Volume 14, Issue 2, Page 52-63, 2025 https://doi.org/10.46810/tdfd.1545596 Research Article

BİNGÖL UNIVERSITY

Türk Doğa ve Fen Dergisi
Turkish Journal of Nature and Science
www.dergipark.gov.tr/tdfd

HealthCraft: A Precision Model for Smart Resource Optimisation in Dynamic Big Data Healthcare Environments

Ümit DEMİRBAGA[1*]

[1] Bartın University, Engineering, Architecture and Design Faculty, Computer Engineering Department, Bartın, Türkiye
Ümit DEMİRBAGA ORCID No: 0000-0001-5159-0723

**Corresponding author: udemirbaga@bartin.edu.tr*

(Received: 10.09.2024, Accepted: 12.04.2025, Online Publication: 27.06.2025)

Keywords
Big data, Cloud computing, Machine learning, Resource utilisation, Prediction

Abstract: Cloud computing offers scalable computing and storage capabilities to handle massive healthcare data. When processing large-scale data, keeping the resource cost reasonable is crucial. Nonetheless, resource utilisation is frequently inefficient because of the inherent complexity and heterogeneity of distributed computing frameworks. In addition, it is challenging to model resource utilisation from real fault-occurring cloud systems. This study proposes an automated online resource utilisation prediction model that combines machine learning (ML) methods with automated log data preprocessing to predict future resource consumption. It allows smart and adaptable allocation of resources in large cloud-based data infrastructures suffering from typical failures like CPU, memory, network, and data locality problems. Using the Hadoop framework on a cloud cluster of 30 worker nodes, our model predicts resource utilisation with up to 97.3% accuracy - outperforming the other baseline models evaluated. In addition, our system accurately recognises resource bottlenecks. It reduces execution time by up to 30%, even in fault-injected environments, implying that it is robust enough for real-time big data analytics.

HealthCraft: Dinamik Büyük Veri Sağlık Hizmetleri Ortamlarında Akıllı Kaynak Optimizasyonu için Hassas Bir Model

Anahtar Kelimeler
Büyük veri, Bulut bilişim, Makine öğrenimi, Kaynak kullanımı, Tahmin

Öz: Bulut bilişim, büyük ölçekli sağlık verilerini işlemek için ölçeklenebilir hesaplama ve depolama yetenekleri sunar. Büyük ölçekli verilerin işlenmesi sırasında, kaynak maliyetini makul seviyede tutmak kritik öneme sahiptir. Bununla birlikte, dağıtık bilişim çerçevelerinin doğasında bulunan karmaşıklık ve heterojenlik nedeniyle kaynak kullanımı sıklıkla verimsiz olmaktadır. Ayrıca, gerçek hata oluşan bulut sistemlerinden kaynak kullanımını modellemek zorlu bir süreçtir. Bu çalışma, otomatik bir çevrimiçi kaynak kullanım tahmin modeli önererek, makine öğrenimi (ML) yöntemlerini otomatik günlük veri ön işleme ile birleştirerek gelecekteki kaynak tüketimini tahmin etmektedir. Önerilen model, CPU, bellek, ağ ve veri yerelliği sorunları gibi tipik arızalardan etkilenen büyük ölçekli bulut tabanlı veri altyapılarında akıllı ve uyarlanabilir kaynak tahsisini mümkün kılmaktadır. Hadoop çerçevesini kullanarak 30 çalışan düğümden oluşan bir bulut kümesinde, modelimiz kaynak kullanımını %97,3'e varan doğruluk oranıyla tahmin etmektedir ve karşılaştırılan diğer temel modellerden daha üstün performans göstermektedir. Ayrıca, sistemimiz kaynak darboğazlarını doğru bir şekilde tespit etmekte ve hata enjekte edilmiş ortamlar dahil olmak üzere çalışma süresini %30'a kadar azaltmaktadır, bu da onu gerçek zamanlı büyük veri analitiği için yeterince sağlam bir çözüm haline getirmektedir.

1. INTRODUCTION

With the increasing data and high internet speed, industries are using cloud computing more widely to meet their data analysis demands. The development of omics sciences like genomics, proteomics, and metabolomics has led to the collection of enormous volumes of data [1]. Data expansion is facilitated by the transition from paper medical records to electronic health records (EHR) [2]. Using such detailed and extensive data, doctors,

epidemiologists, and health policy experts seek to enhance population health and patient care. To properly exploit the created big data within a reasonable time, it is necessary to utilise the system resources efficiently. However, the dependent structure of cloud-based big data frameworks and the complex infrastructure of cloud computing consisting of servers, storage devices, networks, cloud management software and virtualisation result in inefficient use of system resources and loss of time and energy.

Resource utilisation prediction in big data systems estimates the resource requirement required for the successful completion of the application under current conditions, which improves resource utilisation, reduces costs and enhances performance. Statistical techniques and ML methods are the most common to estimate resource utilisation. In statistical methods (e.g. Automatic Regression Modelling), the relationship between variables is determined, and probability distributions are used for assumptions. However, these methods are very general because they make predictions for certain intervals and cannot be estimated accurately [3]. ML models, such as Support Vector Machine (SVM) [4], Genetic Algorithm (GA) [5], and Neural Network (NN) [6], are more widely adopted by researchers to perform a more accurate prediction. Researchers perform resource utilisation predictions of big data systems in simulators and online systems. The simulation models [7-9] create a virtual model by modelling the computer system mathematically and revealing the system controls. Online performance analysis systems [10-12] enable the collection of performance metrics through system logs, providing much more accurate and precise information. However, these technologies cannot offer predictive support for resource usage in environments with different failures. Moreover, most of these studies employ ML techniques to forecast future resource consumption using the historical traces of a data centre as their input. Despite the impressive outcomes of ML-based models, there are still certain limitations. Most models lack a precise method for handling workload non-stationarity. The interactions between characteristics of various sizes should be taken into account. For example, Feedforward Neural Networks (FNN), a part of a multi-layer perceptron, can be used as an efficient approach for dealing with complicated nonlinear systems since they inherit the learning capabilities of neural network models and the inference capabilities of fuzzy systems [13]. As a result, several researchers have built sophisticated controllers and represented complex plants using FNN techniques [14-15].

Considering the above analysis, we propose a novel, robust resource utilisation prediction model for cloud-based big data systems that simultaneously handles multiple resource sources under different fault conditions. Unlike existing models focusing primarily on CPU or memory usage, our approach integrates a multi-resource machine learning framework that simultaneously considers CPU, memory, network, and data locality constraints, enabling more accurate and adaptable resource provisioning. Our approach combines an automated log data preprocessing module, a feature extraction pipeline, and multi-resource machine learning models to predict future resource consumption accurately. We introduce a fault injection-based training mechanism to enhance its adaptability to real-world conditions, allowing the model to learn and generalise from system anomalies.

As a first step, an automated data-driven pipeline is proposed to move the raw data of multiple runs into a form suitable for predicting the resource utilisation of the big data cluster. Finally, the fine-tuned ML Models are used to predict upcoming resource utilisation based on historical system logs and injected fault conditions. We evaluate our model under realistic conditions by injecting four representative faults (CPU, memory, network and data locality) that frequently happen in big data systems. That way, we can test its performance when the environment changes dynamically.

The contributions of this paper can be summarised as follows:

- We introduce a fully automated pipeline to preprocess log data that improves data quality and minimises manual feature engineering.
- We design a multi-resource prediction model, jointly considering CPU, memory, network and data locality factors for more adaptive provisioning of cloud resources.
- We also introduce a fault injection mechanism, which allows the model to be trained and evaluated during real-world faults, greatly improving robustness and accuracy.
- We achieve a 97.3% prediction accuracy, considerably outperforming traditional ML-based resource utilisation models.

2. BACKGROUND

2.1. Navigating EHR Datasets

Significant developments have fuelled the ongoing evolution of data in the dynamic landscape of omics domains, such as proteomics, metabolomics, and genomics [16]. The transition from traditional paper medical records to Electronic Health Records (EHR) highlights this change even more [17]. Due to the ensuing explosion of large-scale healthcare data, healthcare professionals—from doctors and epidemiologists to specialists in health policy—have a critical chance to make well-informed decisions that will eventually improve population health and enhance patient care [18]. Therefore, building strong tools, infrastructure, and methodologies is imperative to fully utilise big data's promise. To this end, the National Early Warning Score (NEWS) was developed in the United Kingdom to identify and treat patients with acute diseases when their clinical condition worsens [19] [20].

2.1.1. Overview of the NEWS dataset

Patients with acute illnesses have seven physiological parameters measured during clinical evaluation: respiration rate, oxygen saturation, temperature, systolic blood pressure, heart rate, consciousness, and oxygen. The NEWS, ranging from 1 to 8 and representing the severity of the illness, is calculated through the following steps: Step 1: Acquiring the patient's score on each of the seven physiological parameters. Step 2: Adding together all of the physiological parameter scores to calculate the NEWS. Step 3: Checking if a single parameter has reached the trigger threshold. We use the NEWS data[*] generated based on the structure and values taken from South Tees Hospitals NHS Foundation Trust[†] . This allows us to confirm the effectiveness and dependability of the proposed system. In addition to the seven distinct physiological indicators, the dataset includes 52 other factors about the health state of the patients, such as weight, dates of admission, duration of hospital stay, pain score, nausea, vomiting, and pulse. An urgent clinical review should be triggered by an aggregate NEWS of 5 or 6, and a NEWS of 7 or above should trigger a high-level clinical alert or emergency clinical review. An "Urgent or emergency reaction" response is appropriate for NEWS values more than 7, indicating "High clinical risk".

2.2. MapReduce Paradigm for Big Healthcare Data

Big data in the healthcare industry refers to heterogeneous, multi-spectral, incomplete, and uncertain observations from primary sources that are presented in structured, semi-structured, and unstructured formats and include observations related to diagnosis, illness, injury, treatment, physical and mental disorders, demography, and disease prevention [21]. Unstructured data comprises medical imaging, notes, environmental, clinical, lifestyle, medication, and health economics data. Structured data contains ICD codes, phenotype, genotype, and genomic information [22]. In addition, the Internet of Things (IoT) has spurred the development of data-driven applications in industries, including transportation, networking, smart cities, and healthcare. Various sensors and devices to track a patient's health have been widely used in the health sector. MapReduce, a distributed programming model, is implemented in Apache Hadoop[‡] and Apache Spark[§] frameworks to analyse petabyte-scale datasets in a parallel manner on computer clusters [23]. The data is divided into smaller chunks using MapReduce, which then turns each piece into a set of tuples known as key-value pairs before eventually reducing these tuples. MapReduce consists of two main phases, namely Map and Reduce, which are carried out by the workers and generate key-value pairs. During the shuffle phase, these pairs are grouped by key and sent to the appropriate Reducer. Subsequently, Reducer groups the key-value pairs and constructs a smaller collection of data tuples from these tuples. The final outputs are ultimately produced and stored in the Hadoop Distributed File System (HDFS). Fig. 1 depicts the working principle of the MapReduce paradigm.

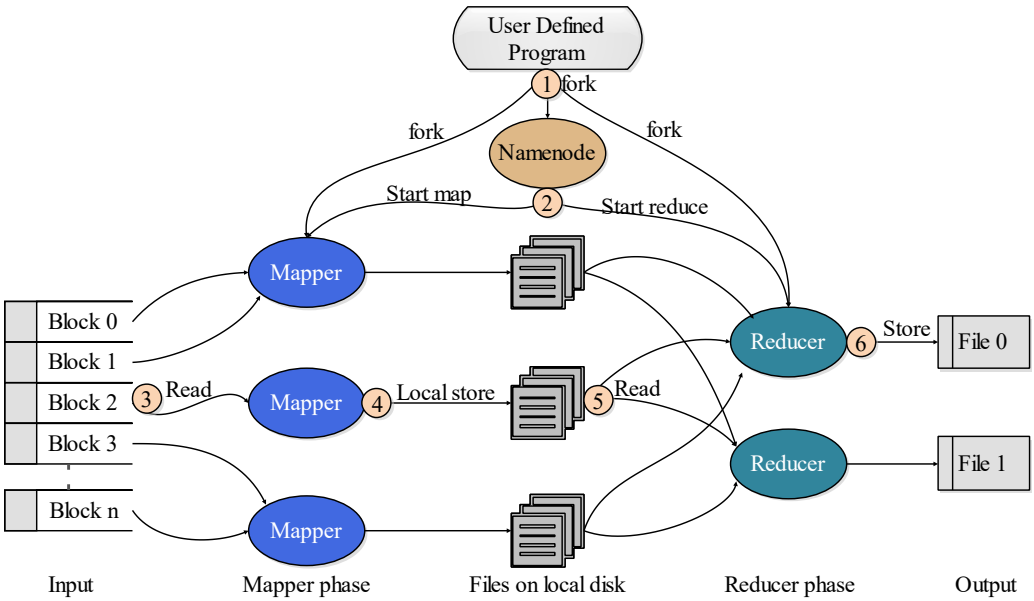


Figure 1. Overview of the MapReduce paradigm: Data processing workflow

2.3. Resource Utilisation Prediction

A resource utilisation prediction model estimates how much resource the system will require in the future to evaluate the system's performance and assess whether the system has adequate resources for users' demands. Resource utilisation prediction is adopted in diverse applications [24-26]. Demand and resource consumption estimates for large-scale data analysis are used to customise workload predictive resource management systems. The workloads, especially streaming data, are dynamic and constantly changing for big data systems [27]. Consequently, a robust and scalable resource

[*] https://github.com/umitdemirbaga/NEWS
[†] https://www.southtees.nhs.uk/
[‡] https://hadoop.apache.org/
[§] https://spark.apache.org/

utilisation prediction model is required to provide reasonable prediction accuracy. Conversely, resource utilisation prediction enables access to free resources and analyses the effects of allocating them to specific workloads [28]. Resource utilisation for predicting the system performance was first used in [29]. A few research studies then created a framework for measuring system performance based on usage patterns and evolution trends in large-scale datasets. Mean Field Theory (MFT), which studies the behaviour of high-dimensional complex systems, has been used to successfully predict and evaluate the performance of highly complex structures such as Hadoop applications [30].

3. PROPOSED SYSTEM

This section introduces our novel resource utilisation prediction model for big data systems. Fig. 2 presents the high-level architecture of the proposed approach for big healthcare data processing, which fundamentally consists of three main components: a big data monitoring system, a fault injection module, and a resource utilisation prediction system.

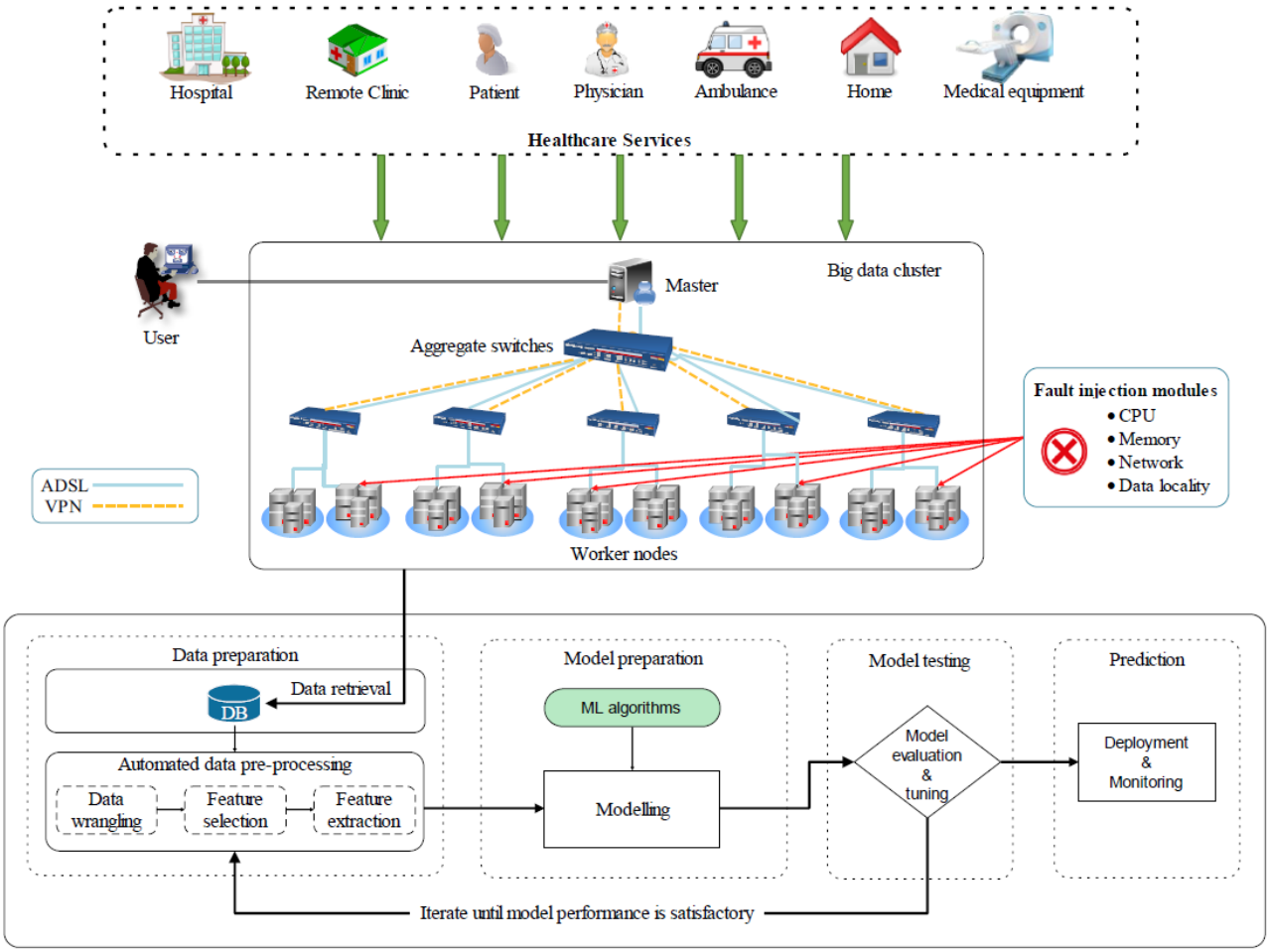


Figure 2. The overview of the proposed system's architecture

3.1. Monitoring Big Data Cluster for Log Collection

In this work, the Apache Hadoop framework is deployed on AWS (Amazon Web Services) EC2 instances, and MapReduce is used as a programming model for processing big healthcare data. We deployed SmartMonit [31], a real-time big data monitoring system, to collect tasks and infrastructure information from the Hadoop cluster in real-time. As shown in Fig. 3, SmartMonit has two sub-agents, TaskAgent and SystemAgent, managed by SmartAgent. TaskAgent employs the ResourceManager REST API[**] to collect the execution status of each task, and SystemAgent employs the Sigar API[††] to gather computing resource metrics, such as CPU/memory utilisation, network throughput, and disk I/O speeds. SmartAgent is responsible for receiving raw data from these two agents and storing it in a time-series database, InfluxDB[‡‡], via RabbitMQ[§§], a message broker system, for further processing.

[**] https://hadoop.apache.org/docs/r3.2.4/hadoop-yarn
[††] https://github.com/hyperic/sigar
[‡‡] https://www.influxdata.com
[§§] https://www.rabbitmq.com/

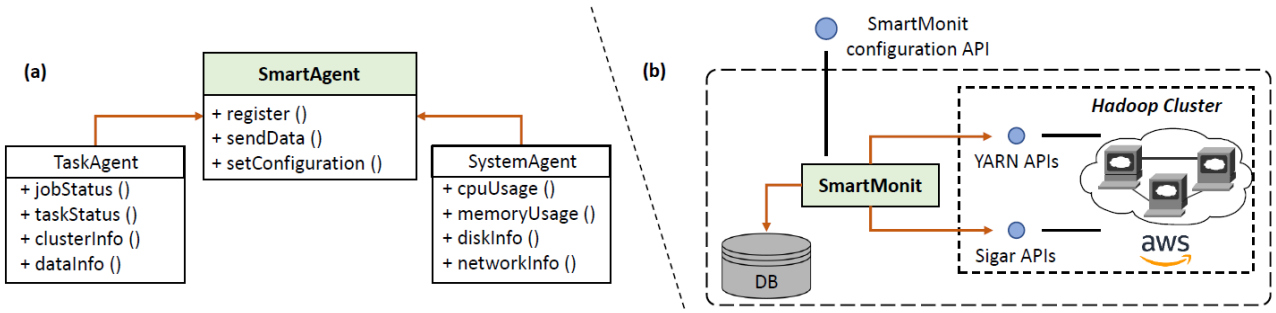


Figure 3. Model of monitoring agents (a); SmartMonit deployment in the proposed system (b)

3.2. Automated Data Pre-Processing

This data analysis pipeline module retrieves time-stamped raw logs stored in a time series database and prepares them for the next phase, implementing ML algorithms. The automated data pre-processing module conceptually consists of three main sections: data wrangling, feature extraction, and feature selection.

3.2.1. Data wrangling

This module includes a comprehensive set of data processing procedures to convert raw data collected from the cloud-based big data cluster into more readily used formats, including cleaning, organising, structuring, and enriching.

- **Regression imputation method for automatic missing value imputation:** The log collector forwards the collected data simultaneously from different APIs to the database or the next step for processing. Due to network congestion, a temporary outage of the Application Programming Interface (API), or an issue with the API itself, API response timeouts occur, in which the log collection process is interrupted. In addition, missing values occur in data sets for other reasons, such as storage limitations, security filters, data loss during data collection or the impossibility of measuring. Missing values in the dataset affect the analysis results, making it difficult to draw meaningful conclusions. To handle this, we propose an automatic missing value imputation to estimate and complete the missing values using the regression imputation method due to collecting large amounts of logs and the strong relationships between the variables. We implemented the regression imputation method by deploying the scikit-learn library in Python.

- **Automatic data encoding:** In this module, we combine one-hot encoding and label encoding techniques to convert categorical or serial data into numerical data that ML algorithms can easily process. Extensive logs are collected from the cloud-based Hadoop cluster deployed on AWS, including numeric and label data (e.g. configuration parameters). Therefore, using the label encoding technique, these parameters are converted into digital forms and made understandable by the machine.

3.2.2. Feature selection

Most of the relevant features are selected in this module to help improve the performance of our ML models. To this end, we implement the correlation analysis method, which measures the linear relationship between each pair of features. The implementation details are discussed in §4.4. Evaluation of Resource Utilisation Prediction.

3.2.3. Feature extraction

This module generates new features based on input data to identify the most significant linear combinations of features. The main objective is to reduce the dimensionality of feature vectors and categorise raw data into distinct groups by facilitating the interpretability and management of the data by ML models. To achieve this, we employ Principal Component Analysis (PCA) in Python, which helps create a new feature to specify the resource utilisation levels, ranging from 0 to 2. These steps are performed by following the steps below:

- **PCA implementation:** PCA is a dimensionality reduction technique that transforms the original set of features into a new set of orthogonal components, known as principal components. In our implementation, PCA efficiently captures the variance within the log data, which enables us to pinpoint the essential features that contribute significantly to the overall resource utilisation patterns. By retaining the principal components that capture the maximum variance, we effectively reduce the dimensionality of the data while preserving the crucial information necessary for accurate resource utilisation prediction.

- **Creation of new feature:** Based on the outcomes of PCA, a new feature is constructed to encapsulate the resource utilisation levels. The determination of these levels is as follows:

  - 0: Low Resource Utilisation
  - 1: Average Resource Utilisation
  - 2: Highest Resource Utilisation

These explicit explanations aim to comprehensively understand the methodology employed in setting and labelling the resource utilisation levels for subsequent predictions.

3.3. Fault Injection Modules

We have developed four fault injection models to experience real design problems in big data systems, which helps to test and improve the reliability and resilience of our proposed system in highly possible fault conditions, such as insufficient CPU and memory resources, network congestion or low bandwidth, and data locality issue in distributed systems, as indicated in our previous work [12].

The fault injection process involves generating realistic workloads and injecting faults into the VMs within a cluster. The algorithm takes into account various parameters, including VM ID (V_m), cluster (C), fault type (F), list of faults (F_l), injection duration (T_m), time interval (T_i), and workload (W_l).

3.3.1. CPU fault injection module

This module dynamically generates workload for the CPU. A set of computations is executed in parallel to increase the number of MIPS (million instructions per second) in the selected VMs. This model creates a high-dimensional Pascal's triangle, where each number is the sum of the two numbers just above it.

3.3.2. Memory fault injection module

It is designed to intentionally induce memory occupancy in the nodes of a computer cluster by creating vector objects in the memory, which enables monitoring the execution times of applications and making accurate and robust resource usage predictions when out-of-memory conditions occur.

3.3.3. Network fault injection module

Intermediate key-value pairs generated during the Map phase are distributed to the relevant nodes to be combined in the Reduce phase. At this stage, the network is overloaded, and delays occur. To improve data flow in MapReduce, Hadoop employs several strategies, including data compression, speculative execution, and pipelining. However, none of these can prevent the transfer of large data sets over the network. Since speculative applications cause data blocks to be transferred entirely to other nodes, delays are experienced. To experience this, the network fault injection module reduces bandwidth by transferring data between nodes, which creates delays similar to those seen in the Hadoop system.

3.3.4. Data locality fault injection module

The ability of a MapReduce job to process data on the same node where the data is stored is known as data locality. Moving data over the network imposes some overheads that lead to delays if a task is assigned to the node where the task's input data is not stored. To create a data locality issue, this module deletes the data blocks of the selected nodes, which causes data transfer from the other nodes that have a copy of the deleted blocks.

Algorithm 1: Fault injection into big data clusters

Input: - V_m: *VM id,*
- *C: cluster,*
- $\mathcal{G}$: *fault type,*
- F_l: *list of faults,*
- T_m: *injection duration,*
- T_i: *time interval,*
- W_l: *workload.*

1 *// Initiate the process of fault injection*
2 ***for*** *selected* V_m *in C* ***do***
3 *// Generate* W_l
4 ***GenerateWorkload***(W_l)
5 *// Execute injection method*
6 $inj \leftarrow$ ***Assign***$(\mathcal{G}, T_m)$
7 *// Inject into the C*
8 $C \leftarrow$ ***Inject***(inj)
9 *// Interrupt the fault injection*
10 ***sleep***(T_i)
11 ***end for***

Algorithm 1 demonstrates the pseudocode of the fault injection modules for big data clusters deployed on cloud environments. First, the user identifies V_m and F; then, based on the fault type, the workload is generated (see Algorithm 1, Line 4). Afterwards, the fault is injected for a predetermined time (T_m) (see Algorithm 1, Line 6). The generated workload is finally loaded into the selected VM (see Algorithm 1, Line 8). This process is suspended once the resource utilisation reaches 90%.

3.4. Resource Utilisation Prediction for Big Data Systems

To forecast the resource utilisation of big data systems, we implement six well-known ML algorithms, namely Support Vector Classifier (SVC), Gaussian Naïve Bayes, Logistic Regression, K-Nearest Neighbors (KNN), Decision Tree, and Random Forest. All these algorithms are evaluated and compared, considering accuracy rates, both online and offline analysis.

We propose an ML-based model with a pipeline that includes automated log data preprocessing, feature extraction, and multi-resource predictive modelling. This model is trained on labelled resource utilisation logs with supervised learning methods such as Decision Trees, k-nearest Neighbors (k-NN), and Random Forest classifiers. This is done so that the fault injection module can help the model learn from real-world failure scenarios and try to make the model robust to rare, unpredictable system anomalies. In contrast to traditional models that consider resource consumption in isolation, we model inter-relations between CPU, memory, network, and data locality constraints to allow for more adaptive and accurate resource provisioning.

3.5. Fault Injection and Testing

The four fault scenarios (CPU, memory, network, and data locality) were chosen because they are commonly encountered in real-world cloud computing and big data systems [12]. CPU overload failures are common in distributed computing workloads, where tasks surpass processing capacity due to high demand and end up degrading system performance or forcing tasks to fail. Memory leaks or too much memory usage can be a

common problem across long-running big data applications, leading to system slowdowns or crashes. Network congestion or packet loss impacts data-intensive applications, and the momentum is even higher in distributed computing frameworks like Apache Spark and Hadoop. Data locality in cloud computing refers to when a task is scheduled on a compute node that is remote from the data it needs to consume, which can greatly increase the cost of I/O and result in a waste of compute resources. These types of failures are well-known in other real-world failures of cloud and big data systems, which are incorporated into our model to ensure our approach is not formulated against a trivial model but rather in facing real-world challenges in large-scale computing environments.

4. RESULTS AND DISCUSSION

This section presents a comprehensive evaluation of the proposed system, including experimental setup and evaluation of resource utilisation prediction.

4.1. Experimental Setup

4.1.1. Environments

To evaluate our system, we set up a Hadoop cluster on AWS, consisting of 1 master and 30 worker nodes, each with 4 cores, 16 GB of memory, and 1TB of SSD storage. We chose Ubuntu Server[***] 20.04 LTS and SSD Volume Type as an operating system, and deployed Apache Hadoop 3.2.4[†††] and Apache Hive[‡‡‡] 3.1.3 versions.

4.1.2. Benchmarks and workload

We use the NEWS data detailed in §2.1. Navigating EHR datasets to evaluate the proposed system, taken from South Tees Hospitals NHS Foundation Trust[§§§] . The data includes 1M patients' healthcare data, including 552 different features, such as pulse, age, sex, nausea, given drugs, weight, length of stay, discharged deceased, etc. The data size for bencmarking the proposed system is 35 GB of healthcare data.

4.1.3. Methodology

Our experiments aim to evaluate the performance and accuracy of the proposed system. To this end, we inject four common faults separately, enabling data analysis under possible scenarios. We perform Hive queries in parallel over the dataset with five repetitions, comprising 25 experiments, including data processing without any faults.

4.2. Hyperparameter Tuning

We train the ML models using default or initial hyperparameters to establish a baseline performance. Following this, we employ a systematic Grid Search approach to optimise hyperparameters for each model, such as maximum depth (*max_depth*) and minimum samples split (*min_samples_split*) in Decision Trees, and the regularisation parameter (*C*) and kernel in SVC while the number of trees (*n_trees*), maximum depth (*max_depth*), and minimum samples split (*min_samples_split*) parameters are tuned for Random Forest model. Once the optimal hyperparameters are identified, the models are re-trained to enhance their capabilities in predicting resource utilisation within cloud-based big data systems.

After conducting hyperparameter tuning, the optimal values for each model were selected based on validation performance. The final hyperparameter values used in our experiments are summarised in Table 1.

Following this, we used Grid Search tuning[****] related to hyperparameters using the various data partitions from the cloud-based big data system logs to assess the tuned hyperparameters' generalisability. Subsets of varying workload patterns were used to train and validate the models to determine their adaptability. We found that optimal hyperparameter values were more or less stable across datasets. Still, some small variations were found in tuning parameters because models like Random Forest and KNN are sensitive to dataset distribution. The models remained highly predictive despite the changes, confirming our hyperparameter selection's stability and performance.

Although the Grid Search optimisation process makes improving our model performance in our study possible, it also applies to many other cloud computing scenarios. Hyperparameter tuning can be performed in real-time big data analytics using online learning approaches that continuously update model parameters as new data is received. Hyperparameter search strategies such as Bayesian Optimisation or Reinforcement Learning-based tuning could be adopted in edge-computing environments to reduce the overhead of exhaustive search methods, as edge devices do not possess unlimited computational resources. On the other hand, in heterogeneous workload scenarios, transfer learning methods can help provision previously optimised hyperparameters across workloads without starting the model training process from scratch. Such emphasises the wider relevance of hyperparameter optimisation beyond the narrative of this paper and leaves space for further research on adaptive tuning techniques.

[***] https://ubuntu.com/
[†††] https://hadoop.apache.org/release/3.2.4.html
[‡‡‡] https://hive.apache.org/
[§§§] https://www.southtees.nhs.uk/
[****] https://scikit-learn.org/stable/modules/grid_search.html

Table 1. Final hyperparameter values after tuning

Model	Hyperparameter	Value
Support Vector Classifier (SVC)	Kernel	RBF
	Regularization (C)	1.0
Gaussian Naïve Bayes	Smoothing factor	1e-9
Logistic Regression	Regularization (C)	0.01
	Solver	Lbfgs
KNN	Number of neighbors (K)	5
	Distance metric	Euclidean
Decision Tree	Max depth	10
	Minimum samples split	2
	Criterion	Gini
Random Forest	Number of trees	100
	Max depth	15
	Minimum samples split	4
	Criterion	Entropy

4.3. Cross-Validation

We conduct a 10-fold cross-validation process to ensure robust and unbiased performance assessment for the developed models. This process divides the dataset into ten equal-sized folds, where nine are used for training and one for validation in each iteration. The process is repeated for all ten folds, and the final reported results represent the average performance metrics (accuracy, precision, recall, F1-score) across all folds to ensure robustness and mitigate bias. In this way, we guarantee to provide more reliable and generalised ML models in predicting resource utilisation within cloud-based big data systems.

4.4. Evaluation of Resource Utilisation Prediction

This section provides a detailed explanation of the experimental evaluation conducted to assess the performance and reliability of the proposed system within real-world cloud computing environments. To this end, we processed the big healthcare dataset over a Hadoop cluster in AWS. We gather different log datasets with and without injection faults. With the help of the automated preprocessing system, logs are labelled, ranging from 0 to 2, based on resource utilisation levels. Fig. 4 demonstrates the data distribution of each stage for resource utilisation.

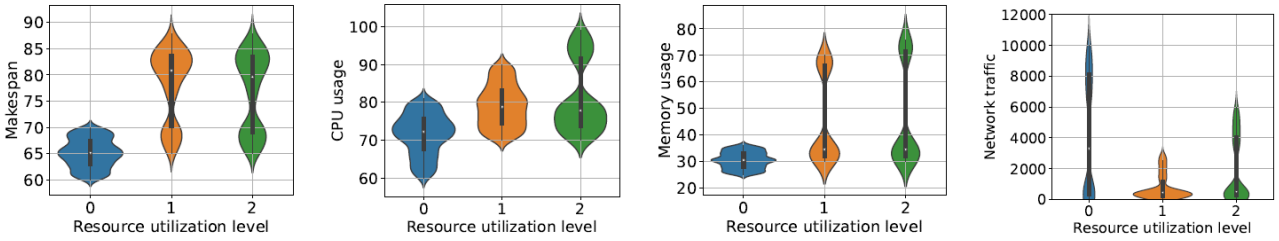
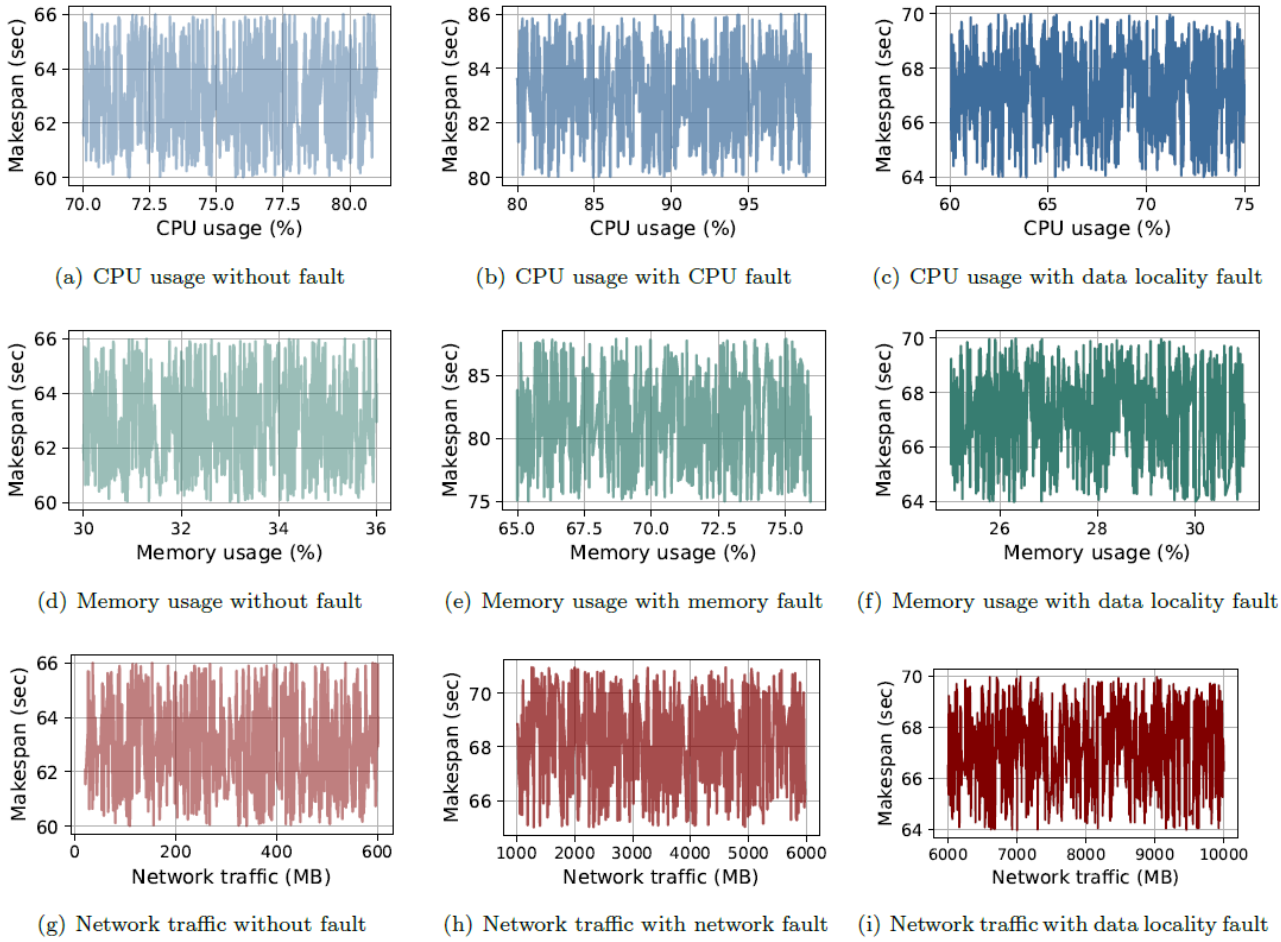


Figure 4. Training data distribution over the features based on resource utilisation levels

In the Hadoop ecosystem, tasks are distributed equally to all compute nodes by default, resulting in less powerful nodes (in terms of resources) needing more time to complete their tasks. To simulate the results of these issues in a heterogeneous big data system, we perform data processing under different scenarios, namely with and without faults. Fig. 5 shows the resource utilisation results for CPU, memory, and network over total execution time (makespan). As seen in Fig. 5(a), CPU utilisation is between 70% and 82%, and makespan varies from 60 to 66 seconds when there are no faults in the system. However, insufficient computation power of the node reaches 100%, increasing the makespan by around 30% (see Fig. 5(b)). Moreover, the data locality issue suspends data processing until the data transfer is complete, reducing CPU usage by 20% and increasing the makespan by 10% as seen in Fig. 6(c). The nodes with less memory resources than others in a heterogeneous big data cluster perform similarly. On nodes with insufficient memory, memory usage and makespan are inversely proportional (Fig. 5(d) and Fig. 5(e)). Moreover, Fig. 5(f) data locality issue decreases memory usage but increases makespan. Compared to Fig. 5(g) (under healthy conditions), Fig. 5(h) and Fig. 5(i) show that the makespan increases when the network usage increases and the number of executions increases dramatically when there is a data locality problem and network fault is injected to simulate the nodes with low bandwidth connections.

For a detailed analysis of our predictive system, we also apply accuracy, precision, recall, and F1-score as performance metrics (as shown in Fig. 6), which bring individual aspects to model assessment. Accuracy is the percentage of correctly classified instances concerning the total number of cases, giving us a descriptor of overall performance. In resource utilisation datasets, imbalanced classes are commonplace, so accuracy is not necessarily a good indicator. Precision is defined as the number of true positives divided by the sum of true positives and false positives, which helps to minimise false alarms. Recall (sensitivity), on the other hand, quantifies the precision of positive classification by determining the fraction of correctly identified adverse cases, which is

important for high-utilisation or failure-prone cases. That is where the F1-score, the harmonic mean of precision and recall, come into play, where a distinct balance is necessary between false positives and false negatives, which serve as a powerful metric in dynamic cloud-based big data environments.



(a) CPU usage without fault (b) CPU usage with CPU fault (c) CPU usage with data locality fault

(d) Memory usage without fault (e) Memory usage with memory fault (f) Memory usage with data locality fault

(g) Network traffic without fault (h) Network traffic with network fault (i) Network traffic with data locality fault

Figure 5. Makespan vs resource utilisation under different scenarios

The proposed algorithms show a high performance in the energy prediction per resource under several online and offline failure situations. Now, we get the most accurate model, k-NN at 97.30%, followed by the random forest at 95.10%. K-NN is relatively successful as it can detect complex patterns and associations in the dataset by evaluating the closeness of examples in the feature space. The degree of sample closeness in this approach is an effective way of presenting how the k-NN structure succeeds in defining nuanced relations. In contrast, it operates using an ensemble of decision trees, merging predictions of the multiple trees to form a comprehensive insight into feature-target connections. Although the accuracy is the same, both these models manage to learn muscle patterns relevant enough to be captured in the prediction, making them appropriate candidates for dependency to ensure the success of our predictive framework. Generally, accuracy is not always the appropriate metric to assess a model's performance, especially when the classes are unbalanced. Metrics like accuracy, recall, or F1 score, which offer a more complex understanding of a model's performance, are better suited to these circumstances. Hence, the reviewed dataset is probably properly balanced if the F1 score, recall, and accuracy are equal.

Our ML-based model provides a 97.3% accuracy rate and does much better than baseline approaches. Incorporating fault injection modules renders the model more resilient, automatically adapting to maintain a high prediction performance level in dynamic workload fluctuations. Our results show the effectiveness of our approach in a real cloud-based big data environment, efficiently handling resource fluctuations, and maintaining high prediction performance across different execution scenarios.

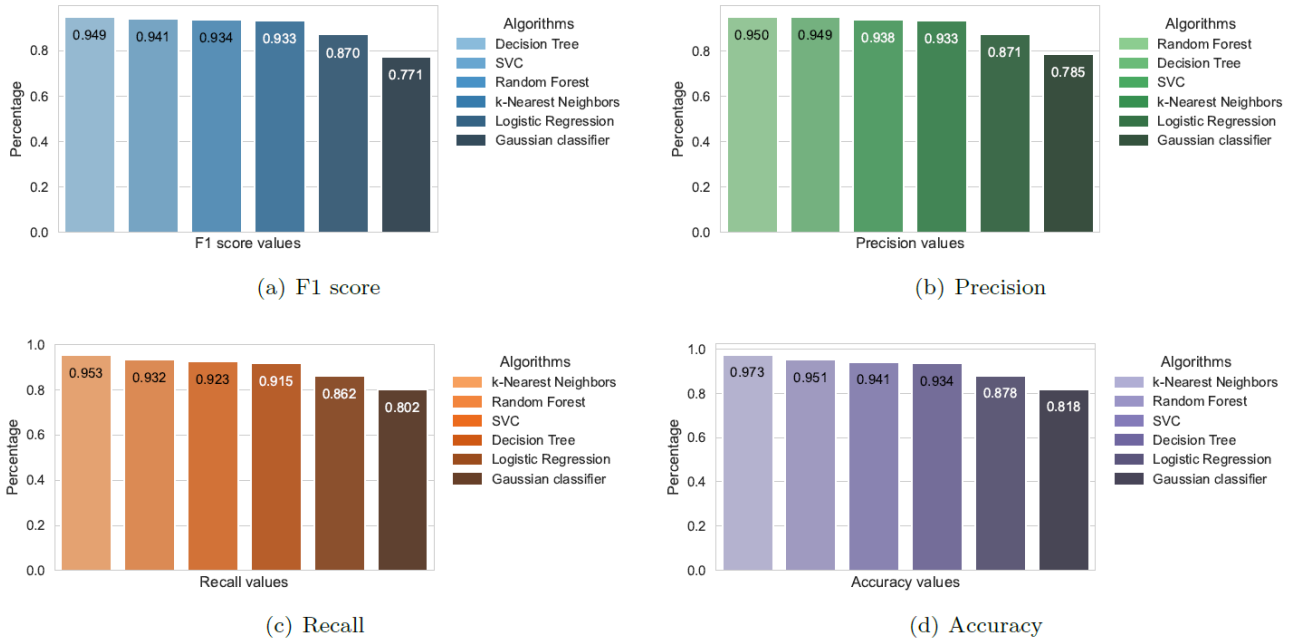


Figure 6 Comparison of performance evaluation metrics for ML algorithms.

5. RELATED WORK

In recent years, researchers have extensively studied resource utilisation prediction for cloud-based big data systems with different aspects, aims, and applications.

5.1. Resource Utilisation Prediction in Cloud Computing

Several recent works have engaged ML models to predict resource utilisation in cloud computing systems. Mehmood et al. developed a hybrid resource utilisation prediction model based on KNN and Decision Trees, resulting in better accuracy for workload prediction [32]. However, this model does not cater to fast-evolving cloud scenarios. Al-Asaly et al. developed a deep learning-based model for autonomic cloud computing environments, intentionally aiming to predict workload. Although their model outperformed traditional approaches, it is not based on real-time system monitoring and, therefore, was not as effective in dynamic cloud-based big data systems [33]. RL algorithms have recently been proposed to better provision cloud resources. For instance, Nguyen et al. developed an RL-based model for decentralised IoMT (Internet of Medical Things) networks that outperforms alternative solutions regarding resource allocation efficiency and reduced latency [38]. Gao et al. introduced a dynamic simulation budget allocation method, and an optimal computing budget allocation (OCBA) strategy was proposed to maximise resource utilisation efficiency [35]; however, this method is designed for passing in cloud environments.

5.2. Fault-Tolerant Resource Management in Big Data Systems

In recent years, fault-tolerant cloud computing strategies have gained much attention in the literature. Rahmanian et al. proposed an ensemble-based resource utilisation prediction model using Learning Automata, which uses the Single and Multiple Window concepts to train a resource utilisation prediction model. Nonetheless, this technique does not provide a real-time adjustment, as it is not accompanied by any live system observant [34]. Khan et al. surveyed blockchain-based edge computing frameworks, which can be applied in IoT applications, pointing out the benefits of using blockchain, including security protection and decentralised resource management [36]. Akbari Zarkesh et al. proposed EdgeLinker, a blockchain-based security mechanism designed for healthcare fog applications to ensure secure data transmission in distributed edge computing networks [39]. Byzantine Fault Tolerance (BFT) techniques have also been employed to maintain reliability in cloud environments. This also addresses the problem of having faulty nodes in a system using these methods, which is common for consensus in blockchain networks [40].

5.3. How Our Model Differs from Prior Work

Our enhanced model, however, is based on an automated structured analysis log data preprocessing pipeline that processes logs in real-time, automating feature extraction and transformation activities before ML model input, allowing for strong input into these models. The second branch is inspired by the characteristics of real-world errors, which are often omitted in prior studies, and we present four fault injection modules (CPU, memory, network, and data locality faults) to assess the system's performance under real-world errors. While most previous work tends to consider only CPU or memory prediction separately, we build a model that can predict multiple usages under changing workloads. Our model outperforms existing methods, achieving up to 97.3% prediction accuracy using hyperparameter-optimised fine-tuned ML algorithms. To clarify these comparisons, we summarise our model and related works in Table 2 below:

Table 2. Comparison of Related Studies

Study	Resource Type	Fault Tolerance	Real-Time Adaptability	Method Used	Accuracy (%)
[32]	CPU & Memory	No	Moderate	KNN & Decision Tree	91.5
[33]	CPU, Memory	No	No	Deep Learning-Based Workload Forecasting	95.4
[34]	CPU & Memory	No	No	Learning Automata	93.0
[36]	Edge/IoT	Yes	Moderate	Blockchain-Based Secure Resource Management	N/A
[39]	Edge Computing & Healthcare	Yes	Yes	Blockchain-Based Secure Communication	N/A
Our Proposed Model	CPU, Memory, Network, Data Locality	Yes	Yes	ML-Based Prediction & Fault Injection	97.3

6. CONCLUSION

Effective resource utilisation is the most critical aspect of processing large amounts of data in an acceptable amount of time. Inefficient resource utilisation is a common problem in cloud-based big data systems due to system heterogeneity, complexity, and unexpected errors, making it challenging to predict resource utilisation for provision resources. This article proposes an automated log data preprocessing-based online resource usage forecasting model using ML algorithms for big data systems. The experiments conducted under different fault scenarios show that our system predicts resource utilisation with a high accuracy rate and can identify the bottlenecks that lead to ineffective resource utilisation in big data systems.

The prediction of resource utilisation plays a crucial role in big data systems regarding time and cost management and provides the necessary provision of resources in case of need. Our system can help predict resource utilisation to embrace this matter for big data systems.

Acknowledgement

The author would like to express their sincere gratitude to South Tees Hospitals NHS Foundation Trust (United Kingdom) for their valuable support in providing access to anonymised healthcare data, which significantly contributed to the development of this study.

REFERENCES

[1] Koppad S, Gkoutos GV, Acharjee A. Cloud computing enabled big multi-omics data analytics. Bioinform Biol Insights. 2021;15.

[2] Wu PY, Cheng CW, Kaddi CD, Venugopalan J, Hoffman R, Wang MD. –omic and electronic health record big data analytics for precision medicine. IEEE Trans Biomed Eng. 2016;64(2):263–73.

[3] Kumar J, Singh AK. Workload prediction in cloud using artificial neural network and adaptive differential evolution. Future Gener Comput Syst. 2018;81:41–52.

[4] Gui B, Wei X, Shen Q, Qi J, Guo L. Financial time series forecasting using support vector machine. In: 2014 Tenth International Conference on Computational Intelligence and Security. IEEE; 2014. p. 39–43.

[5] Kim KJ, Han I. Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. Expert Syst Appl. 2000;19(2):125–32.

[6] Ullrich M, Lässig J. Current challenges and approaches for resource demand estimation in the cloud. In: 2013 International Conference on Cloud Computing and Big Data. IEEE; 2013. p. 387–94.

[7] Alwasel K, Calheiros RN, Garg S, Buyya R, Pathan M, Georgakopoulos D, et al. Bigdatasdnsim: A simulator for analyzing big data applications in software-defined cloud data centers. Softw Pract Exp. 2021;51(5):893–920.

[8] Jung J, Kim H. Mr-cloudsim: Designing and implementing mapreduce computing model on cloudsim. In: 2012 International Conference on ICT Convergence (ICTC). IEEE; 2012. p. 504–9.

[9] Calcaterra C, Carmenini A, Marotta A, Bucci U, Cassioli D. Maxhadoop: an efficient scalable emulation tool to test sdn protocols in emulated hadoop environments. J Netw Syst Manage. 2020;28(4):1610–38.

[10] Datadog [Internet]. New York: Datadog Inc.; [cited 2020 Jul 13]. Available from: https://www.datadoghq.com/

[11] Apache Chukwa [Internet]. [cited 2020 Jul 14]. Available from: https://chukwa.apache.org/

[12] Demirbaga U, Wen Z, Noor A, Mitra K, Alwasel K, Garg S, et al. Autodiagn: An automated real-time diagnosis framework for big data systems. IEEE Trans Comput. 2021;71(5):1035–48.

[13] Zhao K, Li S, Kang Z. Takagi-sugeno fuzzy modeling and control of nonlinear system with adaptive clustering algorithms. In: 2018 10th International Conference on Modelling, Identification and Control (ICMIC). IEEE; 2018. p. 1–6.

[14] Er MJ, Deng C. Obstacle avoidance of a mobile robot using hybrid learning approach. IEEE Trans Ind Electron. 2005;52(3):898–905.

[15] Al-Asaly MS, Bencherif MA, Alsanad A, Hassan MM. A deep learning-based resource usage prediction model for resource provisioning in an autonomic cloud computing environment. Neural Comput Appl. 2021;1–18.

[16] Royal College of Physicians. National early warning score (NEWS): standardising the assessment of acute-illness severity in the NHS. London: Report of working party; 2012.

[17] Ross M, Wei W, Ohno-Machado L. “Big data” and the electronic health record. Yearb Med Inform. 2014;23(1):97–104.

[18] Li S, Kang L, Zhao XM, et al. A survey on evolutionary algorithm based hybrid intelligence in bioinformatics. Biomed Res Int. 2014;2014:1–12.
[19] World Health Organization. Partnering for health early warning systems [Internet]. [cited 2025 Jun 21]. Available from: https://public-old.wmo.int/en/bulletin/partnering-health-early-warning-systems
[20] McGinley A, Pearse RM. A national early warning score for acutely ill patients: A new standard should help identify patients in need of critical care. BMJ. 2012;345(7869):9–9.
[21] Aujla GS, Jindal A. A decoupled blockchain approach for edge-envisioned IoT-based healthcare monitoring. IEEE J Sel Areas Commun. 2021;39(2):491–9.
[22] Mehta N, Pandit A. Concurrence of big data analytics and healthcare: A systematic review. Int J Med Inform. 2018;114:57–65.
[23] Dean J, Ghemawat S. Mapreduce: simplified data processing on large clusters. Commun ACM. 2008;51(1):107–13.
[24] Sun X, Ansari N, Wang R. Optimizing resource utilization of a data center. IEEE Commun Surv Tutor. 2016;18(4):2822–46.
[25] Ikhlasse H, Benjamin D, Vincent C, Hicham M. Multimodal cloud resources utilization forecasting using a bidirectional gated recurrent unit predictor based on a power efficient stacked denoising autoencoders. Alex Eng J. 2022;61(12):11565–77.
[26] Meng Y, Rao R, Zhang X, Hong P. Crupa: A container resource utilization prediction algorithm for autoscaling based on time series analysis. In: 2016 International Conference on Progress in Informatics and Computing (PIC). IEEE; 2016. p. 468–72.
[27] Margara A, Urbani J, Van Harmelen F, Bal H. Streaming the web: Reasoning over dynamic data. J Web Semant. 2014;25:24–44.
[28] Hameed A, Khoshkbarforoushha A, Ranjan R, Jayaraman PP, Kolodziej J, Balaji P, et al. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. Comput. 2016;98:751–74.
[29] Armstrong B, Eigenmann R. Performance forecasting: Towards a methodology for characterizing large computational applications. In: Proceedings of 1998 International Conference on Parallel Processing (Cat. No. 98EX205). IEEE; 1998. p. 518–25.
[30] Benaim M, Le Boudec JY. A class of mean field interaction models for computer and communication systems. Perform Eval. 2008;65(11–12):823–38.
[31] Demirbaga U, Noor A, Wen Z, James P, Mitra K, Ranjan R. Smartmonit: Real-time big data monitoring system. In: 2019 38th Symposium on Reliable Distributed Systems (SRDS). IEEE; 2019. p. 357–72.
[32] Mehmood T, Latif S, Malik S. Prediction of cloud computing resource utilization. In: 2018 15th International Conference on Smart Cities: Improving Quality of Life Using ICT & IoT (HONET-ICT). IEEE; 2018. p. 38–42.
[33] Al-Asaly MS, Bencherif MA, Alsanad A, Hassan MM. A deep learning-based resource usage prediction model for resource provisioning in an autonomic cloud computing environment. Neural Comput Appl. 2021;1–18.
[34] Rahmanian AA, Ghobaei-Arani M, Tofighy S. A learning automata-based ensemble resource usage prediction algorithm for cloud computing environment. Future Gener Comput Syst. 2018;79:54–71.
[35] Gao S, Xiao H, Zhou E, Chen W. Robust ranking and selection with optimal computing budget allocation. Automatica. 2017;81:30–6.
[36] Khan MA, Jan MA, He X. Blockchain-based edge computing frameworks for IoT applications: A comprehensive survey. IEEE Internet Things J. 2021;8(1):22–39.
[37] Aujla M, Jindal R. Blockchain-based healthcare monitoring for edge computing environments: Performance evaluation and analysis. IEEE Trans Ind Inform. 2020;16(3):2204–13.
[38] Nguyen DC, Pathirana PN, Ding M, Seneviratne A. BEdgeHealth: A decentralized architecture for edge-based IoMT networks using blockchain. arXiv. 2021;arXiv:2109.14295.
[39] Akbari Zarkesh M, Dastani E, Safaei B, Movaghar A. EdgeLinker: Practical blockchain-based framework for healthcare fog applications to enhance security in edge-IoT data communications. arXiv. 2024;arXiv:2408.15838.
[40] Cheikhrouhou O, Mershad K, Jamil F, Mahmud R, Koubaa A, Moosavi SR. A lightweight blockchain and fog-enabled secure remote patient monitoring system. arXiv. 2023;arXiv:2301.03551.