National College of Ireland

Forecasting the Future: Enhancing Cloud Efficiency through Dynamic CPU Utilization Allocation

MSc Research Project
MSc Cloud Computing

Prathamesh Warekar
Student ID: x21230196

School of Computing
National College of Ireland

Supervisor: Prof. Sean Heeney

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Prathamesh Warekar
Student ID:	x21230196
Programme:	MSc Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Prof. Sean Heeney
Submission Due Date:	25/04/2023
Project Title:	Forecasting the Future: Enhancing Cloud Efficiency through Dynamic CPU Utilization Allocation
Word Count:	XXX
Page Count:	19

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:	
Date:	24th April 2024

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):	

Forecasting the Future: Enhancing Cloud Efficiency through Dynamic CPU Utilization Allocation

Prathamesh Warekar
x21230196

Abstract

CPU utilization in the cloud describes the percentage of time that a central processing unit (CPU) is actively executing instructions, reflecting the workload demand placed on the CPU. The abstract of this report is the exploration for forecasting CPU utilization in the cloud for dynamic allocation in the cloud. Various traditional methods have been used for forecasting CPU utilization which lack in their RMSE, MSE and failed. However, this study is using few time-series and forecasting algorithms to develop better results and adaptive predictive models. The dataset is of Microsoft Azure which is a publicly accessible dataset for accurately forecasting future CPU utilization patterns and all so that it can enable some sort of cloud service providers and optimize resource allocation as well. For preprocessing the data, this research uses scaling and window rolling techniques, and training multiple machine learning models such as Support Vector Regressor, Extra Trees Regressor, AdaBoost Regressor, and Stacking Regressor, the study aims to identify the most effective approach for CPU utilization forecasting. The stacking regressor has been considered and found the best model for forecasting CPU utilization in the cloud which combined multiple base estimators, including Support Vector Regressor (SVR) and AdaBoost Regressor, with a final estimator, Extra Trees Regressor, to improve prediction accuracy which means evaluation metrics. This has also enhanced cloud efficiency through dynamic CPU utilization allocation.

1 Introduction

In an era of digital transformation, the cloud has emerged as a key component of modern computing infrastructure. Its scalability and flexibility provide organizations with unparalleled opportunities to optimize resources and enhance operational efficiency. Despite this abundance of computing power, resource management remains a challenge to ensure both cost-effectiveness and optimal performancePrasad et al. (2023). The dynamic allocation of CPU utilization is a crucial component of this management and the key to realizing the full potential of cloud environments. Resource allocation is a challenging task because of the dynamic nature of workloads and the limitations of conventional resource allocation strategies. The inability of the current procedures to adapt swiftly to shifting workloads could lead to inefficiencies and wasteful resource usage. It is necessary to study and integrate machine learning techniques—specifically, the Adaboost regressor, support vector regressor and stacking regressor—into cloud computing infrastructures to allocate resources dynamicallyKhan et al. (2022).

1.1 Aim of the study

This report aims to evaluate various forecasting algorithms that this study has used and also to develop them for CPU utilization forecasting in the cloud. Various forecasting and time series models have been developed and used with the help of the Microsoft Azure datasetAzure (n.d.) which is accessible publicly. The main goal is to increase performance scalability and operational efficiency by dynamically allocating computer resources in response to workload needs that are anticipated. The study seeks to determine the most efficient forecasting algorithms for CPU utilization predictions, including Support Vector Regressor, Extra Trees Regressor, AdaBoost Regressor, and Stacking Regressor, through thorough data preparation, model training, and evaluation. Furthermore, the study attempts to evaluate how prediction accuracy and model performance are affected by preprocessing methods such as Window Rolling and Min-Max Normalization. The ultimate goal of this study's conclusions is to offer useful advice and insights for applying predictive modelling methods in cloud environments to maximize resource usage, improve system efficiency, and reduce operating expenses.

1.2 Research Objectives

There are various research objectives in this study which are as follows:

1. This study is forecasting CPU utilization on cloud data for dynamic allocation and for that need to collect the Microsoft Azure dataset which will preprocess historical data of the cloud.

2. To investigate different preprocessing methods to improve the dataset quality, such as Window Rolling and Min-Max Normalization.

3. To train and evaluate multiple forecasting algorithms, including Support Vector Regressor, Extra Trees Regressor, AdaBoost Regressor, and Stacking Regressor, for CPU utilization forecasting.

4. To identify the most effective forecasting model or combination of models for CPU utilization forecasting in the cloud environments.

1.3 Research Questions

What are the different forecasting algorithms and How CPU utilization forecasting be helpful for the dynamic resource utilization allocation on the cloud?

1.4 Research Gaps

There are a few research gaps as well which have been addressed in this study:

1. Limited Exploration of Advanced Algorithms: While various forecasting algorithms have been applied to predict CPU utilization in cloud environments, there may be a gap in the exploration of more advanced algorithms like LSTM, Gradient boosting and all. Investigating these advanced algorithms could potentially lead to improved prediction accuracy and robustness.

2. Lack of Consideration for External Factors: Many of the research that are currently available only consider internal elements that affect CPU utilization, ignoring the influence of external factors such user behavior, software upgrades, and network circumstances. Predictive models that incorporate external factors may yield a more thorough understanding of the dynamics of CPU consumption and improve prediction accuracy.

2 Related Work

2.1 Traditional Resource Allocation Techniques

There are several types of traditional resource allocation techniques in cloud computing which include Static Provisioning, Manual Scaling, Threshold-based Scaling, Load balancing etc. Husaini et al. (2023) Introducing a new paradigm for cloud resource management, the main goal is to meet Quality of Service (QoS) requirements and maximize utilization by allocating resources effectively at the application level. The system uses auto-scaling methods to automatically reallocate virtual resources according to workload, to maximize efficiency and lower expenses.

Similarly, Oladoja et al. (2021) discusses fixed and adaptive threshold-based auto-scaling as it dives into auto-scaling approaches, which are essential for attaining 100% availability and scalability in cloud environments. In the meantime, Khan et al. (2024) addresses computational offloading to effectively manage work between mobile devices and cloud servers as it investigates load-balancing methods in the context of mobile cloud computing.

About 5G networks, Qin et al. (2022) presents a distributed threshold-based offloading algorithm to maximize task and resource allocation. Moving on to Infrastructure as a Service (IaaS) models, Rotter and Van Do (2021) addresses workload balancing difficulties and suggests a queueing model to optimize User Plane Function (UPF) instances in 5G networks.

Additionally, Mishra et al. (2020) offers an analysis of load-balancing algorithms in cloud computing environments, including taxonomy and simulation-based assessments to gauge their effectiveness. Finally, Shafiq et al. (2021) focuses on load balancing in IaaS cloud models, with a novel LB algorithm designed to maximize resource utilization and improve performance. To elaborate, Rehman et al. (2020) provides an extensive analysis of load-balancing strategies in static, dynamic, and cloud contexts inspired by nature, highlighting the significance of preserving application performance while complying with SLA and QoS metrics. The review investigates fault-tolerant frameworks to increase cloud infrastructure dependability and notes research gaps through analytical evaluations and graphical displays.

2.2 Comparison of Literature Reviews on Traditional Techniques on Resource Allocation in Cloud

Table 1: Literature review summary

Study	Focus	Approach	Key Challenges Addressed	Main Results
Husaini et al. (2023)	Cloud resource management framework	Application-level resource allocation, auto-scaling	Predicting client-side experience, optimizing resource utilization	Improved performance, cost minimization
Oladoja et al. (2021)	Auto-scaling techniques in cloud computing	Fixed and adaptive threshold-based auto-scaling	Achieving 100% availability and scalability	Improved fault tolerance, availability, cost management
Khan et al. (2024)	Load balancing algorithms in mobile cloud computing	Computational offloading, distributed threshold-based offloading	Efficient task allocation between mobile devices and cloud servers	Enhanced system resource utilization, performance
Qin et al. (2022)	Distributed threshold-based offloading in 5G networks	Offloading algorithm for UPF instances	Optimization of task and resource allocation	Convergence to Nash Equilibrium, performance gap analysis
Rotter and Van Do (2021)	Optimizing UPF instances in 5G networks	Queueing model for UPF instances	Load balancing in 5G networks, resource optimization	Improved resource utilization, performance
Mishra et al. (2020)	Load balancing algorithms in cloud computing	Taxonomy, simulation-based evaluations	Resource allocation, performance optimization	Understanding algorithm performance, resource utilization
Shafiq et al. (2021)	Load balancing in IaaS cloud models	Novel LB algorithm for IaaS	Workload balancing, resource utilization optimization	Improved resource utilization, performance
Rehman et al. (2020)	Review of load balancing techniques in cloud computing	A comprehensive review of LB techniques	Addressing QoS metrics, SLA requirements, fault tolerance	Identifying research gaps, improving infrastructure reliability

2.3 Advanced techniques used for forecasting CPU utilization in cloud:

The rapid expansion of large-scale cloud data centres has made it increasingly difficult to forecast resource consumption in the future. Numerous studies have suggested creative predictive models that make use of cutting-edge machine-learning techniques to address this problem. Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks are integrated to anticipate CPU use in cloud servers over numerous time steps, as Patel and Kushwaha (2022) highlight.

The pCNN-LSTM model, which combines the temporal correlations captured by LSTM with the pattern extraction capabilities of CNN, shows notable gains in host load prediction accuracy on a range of cloud workload datasets. Similar to this, Karim et al. (2021) introduces BHyPreC, a hybrid Recurrent Neural Network (RNN) model, to forecast multivariate workload components in cloud virtual machines (VMs), such as CPU, memory, and network utilization.

Overtaking conventional statistical models such as Autoregressive Integrated Moving Average (ARIMA), BHyPreC improves non-linear data analysis by combining Bidirec-

tional LSTM (Bi-LSTM) with stacked LSTM and Gated Recurrent Unit (GRU) layers. Moreover, Ouhame et al. (2021) addresses the problem of hardware resource allocation delays during cloud-hosting initiation by putting forth a prediction model that forecasts CPU, memory, and network utilization by fusing CNN and LSTM.

At last, Nashold and Krishnan (2020) have explored models for predicting CPU usage over long and short-term time scales in large-scale cloud which includes LSTM and SARIMA. While SARIMA outperforms LSTM for long-term predictions, LSTM demonstrates greater robustness and adaptability, particularly for short-term forecasting tasks.

3 Research Methodology

3.1 Methodology

This chapter of Crisp DM will establish a structured type of approach for guiding machine learning projects and for data mining as well. This study will also use machine learning models. CRISP-DM stands for the Cross-Industry Standard Process for Data Mining. This methodology has six phases which include Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation, and Deployment. This CRISP-DM offers a systematic framework which enables organizations to work effectively and leverage data for gaining insights, that will make informed decisions and will do drive business value across various industries and domains.

1. Business Understanding: In this phase of business understanding main focus is on enhancing the efficiency of the cloud with the help of dynamic CPU utilization allocation and on defining its objectives as well. This phase also includes about the overarching goals of this study like how they do align the cloud service providers on the basis of their business priorities. Several key considerations have been included like to identifying specific opportunities and their challenges as well to do optimising the utilization of CPU and for the proposed solution as well as the defining success criteria. Additionally, assessing the current state of CPU utilization management and potential areas for improvement is crucial in shaping the direction of the project.

2. Data Understanding: In the second phase of data understanding their aim within cloud environments is to gain analysis of CPU utilization trends and of course patterns. To do this, gather and analyze the Azure dataset, which is made up of time-series CPU use statistics from Microsoft Azure traces. Investigating the dataset to comprehend its quality, organization, and variable relationships is one of the main duties. Finding any outliers, inconsistent data, or missing values in the data that could affect the study is crucial. Furthermore, analyzing how CPU consumption varies over time and among various virtual machine (VM) workloads can reveal important information about usage trends and possible areas for optimization. Comprehending the features and constraints of the dataset will help with the preprocessing and modelling of the data that comes after, guaranteeing that the analysis appropriately captures the fundamental dynamics of CPU consumption in cloud environments.

3. Data Preparation: For transforming, cleaning data and then selecting a relevant amount of data for doing analysis from the dataset of Azure, data preparation comes and does all this. This entails addressing any outliers, discrepancies, or missing numbers that were found during the Data Understanding stage. For missing data, methods like

imputation or removal can be used. To maintain consistency throughout the dataset, it can also be necessary to scale numerical features and encode categorical variables. For additional analysis, subsets of information and characteristics pertinent to the prediction of CPU utilization will be chosen. To aid in modelling and visualization, time-related characteristics like datetime conversion and the extraction of supplementary temporal attributes like month and year may also be carried out. Effective data preparation enables the modelling step that follows to use well-formatted, clean data to create precise predictive models for CPU consumption in cloud environments.

4. Modelling: For doing training and evaluating with the help of Azure dataset for predicting CPU utilization this phase does all that in the cloud environments. There are various machine learning algorithms which this study is going to use Support Vector Regressor, Extra Tree Regressor, AdaBoost Regressor, and Stacking Regressor, will be implemented. It is possible to tune hyperparameters to maximize model performance. We'll investigate ensemble methods like stacking to see if we can increase prediction accuracy. We'll use cross-validation techniques to evaluate the models' resilience. Performance measures including Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared score (R2-score) will be computed as part of the model evaluation process. Additionally, to assess model predictions and pinpoint areas for additional refinement, visualizations like actual versus anticipated graphs will be created. Creating precise and dependable models for forecasting CPU use in cloud systems based on past data is the ultimate objective of the modelling phase.

5. Evaluation: This phase of the Crisp DM which is the evaluation phase is going to check the first performance of the above-trained machine learning models for doing prediction of CPU utilization in cloud. In order to assess the precision and dependability of the models, this entails computing a number of assessment measures, including Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared score (R2-score). To determine the optimal algorithm or ensemble approach, the models will also be contrasted with one another. Visualizations, such as real versus anticipated graphs, will be studied to discover how effectively the models reflect the underlying patterns in the data.

3.2 Libraries Imported

This study uses quite a few libraries for predicting CPU utilisation in the cloud and for that, it needs to facilitate from various stages of the ML pipeline. This study has imported several essential analysis libraries and some data, manipulation libraries like Pandas and NumPy. These types of libraries have been used for handling the dataset for performing some sort of mathematical operations and for manipulating data structures as well. Libraries like Matplotlib, Seaborn, and Plotly are imported for data visualization. While Plotly makes it possible to create dynamic and interactive visuals that improve data display and exploration, Matplotlib and Seaborn provide extensive plotting tools for the construction of static visualizations. One essential library for applying machine learning models is scikit-learn, sometimes known as sklearn. It offers a large selection of tools and algorithms for preprocessing, model selection, training, assessment, and calculating performance indicators. With its versatility and scalability for intricate modelling

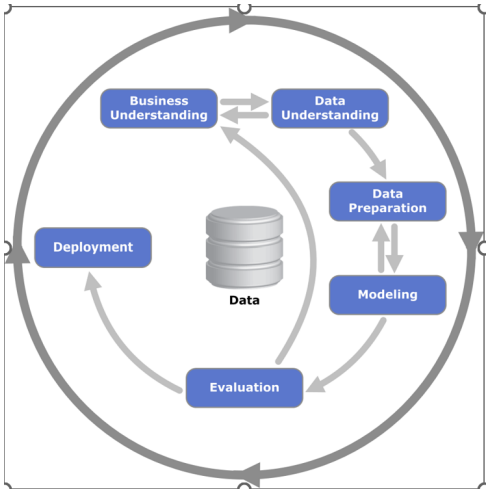

Figure 1: CRISP-DM Methodology (n.d.)

problems, TensorFlow, a potent deep learning framework, is imported for possible use in developing neural network models for CPU use prediction. In addition to that, there is a warning library that has been used to ensure a cleaner type of output for the execution of code.

3.3 Data Preprocessing

Data preprocessing is a very important phase of the project workflow which is used to prepare the respective dataset for the analysis of the model and also for model training. This phase also does the cleaning of the dataset and then it will select the relevant type of features only for CPU utilization prediction. This will demonstrate the creation of a cleaned type of DaatFrame naming 'cleadDF' for selecting the 'timestamp' column as the index and selecting specific columns ('min cpu', 'max cpu', 'avg cpu') representing minimum, maximum, and average CPU utilization, respectively. Temporal analysis is made easier by making the data time-series oriented by assigning the timestamp as the index. The DataFrame is then filtered such that it only contains the pertinent CPU use information. This procedure makes sure that all superfluous columns have been eliminated from the dataset and that it is properly organized for modelling. The preprocessed dataset, known as the 'cleanDF' DataFrame, is prepared for additional analysis, visualization, and model training. These preprocessing procedures are crucial for enhancing data quality, lowering noise, and guaranteeing that the dataset complies with the study's goals. Effective data preparation enables the modelling phase that follows to create precise predictive models for CPU use in cloud environments by utilizing relevant and clean attributes.

3.4 Feature Extraction

This is accomplished using the 'timestamp' column, which is designed to house an object datatype and is subsequently altered by a function entitled 'pd.todatetime'. Consequently, the conversion in question turns the 'timestamp' into a data type that is simpler to manipulate and analyze when working with time-based information. Moreover, the

functions help the researcher to extract two additional columns from the 'timestamp', which include 'year' and 'month' via 'dt.year' and 'dt.month' accessor methods. As a result, both 'year' and 'month' return their respective components of the designated 'timestamp' column, which then allows the following computation and visualization. Lastly, a 'day' column is needed to identify the fourth time component and code for the 'day' within it. Extracting the year, month, and day components enables a more detailed exploration of temporal patterns and trends in CPU utilization data across multiple time segments. I will use such features to plot seasonal patterns and I will also exploit them so that it can detect high-level trends and can decide how time-related features influence CPU utilization in a cloud environment.

3.5 Data Splitting (Training and Testing the Model)

This phase of data splitting evaluates the performance of the machine learning model in an accurate manner. This report is going to explain how much data is for training and how much is for testing the data. So the dataset has been divided into training and testing up of the data. The ratio of training and testing is 80:20 respectively. This ratio gives a significant type of portion which has been used to retain a separate subset for evaluation. There are two variables in the code which are 'training_size' and other one is 'test_size' variables which is used to represent the total number of samples which has been allocated to each type of subset having 80% of the data allocated for training purposes and trest 20% data is allocated for testing the data. This splitting up of data will help so that it can prevent it from overfitting during the testing up of unseen data for allowing the model to generalise in a well-mannered way.

3.6 Dataset Description

The dataset used in this report has been provided by Microsoft Azure which comprises public releases of Azure traces, intended for research and academic purposes. It includes Azure Functions Traces and VM Traces, which are the two main categories of traces. The VM Traces are made up of a special VM request trace intended for the study of packing techniques, as well as sample datasets gathered in 2017 and 2019. These traces, which record data at a granularity of every five minutes, provide insights into the workload of virtual machines in the Azure environment. The Azure Functions Traces also comprise traces of Azure Functions blob accesses that were gathered in November and December of 2020, along with typical traces of Azure Functions invocations throughout two weeks in 2019. Azure (n.d.)

4 Design Specification

This chapter will define and describe the respective approach and the methodology which this study is going to use for forecasting dynamic CPU utilization allocation. This chapter will guide and explain several different components so that we can achieve accurate and reliable predictions, Initially, the system architecture has been designed for efficiently handling large amounts of data. This entails putting in place scalable data processing and storage systems, like distributed databases and parallel processing frameworks, to handle the enormous volume of historical and current CPU utilization data that is gathered from cloud settings. Strong data pretreatment methods should also be included in the system

to clean up and convert the raw data into a format that can be analyzed. Managing missing numbers, identifying outliers, and feature engineering are all part of this process to draw out pertinent information from the data. Moreover, the system's predictive modelling component is to make use of sophisticated machine learning algorithms that can grasp the intricate dynamics and linkages present in CPU use patterns. Models such as Support Vector Regression (SVR), Extra Trees Regressor, and Stacking Regressor can be explored for their ability to provide accurate predictions while accounting for non-linearities and seasonality in the data.

To find the best algorithms for the task, careful testing and validation should be used to inform the modelling technique selection. The system should also have warning and monitoring features to track model performance over time and identify any deviations from predicted behaviour. This entails putting in place real-time monitoring dashboards that show trends in CPU consumption, evaluation data, and model forecasts. Automated warning systems should advise stakeholders in the event of anomalies or declining model performance. This will allow for prompt intervention and, if required, retraining of the model.

To maximize resource efficiency and guarantee performance scalability, the system should also provide dynamic resource allocation based on anticipated CPU utilization. To do this, you must integrate with cloud management platforms so that auto-scaling can begin in response to workload demands that are anticipated. To ensure effective resource allocation and avoid overloads, load balancing techniques can also be used to divide incoming requests among several servers or instances based on anticipated CPU consumption. Figure 2 shows the flowchart of the proposed work.

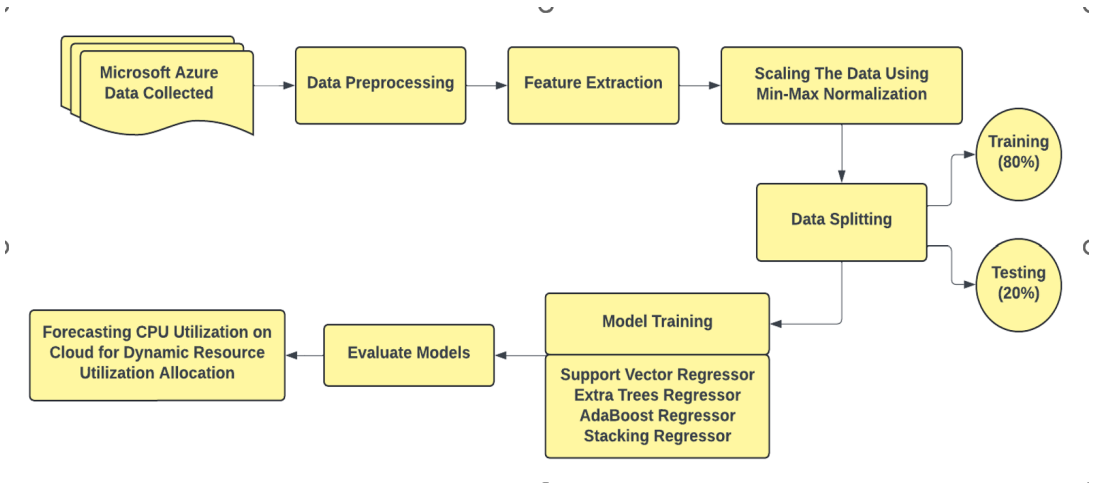

Figure 2: Proposed Workflow

5 Implementation

To implement the machine learning models for this project, Google Colab was utilized as the development environment due to its convenience and access to powerful computational

resources. The Azure Public Dataset is available at **https://github.com/Azure/AzurePublicDatas** was chosen as the dataset for training and testing the models.

The following machine-learning models were employed:

1. Support Vector Regressor (SVR)
2. AdaBoost Regressor
3. Extra Tree Regressor
4. Stacking Regressor

Each model was trained and evaluated using appropriate metrics to ensure its performance and effectiveness in predicting the target variable.

The steps for implementation include:

Data Preparation: The Azure Public Dataset was downloaded and preprocessed to extract relevant features and target variables.

Model Training: Each machine learning model (SVR, AdaBoost, Extra Tree, and Stacking Regressor) was trained using the prepared dataset.

Model Evaluation: The trained models were evaluated using various evaluation metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared score to assess their predictive performance.

Comparison: A comparative analysis of the models was conducted to identify the best-performing model based on the evaluation metrics.

The implementation involved leveraging Google Colab for development, utilizing the Azure Public Dataset for training, and employing SVR, AdaBoost, Extra Tree, and Stacking Regressor models for prediction tasks.

5.1 Support Vector Regressor

The Support Vector Regressor (SVR) is a regression-based of algorithm which is also called SVM in short and this type of framework is used for predicting continuous type of outcomes. In this report, this support vector regressor is going to be used and it plays a vital role in the underlying type of patterns and relationships between input features (such as time, previous utilization values, etc.) and the target variable (CPU utilization). Subject to a predetermined tolerance (epsilon), SVR finds the hyperplane that maximizes the margin between the data points and the hyperplane while also fitting the data the best. SVR focuses on selecting a subset of the training data termed support vectors, which are the data points closest to the hyperplane and impact its position. This is in contrast to classic regression algorithms, which seek to reduce prediction errors. The architecture of SVR entails utilizing a kernel function, where the hyperplane is defined, to transform the input characteristics into a higher-dimensional space. Next, to reduce the prediction error within the given tolerance, the SVR algorithm optimizes the hyperplane's position and orientation.

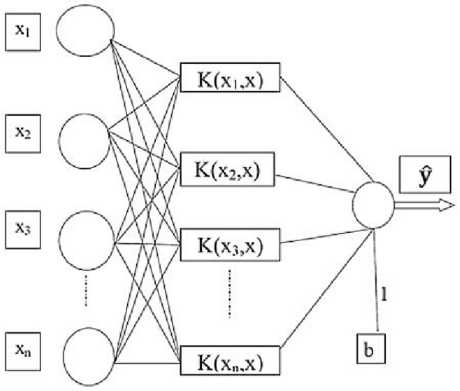

Figure 3: Support Vector Regressor architecture Thomas et al. (2017)

5.2 Extra Tree Regressor

The Extra Trees Regressor (ETR) is an ensemble learning technique that is used to predict CPU utilization in the cloud based on decision tree regressors in this study of course. Its role is to leverage several decision trees for achieving good evaluation metrics and robustness if we compare them to the individual tree for aggregation of predictions. The ETR algorithm in this project builds several decision trees using arbitrary subsets of the features and training data. To lower the chance of overfitting and improve generalization performance, each tree is trained separately and then its predictions are averaged over all the trees' predictions. By choosing feature subsets at each node split, ETR also adds unpredictability, which improves the variety and resilience of the model. An ensemble of decision trees is built as part of the ETR architecture, and each tree is trained using a different subset of the training set's characteristics and data. The final predictive model is created by combining the trees and averaging the estimates from each one to produce a more reliable and accurate estimate of CPU consumption.

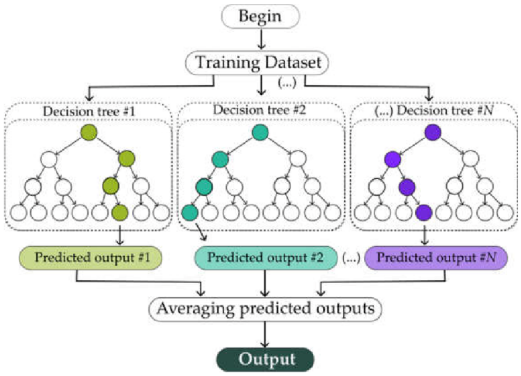

Figure 4: Extra Tree Regressor architecture Araújo et al. (2023)

5.3 AdaBoost Regressor

For combining multiple types of weak learners which means individual regression models for creating a good predictive model that should be accurate and model this study has

employed an ensemble learning method for that which is called the AdaBoost Regressor (ABR). AdaBoost Regressor is used in this research to fit a sequence of weak regression models to the data one after the other. First, the dataset is used to train a weak learner, and every data point is given the same weight. The system then gives data points that the previous weak learner had predicted incorrectly a higher weight. This process of iteration is continued, with each weak learner concentrating more on the incorrectly categorized data points from earlier iterations. AdaBoost Regressor's architecture consists of building a sequence of weak regression models, usually decision trees so that each model can learn from the mistakes of the others. All weak learners' predictions are combined and weighted according to their performance to create the final predictive model. This approach ensures that the final model focuses on the most challenging instances in the data, leading to improved predictive accuracy.

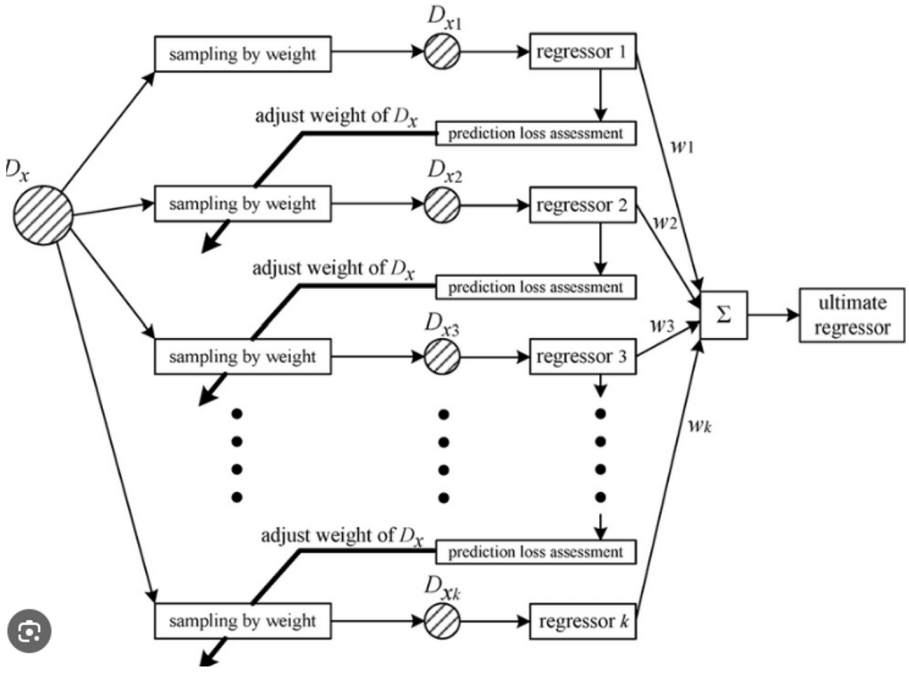

Figure 5: AdaBoost Regressor architecture Min and Luo (2016)

5.4 Stacking Regressor

The Stacking Regressor is an essential component of this project since it enhances the overall accuracy of CPU utilization prediction in cloud environments by combining the predictive powers of numerous base regressor models. Its main purpose is to balance the deficiencies of several individual regressors while integrating their strengths. In this research, the respective Stacking Regressor combines two types of base regressor models which are diverse and include the Support Vector Regressor (SVR) and AdaBoost Regressor (ABR). Support Vector Regressor is known for its ability to capture complex relationships in data, while AdaBoost Regressor excels in correcting the errors of its predecessors through iterative training. By leveraging these complementary strengths, the Stacking Regressor aims to create a more robust and accurate predictive model. The training of several base regressor models, each with distinct qualities and abilities, is

a key component of the Stacking Regressor architecture. Following that, a final meta-estimator learns how to optimally integrate the predictions from these base models as input characteristics to get the final prediction. By utilizing the variety of predictions produced by the base models, the Extra Trees Regressor serves as the last meta-estimator in this setup, thereby augmenting the ensemble's predictive power.

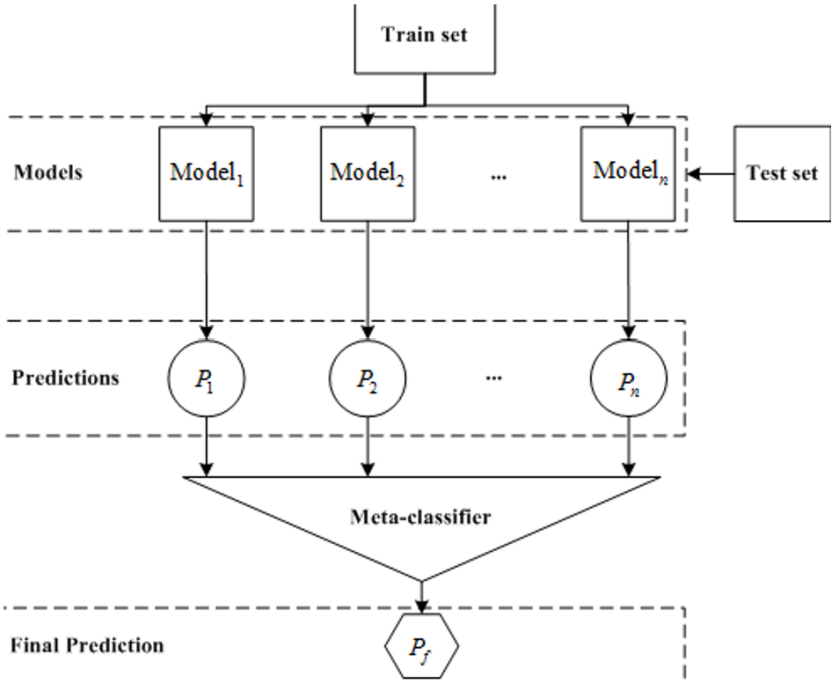

Figure 6: Stacking Regressor architecture Jiang et al. (2019)

5.5 Data Visualization

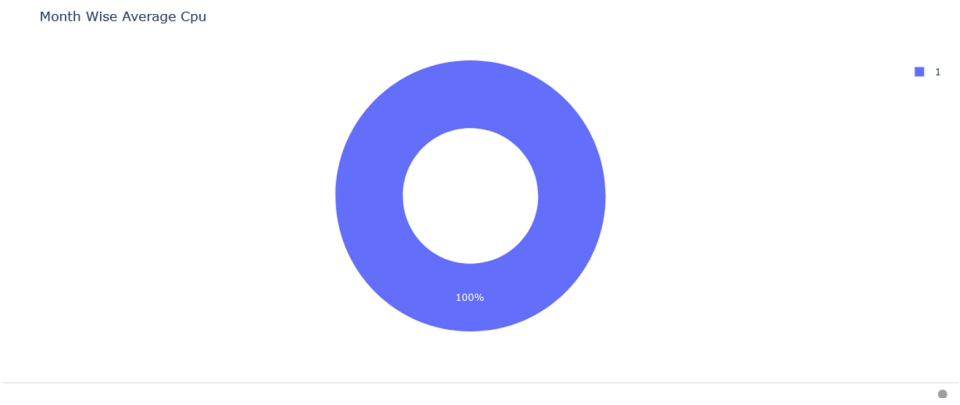

Figure 7: January month CPU Utilization Pie chart

In Figure 7, the January month average CPU utilization is visualized, with a label indicating 100% and a corresponding value of 1,215,661.103. The colour associated with this data point is blue, denoted by colour code 2. This representation provides a clear

visualization of the average CPU utilization across different months, highlighting the specific value for the 100% label and indicating its significance with the blue colour.

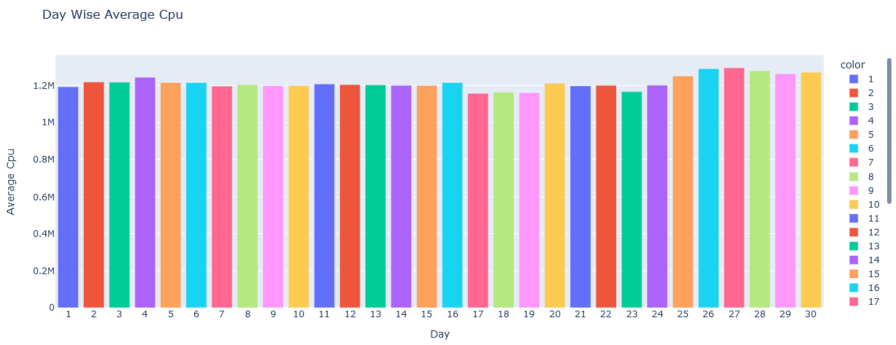

Figure 8: Bar Graph

Figure 8 illustrates a bar graph representing the January 2017 month day-wise average CPU utilization. Each bar in the graph corresponds to a specific day, ranging from 1 to 30. The colour of each bar varies to enhance visual distinction. Additionally, the graph displays the numerical values of the average CPU utilization (v) atop each bar, providing precise information about the CPU utilization level for each day. This visualization effectively captures the daily fluctuations in CPU utilization, allowing for a comprehensive understanding of the patterns and trends over the month.

6 Evaluation

This section lists a set of experiments to validate the proposed machine learning models. The aim is to evaluate the forecasting of CPU utilization in the cloud. This research uses machine learning models to forecast the usage of CPU in the cloud to reduce cost and increase efficiency. The stacking regressor ML model merges the Support Vector Regressor (SVR) and AdaBoost Regressor (ABR).

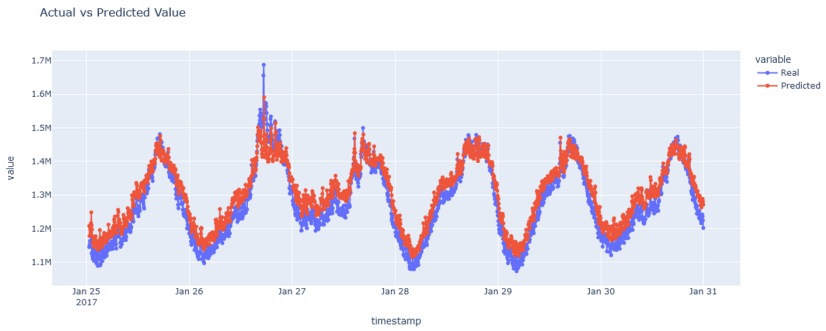

Figure 9: SVR - Actual vs Predicted Graph

Figures 9(Support vector regressor), figure 10(Extra tree regressor) and figure 11(Ada boost regressor) show the comparison graph between the actual and the predicted CPU utilization which varies their value over time. Each data point also represents a specific

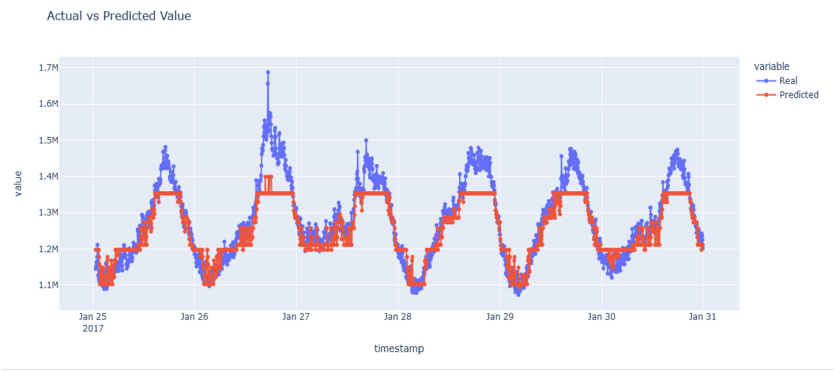

Figure 10: Extra Tree Regressor - Actual vs Predicted Graph

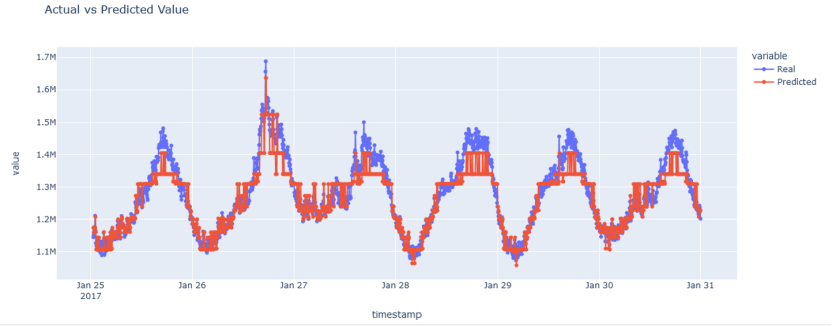

Figure 11: AdaBoost Regressor - Actual vs Predicted Graph

type of timestamp which shows the actual CPU utilization values, which means Real alongside the corresponding predicted values, which are Predicted and generated by the ML model.

Table 2 provides a comparison of the Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R2) score for each regression model

1. **Mean Squared Error (MSE):** MSE is a measure of the average squared difference between the actual and predicted values in a regression problem.Pain (2023)

2. **Root Mean Squared Error (RMSE):** RMSE is the square root of the MSE, providing a measure of the average magnitude of the residuals between predicted and actual values. Pain (2023)

3. **R-squared (R2):** R2 is a statistical measure that represents the proportion of variance explained by the independent variables in a regression model.Pain (2023)

Table 2: Comparative Analysis of Regression Models for Predicting CPU Utilization in Cloud Environments

Model	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)	R-squared (R2) Score
Support Vector Regressor (SVR)	0.0023503667430643947	0.04848058109247861	0.8644565966329448
Extra Trees Regressor	0.003611212476841574	0.060093364665673145	0.7917448284030331
AdaBoost Regressor	0.002473293106314581	0.04973221396956485	0.8573675507690889
Stacking Regressor	0.0010804266354079582	0.032869843860413424	0.9376928287112004

6.1 Discussion

Based on the evaluation metrics provided for each regressor model:

Support Vector Regressor (SVR): Achieves an MSE of 0.00235 and an RMSE of 0.0485, indicating relatively low prediction errors.

The R2 score of 0.8645 suggests that the SVR model explains approximately 86.45% of the variance in the data, indicating a good fit.

Extra Tree Regressor: Exhibits a higher MSE of 0.00361 and a slightly higher RMSE of 0.0601 compared to SVR. The R2 score of 0.7917 indicates that the Extra Tree model explains approximately 79.17% of the variance, showing a slightly lower fit than SVR.

AdaBoost Regressor: Shows an MSE of 0.00247 and an RMSE of 0.0497, similar to the SVR model. With an R2 score of 0.8574, it explains around 85.74% of the variance, indicating a good fit similar to SVR.

Stacking Regressor: Displays the lowest MSE of 0.00108 and the lowest RMSE of 0.0329 among all models, indicating superior predictive performance. The highest R2 score of 0.9377 suggests that the Stacking model explains approximately 93.77% of the variance in the data, indicating the best fit among the evaluated models.

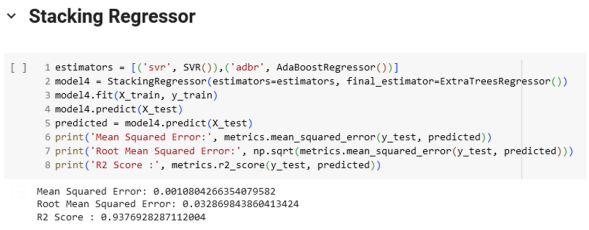

Figure 12: Stacking regressor R2 score

While all models perform reasonably well, the Stacking Regressor stands out with the lowest errors and highest R2 score, indicating superior predictive performance for forecasting CPU utilization.

7 Conclusion and Future Work

7.1 Conclusions

In conclusion, forecasting the CPU utilization for the dynamic resource allocation in the cloud. This study has collected the data from Microsoft Azure which has been analysed and processed and has done some preprocessing steps which include scaling up of the data using Min-Max Normalization and also Window Rolling With 6 Previous Values. These techniques enhanced the quality of the data and facilitated more accurate predictions. Various time series and forecasting algorithms have been used in this study, including Support Vector Regressor, Extra Trees Regressor, AdaBoost Regressor, and Stacking Regressor, were trained and evaluated using metrics such as Mean Squared Error, Root Mean Squared Error, and R-squared Score. The evaluation's findings demonstrated the efficacy of the Stacking Regressor model, which demonstrated the best predictive performance with the lowest MSE, RMSE, and greatest R-squared score. The study also emphasized how important it is to allocate resources dynamically depending on anticipated CPU consumption to maximize resource utilization and guarantee effective cloud system functioning. The study also stressed the significance of ongoing assessment and monitoring to guarantee the correctness and dependability of the prediction models used in cloud environments.

7.2 Future Works

Several limitations have been identified that could be addressed to achieve better results in future research. First off, the study's inability to be applied to other cloud platforms or environments with distinct architectures and workload characteristics may stem from its reliance on historical data from Microsoft Azure. To ensure that predictive models are strong and have a wider range of applications, future research might examine a variety of datasets from other cloud providers. Second, there's a chance that the preprocessing methods used—like Min-Max Normalization and Window Rolling with Six Previous Values—may not be able to fully capture intricate patterns and relationships in the data. Using more sophisticated feature engineering or preprocessing approaches could increase the quality of the input data and the performance of the model.

Furthermore, although the study assessed several machine learning models, such as the Support Vector Regressor, Extra Trees Regressor, AdaBoost Regressor, and Stacking Regressor, it did not investigate any more sophisticated algorithms or ensemble techniques that would have produced superior outcomes. Subsequent studies may look into innovative techniques and algorithms to improve prediction robustness and evaluation metrics even more. Various advanced algorithms can be used in future including Long Short-Term Memory (LSTM) networks which is a type of type of recurrent neural network (RNN) specifically designed to capture temporal dependencies in sequential data. Gradient Boosting, XGBoost and LightGBM can also be used to achieve superior performance and better efficiency in forecasting CPU utilization on the cloud.

References

(n.d.).
URL: *https://commons.wikimedia.org/wiki/File:CRISP-DM_Process_Diagram.png*

Araújo, T., Silva, L., Aguiar, A. and Moreira, A. (2023). Calibration assessment of low-cost carbon dioxide sensors using the extremely randomized trees algorithm, *Sensors* **23**(13): 6153.

Azure (n.d.). Github - azure/azurepublicdataset: Microsoft azure traces.
URL: *https://github.com/Azure/AzurePublicDataset*

Husaini, H., Misbullah, A. and Farsiah, L. (2023). A threshold-based cloud resource allocation framework with quality of services considerations, *Transcendent Journal of Mathematics and Applications* **2**(1): 9–19.

Jiang, W., Chen, Z., Xiang, Y., Shao, D., Ma, L. and Zhang, J. (2019). Ssem: A novel self-adaptive stacking ensemble model for classification, *IEEE Access* **7**: 120337–120349.

Karim, M. E., Maswood, M. M. S., Das, S. and Alharbi, A. G. (2021). Bhyprec: a novel bi-lstm based hybrid recurrent neural network model to predict the cpu workload of cloud virtual machine, *IEEE Access* **9**: 131476–131495.

Khan, A. A., Kumar, S. et al. (2024). A review on fixed threshold based and adaptive threshold based auto-scaling techniques in cloud computing, *MATEC Web of Conferences*, Vol. 392, EDP Sciences, p. 01115.

Khan, T., Tian, W., Zhou, G., Ilager, S., Gong, M. and Buyya, R. (2022). Machine learning (ml)-centric resource management in cloud computing: A review and future directions, *Journal of Network and Computer Applications* **204**: 103405.

Min, H. and Luo, X. (2016). Calibration of soft sensor by using just-in-time modeling and adaboost learning method, *Chinese journal of chemical engineering* **24**(8): 1038–1046.

Mishra, S. K., Sahoo, B. and Parida, P. P. (2020). Load balancing in cloud computing: a big picture, *Journal of King Saud University-Computer and Information Sciences* **32**(2): 149–158.

Nashold, L. and Krishnan, R. (2020). Using lstm and sarima models to forecast cluster cpu usage, *arXiv preprint arXiv:2007.08092* .

Oladoja, I., Adewale, O., Oluwadare, S. and Oyekanmi, E. (2021). A thresholdbased tournament resource allocation in cloud computing environment, *Asian Journal of Research in Computer Science* pp. 1–13.

Ouhame, S., Hadi, Y. and Ullah, A. (2021). An efficient forecasting approach for resource utilization in cloud data center using cnn-lstm model, *Neural Computing and Applications* **33**(16): 10043–10055.

Pain, A. (2023). Understanding evaluation metrics in machine learning: R-squared, adjusted r-squared, mse, mae, and rmse.
URL: *https://www.linkedin.com/pulse/understanding-evaluation-metrics-machine-learning-r-squared-pain*

Patel, E. and Kushwaha, D. S. (2022). A hybrid cnn-lstm model for predicting server load in cloud computing, *The Journal of Supercomputing* **78**(8): 1–30.

Prasad, V. K., Dansana, D., Bhavsar, M. D., Acharya, B., Gerogiannis, V. C. and Kanavos, A. (2023). Efficient resource utilization in iot and cloud computing, *Information* **14**(11): 619.

Qin, X., Li, B. and Ying, L. (2022). Efficient distributed threshold-based offloading for large-scale mobile cloud computing, *IEEE/ACM Transactions on Networking* **31**(1): 308–321.

Rehman, A. U., Ahmad, Z., Jehangiri, A. I., Ala'Anzy, M. A., Othman, M., Umar, A. I. and Ahmad, J. (2020). Dynamic energy-efficient resource allocation strategy for load balancing in fog environment, *IEEE Access* **8**: 199829–199839.

Rotter, C. and Van Do, T. (2021). A queueing model for threshold-based scaling of upf instances in 5g core, *IEEE Access* **9**: 81443–81453.

Shafiq, D. A., Jhanjhi, N. Z., Abdullah, A. and Alzain, M. A. (2021). A load balancing algorithm for the data centres to optimize cloud computing applications, *IEEE Access* **9**: 41731–41744.

Thomas, S., Pillai, G. N. and Pal, K. (2017). Prediction of peak ground acceleration using -svr, ν-svr and ls-svr algorithm, *Geomatics, Natural Hazards and Risk* **8**(2): 177–193.