Comparative Analysis and Implementation of AI Algorithms and NN Model in Process Scheduling Algorithm

by

Maharshi Niloy
19101117
Md. Moynul Asik Moni
19101189
Farah Jasmin Khan
19101239
Aquibul Haq Chowdhury
19101290
Md. Fahmid-Ul-Alam Juboraj
19101618

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
September 2022

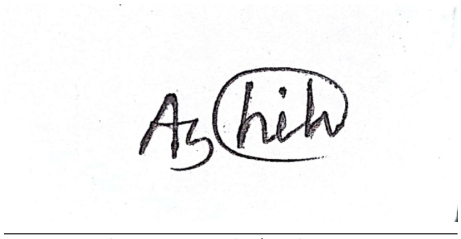
Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

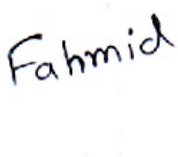
Maharshi Niloy
19101117



Md. Moynul Asik Moni
19101189

Farah Jasmin Khan
19101239

Aquibul Haq Chowdhury
19101290



Md. Fahmid-Ul-Alam Juboraj
19101618

Approval

The thesis/project titled "Comparative Analysis and Implementation of AI Algorithms and NN Model in Process Scheduling Algorithm" submitted by

1. Maharshi Niloy(19101117)
2. Md. Moynul Asik Moni(19101189)
3. Farah Jasmin Khan(19101239)
4. Aquibul Haq Chowdhury(19101290)
5. Md. Fahmid-Ul-Alam Juboraj(19101618)

Of Summer, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on September 20, 2022.

Examining Committee:

Supervisor:
(Member)

Amitabha Chakrabarty, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Moin Mostakim
Senior Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Process scheduling is an integral part of operating systems. The most widely used scheduling algorithm in operating systems is round-robin (RR), but the average waiting time in RR is often quite long. The purpose of this study is to propose a new algorithm to minimize waiting time and process starvation by determining the optimal time quantum by predicting CPU burst time. For burst time prediction, we are using the machine learning algorithms decision tree (DT), k-nearest neighbors (KNN), linear regression (LR) and Neural Network Model Multi-Layer perceptron-MLP. Finally, the obtained accuracy for burst time prediction of DT is 98.64%, KNN is 17.1%, LR is 97.96% and using MLP is 26.01%. Moreover, for 10000 predicted(burst time) processes with the same configuration the average turnaround time (avg TT), the average wait time (avg WT) and the number of context switches (CS) of the proposed algorithm are consecutively 40331930.48, 40312117.96 and 20002, whereas Traditional Round Robin (RR) has 87194390.98 (avg TT), 87174578.46 (avg WT) and 28964 (CS). Self-Adjustment Round Robin (SARR) has 72398064.70 (avg TT), 72378252.18 (avg WT) and 39956 (CS). Modified Round Robin Algorithm (MRRA) has 84924105.36 (avg TT), 84904292.84 (avg WT) and 5208 (CS) and Optimized Round Robin (ORR) has 78508779.73 (avg TT), 78488967.20 (avg WT) and 22470 (CS). Therefore, it is clear that the proposed algorithm is almost 2 times faster than the other algorithm in terms of process scheduling under a huge load of processes.

Keywords: Decision Tree (DT); KNN; Linear Regression (LR); Neural Network (NN); MLP; Prediction; Burst Time; Average Turnaround Time (avg TT); Average Waiting Time (avg WT); Context Switch (CS); Proposed Algorithm; Round Robin (RR); Self-Adjustment Round Robin (SARR); Modified Round Robin Algorithm (MRRA); Optimized Round Robin (ORR)

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption.
Secondly, to our supervisor Dr. Amitabha Chakrabarty and co-supervisor Mr. Moin Mostakim sir for their kind support and advice in our work. They helped us whenever we needed help.
And finally to our parents without their throughout sup-port it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

α Alpha

Σ Summation

σ Sigmoid Function

AI Artificial Intelligence

$avg\,TT$ Turnaround Time

$avg\,WT$ Waiting Time

CPU Central Processing Unit

CS Context Switch

DT Decision Tree

GA Genetic Algorithm

I/O Input/Output

KNN K-Nearest Neighbour

LR Linear Regression

ML Machine Learning

MLP Multi-layer Perceptron

$MRRA$ Modified Round-Robin Algorithm

ORR Optimized Round-Robin

OS Operating System

RR Round-Robin

$SARR$ Self-Adjustment Round-Robin

SCM Supply Chain Management

SJF Shortest Job First

$SMOreg$ Sequential Minimal Optimization

$SRTF$ Shortest Remaining Time First

$WEKA$ Waikato Environment for Knowledge Analysis

Chapter 1

Introduction

In a modern multitasking operating system, efficiency is paramount. Therefore, a good scheduling algorithm is necessary to maximize computing efficiency. These algorithms can be judged on many criteria, such as CPU utilization, throughput, turnaround time (TT), waiting time (WT),Context Switches (CS), response time, etc. Two scheduling algorithms of interest are shortest-remaining-time-first (SRTF) and round-robin (RR). In SRTF, each process is associated with its next CPU burst time, and the process with the smallest next CPU burst is chosen for execution. In RR, a small unit of time called the time quantum is used. The process at the head of the ready queue is allotted at most of the time quantum for execution then preempted and stored at the tail end of the ready queue. If the time quantum is too small, there will be many context switches. On the other hand, if the time quantum is too large, RR will devolve into a first-come-first-serve (FCFS) algorithm [24]. Thus it is evident that both of the above-mentioned algorithms have some shortcomings.

On separate note, we tried to improve the efficiency of the the processor based on our proposed algorithm and made a comparative analysis with the different artificial intelligence algorithms which includes Decision Tree (DT) [1] (previously performed well for applications like: COVID-19 severity prediction [29], table-tennis tactical decision making system [31]), K-nearest Neighbor (KNN) [3] (previously performed well for applications like: biometric image databases [16], industrial fault detection [26], and movie recommendation system [27]), Linear Regression (LR) [5] (previously performed well for applications like: forecasting power load [7], predicting contact temperature in high voltage switchgear [21]) and Multi Layer Perceptron (MLP) in terms of burst time prediction. If it is possible to train the processor to identify and predict the burst time accurately, it will be very much effective to calculate the time quantum more efficiently. So the comparison was done with different algorithms like self-adjustment round-robin (SARR), modified round-robin algorithm (MRRA), and optimized round-robin (ORR) after predicting the burst time with different algorithms and the same was done with our proposed algorithm where the time quantum was made dynamic. The result shows that our algorithm and the analytical process gives complete upper hand in decreasing the TT, WT, and reducing the context switches comparatively,thus making the processors and CPU scheduling algorithms efficient and optimized.Moreover, the machine learning models like DT [1] is a very useful tool in classification. KNN [3] is a widely used ML tool. Another powerful tool in ML is LR [5]. Thus, in this paper, we propose a new process

scheduling algorithm and use ML and neural network (NN) models like DT, KNN, MLP and LR to determine the optimal time quantum dynamically.

1.1 Research Problem

The process scheduling algorithm is one of the most vital aspects of ensuring overall efficiency. With the advent of the operating systems and the processor, there have been a lot of algorithms and processor level modifications to ensure the maximum utilization of the processor and operating system with minimum resource cost. Still, at present most of the algorithms we formulate or micro-level modifications are based on educated assumptions, and those are not exactly correct, because of different processes in different hardware systems and in different computers like cloud, grid [15], high performance, macro or micro along with different operating systems like mac, windows, Linux [6] or different subsystems the process scheduling is very much unpredictable, dynamic and nearly impossible. As a result, it gives rise to different problems like starvation of process, context overhead switching, delay of processor, deadlock, and process and resource allocation, aging, inefficient usage of CPU resources, wastage of internal memory, job pool queuing, etc. Also, another problem is the lack of analytical data based on different CPU scheduling algorithms and parameters like burst time, turnaround time, waiting time, and the estimated and most optimized time slice in the round-robin algorithm. This problem also raises many problems in the feedback queue, because the feedback received by the system is merely based on some presumptions data. Another problem is that the operating system end is not intelligent enough to make the decision or the CPU itself is unaware of the CPU and scheduling parameters. So all these optimizations, maximization, and minimization problems can be solved if the CPU can be made aware of the data or if we can implement our algorithm with a predictive approach to make intelligent decisions.

1.2 Research Objectives

1. To find out the best ML/NN approach for CPU burst prediction.
2. To propose a new CPU scheduling algorithm.
3. To compare the performance between the proposed algorithm and the existing algorithms based on predicted CPU burst time.

Chapter 2

Related Work

Different types of estimation algorithms like exponential average do not always provide accurate results, therefore in the paper [15], the authors proposed an approach where they have used some Machine Learning (ML) methods to estimate the CPU burst time for the process. They traced the most important attributes that are highly correlated with the CPU burst time via some feature selection techniques. They applied various types of ML approaches (Decision Tree-DT, K Nearest Neighbor - KNN, Support Vector Machine - SVM, ANN) on a grid computing dataset. The authors found a rigid linear relation between the CPU burst time and the selected attributes of a process after a series of experiments. Moreover, the authors had shown that KNN outperformed all other methods and approaches. Furthermore, the paper explicitly pointed out that the overall performance of the proposed methods had improved in terms of space and time complexity along with the estimation of CPU burst time, after the refinement of the data via feature selection methodologies. In this paper, the authors mainly focused on four ML-based methodologies (MLP, SMOReg, KNN, and DT). They applied these techniques in six consecutive steps (preparing data, filtering, generating model, prediction, evaluation, and deployment) and finally justified the efficiency by proposing the implementation of SRTF and SJF.

The authors showed in this paper [2] the implementation of the learning mechanism for the handling of the scheduling problem by the processors in a multiprocessor environment. They suggested intelligent heuristics be learned by the intelligent machines learned on their own in different cooperative processes. The authors provided an architecture called expert scheduler on which they recommended knowledge-based along with system inference, learning subsystem, monitor, and finally a scheduler. For the whole process, the authors considered arrival time, time quantum, and set of predecessor processors. Also, they have shown the credit assignment for different heuristic models. This shows the probabilistic feature that is the architecture which is possible to avoid bottlenecks by correcting those and making computations along with the parallelism.

The authors mainly used Fuzzy inference on the user model [8]. The proposed model was mainly composed of two components process log and user feedback the process log is mainly used as data for the classification of the processes and user feedback for the user modeling, on the other hand, it has already been shown that both of the decision parameters from both of the logs are sent to the process priority and the priority was calculated based on the fuzzy inference. The experimentation

was done on the grid on the Linux kernel 2.4.25 and they concluded that based on the multiple processes the system can classify the processes based on the batch, interactive and real-time processes, and based on the type it can recommend the adaptive process and lastly it can provide scheduling algorithm without resetting the operating system and they also proved their model can provide the flexibility of supporting scheduling algorithms to different users.
In the paper [30] the authors claimed a novel approach in which they used some Machine Learning methodologies dynamically to find out the most efficient CPU scheduling algorithm. Through experimentation, they showed that the overall performance of the Decision Tree in terms of accuracy and computational time was better than the other ML techniques. But in the Tarek et al. [15] claimed that KNN gives better and more accurate results, The authors in this paper showed that DT gives high accuracy and less execution time whereas the algorithms like KNN give time-consuming high distance results and naive Bayesian data gives the degraded performance due to highly correlated data based on conditional independence. Their main goal for the research was based on combining the strategies of existing schemes and changing those dynamically based on the arrival time. The constraints that were set high accuracy and the least response time throughout the whole process, though the literature doesn't show predefined data the authors generated the data based on the predefined rules from solving the Gantt chart and use machine learning as predict function on the whole dataset.
In this paper [19] the authors analyzed two important and vital algorithms: The Bayesian decision tree and the modified design scheduling process. They also highlighted the module isolating the kernel modules from the isolated wrappers. Also, they proposed the self-selecting and self-detecting process. Unlike the fuzzy inference, this paper mainly focused on the uniprocessor system and used the Bayesian decision tree as the classifier in two of the main process, the first one mainly comprised of the selection of the static and dynamic properties of the whole system like the process in the queue and secondly the process also comprised of the comparison of the previously executed process. On the other hand, the authors also showed the way to classify and make decisions based on useful and not useful processes. Though the author mainly focused on the low-level assembly environment but mentioned that Linux and UNIX have editable kernels, their model can be implemented in those two kernels as well.
Paper [23] has proposed a median-based modified Round Robin (RR) CPU scheduling algorithm that specifies the time quantum dynamically in each round depending on the process bursts time available in the ready queue. Here, they have worked with five processes with their proposed algorithm and compared the resulting parameters like the required number of context switching, average waiting time, average turnaround time, etc. to the existing available algorithms. They mostly focused on dynamic allocation of the time slice which is called time quantum for RR. Their algorithm illustrates a slightly modified RR algorithm where they approximate this time quantum in each round depending on the process available in the ready queue and their corresponding burst time. In the end, their algorithm reduces the number of context switches and both the turnaround time and waiting time compared to other existing algorithms (e.g. primitive RR, DQRRR, IRRVQ, SARR, etc.).
In this paper [28], the proposed work upholds a dynamic modified Round Robin (RR) algorithm based on K-Means clustering to consolidate the advent of favor

short processes and low scheduling overhead to minimize the average waiting time, turnaround time, and the number of context switching. They first constructed some clusters depending on the process parameters using the K-Means algorithm. A method named Silhouette, they have used here for the evaluation of the processing cluster where it finds out how effectively each data point lies within its cluster. A high average silhouette width indicates a good clustering. Then they have approximated a time slice for RR which directly contributes to the minimization of average waiting time and average turnaround time. Using this pre-calculated time quantum value they have again adjusted the time slice value depending on residual burst time. Surprisingly the proposed approach performs better than other algorithms proposed before in terms of efficient CPU scheduling.

In [20] the author proposed an in-depth survey of some Machine Learning based algorithms for improving the process scheduling algorithm. Here they sequentially reviewed various proposed algorithms. Firstly the authors featured a Machine Learning (ML) based approach referring to the paper [6] for making the CPU scheduling more efficient. Where the main paper [6] was mainly focused on the prediction of the number of required resources of the processes before their execution to provide the CPU with an initial guess about the incoming processes. Here "interactive" and "no interactive" were specified for the categories of the process and the Linux benchmark had used for evaluation purposes. The total execution time of the incoming processes was predicted using a total of 24 attributes by ascertaining the values for the various chosen attributes of a particular process by making required system calls. They first collected the data and put them under 20 classes to be used for machine learning in WEKA by using the "Trees, Lazy, Rules" classifier. The prediction part was accomplished using Decision Trees, K-NN and Decision Tables. For finding the best subset of the attributes for better prediction, the authors used an attribute evaluator which assigned a weight to each subset and search method (i.e. Genetic Search, Best First Search, and Rank Search), which defined the kind of search that would be performed. The "input size" was predicted as the best attribute over others and "page reclaims" was predicted as the second best. In the end, these results were used to improve PBS scheduling algorithms.

Secondly, the paper [20] reviewed a similar sort of Machine learning approach [11] which imposed a data mining technique on the data present in the kernel about each process for the classification of similar type of behavior. Here to create a training base the attributes of the processes present in the Linux context were extracted. Using the unsupervised learning algorithm the groups were made and then manually analyzed. And rest of the steps were quite similar to the previous paper [6], prediction of the best attribute and then using it in the Process Scheduling algorithm.

Then the author of the paper [20] illustrated [12] where the authors proposed a new scheduling technique named "Semantically Cognitive Scheduling" where the scheduler used a cognitive approach for process scheduling. It mainly depended on the utility value of the process for their classification and the rest of the evaluation part was quite similar to the previous. In this manner paper [20] illustrated a few more proposed ML algorithms.

The paper has proposed [9] that the main goal was to get the minimum time quantum to get a minimum turnaround time in this paper. A multilayered perception NN with a hidden layer is used, where a hidden layer activation function is taken and an activation function is taken from the output layer of the neural networking approach,

and the scaled conjugate gradient method is used for the weights of the NN. Here, different service times are taken as input from the ready queue and their avg time has been calculated to get the minimum time quantum. The times have been divided by their maximum service time to get a binary range. And the estimated ideal quantum length and the calculated quantum give us the error function. So, from this experiment instead of a before-thought time quantum, a minimum time quantum is derived from a set of service times.
In the research papers [10][14], a genetic algorithm approach has been used to get a minimum average waiting time for all the time scheduling algorithms FCFS, SJF, and RR. In the genetic algorithm, initially, a random population is initialized, then a fitness evaluation is done, after that from them through roulette wheel selection individuals are selected and crossover & inversion is done. So, in program scheduling, random jobs are taken and from their permuted pool random jobs are selected. So, after the mutation, if the crossover isn't valid then again the selection process is performed so that the jobs don't repeat in the scheduling process. After the selection process, a schedule with a minimum average waiting time is found.
The author in [4], has used the Genetic Algorithm (GA) but with a two-dimensional matrix. Here, the reproduction is used as an operator. Then, a biased roulette wheel is used in the fitness value. A crossover operator mates two randomly selected offspring and does the crossover in the allocation matrix and this does a vertical split of the offspring. After that, if any invalid allocation matrices have been presented then we repair that particular matrix. Then, they used a mutation operator which randomly selects positions of the matrices and compliments their values. After this step, we again have to see if the allocation matrix is valid or not. This replaces the worst-fit individual with the best-fit individual.
The paper has presented [17], a comparison between traditional RR, Improved RR, and a new type of RR named Enhanced RR is given. In the Improved RR, if the burst time of the running job is less than 1 time quantum then the CPU is allocated with the same process again. In the Enhanced RR, if the time quantum of the running process is $<= 1$ then it comes to the same process again and if the time quantum is > 1 then it is submitted to the end of the ready queue tail. So, for these 3 algorithms, 3 types of scenarios are taken where Burst times are taken at random, increasing and decreasing in order. After the experiment, we can see that Enhanced RR gives better output than the traditional RR & Improved RR.
The paper [25] has used Neural Networking on single and multiprocessor systems. For both cases, NN outperforms all the other algorithms like RNN & fixed priority. Both systems have been set to certain constraints. In this scenario, Hopfield-type networks which are a type of recurrent neural network are used. Hard real-time scheduling is used where the system mimics human memory. Also, the nodes are used like neurons, where the neurons have minimum binary storage and regardless of the input they converge back. But many times this system is incorrect. To solve this problem, a solution has been suggested where binary feedback will be given, whether the system is correct or not. And the system has to be rigorously trained. This is called an SCM method without this whole NN system would be pointless.
In the research paper [18] a new algorithm named Improved RR, which is the combination of RR, SJF, and priority. Here, the jobs have a priority and the lowest priority jobs have smaller time slices, medium priority has a bigger time slice than low priority jobs and higher priority has the biggest time slice. Through compari-

son, it is said that the average waiting time, turnaround time & context switches of improved RR are less than the traditional RR.
The research work has proposed [13] a new multi-level scheduling algorithm using fuzzy inference. In the fuzzy inference system, the if-then form is used where binary values 0/1 are taken according to the comparison. Partial Truths are taken in a graph, where they will point to a binary result according to the degree of the truth. Here, the fuzzy system is designed to change the values of the CPU time dynamically. The CPU times are given in different queues, cycle times are given by the users. In this technique, the ready queue is divided into two subparts. In the first part, I/O bounds are taken and in the second part, the CPU-bound processes are taken. Here the 1st process gets the priority during all the processes. The new proposed algorithm improved response time and starvation problems with small increments in waiting time which can be avoided as the average reduced waiting time is less than the increased waiting time.
In the paper [22], a new algorithm named Modified Median round Robin algorithm (MMRRA) where the time quantum is calculated through the root of the multiplication of the median and highest burst time. This algorithm is tested with the SJF algorithm where processes will go to the second round if the first round time quantum is calculated and the outburst time becomes greater than twenty percent of the total burst time; if not, the process will be completed by the CPU. When compared with other algorithms it has provided better results.
In this paper [33], a new type of algorithm MARR (Median Average Round Robin) is proposed where the time quantum is dynamically changed. Here, the burst time of the new ready queue time quantum is calculated. The time quantum is set out to be the running burst time if it is empty, but if the time quantum isn't empty the average of the burst time is taken. It is seen that a new MARR(Median Average Round Robin) is better than other algorithms like RR, AMRR, AN, MMRRA in ATAT (Average Turnaround Time) & AWT (Average waiting Time) but the NCS (number of context switches) is the same. It would have been more efficient if the context switch had been reduced.
In the paper [34], an algorithm named a new novel intelligent CPU Scheduling RR algorithm is proposed where based on a standard deviation of jobs the processes are executed in a single or multiple sub-queue. If the SD (Standard Deviation) is larger than the optimal value of the threshold it will be copied to the SRQ(Single ready queue) otherwise the single sub-ready queue. IQ (Intelligent Quantum) & DIQ (Dynamic Intelligent Quantum) is calculated with the usage of SRQ and the help of mean, median & Standard deviation. IQ is executed by all the processes of SRQ. DIQ will be executed, if the initial SRQ burst time process is less than equal to DIQ's SRQ value. Other SRQ's will be executed if the small time quantum is 1 until the beginning SRQ is empty. If the first SRQ is empty, then the next SRQ will be counted. In this algorithm, ATA, AWT, NCS are better performing than the other algorithms like ADDRR, CRRTQ, MRRA, DMRR, EDRR, MMRRA, etc.
In the paper [32], the Highest Response Ratio Next algorithm (HRRN) is a new algorithm where according to approximate run time, waiting time for each job is prioritized. The longer a job is starved the higher the priority it gets. Longer jobs with shorter life spans are ignored with the starved jobs. According to the largest response ratio, the CPU is given the work of a process. Priority is set by the summation of waiting & estimated time runtime divided by the estimated runtime.

It was seen through tests that the AWT of HRRN was less than the traditional RR due to HRRN's lack of time quantum usage.
From the above discussion, it is observed that most of the research implements various types of algorithms. These models try to propose an optimum solution by implementing these techniques directed upon the burst time, turnaround time, waiting time, context switching, and CPU overhead. For example, Genetic algorithm, Fuzzy inference, and Neural Network approaches have been taken to make the system further efficient regarding their scheduling. All of the research has shown compelling outcomes and also bestowed us with increased methodical solutions. Whilst there have been thousands of works appraising this topic, there are still a few places for improvement where the scheduling can be considerably more efficient.

Table 2.1: Summary of Related Work

References	Task	Model/Algorithm	Summary
[15]	Find the most important attributes of the process and estimate the CPU-burst	Support Vector Machine (SVM), K-Nearest Neighbours (K-NN), Artificial Neural Networks (ANN), Decision Trees (DT)	The authors proposed a Machine Learning (ML) based approach to estimate the length of the CPU-bursts for processes.
[2]	Suggesting the heuristics by machines in cooperative environment, implement expert scheduler	Credit assignment heuristic models along with expert scheduler	Implementation of the learning mechanism about the handling of the scheduling problem by the processors in a multiprocessor environment
[8]	Classification of the processes and user feedback for the user modelling	Fuzzy Inference	The system can classify the processes based on the batch, interactive, and real-time processes; based on the type it can recommend the adaptive process.
[30]	Comparison of the dynamic machine learning models like KNN, DT	Dynamic machine learning algorithms	The decision tree gives better results in terms of accuracy and computational time

Continued on next page

Table 2.1: Summary of Related Work (Continued)

[19]	Analysis of Bayesian Decision Tree and modified scheduling algorithm	Bayesian Decision Tree	Proposal of the self-selecting and self-detecting process.
[23]	Proposal of median-based Round Robin (RR) CPU scheduling algorithm	Modified Round Robin with dynamic time quantum	Approximating the time quantum in each round depending on the process available in the ready queue and their corresponding burst time and reduce the CPU parameters
[28]	Proposal of dynamic modified Round Robin	K-Means clustering	Consolidate the advent of favour short processes and low scheduling overhead in order to minimize the average waiting time, turnaround time, and the number of context switching by K-Means
[20]	Survey of ML-based on process scheduling algorithms	Different scheduling based ML algorithm	Comparison of different ML algorithms
[6]	Prediction of the number of required resources of the processes before their execution	Categorization based on interactive and non-interactive processes	Improving the result of the PBS scheduling algorithm
[11]	Imposition of data mining technique in kernel data	Unsupervised learning algorithm	Predicting the best attributes and improving those
[12]	Proposing Semantically Cognitive Scheduling algorithm	Cognitive process scheduling approach	Showed the dependency on the utility value of the process for their classification.

Continued on next page

Table 2.1: Summary of Related Work (Continued)

[9]	Get the minimum time quantum	Multilayered perceptron NN with a hidden layer	Instead of a before thought time quantum, a minimum time quantum is derived from a set of service times.
[10]	Implementing a genetic algorithm approach to get the minimum average waiting time	Genetic algorithm	A scheduling algorithm with minimum average waiting time was identified.
[4]	Implementing a Genetic algorithm with a two-dimensional matrix for process scheduling algorithms	GA with two-dimensional matrix	The reproduction was used as an operator
[17]	Comparison between traditional RR and enhanced RR	Improved and enhanced RR	Enhanced RR gives better output than the traditional RR & Improved RR.
[25]	Implementation of Neural Networking on single and multiprocessor systems	RNN, fixed-priority scheduling algorithm	A solution has been suggested where binary feedback will be given, whether the system is correct or not. And the system has to be rigorously trained which is the SCM method.
[18]	Implementation of Improved RR	RR+SJF along with priority	Average waiting time, turnaround time & context switches of improved RR are less than the traditional RR.

Continued on next page

Table 2.1: Summary of Related Work (Continued)

[13]	Implementation of a new multi-level scheduling algorithm using fuzzy inference	Fuzzy inference	Proposed algorithm improved response time, the starvation problem with a small increment in waiting time which can be avoided as average reduced waiting time is less than the increased waiting time.
[22]	Efficient Process Scheduling	Modified Median round Robin algorithm (MMRRA)	It is a modified RR where time quantum calculates through the root of the multiplication of the median and highest burst time and compared with other algorithms like SJF.
[33]	Dynamic Time Quantum	MARR (Median Average Round Robin)	It dynamically calculates time quantum and is compared with RR, AMRR, AN, MMRRA by measuring avg TT, avg WT, and number of CS where it performs better except number of CS is same.
[34]	Efficient Process Scheduling	A novel intelligent CPU Scheduling RR algorithm	It is based on a standard deviation of jobs the processes are executed in a single or multiple subqueue.
[32]	Approximation of run time	Highest Response Ratio Next algorithm (HRRN)	It is a new algorithm where according to approximate run time, waiting time for each job is prioritized.

Chapter 3

Methodology

3.1 Workflow

The purpose of the proposed algorithm is to minimize the starvation of processes for an extended period. We are going to do that by predicting the burst/run time of the processes with ML models, MLP, and exponential average. After predicting the burst/run time, we calculate the time quantum. The scheduling algorithm is going to schedule the processes with the calculated time quantum. The following flowchart shows the workflow of our model.
In this paper, we will use a Decision Tree, K Nearest Neighbor, and linear regression as our ML models and Multilayer Perceptron as a Neural Network model to predict the burst/run time from the collected dataset. There are a few stages to complete the training of the model:

1. Input Data Preprocessing: In this stage, the dataset is cleaned and formatted in such a way that makes the models learn and generalize the data better.

2. Processing: After splitting the dataset into training and testing, the training data is used to train the models.

3. Prediction: The testing data is used to make predictions and find the accuracy.

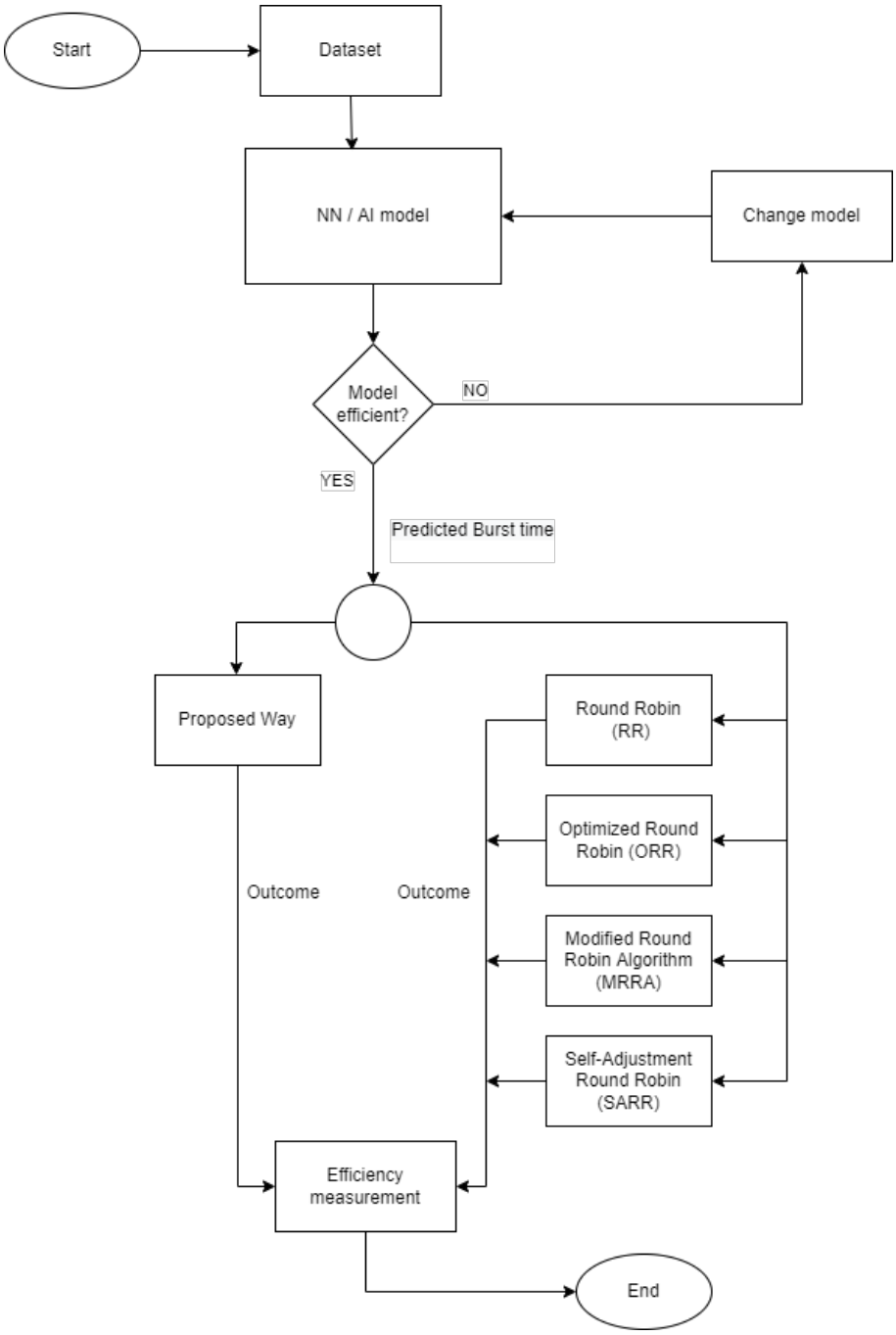


Figure 3.1: Workflow

3.2 Input Data

The input data was collected from the GWA-T-4 AuverGrid dataset. AuverGrid is a production grid platform where there are five clusters geographically located in the AuverGrid region of France. Each cluster has dual 3GHz Pentium-IV Xenons nodes running Scientific Linux. The total number of CPUs is 475 and the number of users is 405. Moreover, the number of jobs in the dataset is traced to be 404176. This dataset and its detailed analysis can be found at GWA-T-4 AuverGrid (tudelft.nl). The dimension of the dataset is 404176x29. We are going to train and test our model with 347611 rows. In this dataset, we take the 'RunTime' column as the target value

as we are predicting the burst/run time of jobs. And all the other columns as feature values.

3.3 Data Preprocessing

Initially, the raw dataset contains 29 columns of which 11 columns contain null values. We drop all these columns: 'JobStructure', 'JobStructureParams', 'UsedNetwork', 'UsedLocalDiskSpace', 'UsedResources', 'ReqPlatform', 'ReqNetwork', 'ReqLocalDiskSpace', 'ReqResources', 'VOID', 'ProjectID' containing the null values. In the dataset, the null values are represented as -1. Moreover, there is one column called 'ReqNProcs' which has one unique value. So, this column would not help in the learning process. So, we discard this column.
Since null values were found in some rows of the 'ReqMemory' column we drop all those rows and keep only rows only with real values.
Furthermore, machine learning models can not work with string values. So, any string value is to be removed or changed. In the AuverGrid dataset, there were categorical string values in the following columns: 'QueueId', 'GroupID', 'ExecutableID', 'OrigSiteID', 'LastRunSiteID', 'UserID' which were encoded with numeric values to avoid disruption in learning. The encoding was done by ranking the unique values by their appearance order in the dataset. The columns names were changed to 'QueueNo', 'GroupNo', 'ExecutableNo', 'OrigSiteNo', 'LastRunSiteNo', 'UserNo'.
Also, 'OrigSiteNo', 'LastRunSiteNo' are fully correlated with each other. So, it was redundant to keep both columns. Therefore, to avoid complexity we removed the 'LastRunSiteID' column. Similarly, 'JobID' was dropped as 'JobID' and 'SubmitTime' columns are totally correlated.

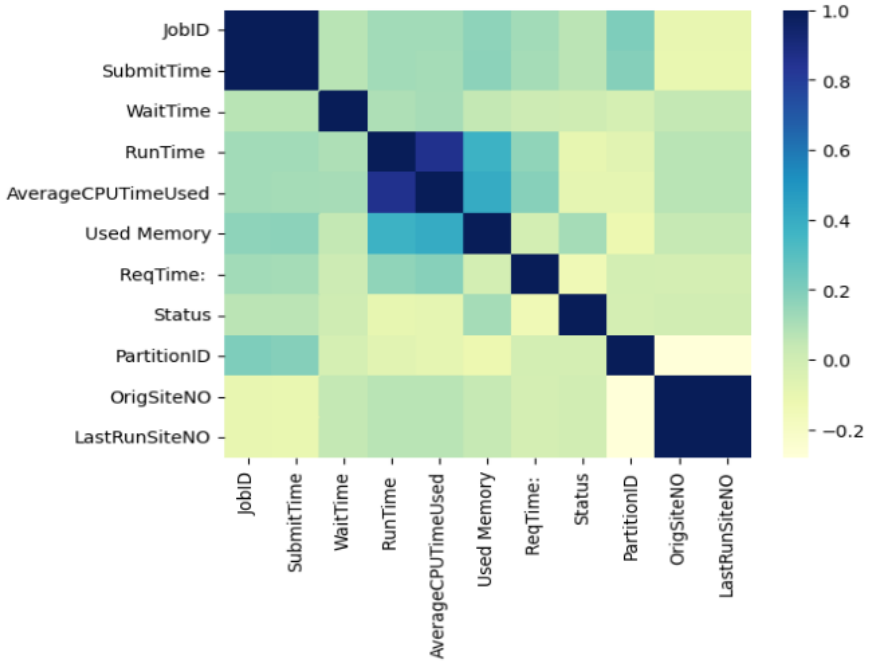


Figure 3.2: Heatmap

Again, we start to figure out which features are more significant in predicting the burst time among all the available features by feature importance measurement.

We find the score for each feature using the linear regression model. The model is trained and tested with each feature to find scores for each corresponding feature. We only keep features whose score is larger than 0.1 and drop all the other columns.

UserNo	0.015196729510913443
LastRunSiteNo	0.5329341568710566
OrigSiteNo	0.5329341568710566
ExecutableNo	0.05874631582963863
GroupNo	0.005515194804284196
QueueNo	0.002129171205123015
PartitionID	0.5455940117741753
Status	0.872746079265907
ReqMemory	0.05597046566995223
ReqTime	2.5662661960198285
ReqNProcs	6.16116002882805e-05
Used Memory	14.046796027826481
AverageCPUTimeUsed	73.58383266335557
NProcs	6.16116002882805e-05
WaitTime	0.9589562873902668
SubmitTime	1.3120721319386353
JobID	1.3977326446768967

Table 3.1: Feature Importance Scores

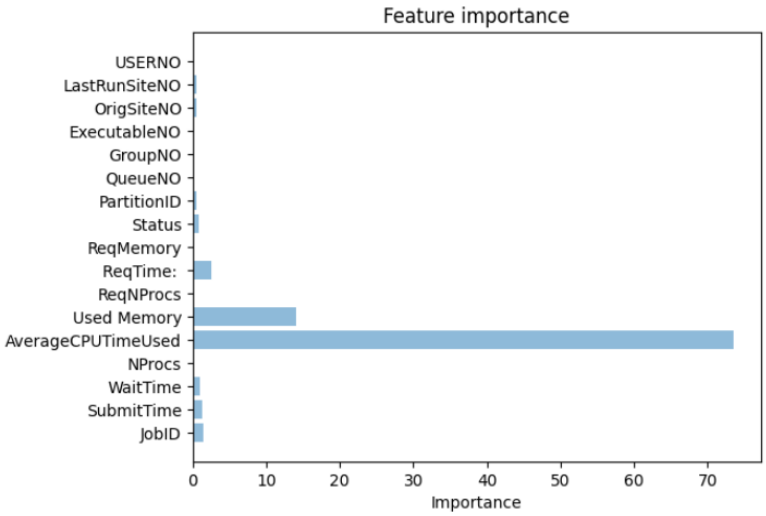


Figure 3.3: Feature Importance Graph

Finally, the resulting dataset had a dimension of 347611x9 where 8 columns were picked as attributes, and 'RunTime' was selected as the target column.

3.4 Implementation

This segment narrates the implementation and result of the proposed scheduling algorithm and the chosen ML and AL models for this work. The whole coding, result analysis and visualization-related workings for this paper were done with Jupyter Notebook (for efficient use of the local resources) although for some cases google collab was also used. The system that is used for this work, has the configuration - processor: AMD Ryzen 7 5800X, ram: 64 GB, GPU: Nvidia GeForce RTX 3060 and operating system: Windows 10. Sometimes a few handwritten calculations were also performed for the manual validation of the proposed algorithm in case of memory and time complexity. The main implementation part of this work is mainly divided into four segments. These are input data preprocessing, prediction of the burst time, choosing the best model, and running the proposed and other existing algorithms on predicted burst time. And the result segment is explicitly focused on comparison and visualization of the outcome and efficiency of the proposed way with some existing and previously proposed algorithms [Traditional Round Robin (RR), Self-Adjustment Round Robin (SARR), Modified Round Robin Algorithm (MRRA) and Optimized Round Robin (ORR)].

3.4.1 Ideology

Most of the process scheduling approaches predict the approximate burst time for a process using the exponential average technique. But in this work, the prediction of the burst time for the proposed algorithm is accomplished via some ML and AI models (KNN, Linear Regression, Decision Tree, etc.). Since it is mostly dependent on Machine Learning and Artificial Intelligence, all of the implementations were done with python 3 on Jupyter Notebook via creating some cells of code for each segment. For instance, cell of code for necessary library importation, cell of code for dataset connection, cell of code for data preprocessing, cell of code for dataset splitting, cell of code for ML, AI model and proposed algorithm, cell of code for outcome visualization.

3.4.2 Input Data Management

The data preprocessing segment of this paper clearly describes the applied data preprocessing techniques for this work in the previous section. For instance, dropping unnecessary data fields, handling null values, finding correlation via the visualization of heat maps, etc.

3.4.3 Burst Time Prediction

Different models of ML and NN work in different fashion for the prediction of the data. For getting appropriate predictions for this work primarily four models and a traditional prediction approach (exponential average) were tested. These are the Linear Regression model (LR), k-nearest neighbor classification model (KNN), Decision Tree model (DT) and Neural Network (NN)[multilayer perceptron-MLP]. For the best fitting of the dataset in these models, the preprocessed dataset is split into 70% training data and 30% testing data. The models have shown different types of behavior in terms of training-validation accuracy and loss and the actual score.

3.4.4 Comparative Analysis of Proposed Way and Other Existing Scheduling Algorithms via Running on the Predicted Burst Time

At this point, the predicted burst times go to the proposed algorithm as well as to other existing and previously proposed algorithms for the execution to determine the 3 comparison parameters; Average Turnaround Time (avg TT) and Average Waiting Time (avg WT) and Context Switch (CS). Turnaround time is the time that is needed to complete the execution of a process and waiting time is the time that is needed for a process to wait to accomplish the whole of its execution. A context switch is a number that shows how many times the CPU is given to the processes throughout the operation. On the basis of these 3 parameters, the efficiency of the proposed way has been estimated. Algorithm 3.1 shows the pseudo-code of the proposed way.
The proposed way works in the below manners:

1. Add incoming process p with associated process_id in process_list
2. Sort the process according to ascending order of the remaining burst time [Sort(process_list)]
3. Add priority process in front of the process_list with process id
4. Calculation of the time quantum:

```
time_quantum = abs(process_list.values()[-1] -
    process_list.values()[0])
if time_quantum == 0 and first round:
     time_quantum = process_list.values()[-1]
if time_quantum == 0:
     time_quantum = time_quantum_previous_round
```

5. Execute process according to time quantum
6. If the process list is empty go to step 7 else go to step 1.
7. End work

The predicted burst times from the decision tree model along with their process ids and priorities go as input to the proposed way as well as in Traditional Round Robin (RR), Self-Adjustment Round Robin (SARR), Modified Round Robin Algorithm (MRRA) and Optimized Round Robin (ORR). And from this section 3 parameters, average turnaround time (avg TT) and average waiting time (avg WT) and the number of the context switch (CS) are deduced for the comparison.
The Existing Models are previously proposed by:

1. Traditional Round Robin [Existing]
2. Modified Round Robin Algorithm (MRRA) (Pradhan and Ray, 2016)
3. Self-Adjustment Round Robin (SARR) (Matarneh and Rami, 2009)
4. Optimized Round Robin (ORR) (Biswas and Saha, 2018)

```
function proposed_algo(burst_time, process_id, arrival_time,
 priority_list)
    process_list ← {};
    previous_time_quantum ← 0;
    flag ← True;
    for e ∈ burst_time do add process p with associated process_id in
     process_list;
    while process_list ≠ ∅ do
        for new_process ∈ burst_time do add process p with associated
         process_id in process_list;
        Sort(process_list);
        if priority_list ≠ ∅ then add priority process in front of the
         process_list with process_id;
        time_quantum ←Abs(process_list.values()[−1] −
         process_list.values()[0]);
        if time_quantum = 0 and flag = True then
            time_quantum ← process_list.values()[−1];
            flag = False;
        end
        if time_quantum = 0 then
         time_quantum ← previous_time_quantum;
        for p ∈ process_list do
            provide CPU to process p for time_quantum unit of time;
            if process execution is done then
                remove it from process_list;
                if p ∈ priority_list then remove it from priority_list;
            else
                decrease burst_time of p for time_quantum unit from
                 process_list;
            end
        end
    end
end
```

Algorithm 3.1: Pseudo-code of Proposed Way

Architecture of the Compared Existing and Proposed Algorithms to the Proposed Way

Round Robin (RR) The process that takes less time to complete suffers, and waiting times are frequently fairly long in the traditional round robin algorithm also I/O-bound processes are preferred over CPU-bound ones. The initial process in this case will receive the CPU before any other processes, and only after the first process has completed running. Also, if additional processes have smaller burst times than the initial process, the processes will have to wait more frequently than necessary, which will increase the average waiting time, or the Convoy effect. Device and CPU utilization are reduced as a result of this effect in the traditional round robin. For time-sharing systems, where it is crucial that each user receive a share of the CPU at regular intervals, the FCFS algorithm is particularly problematic. Time slices are sometimes called as time quantum and usually refer to the method of managing all processes equally and in a circular order (also known as cyclic executive). Scheduling using a round-robin is straightforward, simple to implement, and starvation-free. Other scheduling issues, such as data packet scheduling in computer networks, can be solved with round-robin scheduling. Also in the traditional round-robin algorithm, one queue is maintained without sorting, And repeatedly the time slice is assigned to the remaining process by preemption and again appended to the queue with the remaining time slice.

Self-Adjustment Round Robin (SARR) It is a modified algorithm of classical Round Robin to dynamically adjust the time quantum according to the burst time of the processes in the ready queue. The optimal time quantum is calculated by finding the median[15,16] in each iteration for an existing set of processes in the ready queue; to avoid the overhead of context switching the median is modified to 25 if it is less than 25. The value for time quantum Q consequences for the median is represented by formula 1.

$$Q = \tilde{x} \equiv \begin{cases} Y_{(N+1)/2} & \text{if} N \text{is odd} \\ \frac{1}{2}(Y_{N/2}) + (Y_{1+N/2}) & \text{if} N \text{is even} \end{cases} \tag{3.1}$$

The SARR mainly checks if there is any arrival of process in the queue or not and if the initial process status is 0 it assigns a new counter value for that particular process and if it's false it also goes for the median calculation of the burst time and keeps assigning the median as long as there are process in the queue and when the ready queue becomes empty it leaves the algorithm. There will be more unneeded context switches regardless of whether the process is by itself in the ready queue or not, whereas this issue does not exist at all in the new suggested algorithm because in this scenario the time quantum will be equal to the process's remaining burst time.

Modified Round Robin Algorithm (MRRA) The modification to the CPU scheduling method is the main focus of the suggested algorithm. When compared to other scheduling calculations including plain Round Robin booking calculations, the method significantly reduces waiting times and turnaround times. With a small change, the suggested calculation is comparable as of yet. Instead of using the FCFS's standard Round-Robin computation, it conducts the most limited activity

with the least amount of wasted time first. It also makes use of smart rather than a static time quantum. The approach determines the Smart time quantum itself as per the burst time all factors considered, as opposed to providing static time quantum in the CPU scheduling time. The proposed calculation eliminates the errors that result from implementing fundamental round robin engineering. Each procedure is organized in the expanding request of CPU burst time throughout the primary phase of the creative calculation CPU scheduling method. It means that the need will inevitably be subordinated to the processes. The necessity for a method with a high blast time is greater for processes with low blast times. Then, the calculation then determines the average CPU burst time of the number of processes in the second stage. It will gradually determine the time quantum, i.e. (normal of mean and most elevated burst time), after determining the mean. Then, in the final stage of the algorithm, choose the main process from the prepared line and give it the CPU for up to one Smart time quantum.

Optimized Round Robin (ORR) The maximum difference between two adjacent processes has been presented as the basis for an optimized Round Robin algorithm. The goal is to create a process that is more effective than the previous Round Robin. The suggestion is to classify all processes according to Burst Time in ascending order to achieve effective resource allocation. After that, it had been computed and measured how much could differ between two neighboring processes. The calculations for time quantum were accurate. Time quantum is defined as the largest difference between two adjacent processes plus the burst time of the first process following sorting. The optimized RR algorithm is shown in Algorithm 1. As input, the algorithm uses burst time to find the turn-around-time, waiting-time and context-switching. Inputs and outputs have been clearly stated in the algorithm. The required sorting of all inputs is presented by the maximum difference between two adjacent processes has been calculated. Then, the optimized time quantum is estimated in the algorithm. The entire program has been implemented in a preemptive manner. Finally, the outputs such as turn-around time and waiting time were obtained.

Chapter 4

Results and Analysis

4.1 Accuracy Analysis of Burst Time Prediction

4.1.1 Exponential Average

Traditionally, the old scheduling algorithms use this approach for a rough prediction of the burst time of the incoming processes in the ready queue. It is basically a well defined equation.
Exponential Average: $T_{n+1} = \alpha t_n + (1 - \alpha)T_n$
where, α = is a smoothing factor and $0 \leq \alpha \leq 1$, t_n = actual burst time of nth process and T_n = predicted burst time of nth process. This method scores about 26.08% of accuracy for the burst time prediction on the chosen dataset.

4.1.2 Linear Regression Model

Linear regression is a supervised ML algorithm that finds the best linear equation that finds the correlation between the dependent variable & the explanatory (independent) variable.
Linear Regression Architectural Mechanism:

1. Random weights are chosen
2. According to that hypothesis mean square error function is calculated
3. The Gradient descent function is used to change the weights
4. The third step is repeated until the mean square error is minimized
5. The weight is updated accordingly
6. When the loss function is stable the optimization is done

Linear regression is best used with two continuous variables relationship finding. It is often used in medical sectors like finding weight given height and medicine dosage. Linear regression fails to work with non-linear, non-additive datasets.
This ML model has performed quite well since the data of the dataset are continuous in fashion as well as there is no cluster. Figure 4.2a shows the train and validation accuracy and figure 4.2b shows the train and validation loss of the linear regression

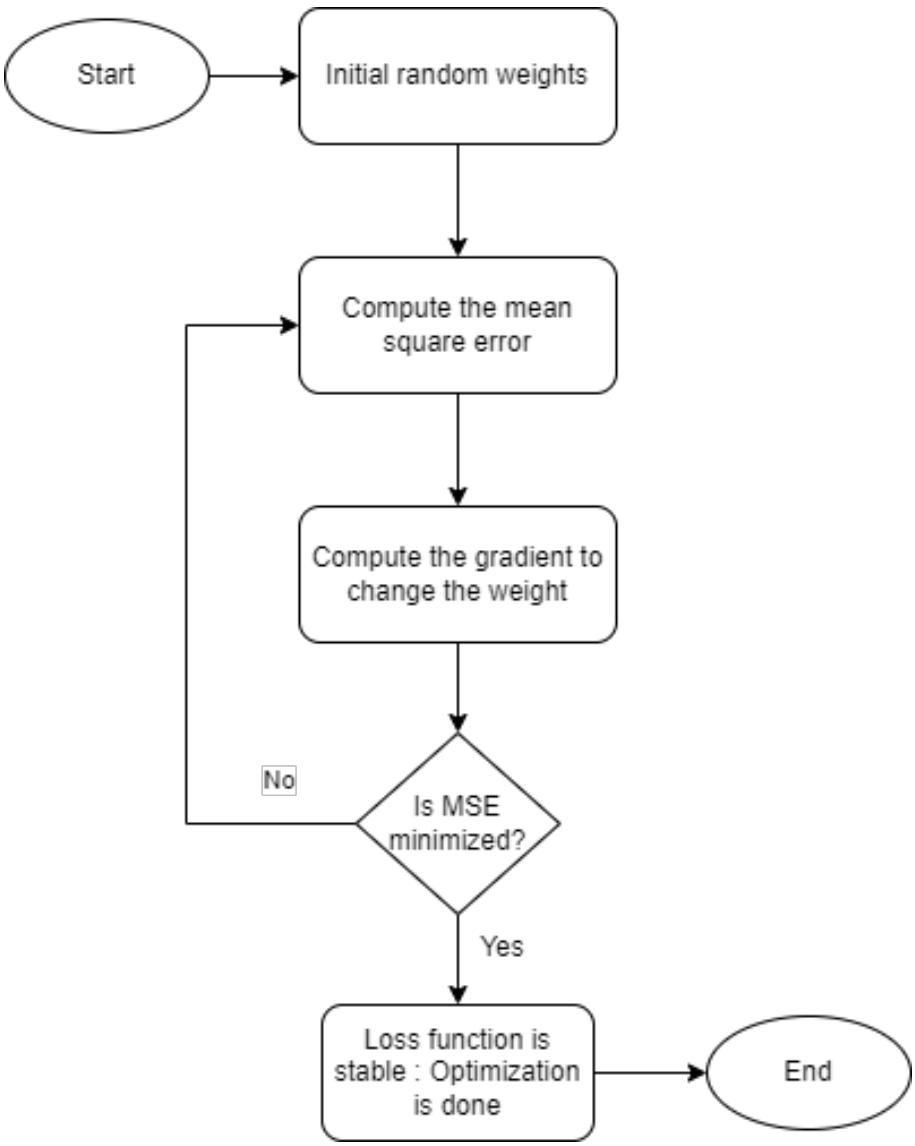
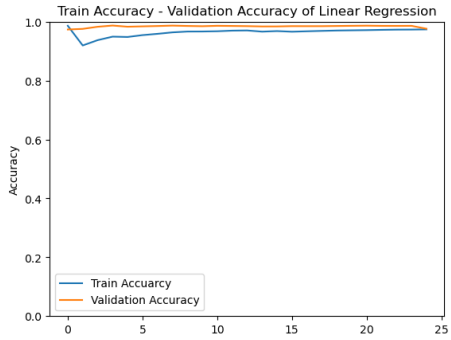
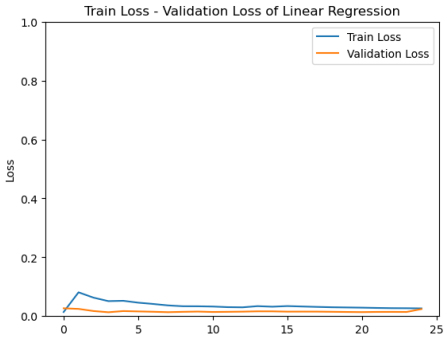


Figure 4.1: Linear Regression Flowchart

model for this work. Figure 4.2a points out that the validation accuracy dominates over train accuracy for the first few iterations where the model is fit and scored depending upon a small number of data points. But with the increment of the data points, the accuracy varies and at point 40 along the X-axis the validation accuracy has fallen dramatically below the train accuracy rate. A similar pattern goes for the training and validation loss curve but in a mirror reflection fashion in terms of the X-axis that is visualized in figure 4.2b.



(a) Train Validation Accuracy of LR



(b) Train Validation Loss of LR

Figure 4.2: Train Validation Accuracy and Loss of LR

Below figure 4.3 highlights the scatter plotting of the actual burst time and the predicted burst time for the Linear Regression Model. The blue points signify the actual burst time from the dataset and the red points signify the predicted burst time by this model. The overlapping segment of blue and red colors explicitly highlights the perfect prediction area via this model. The actual accuracy score of the regression model was more than 97.96% for this work in terms of predicting burst time.

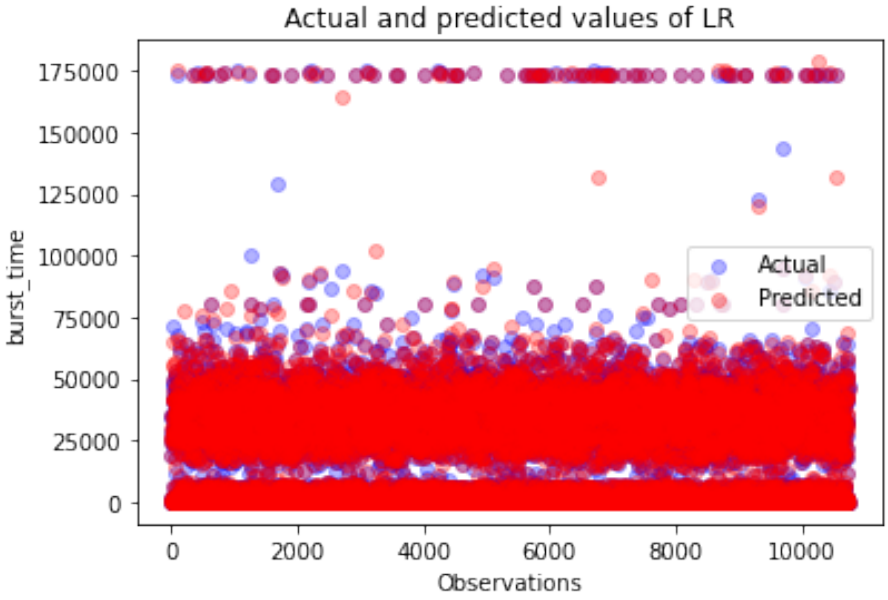


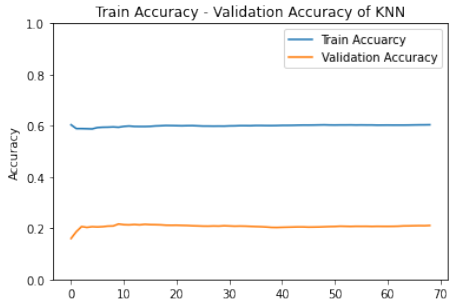
Figure 4.3: Actual-Predicted Data Points of LR

4.1.3 K-Nearest Neighbour (KNN)

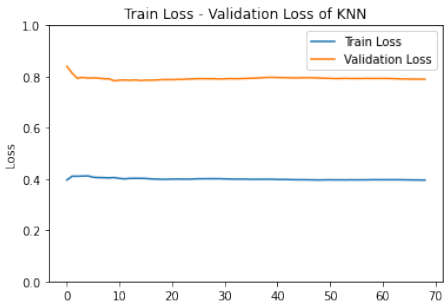
K-Nearest neighbor is an elementary supervised ML algorithm used in both classification & regression problems.
KNN Architecture Mechanism:

1. The dataset is chosen.
2. The value of k in a specific neighborhood of the dataset.
3. Euclidean distance between the currently chosen k and its closest neighbors are calculated.
4. Put the calculated distances in a sorted list.
5. Select top k from the list.
6. If it is regression, then the mean of the k labels ia returned.
7. If it is classification, then the mode of the k labels is returned.

KNN is best used when labeled data isn't available and it doesn't work well when the dataset is too large. It is best for instances like handwriting, image & video detection.
This model has not performed well for this work as this is a non-parametric supervised machine learning. Figure 4.5a shows the train and validation accuracy and figure 4.5b shows the train and validation loss of the KNN model for this work. Figure 4.5a points out that the training accuracy dominates over validation accuracy from the start to the end. Moreover, with the increment of the data points both training and validation accuracy remains constant in manner. A similar pattern goes for the training and validation loss curve but in a mirror reflection fashion in terms of the X-axis that is visualized in figure 4.5b.



(a) Train Validation Accuracy of KNN



(b) Train Validation Loss of KNN

Figure 4.5: Train Validation Accuracy and Loss of KNN

Below figure 4.6 highlights the scatter plotting of the actual burst time and the predicted burst time for the KNN Model. The blue points indicate the actual burst time from the dataset and the red points indicate the predicted burst time by this model. The overlapping segment of blue and red colors explicitly highlights the

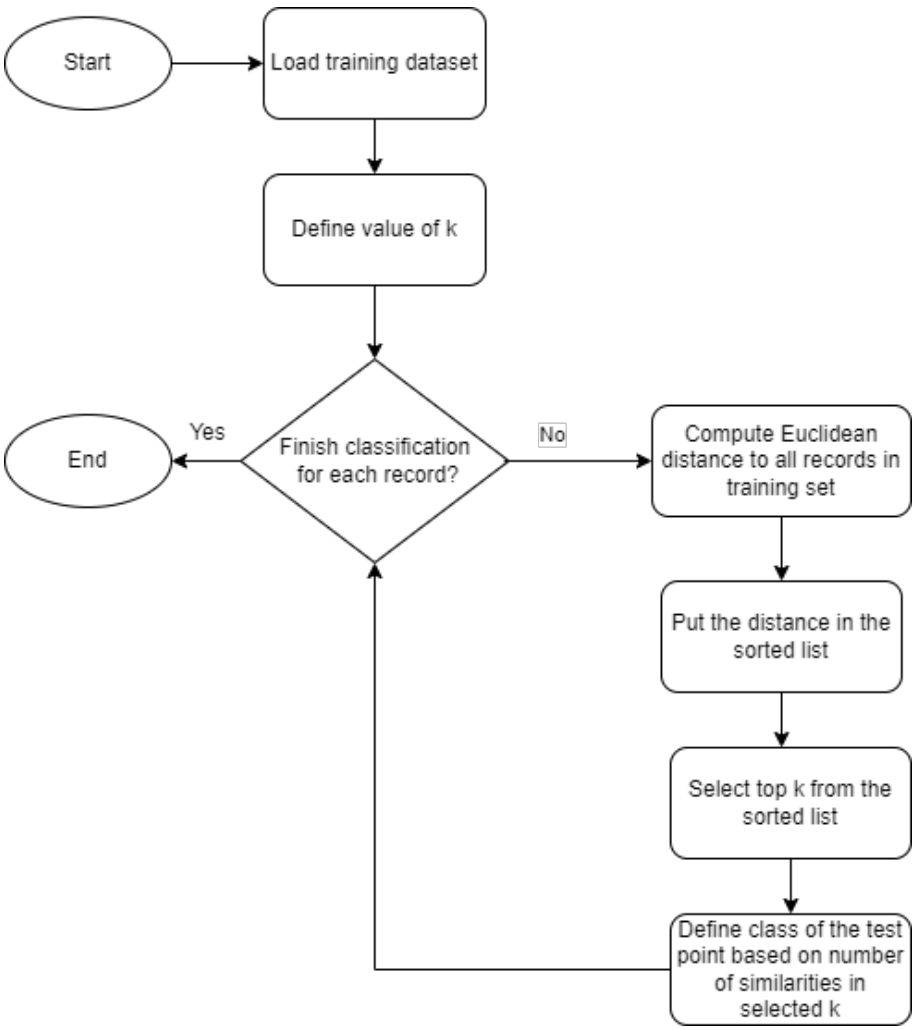


Figure 4.4: KNN Architecture Flowchart

perfect prediction area via this model. The actual accuracy score of the regression model was approximately 17% for this work in terms of predicting burst time.

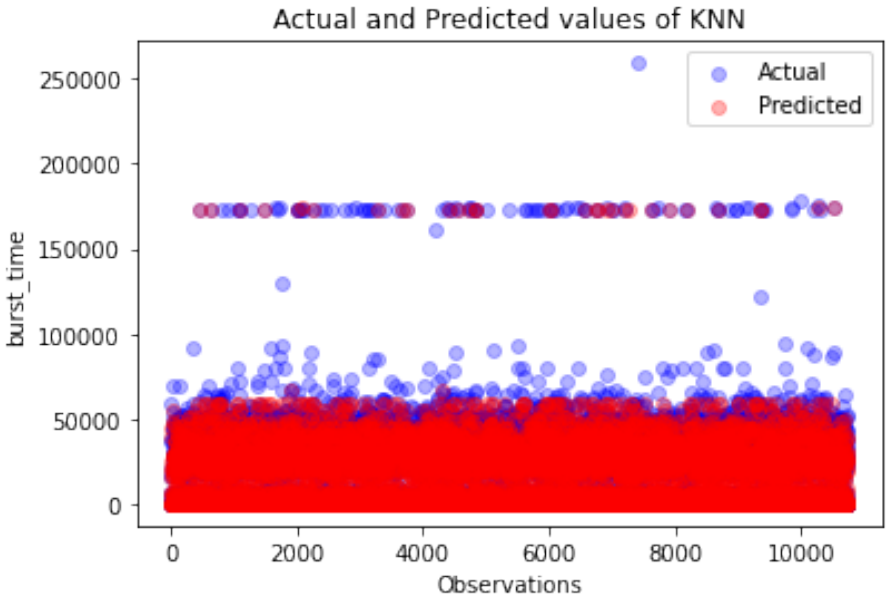


Figure 4.6: Actual-Predicted Data Points of KNN

4.1.4 Neural Network (NN)

An artificial neural network (ANN) or simply, neural network (NN) is a directed weighted graph of neurons. A neuron takes inputs from other nodes via edges in the graph, which could be the inputs to the neural network or other neurons. Each edge has an associated weight. The weighted sum of the inputs is calculated to produce the transfer function. Finally, the transfer function is passed to the activation function to produce the output of the neuron.
Transfer function:

$$v_j = \sum_{i=1}^{n} w_i x_i \tag{4.1}$$

Common activation functions:

$$\text{ReLU}(v_j) = \max(0, v_j) \tag{4.2}$$

$$\sigma(v_j) = \frac{1}{1 + e^{-av_j}} \tag{4.3}$$

A NN may have one or more layers. A NN with a single layer is called a perceptron. On the other hand, a NN with more than one layer is called a multilayer perceptron (MLP).
In an MLP, there are two kinds of layers, hidden layers and the output layer. At first, the inputs are processed via one or multiple hidden layers. Finally, the output layer produces the final output of the MLP.
A MLP is trained through the backpropagation algorithm. It is discussed as follows:

1. The NN is run in a feed-forward fashion and the error of each neuron of the output layer is calculated.

2. The errors are passed backward in the NN, from the output layer to the first hidden layer, by recursively calculating the local gradient of each neuron. The gradients are used to update the weights of the neurons.

One drawback of traditional fully connected NN is that the number of parameters increases exponentially with the number of inputs. This can be mitigated by using convolutional neural networks (CNN). Another drawback is that it cannot detect sequences in the data. This can be mitigated by using recurrent neural networks (RNN).
Multi-layer perceptron (MLP) model is used for this work. But this model is not efficient enough for the chosen dataset. Here for this work, the NN model was built with 20 hidden layers with a learning rate of 0.001. And the score for the prediction of the CPU burst is only 26.01%.

4.1.5 Decision Tree

The decision tree is a supervised learning model based on the tree data structure. It is a way to represent a function that maps from a vector of attributes to a single output value called the "decision." The inputs and outputs can have discrete or continuous values. The properties of a decision tree are:

1. Each non-leaf node represents a test to be performed on an attribute.

2. Each arc from a non-leaf node represents a specific value of the attribute.

3. Each leaf node represents a single output label.

Since the decision tree is a very powerful and expressive machine learning model, it is prone to overfitting. To mitigate this problem, the training algorithm should build a shallow, compact tree by splitting the tree at the most important attributes. For this reason, the model should take into account entropy and information gain. Entropy is a measure of uncertainty and impurity in a dataset. It is given by:

$$\mathrm{H}(P) = \sum_{i=1}^{n} -p_i \log_2 p_i \tag{4.4}$$

On the other hand, information gain is calculated as the reduction of entropy of the label if the decision tree were to be split on a specific attribute. It is a measure of the importance of the attribute. It is given by:

$$\mathrm{Gain}(S, A) = \mathrm{H}(S) - \sum_{i=1}^{n} p(A_i)\, \mathrm{H}(S|A_i) \tag{4.5}$$

The training process of the decision tree is given below: If all the examples of the dataset have the same classification, then return that classification. Else find the attribute with the highest information gain. The tree is split on the attribute with the highest information gain. For each value of the attribute, recursively perform the training algorithm for the subtree.

As we have already mentioned, the decision trees are susceptible to overfitting. Therefore, it should not be used when the dataset is small. Pruning can be used to combat this overfitting in decision tree.

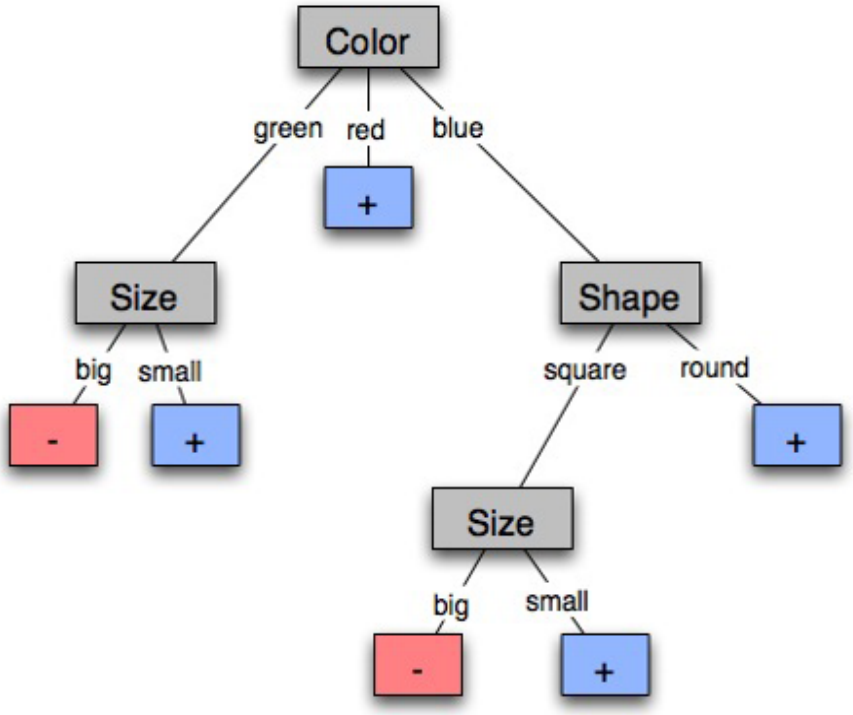
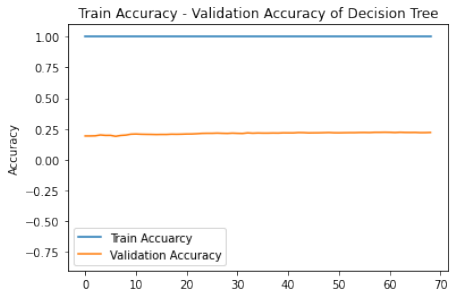
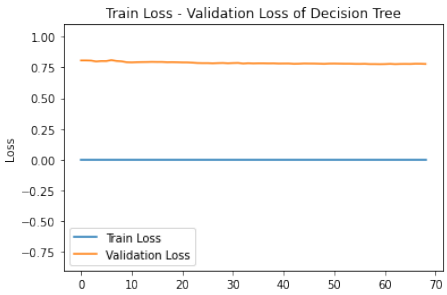


Figure 4.7: Decision Tree Diagram [1]

This model has performed better than the Linear regression model and KNN model for this work. Actually, the decision tree model uses a set of algorithms for making decisions to split a node into required sub-nodes. And the increment of these sub-nodes is directly responsible for the increasing homogeneity of the developed sub-nodes. Figure 4.8a shows the train and validation accuracy and figure 4.8b shows the train and validation loss of the decision tree model for this work. From figure 4.8a it is clear that the accuracy of training is 100% for the data set of this work. It also shows that the training accuracy dominates over the validation accuracy along with the increment of the data points. A similar pattern goes for the training and validation loss curve but in a mirror reflection fashion in terms of the X-axis that is visualized in figure 4.8b.



(a) Train Validation Accuracy of DT



(b) Train Validation Loss of DT

Figure 4.8: Train Validation Accuracy and Loss of DT

Below figure 4.9 highlights the scatter plotting of the actual burst time and the predicted burst time for the Decision Tree Model. The blue points indicate the actual burst time from the dataset and the red points indicate the predicted burst time by this model. The overlapping segment of blue and red colors explicitly highlights the perfect prediction area via this model. The actual accuracy score of the regression model was more than 98.64% for this work in terms of predicting burst time.

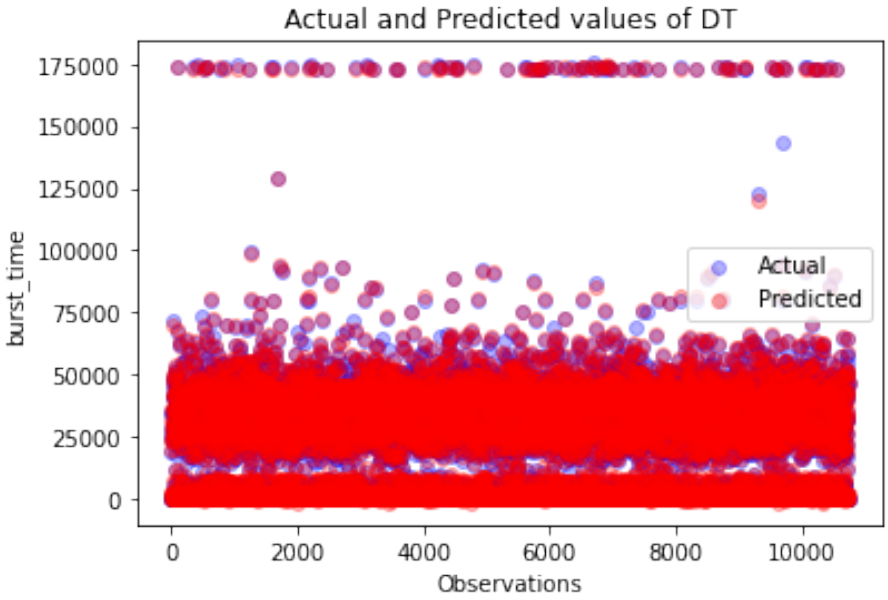


Figure 4.9: Actual-Predicted Data Points of DT

From the above visualization, it is clear that because of the linearity of the dataset for this work the accuracy and loss curves have shown quite a similar sort of curve (straight line in type) without any dramatic increasing and decreasing peak and zig-zag.

4.1.6 Choosing the Best ML/AI Model

Selection of the best model mainly depends on the score of that model for the predefined dataset for any work. For this work and dataset, Decision Tree performs better over exponential average, Linear Regression, KNN, and MLP withholding the

prediction accuracy score of more than 98.64%. Figure 4.10 shows the comparison of the score of the exponential average, linear regression and KNN with the decision tree. Table: 4.1 and Figure 4.10: points the accuracy of the examined ML, NN models.

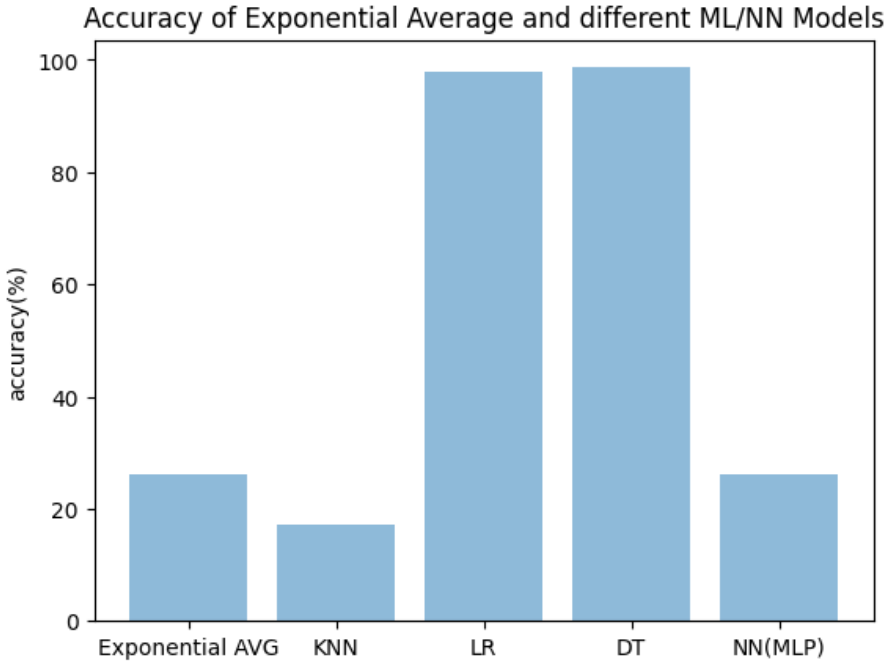


Figure 4.10: Score of Exponential Average, KNN, LR, DT, and NN.

Model Name	Accuracy
K-Nearest Neighbor (KNN)	17.1%
Linear Regression (LR)	97.96%
Decision Tree (DT)	98.64%
Neural Network (MLP)	26.01%

Table 4.1: Score of Exponential Average, KNN, LR, DT, and NN.

4.2 Scheduling Efficiency Analysis of Proposed Algorithm with Others

For the measurement of the efficiency of the proposed algorithm, 10000 processes were put as input for both the proposed method and Traditional Round Robin (RR), Self-Adjustment Round Robin (SARR), Modified Round Robin Algorithm (MRRA) and Optimized Round Robin (ORR) algorithm where a general processor is given 10 to 200 processes at a time by the OS. 5 algorithms run in the same environment and under the same configuration. The final outcome points out that the proposed algorithm ensures almost half of the turnaround time and the waiting time is less than the other 4. Also it ensures less number of context switches. Below Table 4.2 shows the resultant output of the 5 algorithm and figure 4.11, figure 4.12 and

figure 4.13 consecutively shows the comparison of the average turnaround time and waiting time and the number of context switches of the proposed algorithm to the others.

Algorithm	Average Turnaround Time (avg AT)	Average Waiting Time (avg WT)	Number of Context Switch (CS)
Proposed Way	40331930.48	40312117.96	20002
Traditional Round Robin (RR)	87194390.98	87174578.46	28964
Self-Adjustment Round Robin (SARR)	72398064.70	72378252.18	39956
Modified Round Robin Algorithm (MRRA)	84924105.36	84904292.84	25208
Optimized Round Robin (ORR)	78508779.73	78488967.20	22470

Table 4.2: Comparison of the CPU Scheduling Algorithms

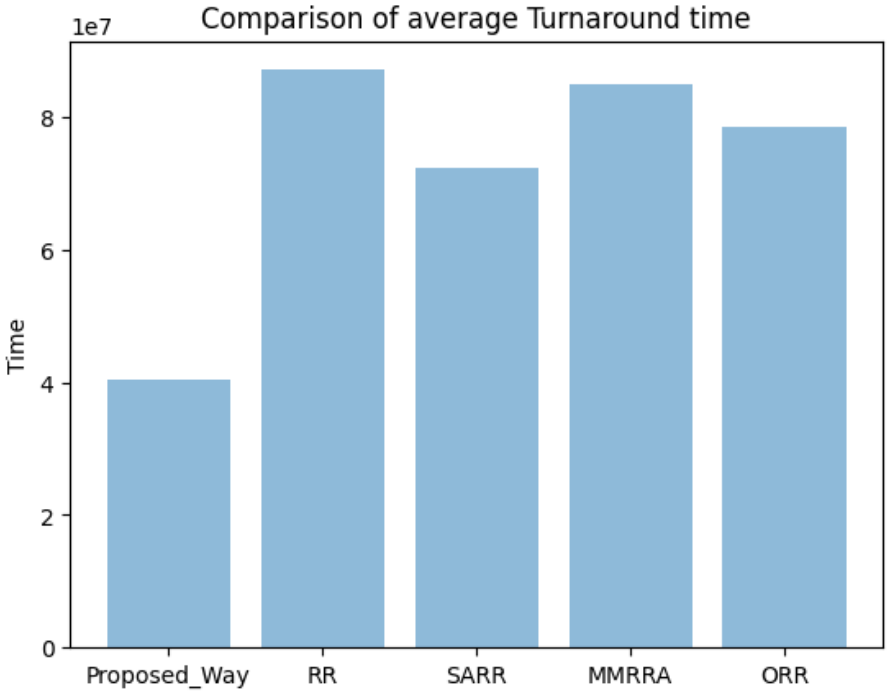


Figure 4.11: Comparison of the average turnaround time

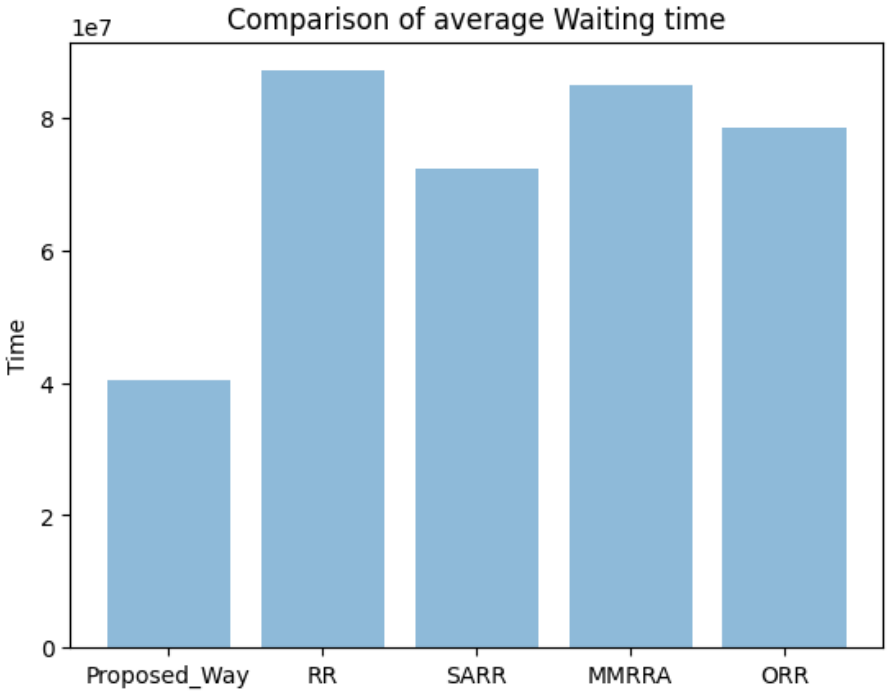


Figure 4.12: Comparison of the average waiting time

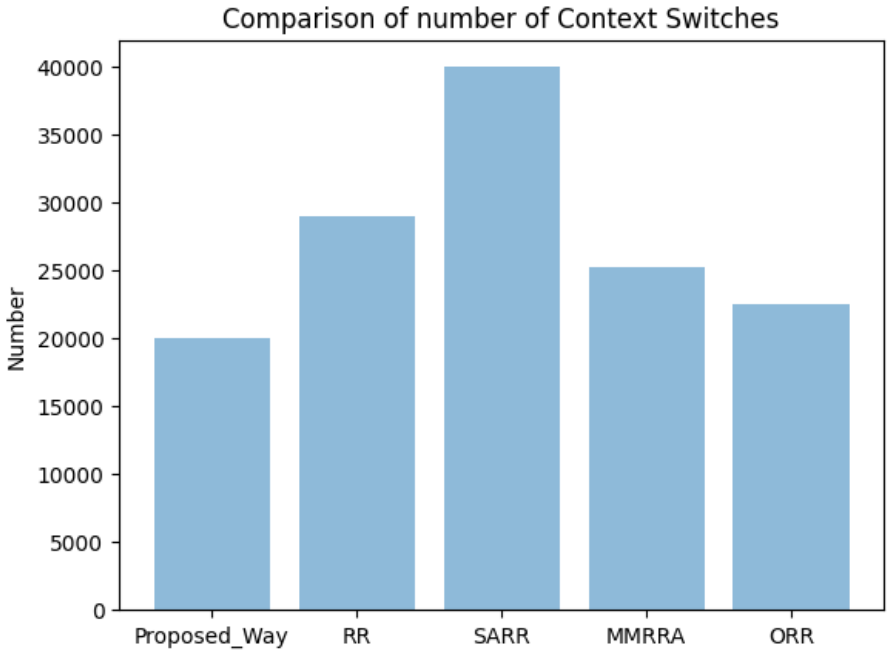


Figure 4.13: Comparison of number of context switches

From the above illustration, it is clear that the proposed algorithm is almost 2 times faster than the other existing and proposed algorithm in terms of process scheduling under a huge load of processes. Also, it can be inferred that although the sorting mechanism of the proposed algorithm create some time complexity, the high performance of the algorithm makes this time complexity negligible.
For better illustration, the proposed algorithm and the other 4 have run on 5000 processes depending upon 10 consecutive epochs wherein each epoch the process

list gets 500 more randomized processes than the previous step. Here Table 4.2.2 illustrates the outcomes of this extensive consecutive test.

Table 4.3: Result of Consecutive Test

Number of Process	Statistic	Proposed Way	SARR	MRRA	ORR	RR
500	avg TT	424.52	458.83	435.18	447.83	454.698
	avg WT	420.08	454.39	430.74	443.39	450.256
	CS	1000	1314	1080	1152	3971
1000	avg TT	7936.57	13925.30	10348.01	14863.83	17353.17
	avg WT	7907.10	13895.84	10318.55	14834.36	17294.24
	CS	2000	3668	2858	9650	5784
1500	avg TT	20267.64	29815.32	23697.23	39071.35	42786.14
	avg WT	20227.01	29774.70	23656.60	39030.73	42704.90
	CS	3000	5526	4222	22354	8229
2000	avg TT	34832.50	49129.28	36845.24	67781.12	74717.89
	avg WT	34783.84	49080.61	36796.58	67732.46	74620.56
	CS	4000	7498	4930	39998	10333
2500	avg TT	52547.49	73284.80	53914.47	101508.70	113031.21
	avg WT	52483.36	73220.68	53850.34	101444.57	112902.95
	CS	5000	9734	5634	41152	12474
3000	avg TT	81761.10	114067.22	89648.77	149688.48	184494.64
	avg WT	81652.24	113958.37	89539.91	149579.63	184276.93
	CS	6000	11866	7332	29092	14595
3500	avg TT	157316.81	228038.63	193686.82	286097.33	373350.66
	avg WT	157038.27	227760.08	193408.27	285818.78	372793.56
	CS	7000	13850	9262	46222	16456
4000	avg TT	316794.73	472188.52	428018.72	600110.13	821582.66
	avg WT	316320.99	471714.78	427544.98	599636.39	820635.18
	CS	8000	15448	11738	100616	18630
4500	avg TT	558382.81	834770.91	731754.71	1070574.47	1436216.40
	avg WT	557654.79	834042.88	731026.69	1069846.45	1434760.36
	CS	9000	17650	12596	109958	20848
5000	avg TT	962821.14	1398027.61	1440608.90	1803293.73	2327200.60
	avg WT	961586.58	1396793.04	1439374.33	1802059.17	2324731.47
	CS	10000	19694	15162	120080	22858

Figure 4.14 and Figure 4.15 describe the overall result in terms of average turnaround

time and average waiting time. From table 4.3 and the figures, it is clear that the proposed algorithm always took less time in terms of average turnaround time and average waiting time than the other algorithms in every situation whether it is a small bath of processes or a large batch of processes.

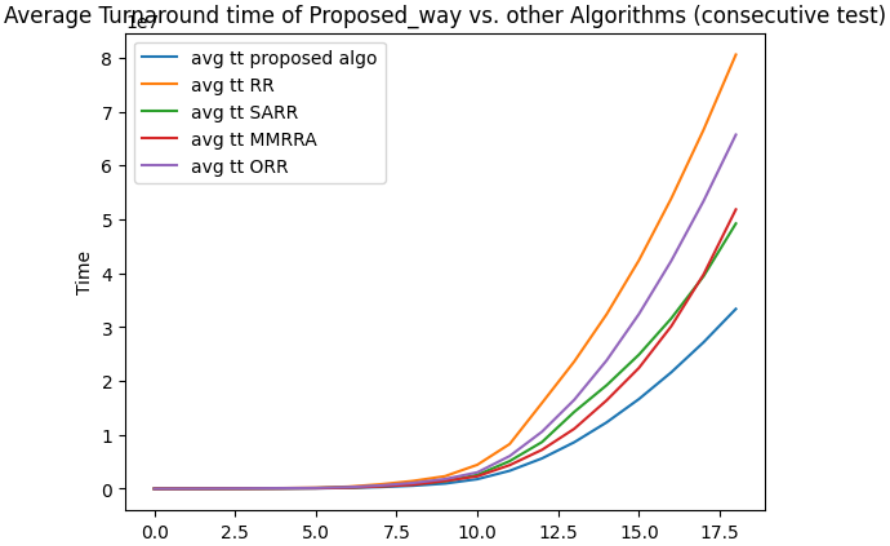


Figure 4.14: Average Turnaround Time (Consecutive Test)

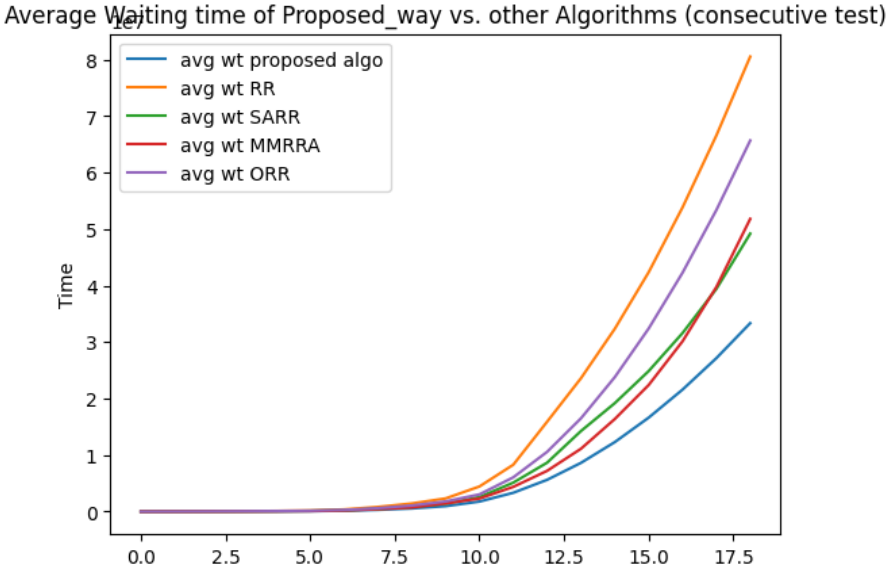


Figure 4.15: Average Waiting Time (Consecutive Test)

Chapter 5

Conclusion

This work will contribute by adding intelligence to the operating system end level of the processor or at least it will be possible to provide necessary information to the processor to take decisions intelligently from the user/software level for the overall optimization. Since we are proposing a new algorithm along with machine learning connectivity for the process scheduling algorithm and evaluating the work done previously. This proposal can be further extended to improved scheduling algorithm implementation along with extending the horizon for new domains in AI and OS. Implementation of learning with the operating system itself is a unique domain and our research will pave the way for future researchers who will be working on this domain.
In the future we would like to continue this research by implementing other AI models and on other different datasets via real implementation of the operating system end.

Bibliography

[1] J. N. Morgan and J. A. Sonquist, "Problems in the analysis of survey data, and a proposal," en, *Journal of the American Statistical Association*, vol. 58, no. 302, pp. 415–434, Jun. 1963, ISSN: 0162-1459, 1537-274X. DOI: 10.1080/01621459.1963.10500855. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/01621459.1963.10500855.

[2] D. Tonogai, "Ai in operating systems: An expert scheduler," EECS Department, University of California, Berkeley, Tech. Rep. UCB/CSD-88-487, Dec. 1988. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/1988/6149.html.

[3] E. Fix and J. L. Hodges, "Discriminatory analysis. nonparametric discrimination: Consistency properties," *International Statistical Review / Revue Internationale de Statistique*, vol. 57, no. 3, p. 238, Dec. 1989, ISSN: 03067734. DOI: 10.2307/1403797. [Online]. Available: https://www.jstor.org/stable/1403797?origin=crossref.

[4] P.-C. Wang and W. Korfhage, "Process scheduling using genetic algorithms," in *Proceedings.Seventh IEEE Symposium on Parallel and Distributed Processing*, San Antonio, TX, USA: IEEE Comput. Soc. Press, 1995, pp. 638–641, ISBN: 9780818671951. DOI: 10.1109/SPDP.1995.530742. [Online]. Available: http://ieeexplore.ieee.org/document/530742/.

[5] J. M. Stanton, "Galton, pearson, and the peas: A brief history of linear regression for statistics instructors," en, *Journal of Statistics Education*, vol. 9, no. 3, p. 3, Jan. 2001, ISSN: 1069-1898. DOI: 10.1080/10691898.2001.11910537. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/10691898.2001.11910537.

[6] K. K. P. and A. Negi, "Characterizing process execution behavior using machine learning techniques," in *In DpROMWorkShop Proceedings, HiPC 2004 International Conference*, 2004.

[7] K.-B. Song, Y.-S. Baek, D. Hong, and G. Jang, "Short-term load forecasting for the holidays using fuzzy linear regression method," en, *IEEE Transactions on Power Systems*, vol. 20, no. 1, pp. 96–101, Feb. 2005, ISSN: 0885-8950. DOI: 10.1109/TPWRS.2004.835632. [Online]. Available: http://ieeexplore.ieee.org/document/1388498/.

[8] S. Lim and S.-B. Cho, "Intelligent os process scheduling using fuzzy inference with user models," in *New Trends in Applied Artificial Intelligence*, H. G. Okuno and M. Ali, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 725–734, ISBN: 978-3-540-73325-6.

[9] O. AlHeyasat. and R. Herzallah., "Estimation of quantum time length for round-robin scheduling algorithm using neural networks," in *Proceedings of the International Conference on Fuzzy Computation and 2nd International Conference on Neural Computation - ICNC, (IJCCI 2010)*, INSTICC, SciTePress, 2010, pp. 253–257, ISBN: 978-989-8425-32-4. DOI: 10.5220/0003058002530257.

[10] R. Kumar, E. R. Kumar, E. S. Gill, and E. A. Kaushik, "Genetic algorithm approach to operating system process scheduling problem," *International journal of Engineering science and Technology*, vol. 2, no. 9, pp. 4247–4252, 2010.

[11] P. V. Araujo, "Classificação automática de processos em sistemas operacionais," 2011.

[12] S. Dolev, A. Mendelson, and I. Shilman, "Semantical cognitive scheduling," Nov. 2012 Technical Report, Tech. Rep., 2012.

[13] V. Chahar and S. Raheja, "Fuzzy based multilevel queue scheduling algorithm," in *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Mysore: IEEE, Aug. 2013, pp. 115–120, ISBN: 9781467362177. DOI: 10.1109/ICACCI.2013.6637156. [Online]. Available: http://ieeexplore.ieee.org/document/6637156/.

[14] M. Sharma, P. Sindhwani, and V. Maheshwari, "Genetic algorithm optimal approach for scheduling processes in operating system," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 14, no. 5, p. 91, 2014.

[15] T. Helmy, S. Al-Azani, and O. Bin-Obaidellah, "A machine learning-based approach to estimate the cpu-burst time for processes in the computational grids," in *2015 3rd International Conference on Artificial Intelligence, Modelling and Simulation (AIMS)*, Kota Kinabalu, Malaysia: IEEE, Dec. 2015, pp. 3–8, ISBN: 9781467386753. DOI: 10.1109/AIMS.2015.11. [Online]. Available: http://ieeexplore.ieee.org/document/7604542/.

[16] H. b. Jaafar, N. b. Mukahar, and D. A. Binti Ramli, "A methodology of nearest neighbor: Design and comparison of biometric image database," in *2016 IEEE Student Conference on Research and Development (SCOReD)*, Kuala Lumpur: IEEE, Dec. 2016, pp. 1–6, ISBN: 9781509029488. DOI: 10.1109/SCORED.2016.7810073. [Online]. Available: https://ieeexplore.ieee.org/document/7810073/.

[17] J. Khatri, "An enhanced round robin cpu scheduling algorithm," *IOSR Journal of Computer Engineering*, vol. 18, no. 04, pp. 20–24, Apr. 2016, ISSN: 22788727, 22780661. DOI: 10.9790/0661-1804022024. [Online]. Available: http://iosrjournals.org/iosr-jce/papers/Vol18-issue4/Version-2/D1804022024.pdf.

[18] H. B. Parekh and S. Chaudhari, "Improved round robin cpu scheduling algorithm: Round robin, shortest job first and priority algorithm coupled to increase throughput and decrease waiting time and turnaround time," in *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, Jalgaon, India: IEEE, Dec. 2016, pp. 184–187, ISBN: 9781509004676. DOI: 10.1109/ICGTSPICC.2016.7955294. [Online]. Available: http://ieeexplore.ieee.org/document/7955294/.

[19] N. Sarwar, N. Aslam, and A. Batool, "Designing a model for improving cpu scheduling by using machine learning," *International Journal of Computer Science and Information Security,*, vol. 14, pp. 201–204, Oct. 2016.

[20] S. Dias, S. Naik, S. K, S. Raman, and N. M, "A machine learning approach for improving process scheduling: A survey," *International Journal of Computer Trends and Technology*, vol. 43, no. 1, pp. 1–4, Jan. 2017, ISSN: 22312803. DOI: 10.14445/22312803/IJCTT-V43P101. [Online]. Available: http://www.ijcttjournal.org/archives/ijctt-v43p101.

[21] X. Feng, Y. Zhou, T. Hua, Y. Zou, and J. Xiao, "Contact temperature prediction of high voltage switchgear based on multiple linear regression model," in *2017 32nd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, Hefei, China: IEEE, May 2017, pp. 277–280, ISBN: 9781538629017. DOI: 10.1109/YAC.2017.7967419. [Online]. Available: http://ieeexplore.ieee.org/document/7967419/.

[22] H. Mora, S. E. Abdullahi, and S. B. Junaidu, "Modified median round robin algorithm (mmrra)," in *2017 13th International Conference on Electronics, Computer and Computation (ICECCO)*, Abuja, Nigeria: IEEE, Nov. 2017, pp. 1–7, ISBN: 9781538624999. DOI: 10.1109/ICECCO.2017.8333325. [Online]. Available: http://ieeexplore.ieee.org/document/8333325/.

[23] M. Tajwar, M. Pathan, L. Hussaini, and A. Abubakar, "Cpu scheduling with a round robin algorithm based on an effective time slice," *Journal of Information Processing Systems*, vol. 13, Jan. 2017. DOI: 10.3745/JIPS.01.0018.

[24] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating system concepts*, eng, Tenth edition. Hoboken, NJ: Wiley, 2018, ISBN: 9781119439257.

[25] D. Hureira and C. Vartanian, "Machine learning and neural networks for real-time scheduling," 2019.

[26] G. S. K. Ranjan, A. Kumar Verma, and S. Radhika, "K-nearest neighbors and grid search cv based real time fault monitoring system for industries," in *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, Bombay, India: IEEE, Mar. 2019, pp. 1–5, ISBN: 9781538680759. DOI: 10.1109/I2CT45611.2019.9033691. [Online]. Available: https://ieeexplore.ieee.org/document/9033691/.

[27] C. Cai and L. Wang, "Application of improved k-means k-nearest neighbor algorithm in the movie recommendation system," in *2020 13th International Symposium on Computational Intelligence and Design (ISCID)*, Hangzhou, China: IEEE, Dec. 2020, pp. 314–317, ISBN: 9781728184463. DOI: 10.1109/ISCID51228.2020.00076. [Online]. Available: https://ieeexplore.ieee.org/document/9325794/.

[28] S. M. Mostafa and H. Amano, "Dynamic round robin cpu scheduling algorithm based on k-means clustering technique," en, *Applied Sciences*, vol. 10, no. 15, p. 5134, Jul. 2020, ISSN: 2076-3417. DOI: 10.3390/app10155134. [Online]. Available: https://www.mdpi.com/2076-3417/10/15/5134.

[29] N. Rochmawati, H. B. Hidayati, Y. Yamasari, *et al.*, "Covid symptom severity using decision tree," in *2020 Third International Conference on Vocational Education and Electrical Engineering (ICVEE)*, Surabaya, Indonesia: IEEE, Oct. 2020, pp. 1–5, ISBN: 9781728174341. DOI: 10.1109/ICVEE50212.2020.9243246. [Online]. Available: https://ieeexplore.ieee.org/document/9243246/.

[30] S. Tehsin, Y. Asfia, N. Akbar, F. Riaz, S. Rehman, and R. C. D. Young, "Selection of cpu scheduling dynamically through machine learning," in *Pattern Recognition and Tracking XXXI*, M. S. Alam, Ed., Online Only, United States: SPIE, Apr. 2020, p. 24, ISBN: 9781510635777. DOI: 10.1117/12.2559540. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11400/2559540/Selection-of-CPU-scheduling-dynamically-through-machine-learning/10.1117/12.2559540.full.

[31] B. Zhang, "Tactical decision system of table tennis match based on c4.5 decision tree," in *2021 13th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, Beihai, China: IEEE, Jan. 2021, pp. 632–635, ISBN: 9781665438926. DOI: 10.1109/ICMTMA52658.2021.00146. [Online]. Available: https://ieeexplore.ieee.org/document/9410189/.

[32] B. Richardson and W. Istiono, "Comparison analysis of round robin algorithm with highest response ratio next algorithm for job scheduling problems," *International Journal of Open Information Technologies*, vol. 10, no. 2, pp. 21–26, 2022.

[33] Sakshi, C. Sharma, S. Sharma, *et al.*, "A new median-average round robin scheduling algorithm: An optimal approach for reducing turnaround and waiting time," en, *Alexandria Engineering Journal*, vol. 61, no. 12, pp. 10 527–10 538, Dec. 2022, ISSN: 11100168. DOI: 10.1016/j.aej.2022.04.006. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1110016822002599.

[34] P. S. Sharma, S. Kumar, M. S. Gaur, and V. Jain, "A novel intelligent round robin cpu scheduling algorithm," en, *International Journal of Information Technology*, vol. 14, no. 3, pp. 1475–1482, May 2022, ISSN: 2511-2104, 2511-2112. DOI: 10.1007/s41870-021-00630-0. [Online]. Available: https://link.springer.com/10.1007/s41870-021-00630-0.