-SOA and ISMOP. He is a member of the Polish Information Processing Society (PTI).
https://orcid.org/0000-0003-4512-9337

Mateusz Smendowski M.Sc. is a Ph.D. student at the AGH University of Krakow. His interests encompass machine learning, software engineering, and cloud computing, with a specific focus on cloud resource usage optimization and time series forecasting.
https://orcid.org/0009-0004-0946-2233