Enhancing Energy-Efficiency in Data Centers by Leveraging Computational Steering and Dynamic Task Allocation

Aditya Vijjapu, Alphonsa Jose, Achal Baniya, Alladi Sai Revanth and Dr Beena B.M*
Dept. of Computer Science and Engineering, Amrita School of Computing, Bengaluru, Amrita Vishwa Vidyapeetham, India
*corresponding author: bm_beena@blr.amrita.edu

***Abstract*—Data centers are critical components of the infrastructure of digital services but have a number of problems related to the management of resources, energy consumption, and organizational efficiency. The following paper describes a computational steering approach for resource management and energy consumption in cloud computing data centers. The system combines Round Robin and SJF scheduling with a Random Forest Regressor model for job duration prediction. Performance evaluation demonstrates significant improvements: SJF provided lesser energy utilization where it was brought down to 17.44 kW from 18.83 kW, enhanced CPU utilization from 16.31% to 15.82% and energy consumption variation was lowered to 5.79 from 39.4. Similarly, Round Robin scheduling minimized energy consumption to 17.76 kW, and maximized CPU utilization to 15.64%. Random Forest Regressor model gave an R-squared of 0.98, showing the model's efficiency in estimating job duration. It involves a Streamlit frontend that emulates real-time monitoring and control for scheduling tasks, and shows possible improvements in resource usage and energy consumption. This research work helps to effectively manage data center energy usage and is anchored on United Nation Sustainable Development Goals to provide a practical solution to energy efficient data center management.**

***Index Terms*—Computational Steering, Energy Efficiency, Machine Learning, Task Scheduling, Resource Allocation**

I. INTRODUCTION

Data centers are integral for modern digital infrastructure, supporting a wide range of services from data storage to high-performance computing. However, as computing demands grow, they face significant challenges in resource management, energy consumption, and operational efficiency [3]. Inefficient task allocation, uneven workload distribution, and fluctuating server performance can result in high energy consumption, increased costs, and degraded system performance. Optimizing these factors is crucial for enhancing operational efficiency and achieving sustainability goals.

The computational steering algorithm proposed in this paper combines Round Robin and SJF scheduling methods with a Random Forest Regressor model to predict the duration of jobs. The system performs load balancing on the fly to distribute workloads across the servers in order to fully utilize the CPU and at the same time decrease on energy consumption. A user interface built with Streamlit is used; users can input the job parameters, choose the scheduling algorithms, and then see how the performance of the system in terms of energy consumption and CPU usage changes.

Key contributions of this study include:

1) Development of a novel architecture that integrates computational steering, task allocation and machine learning to optimize energy efficiency.
2) Combination of multiple datasets and feature engineering for predictive modeling.
3) Automation of job scheduling using machine learning and computational steering for resource allocation.
4) Simulated real-time insights through a user-friendly Streamlit interface.

The proposed system complies with Green Computing principles which are the power saving, reducing the impact on the climate, and environmental conservation in data centers. Also, the system supports the UN Sustainable Development Goals (SDGs): SDG 7 (Clean and Affordable Energy), SDG 9 (Industry, Innovation, Infrastructure), and SDG 12 (Responsible Consumption).

The paper is structured as follows: Section II discusses related work in the field, Section III provides details of the dataset, Section IV presents the methodology, Section V discusses the results, and Section VI concludes the study.

II. LITERATURE SURVEY

In order to find areas where the study could be innovative, the study examines a number of previous researches that have been conducted on Datacenter energy efficiency. Research gaps were found during this investigation, opening up possibilities for inclusion in the report.

Srikanth and Geetha [1] examined the issues of Virtualization and Dynamic Resource Scheduling for Cloud Computing. They stressed that the scheduling algorithm is crucial for QoS, energy consumption and data resources management. Likewise, Magotra et al. [2] provided a survey of computational methods for minimizing energy usage and operational cost in cloud infrastructures where the models discussed include statistical, deterministic, probabilistic,

machine learning, and optimization-based models with their advantages and disadvantages.

Katal et al. [3] dealt with the software layer energy management at cloud data centers including virtualization layer, OS layer, and application layer. They found shortcomings in controlling CO2 emission and managing e-waste and presented futuristic green data center strategies. Belgacem et al. [4] proposed a machine learning VM migration model known as VMLM which reduces energy consumption and outperformed other benchmarks. Finally, there are works on clustering, optimization, and ML in energy efficient cloud networks by Jayprakash et al. [5]. The authors informed that despite the problems like low convergence, local optima, and overfitting these methods are promising.

The energy problem in cloud computing was discussed by Zhong et al. [6] with the emphasis on the server task scheduling, developing the dynamic fusion task scheduling algorithm for green cloud computing. Adamou et al. [7] proposed a biologically inspired approach in a modified ant colony optimization algorithm addressing energy consumption and task scheduling optimization with minimal makespan, optimal usage of resources, and increased green computing. To address the issue of high energy cost and energy wastage Lu and Sun [8] proposed a Resource-Aware Load Balancing Clonal Algorithm for Task Scheduling which integrates load balancing policy and the clonal selection algorithm for efficient energy management and load distribution. Yuan et al. [9] proposed the STSRO mechanism to minimize the total cost of providers in DGCDCs environments.

To address the dynamic market environments, STSRO coordinates tasks from various applications and geographical locations, and applies the SBA to solve constrained optimization problems for each time frame. The experimental outcomes show that the proposed STSRO is better than the traditional methods in terms of low cost and high flow efficiency. Likewise, Mao et al. [10] discussed the drawbacks of previous work, which either aimed at energy saving or performance enhancement only. They proposed two efficient algorithms and integrated them into the Energy-Performance Trade Off Multi-Resource Cloud Task Scheduling Algorithm (ETMCTSA) that allows tuning of power usage and performance by means of a probability factor (α).

Virtualization technology was used to solve the problems of resource utilization and throughput optimization in cloud services by Shu et al [11]. They developed an agile response task scheduling optimization algorithm using peak energy usage and optimum task scheduling time span. Also, Zhong [13] and Buyya et al [14] proposed an efficient technique of container orchestration for cluster management systems. They also use dynamic cluster resizing, underutilized VM rescheduling, and heterogeneous task allocation. Specific cost savings were observed when running evaluations on the Australian National Cloud Infrastructure (Nectar), compared to the default Kubernetes architecture for diverse cloud workload patterns.

Hussein et al., [13] on the other hand, reviewed the CaaS, with specific reference to flexibility of resource use in cloud systems. Their study suggested container placement algorithms that can accommodate both instantiated virtual machines and physical machines. Based on the increasing importance of CaaS, Al-Moalmi et al. [14] discussed the issues emerging from the interaction of containers, VMs, and platform managers. They put forward a core solution based on the WOA for improving the container deployment modeling. Gholipour et al. [15] have stressed that CaaS model has the potential to enhance the development of cloud computing; the authors proposed an integrated model that integrates both the virtual machine migration and the container migration. These improvements were achieved using Container CloudSim, where their simulations shown marked enhancements in energy efficiency, SLA violations and migration mishaps. Similarly, Beena et al. [16] have described various power management methodologies in high-performance computing systems and data centers and observed that the problems of efficient power management still remain the same. David et al. [17] analyzed energy efficiency concerns in IoT-healthcare applications where the concern is on privacy breach and high energy intake in health care facilities

Finally, Tiwari et al. [18] presented a new task scheduling algorithm, NIMS (Neighborhood Inspired Multiverse Scheduler), for managing two conflicting factors that are commonly associated with cloud computing: makespan and energy consumption. From the above literature review, it can be noted that current research lacks recent advances in the field of computational steering and energy efficiency in data centers.

III. Methodology

The flow of the application is explained below with reference to the flow chart shown in Fig. 1. It describes the general approach of the work starting from the idea and design to the integration of the key components of the system, including scheduling algorithms, machine learning models, and the Streamlit interface. The figure is to illustrate the workflow and the relations between the modules, which can give a overview of the application approach.

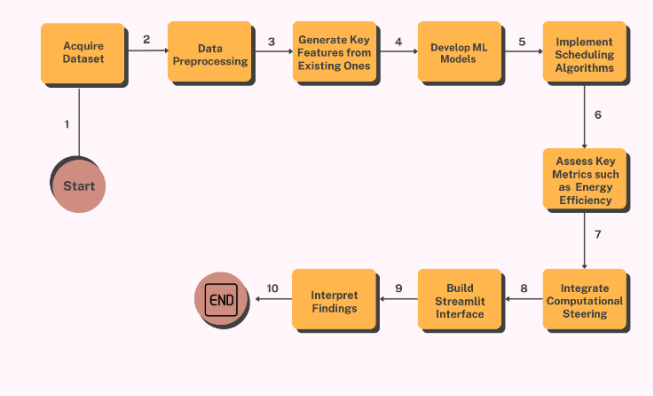

Fig. 1. Flowchart of Methodology

A. Acquire Dataset

The dataset collected is confidential and includes data from a high-performance computing center consisting of 1,160 CPU servers and 128 GPU nodes distributed across 20 racks. It encompasses information on server loads, energy consumption, and environmental factors, gathered over a month (July 1–31, 2018) at a minute-level granularity, resulting in 892,800 observations. The Power dataset logs energy consumption from rack power distribution units. The Jobs dataset tracks job activities, including job times and core allocations, comprising. The final dataset is combined dataset featuring both Power and Job Dataset Features. Due to the large size of the dataset we have taken a subset of the final dataset for our analysis that consists of 12 features and 4259 records.

B. Analyse Data Distribution

The data ingestion process involves importing large datasets from various sources into the database. The first step is to perform data wrangling, which ensures that the data is clean, consistent, and ready for analysis [22]. During this phase, unnecessary noise and irrelevant Figures are removed, and only meaningful and accurate data is selected for analysis. This step ensures the following:

1) **Data Cleansing**
   Removing any invalid, missing, or inconsistent data points.
2) **Data Transformation**
   Standardizing formats for easier analysis, such as converting dates into a uniform format.
3) **Handling Missing Data**
   Identifying and appropriately dealing with any missing or null values in critical columns (e.g., filling in missing energy readings or timestamps).
4) **Data Filtering**
   Excluding irrelevant information (e.g., unnecessary columns or duplicate rows) to focus only on the data that's relevant for analysis.
5) **Visualizations**
   After the data is cleaned and transformed, visualization is created in order to gain useful perspectives on the data. They are useful in interpreting the distribution of key metrics like power consumption, task duration, and CPU utilization and so on in order to facilitate decision making.

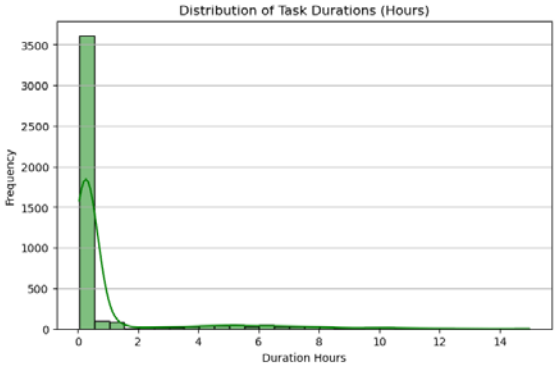

Fig. 2. Distribution of Task Durations

The above figure, Fig. 2, contains a histogram depicting the distribution of task durations. It is observed that the task durations mainly focus on the time duration between 0 and 1 hour with the highest density at the middle of this range. While starting with a small decline as the duration goes above 1 hour, the frequency of activities drops sharply for operations that take more than 2 hours. Further, the distribution of the number of hours is positively skewed showing that most of the tasks are accomplished within a short period of time and only a few tasks take more than 10 hours.

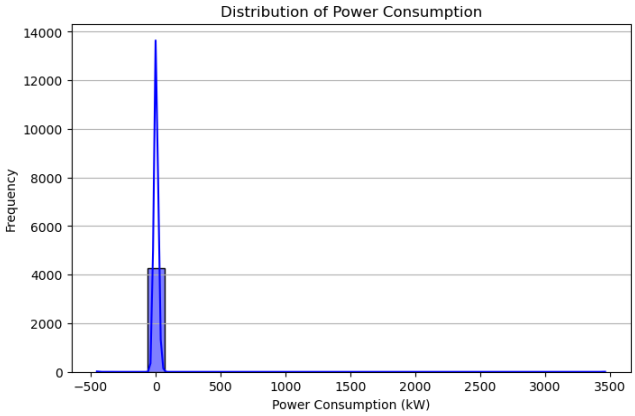

Fig. 3. Distribution of Power Consumption

The histogram in Fig. 3, demonstrates the power consumptions in kilowatts (kW) of the different areas. Peculiarities Most values are concentrated near 0 kW, which means that the majority of tasks does not consume much power. The distribution is positively skewed meaning that there are more values of lower power consumptions as compared to higher values. Values greater than 500 kW are quite limited, and the spike at 0 kW underlines the fact that the majority of the data points represent small power usage.

C. Generate Key Feature from Existing ones

Once the data has been cleansed and pre-processed, important features are extracted from the raw dataset for use in further analysis. These features will provide valuable insights into the operation of the data center. After the feature extraction, the Job and Energy Used data collections are combined to create a unified data structure, storing key information for each job/task in the data center. The following features are extracted as shown in Table I.

TABLE I
THIS A LIST OF FEATURES UTILIZED

Feature	Description
Rack Number	Identifies the physical rack in the data center.
Server Number	Identifies the server within the rack.
Task Number	Unique identifier for the task or job executed.
CPU Cores	The number of CPU cores allocated to each task, representing the computational resources being used.
Duration	The time difference between the start and end timestamps of the task, calculated by subtracting the Start Datetime from the End Datetime.
Start Datetime	Timestamp when the task starts.
End Datetime	Timestamp when the task ends.
Start Energy	Energy consumption at the start of the task.
End Energy	Energy consumption at the end of the task.
Power Consumption	The total energy consumed during the task, calculated as the difference between End Energy and Start Energy.
RPDU Number	Identifies the RPDU (Rack Power Distribution Unit) handling the task, linking the energy consumption data to the specific RPDU.

After feature extraction and calculation, the data is stored in MongoDB. MongoDB is chosen due to its ability to handle large datasets with high flexibility. The combined dataset, now containing all the relevant features, is stored in a structured format where each document in the MongoDB collection corresponds to a unique task/job in the data center.

D. Scheduling Algorithms Used

Three scheduling algorithms, namely First-Come, First-Served (FCFS), Shortest Job First (SJF), and Round Robin (RR) are considered in this work to determine the best scheduling method to allocate jobs to servers so as to enhance the use of resources and reduce energy consumption. FCFS is a simple algorithm, which assigns tasks according to their arrival time but it may cause the inefficient usage of the CPU and energy consumption [23]. SJF works on the basis of short jobs to minimize average waiting time and maximize system utilization but it needs information about the time required for each job. RR makes the CPU usage equal and regular with the help of fixed time quanta of tasks, which makes the load balancing better. All algorithms are then evaluated to find out how well each algorithm performs in different task conditions.

E. *ML Models Used*

In this study, several machine learning models were developed to predict the Duration of tasks in a data center environment. The models used include Random Forest Regression, Gradient Boosting Regression, Support Vector Regressor (SVR), Decision Tree Regression, and Linear Regression. The Duration was treated as the target variable, while various features such as Rack Number, Server Number, Task Number, CPU cores, Start and End Datetime, Start and End Energy, Power Consumption, and RPDU Number were used as input parameters. The models were evaluated based on three key performance metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R^2), which helped assess the accuracy and effectiveness of each model in predicting the task Duration. The model with the best performance, as determined by these metrics, was selected for further analysis [20, 21].

F. Integrate Computational Steering

In this study, Computational Steering was used to facilitate the changes of model parameters during the prediction process. This approach meant that one could input the parameters and immediately get the output, and then adjust the parameters as needed in a dynamic modeling process. With help of computational steering, the parameters like Rack Number, Server Number, Task Number, CPU cores and other characteristics could be changed and the models would give new suggestions for Duration [19]. This capability made it possible to base the analysis on different conditions or needs and made the machine learning models more versatile in predicting the task Duration depending on the conditions that prevailed.

G. Streamlit Interface

1) **User Inputs**
Users have option of entering the necessary input data for the web application to calculate and redistribute job scheduling. The inputs will include:
a) The number of CPU cores that each job will use. Users can select or input the number of cores available.
b) Users specify the power consumption of each job, which will be used in energy consumption calculations.
c) A dropdown or input field for selecting the server where the job will be assigned.
d) When user click on this the job which they wish to add will be added to existing job queue.

2) **Job Scheduling Options**
Users can choose between different scheduling algorithms. The two algorithms are:
a) *SJF (Shortest Job First):* A scheduling algorithm that prioritizes jobs based on their duration. The job with least duration is given highest priority.
b) *Round Robin:* A simple scheduling algorithm where jobs are assigned CPU time in a cyclic order.
The user will select one of these algorithms, and the app will apply it to redistribute the jobs accordingly.

3) **Duration Calculation**
In this part of the application, Duration will be calculated for each job based on the start and end time provided by the user. The formula for duration is:
Duration=End Time−Start Time
This value will be used in subsequent calculations for CPU utilization and energy consumption.

4) **CPU Utilization Calculation**

a) CPU Utilization will be calculated to evaluate how efficiently the CPU cores are being used. The formula is given below.

$$\text{CPU Utilization} = \frac{\text{Total CPU Cores used} \times 100}{\text{Total Hours} \times 24} \quad (1)$$

This calculation will be performed before and after job scheduling (depending on the algorithm selected), and results will be shown to the user.

b) *Threshold Check:* If the CPU utilization exceeds 80%, it will be considered as "High Loaded" , otherwise, it will be marked as "Normal Loaded" and "Low loaded" if utilization is below threshold.

5) **Energy Consumption Calculation**

Energy Consumption is calculated to measure how much power is consumed during the job's execution. The formula is given below.

$$\text{Energy Consumption (Wh)} = \text{Power Consumption (W)} \times \text{Duration in Hours} \quad (2)$$

$$\text{Total Energy Consumption (Wh)} = \sum(\text{Power Consumption} \times \text{Duration in Hours}) \quad (3)$$

This calculation will be performed for both the before and after scheduling and displayed to the user for comparison.

6) **Before and After Comparison**

In this section, the application will show a comparison of the Before and After values for both CPU Utilization and Energy Consumption:

a) *Before Scheduling:* The CPU utilization and energy consumption before applying the selected scheduling algorithm.

b) *After Scheduling:* The CPU utilization and energy consumption after applying the scheduling algorithm. This allows the user to see the effects of the scheduling algorithm on resource utilization and efficiency.

7) **Graphical Representation**

To enhance the visual representation of the results, the web application will display graphs:

a) *CPU Utilization Graph:* A bar chart showing the Before and After CPU utilization, helping users visualize how the scheduling algorithm impacted CPU resource usage.

b) *Energy Consumption Graph:* A bar chart that compares the Before and After energy consumption, providing insights into the efficiency of the scheduling algorithm in terms of power usage.

8) **Output Summary**

This section will summarize the key results, making it easy for users to understand the overall impact of the scheduling algorithm:

a) *Total CPU Utilization (Before and After):* A summary of the CPU utilization percentages.

b) *Energy Consumption (Before and After):* A summary of the energy consumed before and after scheduling.

c) Status (Overloaded/Normal): A final indication of whether the system is Overloaded (CPU utilization > 80%) or Normal (CPU utilization = 80%) and Low-Loaded (CPU utilization < 80%).

9) **Interactivity**

The web application, through computational steering, will allow users to dynamically modify inputs such as CPU cores, power consumption, server number. This interactivity allows users to instantly see the updated calculations and graphs in real time. The model will immediately recalculate the CPU utilization and energy consumption based on the new inputs.

10) **Optimization**

In this part, the user can evaluate the effect of different scheduling algorithms on system performance:

a) *Compare Algorithms:* Users can compare the effects of SJF and Round Robin on CPU utilization and energy consumption.

b) *Best Scheduling Choice:* The application will allow users to identify which scheduling algorithm results in the most efficient use of CPU resources and the lowest energy consumption.

This Streamlit web application will serve as an interactive tool to analyze and optimize job scheduling, helping users to assess CPU utilization and energy efficiency under different scenarios.

IV. IMPLEMENTATION

This section briefly explains about implementation of this study as depicted in Fig. 4.

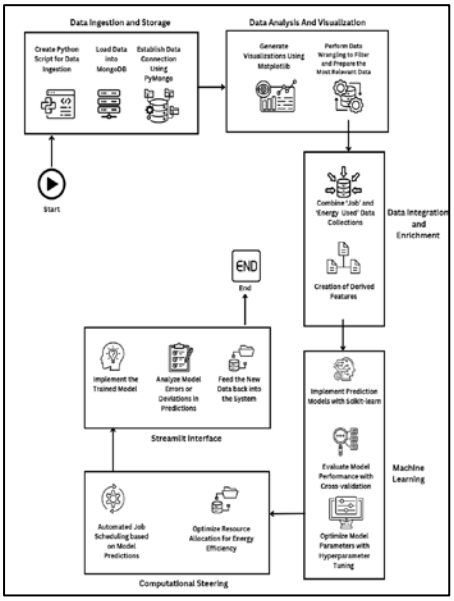

Fig. 4. Architecture of Proposed Study

A. Data Ingestion and Storage

In the initial phase, the focus is on ingesting data into a storage system for further analysis.

1) **Python**
   Python scripts are used to automate the data ingestion pipeline. Libraries such as pandas and csv are utilized to read, clean, and preprocess the raw datasets.
2) **MongoDB**
   MongoDB, a NoSQL database, is used for storing the ingested data due to its scalability and ability to handle unstructured and semi-structured data.
3) **PyMongo**
   PyMongo is the Python driver for MongoDB. It enables seamless data transfer and connection between Python scripts and the MongoDB database

B. Data Analysis and Visualization

The data stored in MongoDB is fetched for analysis and visualization.

1) **Matplotlib**
   Used to generate insightful visualizations, such as line charts, bar graphs, and scatter plots, to identify patterns and trends in the data.
2) **NumPy and Pandas**
   These libraries facilitate numerical computations and data manipulation, including handling missing data and filtering data records based on conditions.

C. Data Integration and Enrichment

To prepare the data for machine learning models, derived features are created by combining relevant datasets.

1) **Data Fusion**
   The "Job" and "Energy Used" datasets are integrated using Python libraries (pandas for merging datasets).
2) **Feature Engineering**
   Derived features are created by applying mathematical transformations and aggregations to enrich the dataset for predictive modeling [22].

D. Machine Learning

The enriched and preprocessed data is used for training machine learning models.

1) **Scikit-learn**
   The primary library for machine learning tasks. The following steps are performed:

   a) *Model Implementation:* Models such as Linear Regression, Decision Trees, or Gradient Boosting are implemented to predict energy usage or job scheduling outcomes.
   b) *Cross-Validation***:** cross_val_score is used for evaluating model performance and ensuring generalizability.
   c) *Hyperparameter Tuning***:** GridSearchCV or RandomizedSearchCV is employed for optimizing model parameters to improve accuracy and efficiency

E. . Computational Steering

To automate and optimize resource allocation, predictions are utilized to streamline operations.

1) **Automated Job Scheduling**
   The machine learning predictions are used to schedule jobs dynamically for efficiency.
2) **Resource Optimization**
   Predictions are leveraged to optimize resource allocation, ensuring energy efficiency and reduced computational costs.

F. Streamlit Interface

A user-friendly interface is built to integrate the machine learning results and provide real-time insights.

1) **Streamlit**
   Streamlit is a Python-based library used to create interactive dashboards. Features implemented include:

   a) Displaying predictions and performance metrics of trained models.
   b) Real-time analysis of deviations in predictions and model errors.
   c) Updating the data back into the system for iterative improvements.

V. Results And Analysis

This section summarizes the results achieved through executing the steps defined earlier in both Methodology and Implementation sections. The section presents main findings from experimental data collection together with performance indicators and analytical conclusions drawn from the setup. The research findings go through analysis to evaluate the effectiveness of the proposed method. The results are analyzed against existing methods and baseline models to determine both the strengths and weaknesses of the proposed approach.

A. Streamlit Interface

For the lack of connectivity between the high-end computations and the user interface, a dynamic and user friendly interface is under development through Streamlit. This interface is a user friendly platform that acts as an interpreter between the automated machine learning models and the final graphical images that people work with when using the system. It gives the results and provides features like graphs as well as tables for enhanced usage and better decision-making. Also, the interface has been developed with the consideration of users' needs; it has a responsive design, which makes it easier for the users to perform schedule and task operations. In this way, this interface successfully translates complex computational predictions into easily understandable forms of input that can be used to address actual problems in the field.

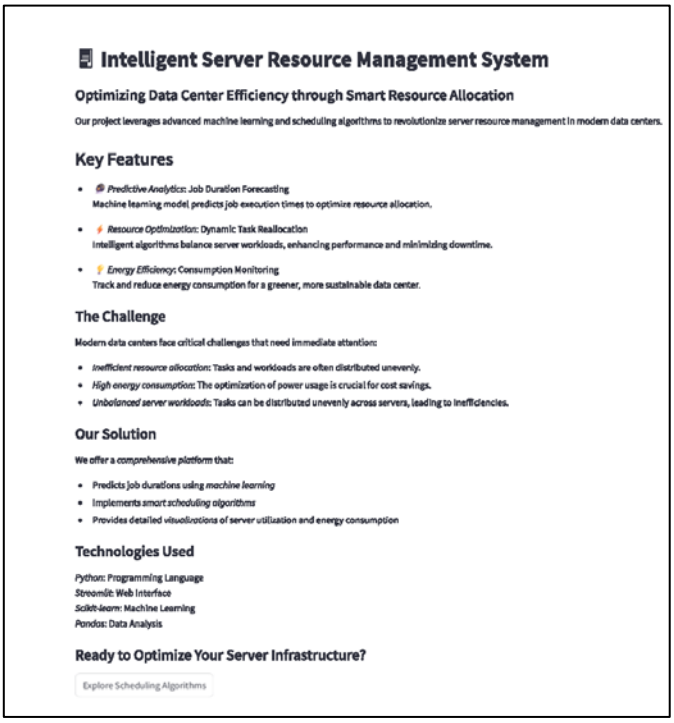

Fig. 5. Main Page of the Streamlit Interface

The first page as shown in Fig. 5, of the Streamlit interface presents the major functionalities of the application, the difficulties met during the dealing with modern datacenters and also, the technologies used in this study. There is a button below that leads the user to the main dashboard where the user can schedule his/her tasks upon clicking it.

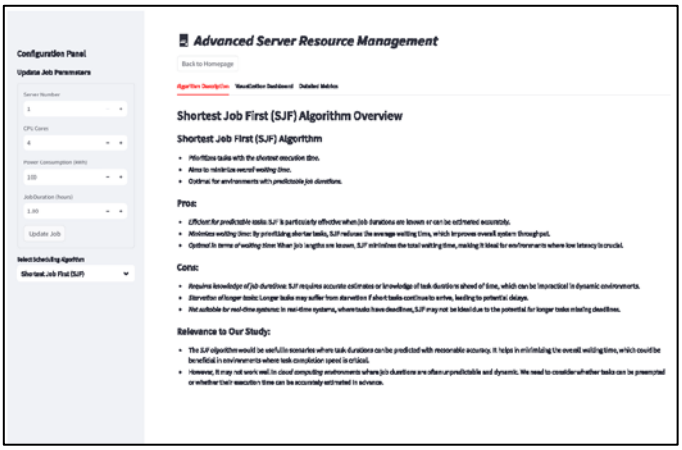

Fig. 6. Scheduling Page of the Streamlit Interface

On the Scheduling page of the application as shown in Fig. 6, there is a side bar where users can add new task manually and the task will be integrated into the present job pool through machine learning and computational steering. There is a button below to select the preferred scheduling algorithm for the task rescheduling. The main panel also contains three subpanels which represent the description of the selected scheduling algorithm, the comparison of the task states before and after the reallocation as well as the statistical report of the reallocation process.

Fig. 7. A Panel to add new job by modifying parameters

Fig. 7 shows that the web application has a sidebar in a webpage where users get an interface that allows them to put new tasks. From this panel, the users can enter Server Number, CPU Cores, Power Consumption (W), and Job Duration. This functionality can be used to enable change of settings with ease as well as dynamic setting of tasks. When modifying these parameters, users are able to regulate their work load, manage organizational resources, and make certain that the system is running along certain parameters.

B. Visualizations and Summary Statistics

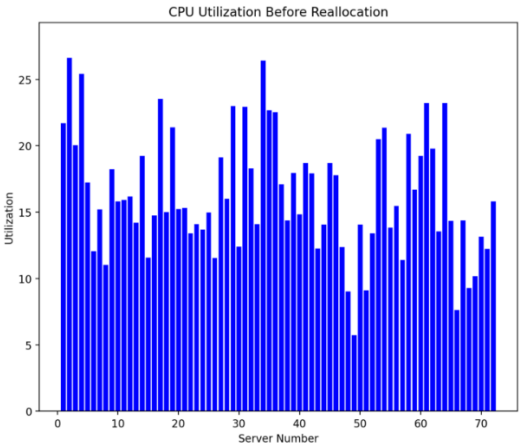

Fig. 8. CPU Utilization before reallocation using SJF

Before the job reallocation, as shown in Fig. 8, the CPU

utilization across the servers was uneven. Some servers were heavily loaded, while others remained underutilized. This discrepancy in usage indicates that jobs were not distributed efficiently, leading to certain servers being overworked and others sitting idle. The uneven CPU usage can be clearly seen in the graph, where some servers show high CPU usage while others show minimal usage.

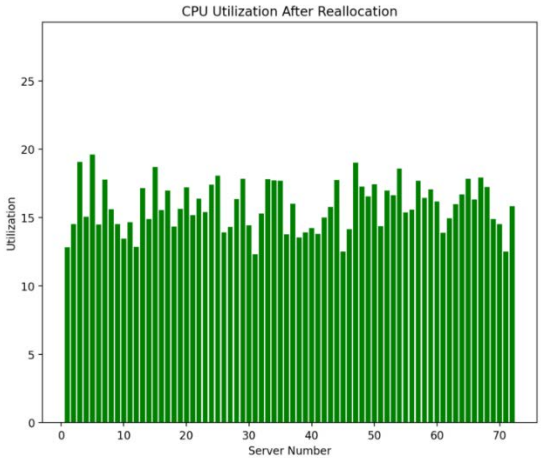

Fig. 9. CPU Utilization after reallocation using SJF

However, after the reallocation of jobs using the SJF (Shortest Job First) algorithm, the load distribution improved significantly. Fig. 9 shows that after reallocation, the CPU utilization became more balanced. All servers are now being utilized similarly, with the load distributed more evenly across the available servers. This shift indicates a more efficient distribution of tasks, preventing any single server from being overwhelmed and ensuring a more stable system performance.

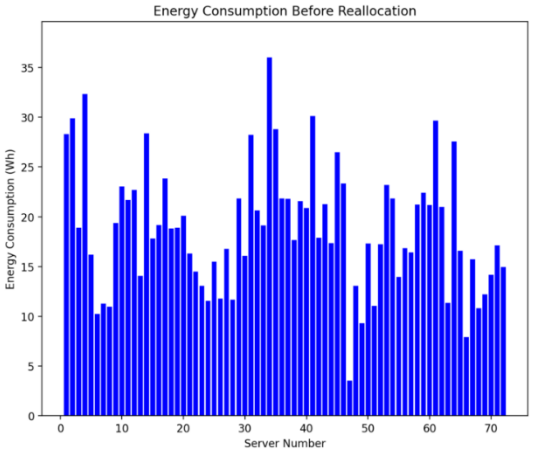

Fig. 10. Energy Consumption before reallocation using SJF

Before the reallocation process, the energy consumption was considerably higher which proves the inefficiency of the load distribution between the servers. As can also be observed in Fig. 10, the energy consumption levels were high, mainly because some servers were overloaded with too many tasks. These servers, because of the high traffic that they received, consumed more power as they worked hard to optimize their operations, thus becoming less efficient. This led to some servers being overutilized, thereby wasting resources by consuming more energy than they require, while others are underutilized. In this scenario, the case pointed out the need for the system to move resources around to optimize the distribution of energy consumption.

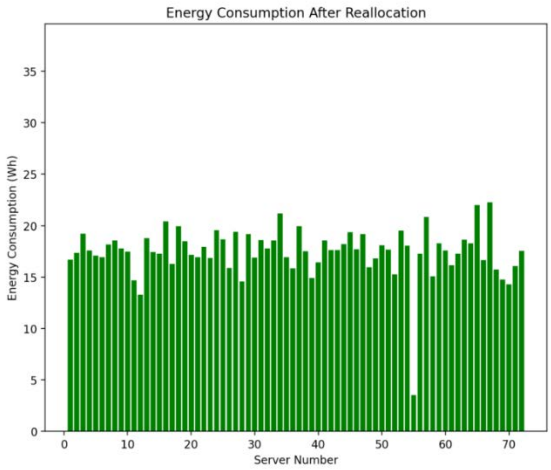

Fig. 11. Energy Consumption after reallocation using SJF

However, after job reallocation as shown in Fig. 11 the energy consumption was observed to have improved after the reallocation of the jobs. This reduction is the direct outcome of the fact that CPU usage had been balanced and none of the servers experience high CPU usage and all the servers were working up to their capacity. The energy consumption graph depicted below confirms that a balanced usage of the CPU results in more efficient use of energy: It also ensured great energy savings and emphasized the great role of proper distribution of tasks for best utilization of resources. Consequently, the system also achieves more cost-effective operations to reduce its cost of operations and increase efficiency.

TABLE II
CPU UTILIZATION STATISTICS (SJF)

Before		**After**	
Metrics	**Value**	**Metrics**	**Value**
Min	5.73	Min	12.33
Max	26.63	Max	19.6
Average	16.3113	Average	15.8243
Std	4.5344	Std	1.76
Variance	20.5607	Variance	3.0975

From table II, it is clear that by implementing the Shortest Job First (SJF) scheduling algorithm, it was observed that CPU utilization had increased significantly. The minimum CPU utilization is raised from 5.73% to 12.33%, and the maximum

is reduced from 26.63% to 19.6%. Besides, the coefficient of variation and standard deviation/ variance also declines, showing a more efficient and stable system after the reallocation.

TABLE III
ENERGY CONSUMPTION STATISTICS (SJF)

Before		After	
Metrics	**Value**	**Metrics**	**Value**
Min	3.54	Min	3.55
Max	36.01	Max	22.28
Average	18.8268	Average	17.4432
Std	6.2769	Std	2.4055
Variance	39.3993	Variance	5.7862

As seen from Table III, the energy consumption statistics reveal an improvement after the reallocation using the SJF scheduling algorithm. The minimum energy consumption remains almost the same, 3.54 kWh to 3.55 kWh. However, the maximum energy consumption reduces by a large margin from 36.01 kWh to 22.28 kWh. Also, the average energy consumption decreases from 18.8268 to 17.4432 kWh with decrease in both the standard deviation of energy consumption from 6.2769 to 2.4055 and the variance of energy consumption from 39.3993 to 5.7862 respectively, which also shows a more stable and efficient energy consumption pattern after reallocation.

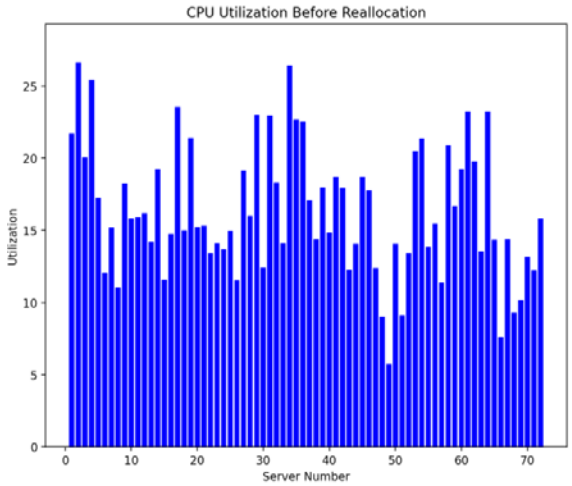

Fig. 12. CPU Utilization before reallocation using Round Robin

The same was observed when applying Round Robin scheduling algorithm of the system. As it was demonstrated in Fig. 12 prior to the reallocation, the CPU load distribution was rather unbalanced with some less loaded servers and others heavily loaded. Due to this, there was unbalanced distribution of tasks to the organizational CPU and this caused irregular utilization of CPU thus slow processing and poor use of resources. Reallocation resulted in a more balanced spread of the load in all the servers which implied that the use of the resources was very efficient. This made it possible to improve the response and balance of the overall system and not only cut down the time it takes to complete the tasks at hand.

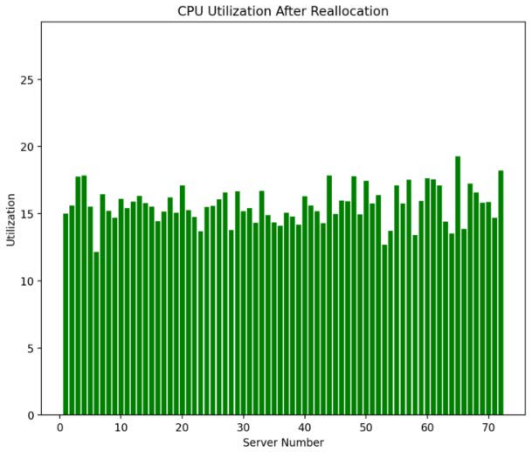

Fig. 13. CPU Utilization after reallocation using Round Robin

As indicated by the Round Robin of jobs shown below in Fig. 13, the CPU utilization was made more balanced. All the servers are utilized more effectively in the proposed solution, and the workload is distributed equally across the system. This leads to better utilization of resources in each server where none of the servers will be overwhelmed or idle. The enhanced load distribution capability makes the system more efficient and competent in handling load, thus minimizing the incidence of resource contention and improving system performance. The enhancement here does not only make best use of the available resources but also helps implement proper energy control in the servers.

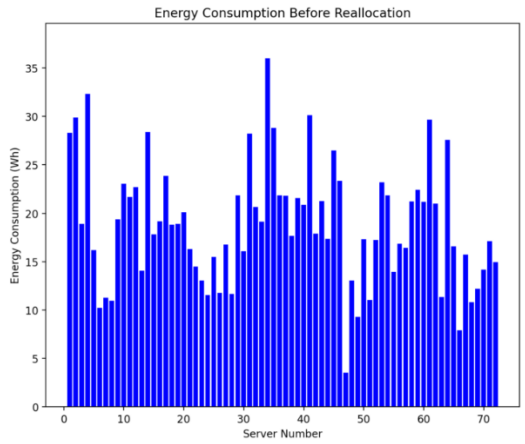

Fig. 14. Energy Consumption before reallocation using Round Robin

As seen in Fig, 14, the graph of energy consumptions before Round Robin reallocation shows that servers were unevenly loaded. A vast majority of the servers still operate below 20 kWh with several major peaks up to 35 kWh mainly around servers 20-25. This status shows resource imbalance, or in other words, that some servers can be overloaded while the others can remain free. This imbalance of energy consumption from server to server points to a conspicuous call for workload rebalancing for optimal distribution of the important resource with the

likelihood of decreasing cost on energy.

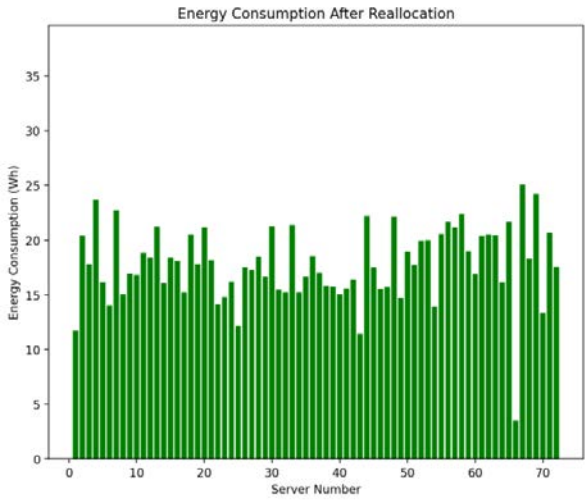

Fig. 15. Energy Consumption after reallocation using Round Robin

In Fig. 15, the energy consumption patterns observed after implementing the Round Robin job reallocation strategy are depicted. The measurements, spanning over 70 intervals, demonstrate energy usage predominantly fluctuating between 15 and 25 units, with occasional peaks reaching approximately 30 units. The consistent pattern of green vertical bars indicates stable system behavior, though periodic variations suggest the system's dynamic response to varying workload demands. Notable dips in energy consumption, particularly towards the latter measurement periods, reflect the effectiveness of Round Robin in optimizing task distribution and minimizing unnecessary power usage, thereby achieving a more even and energy-efficient operational state.

TABLE IV
CPU UTILIZATION STATISTICS (ROUND ROBIN)

Before		**After**	
Metrics	**Value**	**Metrics**	**Value**
Min	5.73	Min	12.15
Max	26.63	Max	19.27
Average	16.3113	Average	15.6426
Std	4.5344	Std	1.3698

From Table IV, it is clear that the utilization of the Round Robin scheduling algorithm increases the CPU utilization by a considerable percentage. The minimum value rises from 5.73% to 12.15% and the maximum utilization falls from 26.63% to 19.27%. Furthermore, the standard deviation decreases from 4.5344 to 1.3698 meaning that the usage of CPUs will be more balanced after reallocation.

TABLE V
ENERGY CONSUMPTION STATISTICS (ROUND ROBIN)

Before		**After**	
Metrics	**Value**	**Metrics**	**Value**
Min	3.54	Min	3.51
Max	36.01	Max	25.05
Average	18.8268	Average	17.7628
Std	6.2769	Std	3.4469
Variance	39.3993	Variance	11.8813

Table V shows the energy consumption statistics, which have been enhanced after the reallocation using Round Robin algorithm. The minimum energy consumption is kept nearly constant and slightly decreased from 3.54 kWh to 3.51 kWh. The maximum value is reduced significantly from 36.01 kWh to 25.05 kWh and the average consumption is reduced from 18.8268 kWh to 17.7628 kWh. In addition, the standard deviation and variance prove a significant reduction after the reallocation, which means that the energy consumption is more stable and efficient.

C. Machine Learning Results

This section contains the results of the assessment of different machine learning models for estimating the time required for tasks. The performance of the models was evaluated with the help of MSE, MAE, RMSE, and R^2 which are presented in Table VI. According to the results, the best model of the application was chosen for further use in the application.

TABLE VI
MACHINE LEARNING MODEL PERFORMANCE METRICS

Model	MSE	MAE	RMSE	R2
Random Forest Regressor	11.98	2.0	3.46	0.98
Gradient Boosting Regressor	17.30	2.68	4.16	0.96
Linear Regression	30.92	3.95	5.56	0.94
Decision Tree Regressor	18.66	2.51	4.32	0.96
SVR	405.53	5.90	20.14	0.17

From Table VI, it is seen that the proposed Random Forest Regressor model has better values of key performance indicators like MSE, MAE, RMSE, and R^2 than other models. Having an MSE of 11.98, an MAE of 2.0, and an R^2 of 0.98, it

is clearly seen that the model has better predictive accuracy and lesser error than the other models. The Gradient Boosting Regressor, Decision Tree Regressor and Linear Regression models have comparatively higher values of MSE and RMSE, therefore specifying low accuracy of predictions. The Support Vector Regressor (SVR) has the highest error metrics and therefore the worst performance. Hence, the Random Forest Regressor model was used to determine the duration for the jobs added by the user as it came out as the most reliable.

VI. Conclusion

In conclusion, the developed framework effectively optimizes job scheduling and resource allocation by combining machine learning models with classical scheduling algorithms such as Shortest Job First (SJF) and Round Robin. The integration of computational steering allows for dynamic user interaction, enabling the addition of new jobs to the queue and real-time resource reallocation. A comparative analysis of CPU utilization and energy consumption metrics demonstrates significant improvements in system efficiency. These benefits were clearly illustrated through graphical and tabular representations, highlighting enhanced energy usage and more balanced CPU utilization. The inclusion of Streamlit further enhances the system by providing an intuitive user interface, enabling stakeholders to interactively visualize results and manage resource utilization. Overall, this approach showcases the potential for scalable and flexible resource management in computing environments, with future possibilities for improvement through more sophisticated scheduling algorithms, larger dataset evaluations, and deployment in real-world applications.

References

[1] Srikanth, G. Umarani, and R. Geetha. ”Effectiveness review of the machine learning algorithms for scheduling in a cloud environment.” Archives of Computational Methods in Engineering 30.6 (2023): 3769-3789.

[2] Magotra, Bhagyalakshmi, Deepti Malhotra, and Amit Kr Dogra. ”Adaptive computational solutions to energy efficiency in cloud computing environments using VM consolidation.” Archives of Computational Methods in Engineering 30.3 (2023): 1789-1818.

[3] Katal, Avita, Susheela Dahiya, and Tanupriya Choudhury. ”Energy efficiency in cloud computing data centers: a survey on software technologies.” Cluster Computing 26.3 (2023): 1845-1875.

[4] Belgacem, Ali, Sa¨ıd Mahmoudi, and Mohamed Amine Ferrag. ”A machine learning model for improving virtual machine migration in cloud computing.” The Journal of Supercomputing 79.9 (2023): 9486- 9508.

[5] Jayaprakash, Stanly, et al. ”A systematic review of energy management strategies for resource allocation in the cloud: Clustering, optimization and machine learning.” Energies 14.17 (2021): 5322.

[6] Zong, Zhong. (2020). An Improvement of Task Scheduling Algorithms for Green Cloud Computing. 10.1109/ICCSE49874.2020.920178

[7] Ari, Ado Ir’epran, Damakoa Titouna, Chafiq Labraoui, Nabila Gueroui, Abdelhak. (2017). Efficient and Scalable ACO-Based Task Scheduling for Green Cloud Computing Environment. 66-71. 10.1109/Smart-Cloud.2017.17.

[8] Lu, Yong, and Na Sun. ”An effective task scheduling algorithm based on dynamic energy management and efficient resource utilization in green cloud computing environment.” Cluster Computing 22.Suppl 1 (2019): 513-520.

[9] Yuan, Haitao, Jing Bi, and MengChu Zhou. ”Spatial task scheduling for cost minimization in distributed green cloud data centers.” IEEE Transactions on Automation Science and Engineering 16.2 (2018): 729- 740.

[10] Mao, Li Li, Yin Peng, Gaofeng Xu, Xiyao Lin, Weiwei. (2018). A Multi-Resource Task Scheduling Algorithm for Energy- Performance Trade-offs in Green Clouds. Sustainable Computing: Informatics and Systems. 19. 10.1016/j.suscom.2018.05.003.

[11] Shu, Wanneng Cai, Ken Xiong, Naixue. (2021). Research on strong agile response task scheduling optimization enhancement with optimal resource usage in green cloud computing. Future Generation Computer Systems. 124. 10.1016/j.future.2021.05.012.

[12] Zhong, Z. and Buyya, R., 2020. A cost-efficient container orchestration strategy in kubernetes-based cloud computing infrastructures with heterogeneous resources. ACM Transactions on Internet Technology (TOIT), 20(2), pp.1-24.

[13] Hussein, Mohamed K., Mohamed H. Mousa, and Mohamed A. Alqarni. ”A placement architecture for a container as a service (CaaS) in a cloud environment.” Journal of Cloud Computing 8 (2019): 1-15.

[14] Al-Moalmi, A., Luo, J., Salah, A., Li, K. and Yin, L., 2021. A whale optimization system for energy-efficient container placement in data centers. Expert Systems with Applications, 164, p.113719.

[15] Gholipour, N., Arianyan, E. and Buyya, R., 2020. A novel energy-aware resource management technique using joint VM and container consolidation approach for green computing in cloud data centers. Simulation Modelling Practice and Theory, 104, p.102127.

[16] B.M. Beena and Dr. C.S.R. Prashanth, ”Green Computing-A Technology for High Performance Computing Systems,” World Applied Sciences Journal, vol. 34, no. 6, pp. 839-844, 2016, ISSN: 1818-4952, DOI: 10.5829/idosi.wasj.2016.34.6.15666.

[17] Thomas, D.M., 2023, July. Optimization of Green Computing in Data Privacy in Healthcare BCT Framework. In 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT) (pp. 1-6). IEEE.

[18] S. Tiwari and B. B. M., "A Neighborhood Inspired Multiverse Scheduler for Energy and Makespan Optimized Task Scheduling for Green Cloud Computing Systems," in IEEE Access, vol. 12, pp. 157272-157298, 2024, doi: 10.1109/ACCESS.2024.3484388.

[19] Kumar, K. Dinesh, et al., eds. Computational Intelligence for Green Cloud Computing and Digital Waste Management. IGI Global, 2024.

[20] T. N. Varunram, M. B. Shivaprasad, K. H. Aishwarya, A. Balraj, S. V. Savish and S. Ullas, "Analysis of Different Dimensionality Reduction Techniques and Machine Learning Algorithms for an Intrusion Detection System," 2021 IEEE 6th International Conference on Computing, Communication and Automation (ICCCA), Arad, Romania, 2021, pp. 237-242, doi: 10.1109/ICCCA52192.2021.9666265

[21] Harigaran, R., et al. "Fake news detection using a stacked ensemble of machine learning models." 2024 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT). IEEE, 2024.

[22] Krishna, ER Adwaith, et al. "Efficient Data Management through ML-Based Feature Extraction." 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT). IEEE, 2024.

[23] Reddy, Padigela Srinithya, et al. "Enhanced Task Scheduling in Cloud Computing: A Comparative Analysis and Hybrid Algorithm Implementation Using CloudSim." 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT). IEEE, 2024.