National College of Ireland

# Enhancing Cloud Efficiency using Intelligent Autoscaling Algorithms

MSc Research Project
MSc in Cloud Computing


Pushkar Rahane
Student ID: X21177279

School of Computing
National College of Ireland


Supervisor: Dr. Aqeel Kazmi

**National College of Ireland**
**Project Submission Sheet**
**School of Computing**

National College of Ireland

| | |
|---|---|
| **Student Name:** | Pushkar Rahane |
| **Student ID:** | X21177279 |
| **Programme:** | MSc Cloud Computing |
| **Year:** | 2023 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Dr. Aqeel Kazmi |
| **Submission Due Date:** | 18/09/2023 |
| **Project Title:** | Enhancing Cloud Efficiency using Intelligent Autoscaling Algorithms |
| **Word Count:** | 6300 |
| **Page Count:** | 21 |

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

**ALL** internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

| | |
|---|---|
| **Signature:** | Pushkar Rahane |
| **Date:** | 18th September 2023 |

## PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies). | ☐ |
| **Attach a Moodle submission receipt of the online project submission**, to each project (including multiple copies). | ☐ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | ☐ |

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Enhancing Cloud Efficiency using Intelligent Autoscaling Algorithms

Pushkar Rahane
X21177279

**Abstract**

Cloud computing has revolutionized the way data and applications are managed, offering scalable, flexible, and cost-effective solutions. A critical aspect of cloud computing is autoscaling, the ability to dynamically adjust computing resources to match demand. In this research, we explored the challenges of autoscaling by implementing and evaluating three distinct algorithms: Decision Tree, Random Forest, and LSTM, within a simulated cloud environment. The study utilized metrics such as Total Response Time, Total Delayed Load, Prediction Error of Models, and Total Wastage of Resources to evaluate the algorithms' effectiveness. The results revealed that LSTM achieved the best performance in response time and prediction accuracy, while Random Forest excelled in resource utilization. Our research contributes valuable insights into the dynamics of autoscaling, paving the way for more efficient and responsive cloud systems. The findings also highlight potential avenues for future research and optimization in autoscaling techniques.

## 1 Introduction

Cloud computing is identified as a "service-driven computing model" which provides end-users with computing resources (or virtual resources) by considering cloud service providers as the medium under the "service-level agreement" (SLA) to prevent violation of services. The importance of cloud service is well-recognized and demonstrated widely by researchers and taken into configuration to study precisely how the tasks are performed in terms of computing services. As explained by Fé et al. (2022) cloud services hosted by CSPs can be identified in different forms, which highlight networking, storage, and computational components. Generally, the cloud environment is multilayered, although the composition differs based on the infrastructure of the CSP, applications' usability, and specific models used for analysis. Cloud computing typically provides on-demand accessibility to shared computing or virtual resources that serve a range of operations in users' devices. These resources, as well as mechanisms often observed to be released or acquired with minimum effort of management. From a wide understanding, these features are identified to enable administrators to manifest knowledge and focus only on the business model rather than worrying about the details of infrastructures. Oftentimes, these resources and cloud service experiences are compared to the exact consumption of various public utilities. Fé et al. (2022) marked that the two features based on which cloud computing services are measured are the elasticity in resource acquisitions and cost-effectiveness.

Bestowing the understanding of the effectiveness of cloud computing services, experts have highlighted the cost reduction and enhanced performance of clouds subjective to variable loads. In this regard, the concept of auto-scaling serves as a process to manage computing resources, and trade-off between cloud performance and costs through an automatic adjustment of resource counts and variable demands of users. The information perceived herein amplifies ideas on how cloud computing resources are managed wisely and suggestively provide users with reliable services. It has often been observed that users' applications' demands for resources can be seasonal under the implication of variable workload demands in clouds. Over the years, different concepts, as well as technologies, have been developed and implemented to support cloud computing in terms of elasticity wherein automatic scaling capacity has gained tremendous attention.

Cloud computing services have been well-known for years and various integrated applications are provided to users. As already implored, deliverables of these services and resources are performed by the process of autoscaling to simulate the computing capacity and hence provide reliable services to users. However, various concepts and illustrations are constructed to acknowledge the context of computing issues and resource optimization. Virtualization is identified as the process that allows users to work on dedicated machines with a complete OS infrastructure. Herein, the application of computing services is recognized to be enhanced with the emulation of hardware resources in order to offer users virtual machine applications. When understanding virtual machine (VM) migration, it is mostly identified that cloud service providers typically offer general-purpose virtual machine classes that contain generic software as well as resource configurations. In this regard, it is identified that installation of virtual machines with proper optimization certainly requires peak demands although overprovisioning of resources might incur high-cost configurations even in low-demand periods. The autoscaling of cloud resources is important since cloud computing provides on-demand self-services. On the basis of understanding its applications, a cloud user enjoys service and resource elasticity by ensuring the ability to acquire or release resources as per the demand. Auto-scaling techniques have been divided into predictive and reactive processes wherein the former technique typically determines the requirement for future resources and accordingly provides the resource in advance. On the contrary, the reactive technique is a widely used process in commercial systems that follows rules in responding to systems changing configuration when reaching a pre-defined threshold. An illustration of Amazon's "Elastic Compute" (EC2) Cloud services enables the allocation of resources by providers, who can accordingly monitor the same or log prior to manually reacting towards pre-defined alerts. In Amazon, the cloud service provider enables users to apply autoscaling for the addition of new virtual machines.

The purpose of reactive auto-scaling techniques has already been explored, which is based on two distinct parameters: a threshold for instantiating new VMs (or scaling up) and a threshold to destroy VMs (or scaling down). In this regard, it is often identified that the proper selection of parameters during auto-scaling is a common problem due to which the accuracy in resource allocation, eliminating delay time, and error reduction phenomenon is disturbed. It is understandable that the process as well as the instantiation duration for virtual machines is not necessarily constant. Therefore, a potential influence of the identified random variables is necessary to be predicted upon envisioning effective

system adjustments. An appropriate acknowledgment of the issue is identified in VM configuration optimization as well as auto-scaling during vast possibilities. This issue is aligned with the time-consuming and complexities that, therefore, demand effective implementation of models that can necessarily enhance the autoscaling process in the cloud computing environment.

Over the years, different methods and techniques have been developed and experimentally explored to simulate an effective prediction of the autoscaling process while considering three important parameters - elimination of delays, proper resource allocation without wastage, and error analysis. Moreover, it is highly acknowledged that VM configuration optimization leads to SLA violations which is really a concerning matter. Therefore, to mitigate the issues, exports have drawn their focus on the integration of efficient models into cloud computing resource configuration and autoscaling approaches. In the current study, the focus has shed light on different improved models, especially machine learning and deep learning methods that have gained tremendous attention due to distinguished prediction performance and accuracy. An ideal approach to determine the prediction accuracy of the autoscaling process can be established with advanced deep neural architecture such as LSTM models, which has been discussed and contrasted with other algorithms and prediction techniques. Thus, it can be stated that the current study has predominantly served the purpose of understanding the importance of cloud computing resources and their optimization and accordingly introduced as well as discussed predictive models that can provide a comprehensive result with higher performance accuracy.

## 1.1 Research Objectives

In the evolving landscape of cloud computing, dynamic resource allocation is critical. This research delves into various autoscaling algorithms to determine their efficacy in a cloud environment. Outline the following research objectives.

- To critically analyze and compare the performance of three distinct autoscaling algorithms - Decision Tree, Random Forest, and LSTM - in a simulated cloud computing environment, assessing their effectiveness in managing dynamic resource allocation.
- To examine essential metrics such as Total Response Time, Total Delayed Load, Prediction Error of Models, and Total Wastage of Resources, aiming to identify the strengths and weaknesses of each algorithm in optimizing cloud efficiency.
- To derive meaningful insights from the comparative study and highlight potential avenues for future research, innovations, and enhancements in cloud autoscaling techniques, contributing to the broader field of cloud computing efficiency.

## 1.2 Research Question

*Question: How do the autoscaling algorithms - Decision Tree, Random Forest, and LSTM - perform in a cloud computing environment in terms of efficiency and effectiveness, and which among them offers the most optimal solution for dynamic resource allocation as evaluated by key metrics like Total Response Time, Total Delayed Load, Prediction Error of Models, and Total Wastage of Resources?*

# 2 Related Work

## 2.1 Chapter Overview

In the chapter, the study has highlighted contextual information from various studies that have demonstrated the significance of different prediction models in improving the autoscaling of cloud computing resources. The information is specific to particular models that are illustrated by authors to demonstrate how each model has performed and how their accuracy level has determined their importance in the autoscaling process. A comparative discussion is produced with distinct insights that have been put forward to acknowledge the reliability of the model under the convenience of three identified parameters in resource optimization - delay time, error analysis, and efficient resource allocation.

## 2.2 Autoscaling of cloud computing resources using Machine Learning Methods

Cloud computing in recent times along with its native applications has gained significant standards to enhance new developments in organizations. As explained by Marie-Magdelaine and Ahmed (2020) organizations nowadays are ensuring that their workloads are maintained through microservices and accordingly retain profit in cloud-native paradigms. While stating this fact, the author has further delineated challenges in optimizing the "Quality of Services" (QoS) during the autoscaling of resources da Silva et al. (2022). To address the challenge and enhance the resource optimization process, machine learning models are introduced which are identified as a proactive-scaling framework. One of the apparent recognition is the long short-term memory (LSTM) model that has been found to improve the "end-to-end latency" regarding cloud applications. The approach is well-developing in the concerned study although more insights are necessary to develop with further research implications. In another study conducted by Alipour, (2019), a novel framework of model-driven machine learning is developed which highlights the technique to learn multiple parameters during model training as well as forecasting of workloads. As per the understanding of the experimental outcome, it has been observed that workload forecasting has automatically triggered the process of autoscaling. Apart from this, the serverless function observed in the developed ML-based framework has demonstrated an architectural implementation that serves reliable and highly accurate prediction results. Thus, it can be stated that the information present in the focus study on the implemented model is a cost-effective solution and minimizes the delay time compared to previous threshold-based autoscaling methods.

In the modern cloud computing system, it has been demonstrated in the literature that the approach has served a greater aptitude to provide accessibility, cost-effective, and scalable computing processes to users. However, an apparent recognition of challenges in effective resource optimization is observed; thereby studies have retained information on various predictive models that can improve the autoscaling process. Lemke & Majumdar, (2023) provided a survey-based interpretation of machine learning models and put forward a contrasting view between MLScale, which is a supervised-learning algorithm based on neural architectures, and RLPAS, which enforces a SARSA reinforcement solution. Based on the suggestive interpretation of the experimental outcome, it has been

observed the RLPAS has served a better predictive ability in spiking and provisioning proactive resources compared to the MLScale reactive model. This information has retained a distinct positivity in understanding the effectiveness of machine learning models in resource optimization and scalability. Over the years, many studies have evaluated the significance of machine learning models in minimizing the obstacles to performing effective resource autoscaling. Apart from previous approaches delineated, Qassem et al. (2023) introduced a random forest (RF) model to forecast workload and future resource utilization. The predicted values have been adjusted based on the pool of resources horizontally and vertically. According to the experimental result obtained with the increase in allocated resources, the accuracy level is estimated to be 90% and the end-to-end latency is 95% respectively. Depending on the perceived information, it can be explained that machine learning algorithms explored in the autoscaling process provide improved accuracy over the others however, it is recommended to choose a wide range of parameters to increase the model reliability under different conditions of autoscaling.

## 2.3 Autoscaling of cloud computing resources using Deep Learning Methods

With the changing landscape of network systems in recent times, a flexible yet scalable approach for real-time data processing has been identified to be increasingly important. Therefore, the prediction of workloads with the integration of future resources is enhanced with the help of novel frameworks. As explained by Dang-Quang and Yoo (2022) cloud service providers (CSPs) based on this approach can easily provision or de-provision users' resources beforehand in order to ensure non-violation of service level agreements (SLA). Previously, it is identified that time-series forecasting methods are only used in the prediction of workload. As has been already explored with machine learning models, most of the models only depend on a single feature (univariate) or measure limited features. Dang-Quang and Yoo (2022) introduced an improved model integrating bidirectional LSTM (Bi-LSTM) that exhibited a "root mean square error" (RMSE) to predict error in the autoscaling process. According to the observation, it has been observed that the error is 1.84 times lower than in univariate models. Moreover, in the case of resource allocation and reduced wastage, the Bi-LSTM autoscaler is 47.2% & 14.7% efficient compared to other LSTM and multivariate models.

In recent years, with the increased resource utilization, cloud service providers are facing immense challenges in the autoscaling process in terms of different parametric values. Radhika and Sadasivam (2021) explained that to reduce the dynamism in handling a wide range of requests from users regarding resource allocation improved autoscaling techniques are developed wherein deep learning algorithms and neural architectures have gained good attention. It is already acknowledged beforehand, that resource optimization and scalability are important characteristics in cloud computing. In recent times, with the setting of current cloud infrastructures, cloud service providers deliver "threshold-based autoscaling techniques" for users' convenience. However, given the fact of effective resource utilization and cost minimization, improved methods are needed to autoscale the right values considering the variance and workload dynamism present. A. (2016) in this regard has delineated the significance of the LSTM-RNN model that has shown its efficiency in autoscaling of virtual resources on the basis of predicted values. Further,

based on the experimental values obtained from the study, it can be stated that the model has outperformed existing algorithms. The effective prediction of autoscaling of workload with proper forecasting is a challenging task with the increased dynamism observed in users' demands. Reinforcement learning is introduced as a promising solution in the resource management process that can guide scaling actions in a dynamic and unstable cloud environment. However, Xue et al. (2022) implied that reinforcement learning (RL) faced certain challenges in steering the predictive autoscaling and thus lacking accuracy in performance. In the concerned study, therefore, a meta-reinforcement learning (RL) model is introduced which embeds a neural architecture and enhances the scaling process with higher accuracy. Contrasting to this study, Cheng et al. (2018) have developed a perspective on resource provision and workload management without wastage by presenting a deep reinforcement learning (DRL) model, which is considered to be a novel approach. Based on the observation of the outcome, this model serves a better prediction accuracy and energy efficiency compared to existing state-of-the-art techniques. It has achieved nearly 320% of energy efficiency and cost-effectiveness while managing the rate of rejection on average. Moreover, it has achieved approximately 144% of reduction in runtime. Based on the overall interpretation of DL-based methods and neural architectures, it can be stated that both LSTM and reinforcement learning have served a greater advantage in regulating cloud resources upon measuring multiple features. Thus, it is necessary to contribute further research to reflect on these models.

## 2.4 Autoscaling of cloud computing resources using Hybrid Methods

Over the past decade, there has been a wide escalation in computational services witnessed with the progression of modern technologies. In this regard, the cloud computing environment has gained enormous attention that has eased access to virtual resources for users, platforms as well as software-based services. However, it is already discussed that while providing exclusive services to users, several challenges are embarked upon with the approach, especially with resource management, reducing error and delay. Previously, different machine learning and deep learning models have been explored to identify an improved autoscaling process Ahamed et al. (2023). However, it is better to exclaim that more information and enlightenment are needed to better acknowledge the significance of these models. Apart from the consideration of these models, some studies have empirically elucidated the importance of hybrid autoscaling techniques that precisely improved the advent of resource management. In a study conducted by J.V. and Dharma (2018) the author introduced a hybrid auto-scaler that has been proposed for adjusting resources automatically in order to serve users' demands without delay, error, and wastage of unnecessary resources. The model essentially predicts the future behavior of users' systems through time-series analysis while deploying anticipated resources. This deployment is based on computing only the required resource capacity through a "queuing model". As per the understanding of the model validation, it can be explained that the model has achieved significant improvement with various benchmarked workloads in terms of CPU utilization & response time.

In another study conducted by Do and Tan (2022) a similar approach to introduce a hybrid auto-scaler is explained which uses container-based cloud platforms to manage virtual resources. The hybrid solution presented in the study typically combines the

benefit of autoscaling and determines a "bandwidth-efficient scheduler strategy" which strategically manages cost and end-to-end latency compared to VPA approaches. Over the years, hybrid autoscaling has gained wide attention due to the elasticity and scalability options it provides in resource management. Biswas et al. (2017) elucidated a hybrid autoscaling approach that has typically combined reaction & proactive techniques to enhance resource scalability depending on users' demands. The approach is highly significant to improve resource scaling and optimization by reducing error and delay time while increasing elasticity. Thus it can be suggested that the hybrid auto-scaling models are improved methods that have been designed and achieved improved accuracy with better measurement of parameters. Autoscaling techniques for cloud resources have gained good attention from cloud service providers and users due to their improved resource configuration and reduction of runtime with guaranteed SLAs and cost-effectiveness. In a study conducted by Zhong et al. (2019) a hybrid autoscaling model is introduced which is based on LSTM and reinforcement learning (RL). The model proposed herein has obtained an optimal action for the scalability of virtual machines. The proposed model is further validated and deciding to the result obtained from the study, it shows that the model is highly significant in reducing workload and managing work steadily. Moreover, it reduces SLA violations by 10-30% compared to other models.

## 2.5 Autoscaling of cloud computing resources using other approaches

There has been a marked increase in computing services, especially with fog and cloud computing. The extension of cloud networks, processing management of data, and network nodes storage has been intended to enhance to bridge the identified gap between cloud & IoT devices. With the advent of understanding the necessity of virtualization and cloud resources management, different methods have been devised to integrate an improved auto-scaling solution Dogani et al. (2023). In this regard, Fe et al. (2017) have introduced a stochastic model that can assist in cloud resources planning. The model has served improved benefits in autoscaling configurations and measures different parameters and reduces delay time. A similar approach is further explained by Fé et al. (2022) by introducing a Stochastic Petri Net (SPN) model upon employing an adaptive "meta-heuristic" search for the discovery of trade-offs that exist between cloud performance and the cost structure. The model serves a better-autoscaled performance with an estimation of 95% confidence upon testing on the "real test-bed scenarios" containing 18,000 samples. Based on the information on different approaches over the years, it can be stated that research has extensively viewed the necessity to enhance flexibility in managing cloud computing resources ZargarAzad and Ashtiani (2023). The overall assessment of the information established from the literature has served as a basis for understanding the importance of the autoscaling of cloud computing resources. Based on the simulated research presented and information drawn from the study, it can be explained that different models and methods are integrated into the autoscaling process and mitigate the challenges although it is essential to explore the propensity of some models such as LSTM and its hybrid configurations to properly acknowledge its significance as a multivariate model ineffective autoscaling of cloud computing resources.

# 3 Methodology

Autoscaling is a critical feature in cloud computing environments due to the dynamic and often unpredictable nature of workloads. An optimal autoscaling algorithm can optimally utilize the resources, reduce the operational cost, and enhance scalability and performance. With our proposed methodology in a simulated environment, we will identify the most optimal autoscaling algorithms that can resolve all the challenges associated with traditional autoscaling methods and enhance the productivity of the system. The flow diagram of methodology is shown in Figure 1.

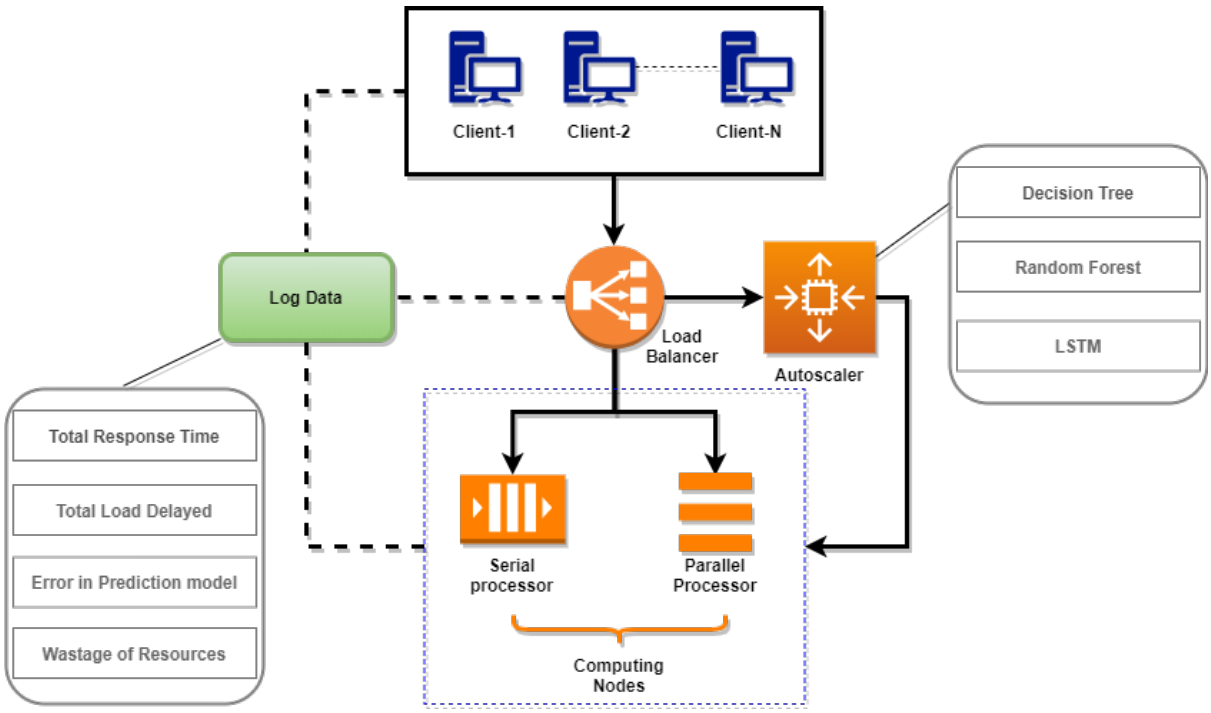


Figure 1: Simulated Cloud Computing Environment for Load Balancing

## 3.1 Clients

The client produces various load types. These are presumably distinct classes that determine the characteristics or requirements of the tasks to be processed. Along with the type, each load has a specific magnitude or size. This indicates how much computing resources or time might be required to process this load. The client sends these loads as requests to the load balancer. All the communication between components is happening using HTTP requests with Flask micro-service API.

## 3.2 Load Balancer

The load balancer in our architecture acts as an intermediary between the client and the computing nodes. It is responsible for distributing the load to the computing nodes. The load balancer uses a simple rule-based approach to distribute the load. If the size of the load is more than 10, then it will be transmitted to the parallel processor. Otherwise, it will be transmitted to the serial processor. To communicate and transmit loads, the

load balancer utilizes HTTP request APIs, made possible through the Flask micro-service API. The load balancer, in the architecture, maintains its own log information.

## 3.3 Computing Nodes

The computing nodes represent the fundamental processing units that handle the various loads sent by the client and distributed by the load balancer. There are two primary types of computing nodes.

### 3.3.1 Serial Processor

The serial processor is designed to handle one load at a time. This means it processes tasks sequentially, completing one before moving on to the next. A serial processor is a node that has its own computing capability, which is managed by Autoscaler. For handling tasks with a small amount of load Serial processor is an optimal choice.

### 3.3.2 Parallel Processor

Unlike the serial processor, the parallel processor can handle multiple loads simultaneously. This makes it inherently faster for specific tasks, especially when there's a need to handle multiple loads concurrently. It's designed to handle loads that are larger or more complex, making it especially valuable when the load size exceeds a certain threshold (in our case, more than 10). The autoscaler also has authority over the parallel processor, adjusting its computing capability based on the expected future loads. Both computing nodes, serial and parallel processors, communicate using the HTTP requests API via the Flask micro-service API. They also maintain their own log information,

## 3.4 Autoscaler

The autoscaler in our architecture plays a pivotal role in optimizing resource utilization based on the incoming load and predicted future load. It dynamically adjusts the computing capability of both the serial and parallel processors in real-time. Adjustment of Computing capability in computing nodes is dependent on the prediction calculated by the algorithm. This predictive capability allows the autoscaler to proactively allocate resources in anticipation of upcoming demands. By doing so, it ensures that the system can handle varying loads efficiently without unnecessary resource consumption or wastage. We have implemented 3 different algorithms in autoscaler, these algorithms are Decision Tree, Random Forest and LSTM, all these algorithms are used for performing the classification. We will discuss about each of autoscaling algorithms and their architecture in Chapter-4. Like other components in our architecture, the autoscaler communicates using the HTTP requests API facilitated by the Flask micro-service API. The autoscaler maintains its own log information, crucial for performance analysis and for refining its predictive and scaling strategies over time.

## 3.5 Log Information

The log information in our architecture serves as a detailed chronicle of operations, decisions, and outcomes. Each component's logs capture its activities, and collectively, these logs provide a comprehensive picture of system performance, scalability decisions,

and algorithm effectiveness. This data is foundational for monitoring, troubleshooting, and refining our system. Based on the log information we will calculate the performance metrics of autoscaling algorithms for evaluation. The metrics that will be used for this research are as follows Total Response time, Total Delayed Load, Prediction error of models, and Total wastage of resources.

# 4 Design Specification

This chapter discusses about the architectural part of all the algorithms used for autoscaling in this research. Understanding the architecture of each algorithm is crucial not only for the implementation and integration into the cloud computing environment but also for gauging the efficiency, accuracy, and overall effectiveness of the autoscaling process. The detailed explanation for each autoscaling algorithm is as follows.

## 4.1 Decision Tree Algorithm

Decision tree algorithms are a type of supervised learning algorithm that can be used to classify or predict outcomes. They work by creating a tree-like structure of decisions, where each decision is based on a single feature of the data. The decision tree is trained on a set of labeled data, and it learns to make predictions by finding the best path through the tree for each new data point. In our architecture, the Decision Tree is trained on the historical data about loads to predict the type or amount of load that will be encountered next based on patterns in the provided features. Once the predicted load type or magnitude is known, decisions can be made to either scale up (allocate more resources) or scale down (deallocate resources) to ensure optimal performance and resource utilization. The Decision Tree can quickly evaluate the conditions of the provided 5 load types and traverse its tree structure to make a prediction for the 6th load. The architecture diagram of a decision tree is shown in Figure 2.

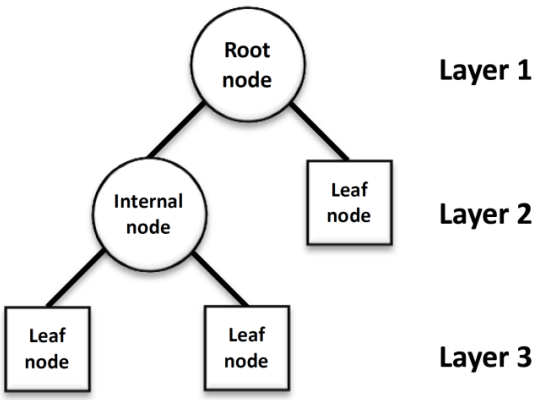


Figure 2: Architecture of Decision Tree algorithm Chiu et al. (2016)

## 4.2 Random Forest Classifier

A random forest algorithm is a type of ensemble learning algorithm that is made up of a collection of decision trees. Each decision tree in the random forest is trained on a

different subset of the training data, and the predictions of the individual decision trees are then combined to make a final prediction. Similar to the Decision Tree, the Random Forest uses historical load data to predict future load types or magnitudes. However, due to its ensemble nature, it may capture more complex patterns or anomalies than a single tree. Once the type or magnitude of the predicted load is known, the system can make scaling decisions. The algorithm evaluates the conditions of the previous loads (e.g., the last 5 load types) using its ensemble of trees and then aggregates their predictions for the upcoming (6th) load. based on this prediction, scaling decisions are made, be it scaling up or down. The ensemble nature of the Random Forest ensures a more diversified and, often, more accurate prediction than single models. The architecture diagram of the Random forest classifier is shown in Figure 3.

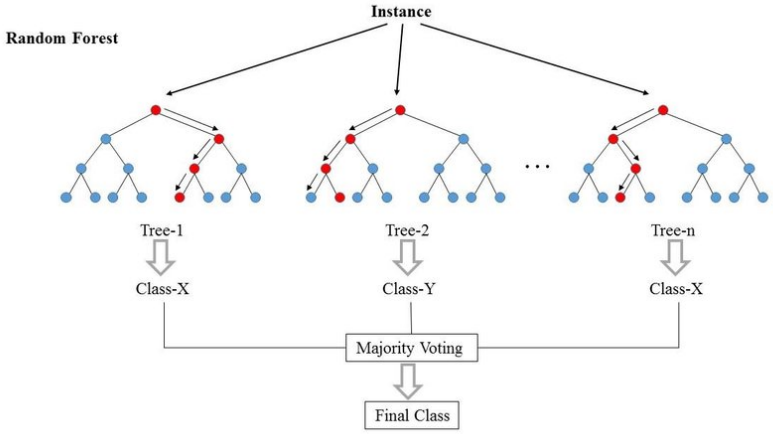


Figure 3: Architecture diagram of Random Forest Classifier Dimitriadis and Liparas (2018)

## 4.3 Long Short Term Memory (LSTM)

Long short-term memory (LSTM) is a type of recurrent neural network (RNN) that is commonly used for time series prediction. LSTM networks are able to learn long-term dependencies in the data, they have a unique structure called cells which contain three gates: input, forget, and output gates. These gates determine how information flows, is stored, or is discarded in the cell. LSTM takes the sequence of previous loads as input and provides a prediction for the upcoming load, capturing temporal dependencies that might be missed by other algorithms. Once the predicted load type or magnitude is obtained from the LSTM, scaling decisions can be made. Based on the scaling decision, the autoscaler allocates or deallocates the resources to the processing nodes. The architecture of LSTM is shown in Figure 4.

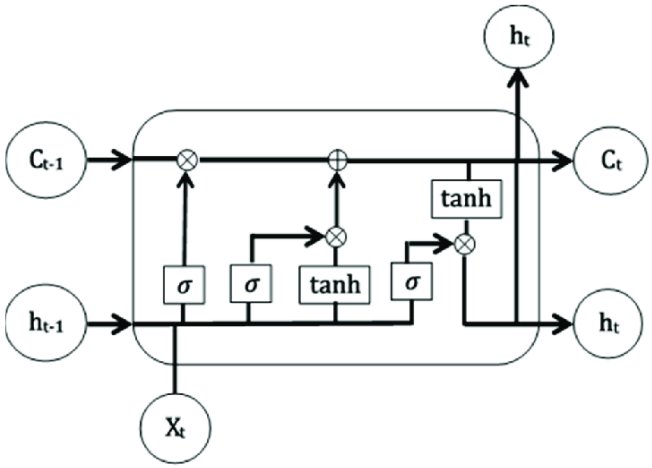


Figure 4: Architecture diagram of LSTM Algorithm Hasan et al. (2019)

# 5 Implementation

In order to achieve scalability in the cloud computing environment, an optimal autoscaling mechanism is highly required. There are an enormous number of autoscaling algorithms available in the market, so selecting the most appropriate autoscaling is a vital task. In this research, we have developed a simulated cloud computing environment that is inspired by the real-world cloud infrastructure and thus consists of all the necessary components, the best part about our simulated environment is that the performance of any machine learning or deep learning algorithms can be evaluated at our platform in order to identify the most suitable autoscaling algorithm as per the requirement. In this research, we have deployed 3 different algorithms for autoscaling which include the Decision Tree, Random Forest, and Decision Tree. In order to develop the Simulated Cloud environment python is used as a programming language where the different Python libraries such as flask, requests, JSON, numpy, pickle, seaborn, matplotlib, and many more are utilized to achieve our objective. Scikit-learn, Tensorflow, and Keras libraries are used to train machine learning and deep learning model, save them, and make the prediction from them. Matplotlib and seaborn libraries are used to visualize the metrics calculation. To communicate between the cloud components we use the HTTP requests API from Python and in order to simulate the system components we use Flask micro-service API. As the entire environment is developed using Python, it can be deployed or run on any operating system. We have deployed the entire environment to Cloud 9. Deploying the environment to Cloud 9 allows us to emulate a real-world cloud computing scenario, providing authenticity to our simulated environment. In order to run minimum system configuration requirement is as follows.

| Minimum System Configuration | Value |
|---|---|
| **Operating System** | Any (Linux, Windows) |
| **RAM** | 8GB |
| **Hard Disk** | 100GB |
| **Number of CPU Cores** | 8 Cores |
| **Platform** | AWS Cloud9 |
| **Python Libraries** | Numpy, Requests, Flask, Scikit-learn, TensorFlow, Keras, pickle |

# 6 Evaluation

This section describes all the evaluation metrics used for this research. To determine the efficiency and effectiveness of the implemented autoscaling algorithms, a systematic and robust evaluation approach is crucial. This ensures that the results are not only reliable but also reproducible, allowing for consistent comparison and benchmarking. In order to achieve this objective we will be calculating the total response time, delayed load, error in model prediction and wastage of resources, the algorithm which will be achieving the minimal value for all the described metrics will be considered as the most optimal autoscaling algorithm.

## 6.1 Experiment-1 / Total Response Time

To calculate the Total response time measures, we will be calculating the total time taken from the moment a request is initiated by the client until a response is received. It is used to calculate the efficiency and speed of the system. A low response time is desirable, as it means that the system is responsive and can handle a high volume of load. On the other hand, an algorithm with a high response time indicates a delay or lag in processing a request. The total response time obtained from each algorithm is represented with the help of a bar graph as shown in Figure 5.

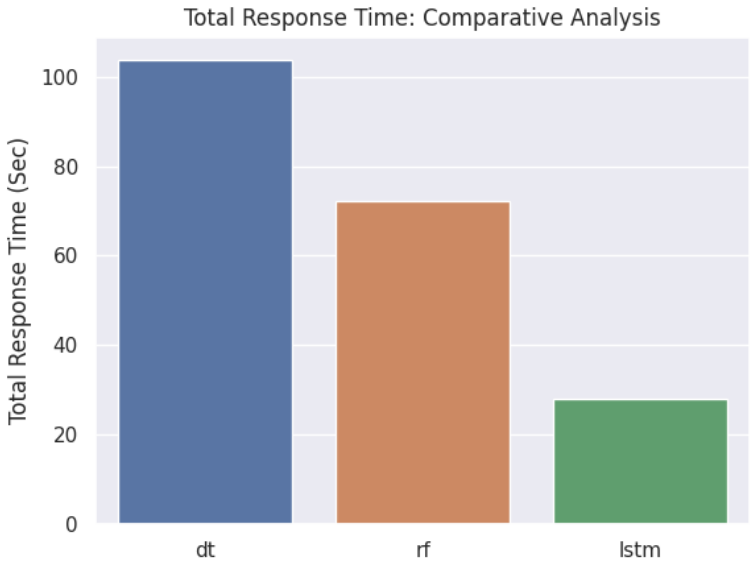


Figure 5: Comparison of Total Response time Casalicchio (2019)

Figure 5, represents the total response time of all the 3 autoscaling algorithms, these algorithms are executed in our simulated environment for 100 requests then the total response time has been calculated. After analyzing the graph it has been found that the LSTM algorithm is able to achieve the minimum response time close to 30 seconds, whereas, the response time obtained for the random forest algorithm is 70 Seconds and for the decision tree it's approximately 110 seconds. In terms of response time, the LSTM algorithm is 4 times more efficient than the decision tree and 2 times more efficient than the random forest algorithm. Figure 6 represents the distribution of response time over each request for all the algorithms and has been represented with the help of a scatter plot.

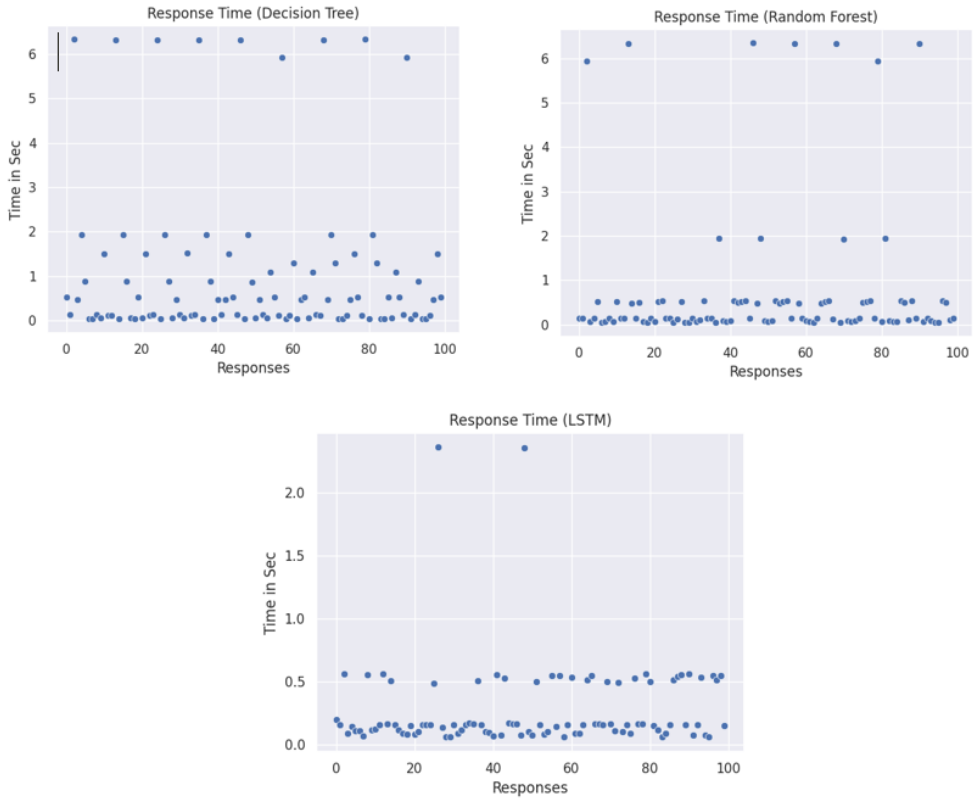


Figure 6: Distribution of Response time with respect to requests Casalicchio (2019)

## 6.2 Experiment-2 / Comparison of Delayed Load

Delayed load represents the total amount of load that experiences a delay before processing due to the overloading of tasks to the processing node. A high amount of delayed load can signal bottlenecks or inefficiencies within the system. If a task is getting delayed it clearly indicates that there is a prediction error from the model end which is either under-provisioning or over-provisioning the resources. Underprovisioning of resources is one of the possible reasons for delayed load. Delayed load can be avoided by adding more resources to the computing nodes. In this experiment total delayed load for each algorithm has been calculated and visualized with the help of a bar graph as shown in Figure 7. The total delayed load obtained using the decision tree algorithm is very high as compared to the random forest and LSTM model, which clearly represents that the decision tree autoscaling algorithm is not able to make correct predictions and which results in the underprovisioning of resources. On the other hand, with the help of LSTM minimum amount of load has been delayed which clearly indicates that the algorithm is correctly able to perform the capacity planning of resources for processing the load.

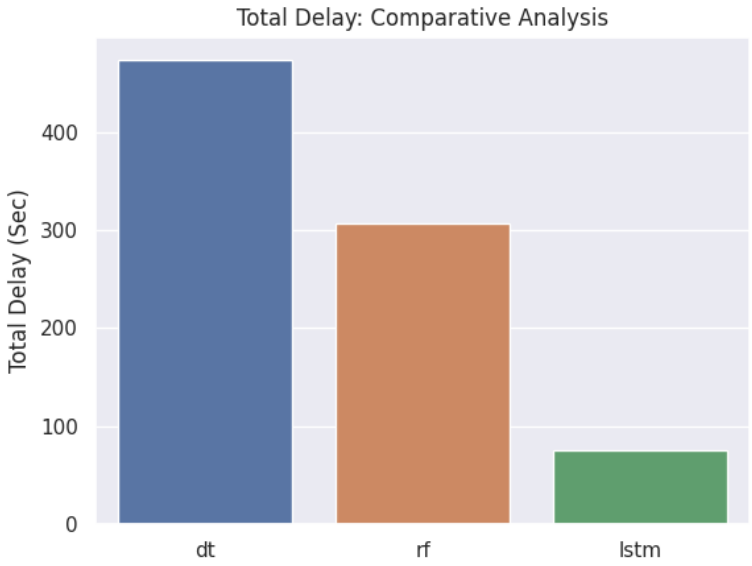


Figure 7: Comparison of Delayed Load

## 6.3 Experiment-3 / Error in Model Prediction

In this experiment, we will be calculating the Error in model prediction for each autoscaling algorithm. This metric quantifies the discrepancy between the predicted load type (or amount) by the autoscaler's algorithms and the actual load observed. Error in model prediction will help in assessing the accuracy and reliability of the implemented algorithms for autoscaler. Accurate predictions ensure that resources are appropriately scaled. High prediction errors could lead to over-provisioning (leading to resource wastage) or under-provisioning (leading to delays). The optimal autoscaling algorithm should have minimal prediction error. On analyzing the graph, it is clearly evident that the LSTM algorithm calculates the minimum model prediction error which represents that it is optimally able to allocate and reallocate the resources as compared to the random forest and decision tree algorithm. The performance of the random forest is better than the decision tree algorithm. However, still, the random forest algorithm is 3 times less efficient than the LSTM algorithm. The error comparison among the models has been represented with the help of a horizontal bar graph as shown in Figure 8.

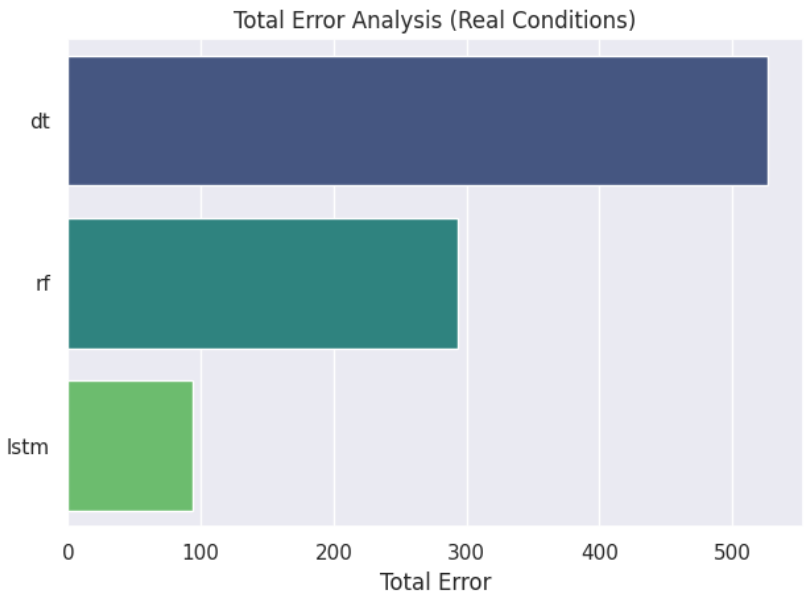


Figure 8: Comparison of Error in Model Prediction Dang-Quang and Yoo (2021)

## 6.4 Experiment-4 / Wastage of Resources

This metric indicates the additional amount of computing resources (like processing power, memory, etc.) that were allocated but remained unused or were underutilized. Wasted resources equate to wasted money, especially in cloud environments where we pay for provisioned resources. Wastage of Resource metric can guide the autoscaler's decision-making process, encouraging it to scale down when excess resources are consistently detected. The following graph as shown in Figure 9, represents the total wastage of computing resources by each algorithm. On analyzing the graph very carefully, it has been found that Random forest minimizes the over-provisioning of resources and thus reduces resource wastage. However, the resource wastage with the LSTM algorithm is higher than the random forest algorithm but lower than the decision tree algorithm.

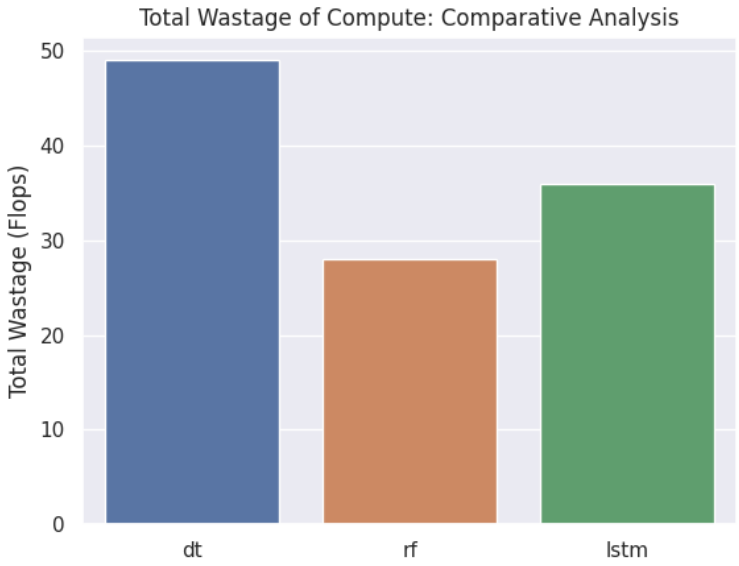


Figure 9: Resource Wastage Comparison Meng et al. (2016)

## 6.5 Discussion

In this section, we will be discussing the overall summary of our evaluation analysis. Comparing all 3 algorithms with respect to different metrics has been identified that LSTM achieves the minimum response time, delayed load, and error in model prediction. This is because LSTM is a powerful algorithm that can learn long-term dependencies in the data. This allows LSTM to predict the future load with a high degree of accuracy, which helps to ensure that the system is always able to handle the load without delay. The random forest has the minimum resource wastage followed by LSTM. This is because the random forest is a relatively conservative algorithm and less likely to overfit compared to a single decision tree that does not overprovision resources as much as LSTM. However, this can also lead to underprovisioning of resources, which can result in delayed load. The Decision Tree algorithm performed sub-optimally across all metrics and obtained the highest metric value as the decision tree is not as good at handling long-term dependencies in the data, which can lead to delayed load and error in model prediction. While LSTM demonstrated an edge in response time, delayed load, and prediction accuracy, it's crucial to weigh these advantages against the slightly increased resource wastage. Random Forest's strength in optimal resource provisioning showcases its potential in environments where resource conservation is a priority. In this research accuracy of the algorithm and minimizing service disruption is our first priority therefore, we are considering LSTM as most efficient algorithm for autoscaling. However, if minimizing resource wastage is a concern then Random Forest is a good compromise. Our research can be very helpful for cloud providers as it can help them in Optimized resource management, maintaining the service level agreements (SLAs), cost saving, and Informed Decision-Making.

# 7 Conclusion and Future Work

Autoscaling in cloud environments is faced with challenges such as predictive accuracy, resource management, latency concerns, and cost efficiency. While many prior studies have narrowed their focus to a single algorithm or a limited set of techniques, our research took a broader view, encompassing three distinct algorithms: Decision Tree, Random Forest, and LSTM. The inclusion of LSTM, in particular, underscores the significance of time-series analysis in this field. Our findings reveal that LSTM is superior in metrics like response time, delayed load, and prediction error. However, it's vital to note that LSTM can sometimes lead to resource wastage. Random Forest strikes a balance between response time and resource conservation, whereas Decision Tree, despite its simplicity and ease of implementation, does not perform as effectively. A defining feature of our study was the application of multiple evaluation metrics like Total Response Time, Total Delayed Load, Prediction Error of Models, and Total Wastage of Resources. This approach offers a more holistic view, addressing a common limitation in existing research that often zeroes in on a single metric, thus missing the bigger picture of autoscaling impacts. Looking to the future, we believe there's great potential in integrating different models to leverage their respective strengths. Further exploration into various neural network architectures could also yield improved performance. The adoption of reinforcement learning, refining current monitoring techniques, a sharper focus on cost implications, and the development of dynamic feedback systems should also be at the forefront of autoscaling research. In essence, while our study has made significant strides, the autoscaling domain beckons with ample opportunities for innovation and optimization.

# References


A., A. (2016). Automatic cloud resource scaling algorithm based on long short-term memory recurrent neural network, *International Journal of Advanced Computer Science and Applications* **7**(12).
**URL:** *https://doi.org/10.14569/ijacsa.2016.071236*

Ahamed, Z., Khemakhem, M., Eassa, F., Alsolami, F. and Al-Ghamdi, A. S. A.-M. (2023). Technical study of deep learning in cloud computing for accurate workload prediction, *Electronics* **12**(3): 650.
**URL:** *https://doi.org/10.3390/electronics12030650*

Biswas, A., Majumdar, S., Nandy, B. and El-Haraki, A. (2017). A hybrid auto-scaling technique for clouds processing applications with service level agreements, *Journal of Cloud Computing* **6**(1).
**URL:** *https://doi.org/10.1186/s13677-017-0100-5*

Casalicchio, E. (2019). A study on performance measures for auto-scaling CPU-intensive containerized applications, *Cluster Computing* **22**(3): 995–1006.
**URL:** *https://doi.org/10.1007/s10586-018-02890-1*

Cheng, M., Li, J. and Nazarian, S. (2018). DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers, *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, IEEE.
**URL:** *https://doi.org/10.1109/aspdac.2018.8297294*

Chiu, M.-H., Yu, Y.-R., Liaw, H. and Hao, L. (2016). The use of facial micro-expression state and tree-forest model for predicting conceptual-conflict based conceptual change.

da Silva, T. P., Neto, A. R., Batista, T. V., Delicato, F. C., Pires, P. F. and Lopes, F. (2022). Online machine learning for auto-scaling in the edge computing, *Pervasive and Mobile Computing* **87**: 101722.
**URL:** *https://doi.org/10.1016/j.pmcj.2022.101722*

Dang-Quang, N.-M. and Yoo, M. (2021). Deep learning-based autoscaling using bidirectional long short-term memory for kubernetes, *Applied Sciences* **11**(9): 3835.
**URL:** *https://doi.org/10.3390/app11093835*

Dang-Quang, N.-M. and Yoo, M. (2022). An efficient multivariate autoscaling framework using bi-LSTM for cloud computing, *Applied Sciences* **12**(7): 3523.
**URL:** *https://doi.org/10.3390/app12073523*

Dimitriadis, S. and Liparas, D. (2018). How random is the random forest? rf algorithm on the service of structural imaging biomarkers for ad: from adni database, *Neural Regeneration Research* **13**: 962–970.

Do, T.-X. and Tan, V. K. N. (2022). Hybrid autoscaling strategy on container-based cloud platform, *International Journal of Software Innovation* **10**(1): 1–12.
**URL:** *https://doi.org/10.4018/ijsi.292019*

Dogani, J., Namvar, R. and Khunjush, F. (2023). Auto-scaling techniques in container-based cloud and edge/fog computing: Taxonomy and survey, *Computer Communications* **209**: 120–150.
**URL:** *https://doi.org/10.1016/j.comcom.2023.06.010*

Fe, I., Matos, R., Dantas, J., Melo, C. and Maciel, P. (2017). Stochastic model of performance and cost for auto-scaling planning in public cloud, *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE.
**URL:** *https://doi.org/10.1109/smc.2017.8122926*

Fé, I., Matos, R., Dantas, J., Melo, C., Nguyen, T. A., Min, D., Choi, E., Silva, F. A. and Maciel, P. R. M. (2022). Performance-cost trade-off in auto-scaling mechanisms for cloud computing, *Sensors* **22**(3): 1221.
**URL:** *https://doi.org/10.3390/s22031221*

Hasan, M. N., Toma, R. N., Nahid, A.-A., Islam, M. M. M. and Kim, J.-M. (2019). Electricity theft detection in smart grid systems: A CNN-LSTM based approach, *Energies* **12**(17): 3310.
**URL:** *https://doi.org/10.3390/en12173310*

J.V., B. B. and Dharma, D. (2018). HAS: Hybrid auto-scaler for resource scaling in cloud environment, *Journal of Parallel and Distributed Computing* **120**: 1–15.
**URL:** *https://doi.org/10.1016/j.jpdc.2018.04.016*

Marie-Magdelaine, N. and Ahmed, T. (2020). Proactive autoscaling for cloud-native applications using machine learning, *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, IEEE.
**URL:** *https://doi.org/10.1109/globecom42002.2020.9322147*

Meng, Y., Rao, R., Zhang, X. and Hong, P. (2016). Crupa: A container resource utilization prediction algorithm for auto-scaling based on time series analysis, *2016 International Conference on Progress in Informatics and Computing (PIC)*, pp. 468–472.

Qassem, L. M. A., Stouraitis, T., Damiani, E. and Elfadel, I. A. M. (2023). Proactive random-forest autoscaler for microservice resource allocation, *IEEE Access* **11**: 2570–2585.
**URL:** *https://doi.org/10.1109/access.2023.3234021*

Radhika, E. and Sadasivam, G. S. (2021). A review on prediction based autoscaling techniques for heterogeneous applications in cloud environment, *Materials Today: Proceedings* **45**: 2793–2800.
**URL:** *https://doi.org/10.1016/j.matpr.2020.11.789*

Xue, S., Qu, C., Shi, X., Liao, C., Zhu, S., Tan, X., Ma, L., Wang, S., Wang, S., Hu, Y., Lei, L., Zheng, Y., Li, J. and Zhang, J. (2022). A meta reinforcement learning approach for predictive autoscaling in the cloud, *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ACM.
**URL:** *https://doi.org/10.1145/3534678.3539063*

ZargarAzad, M. and Ashtiani, M. (2023). An auto-scaling approach for microservices in cloud computing environments.
**URL:** *https://doi.org/10.21203/rs.3.rs-3020374/v1*

Zhong, J., Duan, S. and Li, Q. (2019). Auto-scaling cloud resources using LSTM and reinforcement learning to guarantee service-level agreements and reduce resource costs, *Journal of Physics: Conference Series* **1237**(2): 022033.
**URL:** *https://doi.org/10.1088/1742-6596/1237/2/022033*