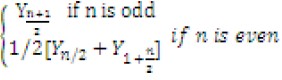
CPU SCHEDULING WITH DYNAMIC TIME

ALLOCATION AND PREDICT BURST TIME USING ML

**II. METHODOLOGY USED**

In our proposed algorithm, we have to arrange the processes in ascending order according to their burst time present in the ready queue. Then the time quantum is calculated. For finding the optimal time quantum, median method is followed. The median can be found out by using the following formulae:

 Median =

y = number located in the middle of a group of numbers

arranged in ascending order

n = number of processes

The standard deviation of the burst times is also a factor used in calculation of the dynamic time quantum.

**III. ALGORITHM**

1. Take the input of the processes and sort them according to their burst times.
2. Take the median and standard deviation of the burst times of the processes.
3. Calculate the dynamic quantum using the formula: Median – 0.5\*(Standard Deviation of burst time)
4. Perform normal round robin scheduling on the given processes.
5. If a process is preempted at the end of the time quanta, send the process to a waiting queue.

1. After all processes are gone through, make the waiting queue into the executing queue and repeat steps 4,5 and 6 until all processes are completed.

**IV. SOURCE CODE (C LANGUAGE)**

#include<stdio.h>

#include<stdlib.h> #include<math.h> struct process { int process\_id; int waiting\_time,turnaround\_time,burst\_time,remaining\_

time; }temp; void ins(struct process \*p,int n) {

int i;

for(i=0;i<n;i++) { printf("Enter the burst time of process P[%d]: ",i+1); scanf("%d",&p[i].burst\_time); p[i].remaining\_time=p[i].burst\_time; p[i].process\_id=i+1;

}

} int fcfs(struct process \*p,int n) {

int i;

int total\_time=0; printf("\nProcess Scheduling Order: "); for(i=0;i<n;i++) { p[i].waiting\_time=total\_time; total\_time+=p[i].burst\_time; p[i].turnaround\_time=total\_time; printf("P[%d], ",p[i].process\_id);

} printf("\n"); display(p,n,total\_time); return 1; } int sjf(struct process \*p,int n) { int i,j;

for(i=0;i<n-1;i++)

for(j=0;j<n-1-i;j++)

if(p[j].burst\_time>p[j+1].burst\_time){ temp=p[j];

p[j]=p[j+1]; p[j+1]=temp;

}

int total\_time=0; printf("\nProcess Scheduling Order: "); for(i=0;i<n;i++) { p[i].waiting\_time=total\_time; total\_time+=p[i].burst\_time; p[i].turnaround\_time=total\_time; printf("P[%d], ",p[i].process\_id);

} printf("\n"); display(p,n,total\_time); return 1; } int rr(struct process \*p,int n) { int time\_quanta=10; int total\_time=0; int n1=n,count=0; printf("\nProcess Scheduling Order: "); while(n1) { if(p[count].remaining\_time<=time\_quanta && p[coun

t].remaining\_time>0){

total\_time+=p[count].remaining\_time; p[count].remaining\_time=0; p[count].turnaround\_time=total\_time; p[count].waiting\_time=total\_time-

p[count].burst\_time;

printf("P[%d], ",p[count++].process\_id); n1--;

}

else if(p[count].remaining\_time>0) { p[count].remaining\_time-=time\_quanta; total\_time+=time\_quanta; printf("P[%d], ",p[count++].process\_id);

} else count++; if(count==n) count=count%n;

} printf("\n"); display(p,n,total\_time); return 1; } int modifiedSch(struct process \*p,int n) { int i,j,x,sum=0,t=n,n1=n,n2,total\_time=0; struct process \*ex\_que=p,\*wt\_que; float mean,median,standard\_deviation=0; for(i=0;i<n-1;i++) for(j=0;j<n-1-i;j++) if(p[j].burst\_time>p[j+1].burst\_time){ temp=p[j]; p[j]=p[j+1]; p[j+1]=temp;

}

for(i=0;i<n;i++) sum+=p[i].burst\_time; mean=(float)sum/n; if(n%2==0) median=(float)(p[n/2].burst\_time); else

median=(float)(p[(n-

1)/2].burst\_time+p[(n+1)/2].burst\_time)/2; for(i=0;i<n;i++) standard\_deviation+=(p[i].burst\_time-

mean)\*(p[i].burst\_time-mean);

standard\_deviation=sqrt(standard\_deviation/n); int time\_quanta=(int)(median-0.5\*standard\_deviation); printf("\nProcess Scheduling Order: "); while (t) { wt\_que=(struct process\*)malloc(sizeof(struct process) \*n1); n2=0; for(i=0;i<n1;i++) {

if(ex\_que[i].remaining\_time<time\_quanta) { total\_time+=ex\_que[i].remaining\_time; x=returnIndex(p,n,ex\_que[i].process\_id);

p[x].turnaround\_time=total\_time; p[x].waiting\_time=total\_time-p[x].burst\_time;

ex\_que[i].remaining\_time=0;

t--; } else{ total\_time+=time\_quanta; ex\_que[i].remaining\_time-=time\_quanta; wt\_que[n2++]=ex\_que[i];

}

printf("P[%d], ",ex\_que[i].process\_id);

}

ex\_que=wt\_que; n1=n2; } printf("\n"); display(p,n,total\_time); return 1; } int returnIndex(struct process \*p,int n,int a) {

int i;

for(i=0;i<n;i++ ) { if(p[i].process\_id==a) return i; } return -1; } int display(struct process \*p,int n,int t) { float avg\_wt=0,avg\_tt=0; int i,j;

for(i=0;i<n-1;i++) for(j=0;j<n-i-1;j++) if(p[j].process\_id>p[j+1].process\_id) { temp=p[j]; p[j]=p[j+1]; p[j+1]=temp;

}

printf("\nProcess\_id\twaiting Time\tTurnaround Time\n"

);

for(i=0;i<n;i++) { }

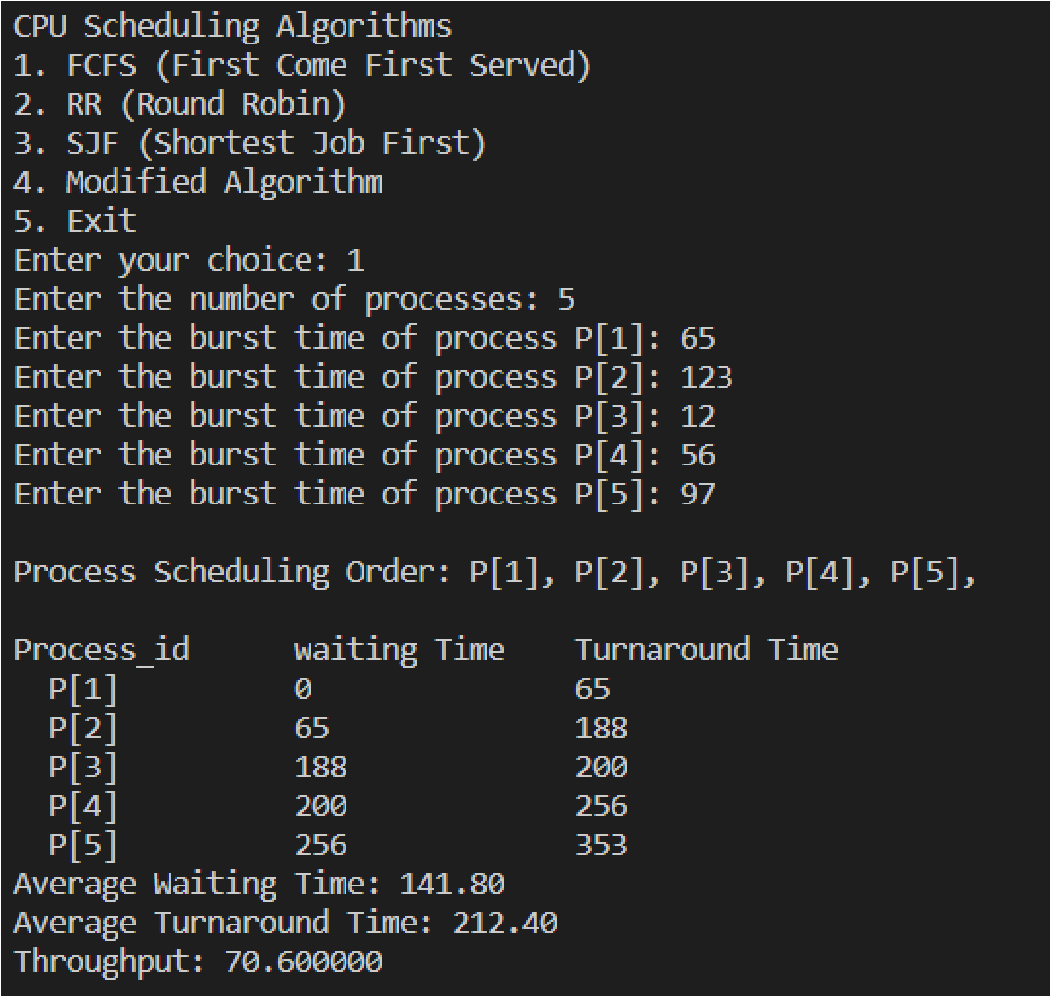
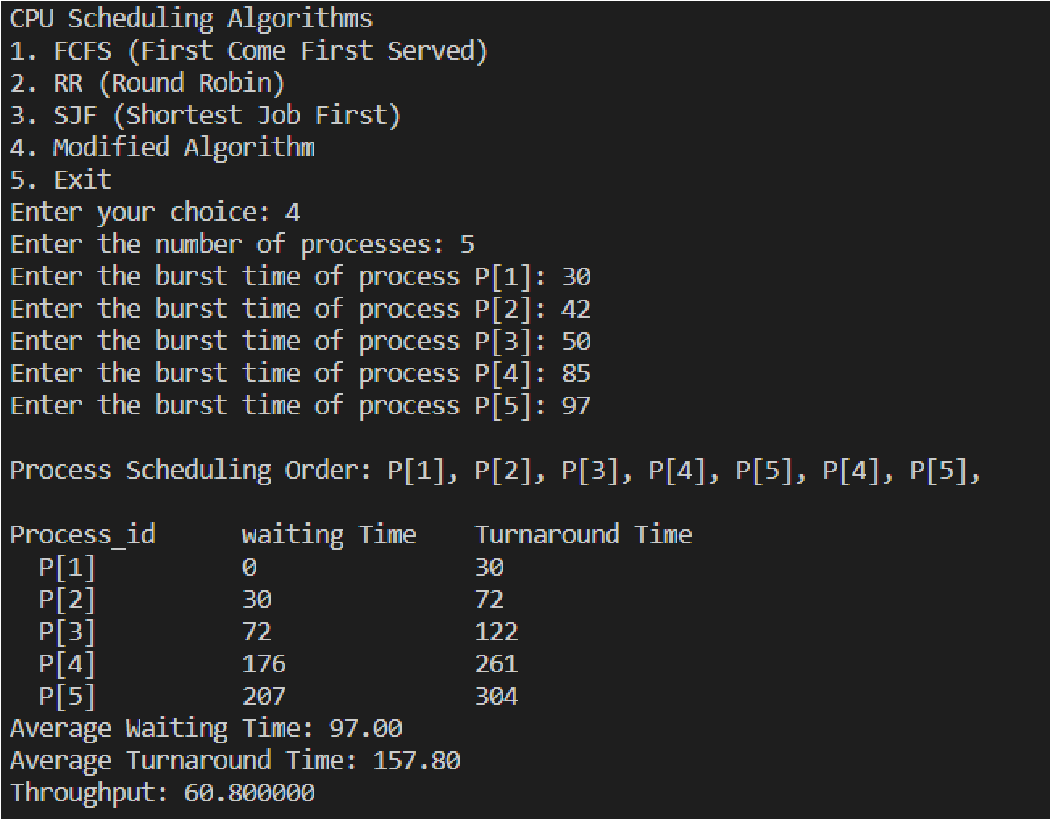
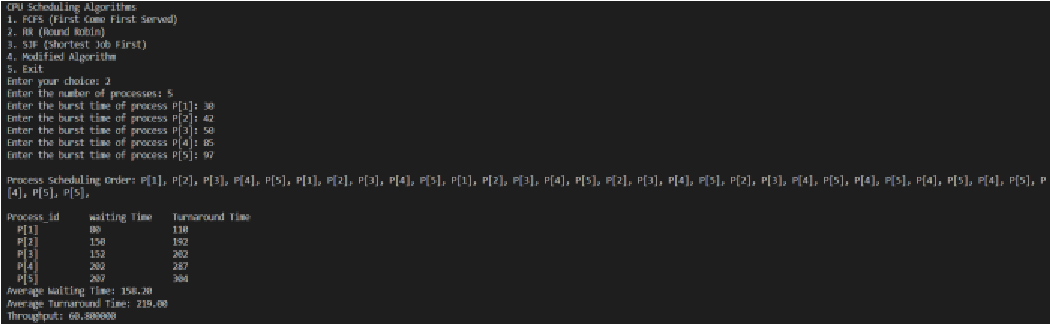
avg\_wt+=p[i].waiting\_time; } avg\_tt+=p[i].turnaround\_time; return 0;

printf(" P[%d]\t\t%d\t\t%d\n",p[i].process\_id,p[i].wait } ing\_time,p[i].turnaround\_time);

}

# **RESULTS**

avg\_wt=(float)avg\_wt/n;  avg\_tt=(float)avg\_tt/n; printf("Average Waiting Time: %.2f\n",avg\_wt); printf("Average Turnaround Time: %.2f\n",avg\_tt); printf("Throughput: %f\n",(float)t/n);



} int main() { int choice,n; while(1) {

printf("\nCPU Scheduling Algorithms\n1. FCFS (First Come First Served)\n2. RR (Round Robin)\n3. SJF (Shortest J ob First)\n4. Modified Algorithm\n5. Exit\nEnter your choice:

"); scanf("%d",&choice); if(choice==5) { printf("\nThank you\n"); exit(0);

}

printf("Enter the number of processes: "); scanf("%d",&n); struct process p[n]; switch(choice) { case 1: ins(p,n); fcfs(p,n); break; case 2: ins(p,n); rr(p,n); break; case 3: ins(p,n); sjf(p,n); break; case 4: ins(p,n); modifiedSch(p,n);

break;

default: printf("Wrong option selected");

For getting the predicted class, iterate from 1 to total number of training data points.

**VI.**

**G**

**RAPHS**

**F**

**OR**

**A**

**NALYSIS**

**VII.**

**I**

**MPLEMENTA**

**TION USING**

**M**

**ACHINE**

**L**

**EARNING**

S

o

, the implementation of CPU scheduling algorithms such as

shortest job first (SJF) and

shortest remaining time first

(

S

RTF

)

is relying on knowing the length of the CPU burst time.

So,

we

will propose a Machine Learning based ap

proach on predicting

the CPU burst time using Support Vector Machines and KNN

algorithm.

KNN CLASSIFICATION

Let’s tak

e a simple example to understand the algorithm.

Following is the spread of red circles and green squares.

The classification of the blue

star has to be predicted.

So,

in

KNN classification K is the nearest neighbour through which we

have to take vote from

. Let’s say K=3 so we see the nearest 3

datapoints from the blue star and the maximum number of

datapoints belong to the red circle so the

blue star belongs to the

red circle class.

**KNN Algorithm:**

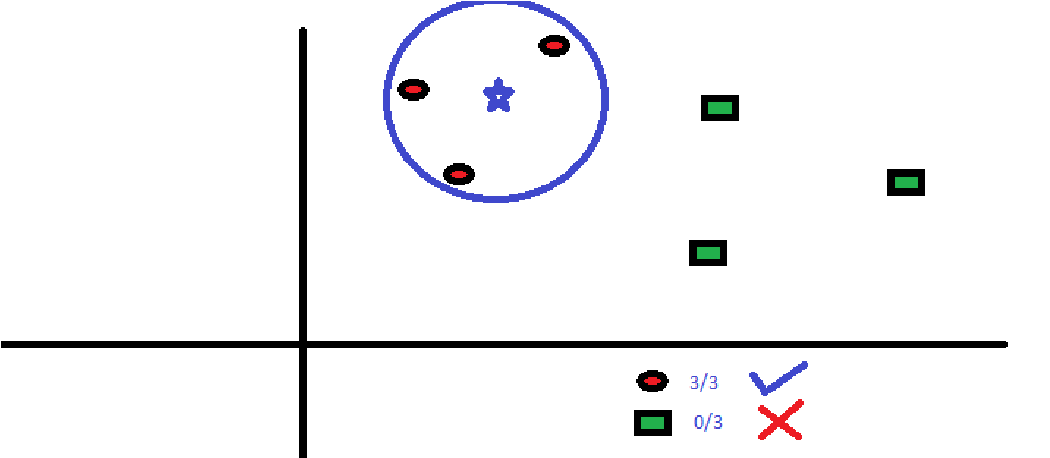
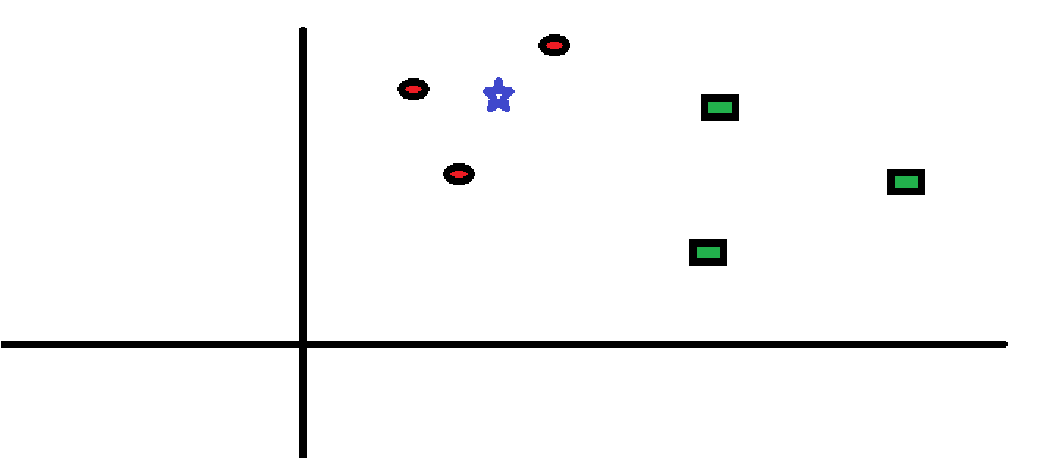
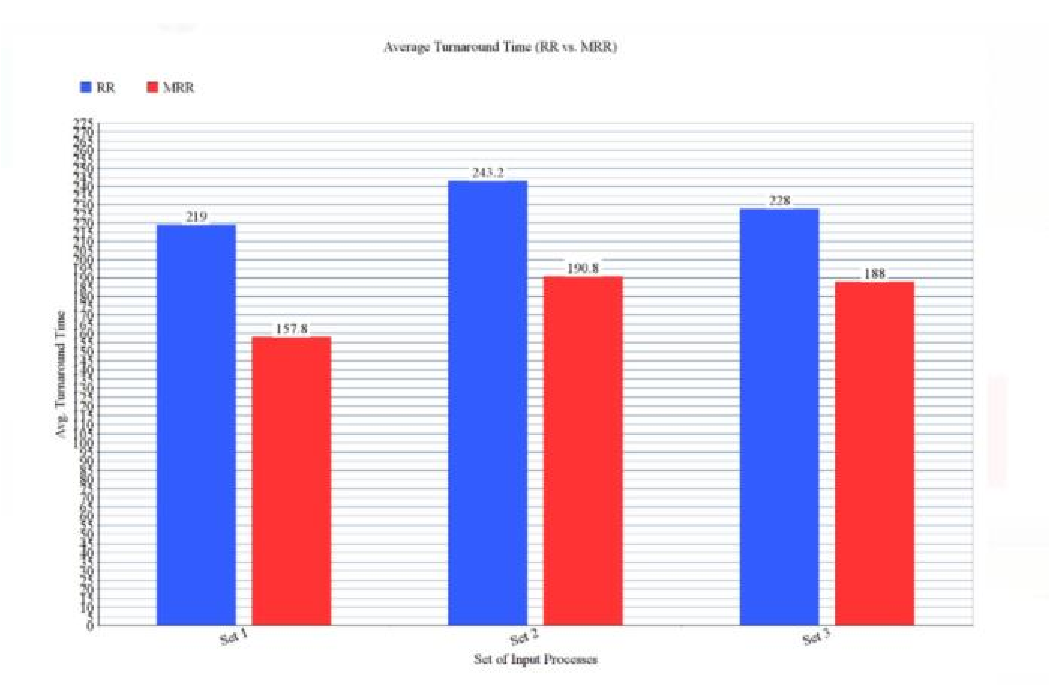
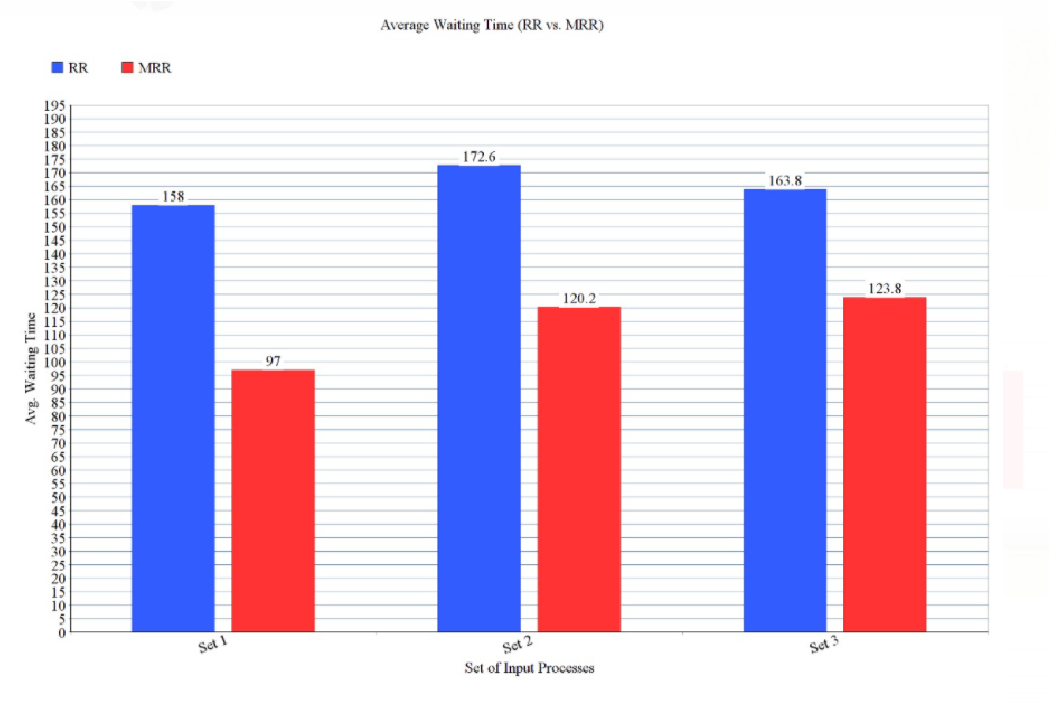
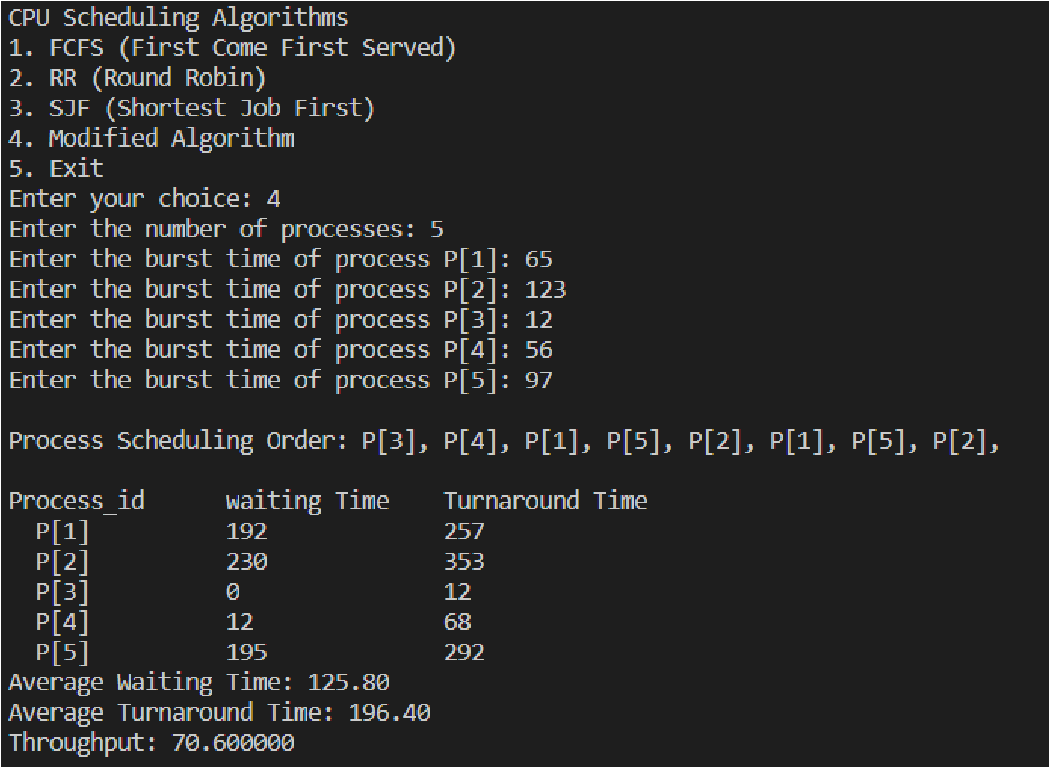
•

load the data

•

Initialise the value of K.

•



1.Calculate the distance between the data points and each row of training data. Here we will use the Euclidian distance metric. The other metrics can be used are Chebyshev, cosine, etc.

2.Sort the calculated distances in ascending order based

on distance values.

3.Get top k rows from the sorted array 4.Get the most frequent class of these rows.

5.Return the predicted class.

import pandas as pd

**VIII.**

**M**

**ACHINE**

**L**

**EARNING**

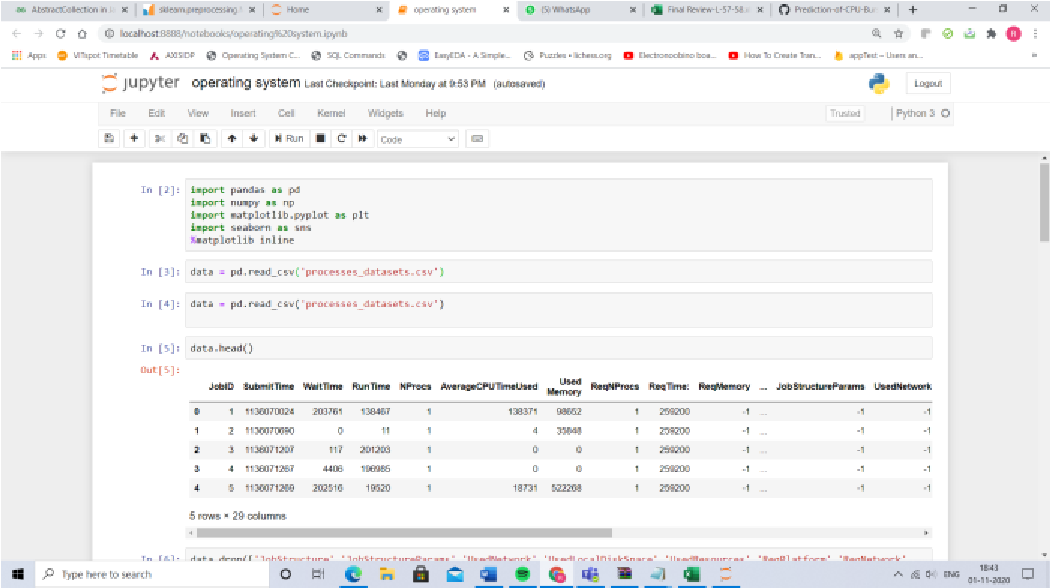
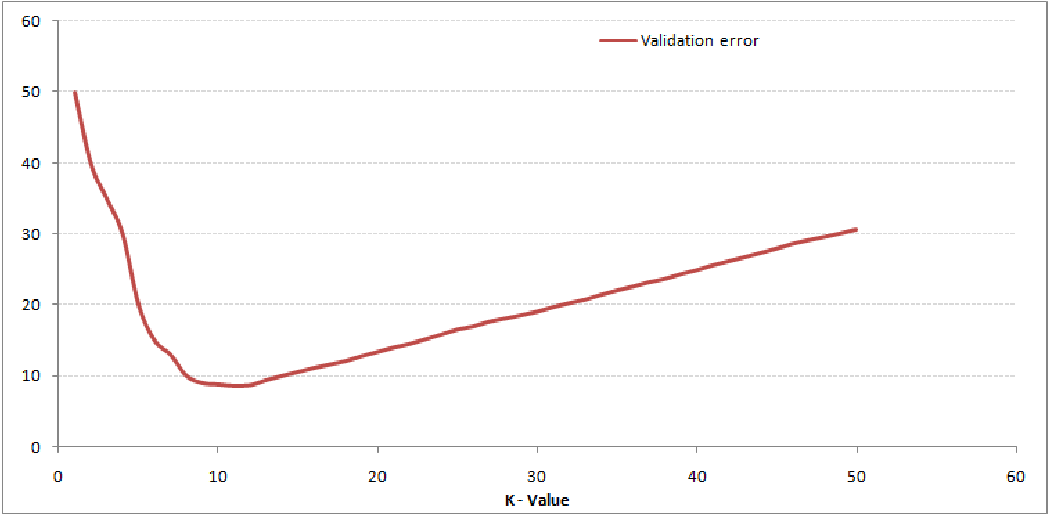
**C**

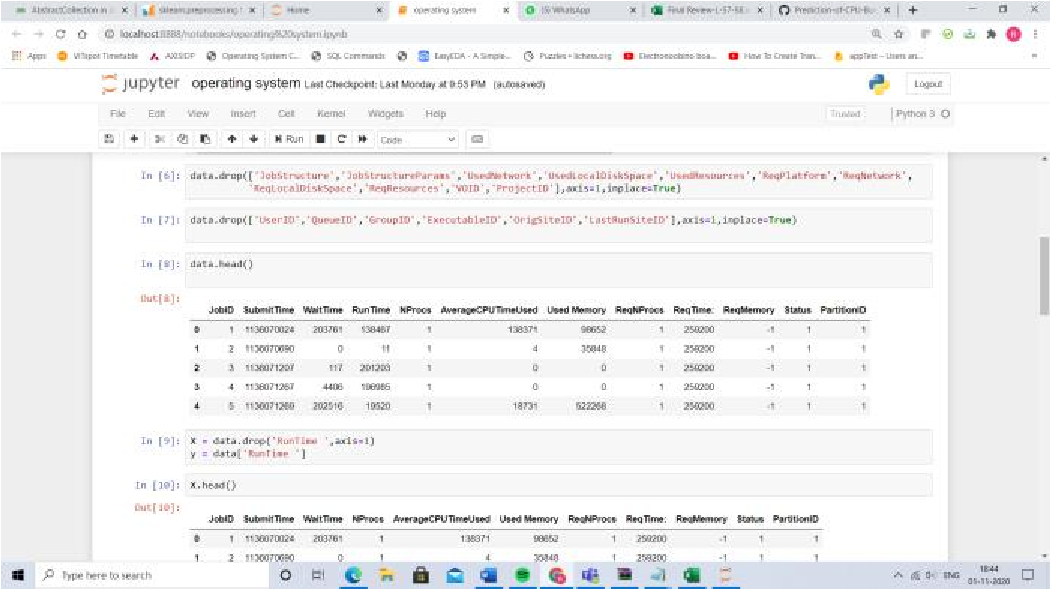
**ODE**

from sklearn.metrics import accu

racy\_score, r2\_score

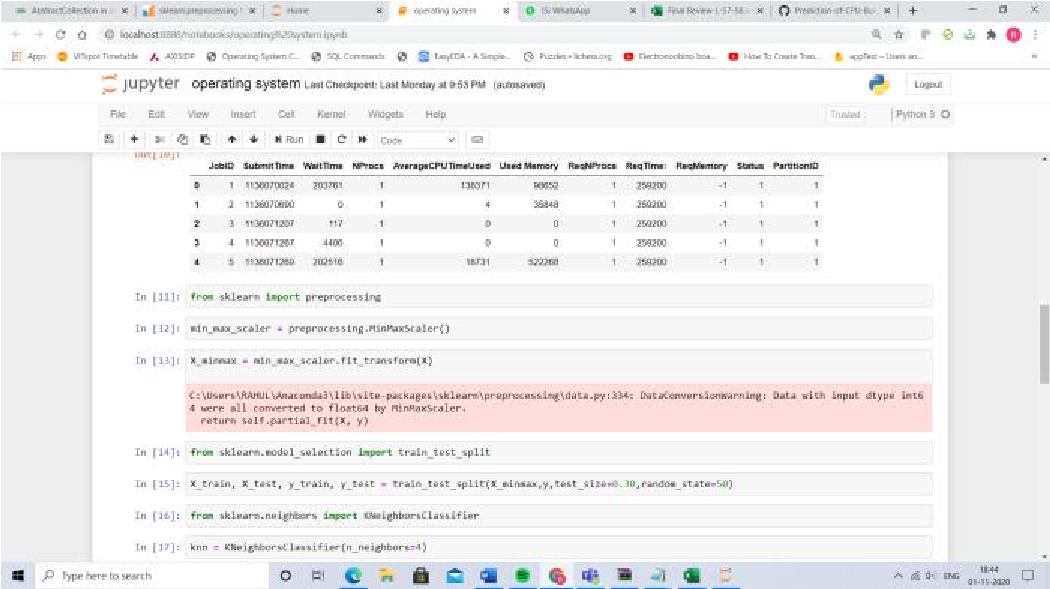
r2\_score(y\_pred\_knn, y\_test)



import numpy as np import matplotlib.pyplot as plt import seaborn as sns %matplotlib inline data = pd.read\_csv('processes\_datasets.csv') data.head() data.drop(['JobStructure','JobStructureParams','UsedNetwor

k','UsedLocalDiskSpace','UsedResources','ReqPlatform','ReqN etwork',

'ReqLocalDiskSpace','ReqResources','VOID','ProjectID'],axis=

1,inplace=True) data.drop(['UserID','QueueID','GroupID','ExecutableID','Or

igSiteID','LastRunSiteID'],axis=1,inplace=True) data.head()

X = data.drop('RunTime ',axis=1) y = data['RunTime '] X.head()

from sklearn import preprocessing

min\_max\_scaler = preprocessing.MinMaxScaler()

X\_minmax = min\_max\_scaler.fit\_transform(X) from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test= train\_test\_split(X\_minmax,y,test\_size=0.30,random\_state=50) from sklearn.neighbors import KNeighborsClassifier knn = KNeighborsClassifier(n\_neighbors=4) knn.fit(X\_train, y\_train) y\_pred\_knn = knn.predict(X\_test) y\_pred\_knn

**ALGORITHM WITH DYNAMIC TIME**

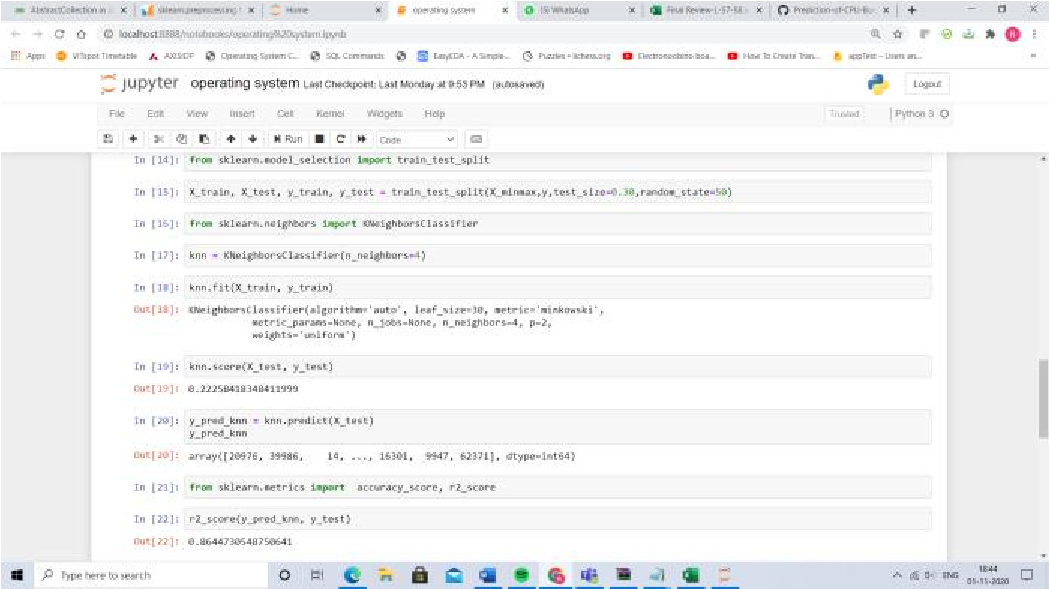
**QUANTUM”**. *Int. J. of Adv. Res.* **7** (Feb). 422429] (ISSN 2320-5407).

Link:

<http://www.journalijar.com/uploads/218_IJAR-26194.pdf>

• **“A Novel Amended Dynamic Round Robin Scheduling Algorithm for Timeshared Systems”**-Uferah Shafi, Munam Shah, Abdul

Wahid, Kamran Abbasi, Qaisar Javaid,

 Muhammad Asghar, and Muhammad Haider

Link: <https://iajit.org/PDF/%20Vol%2017,%20No.%201/11166.pdf>

**IX. TOOLS USED:** • Asst. Prof. M.K.Srivastav, Sanjay Pandey,

* Code blocks (C programming IDE) or any C Indresh Gahoi, Neelesh Kumar Namdev – programming IDE **“Fair Priority Round Robin with Dynamic**
* Jupiter Notebook (Python) for implementation of ML. **Time Quantum:FPRRDQ”**, International

Journal of Modern Engineering Research

(IJMER) Link:

[http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.6](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.640.2907&rep=rep1&type=pdf)

**X. REFERENCES**

**Operating System – Research Papers:**

* Dash, Amar & Sahu, Sandipta & Samantra, Sanjay. (2015). **“An Optimized Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum”**. International Journal of Computer Science, Engineering and

Information Technology. 5. 07-26.

10.5121/ijcseit.2015.5102. Link:

[https://www.researchgate.net/publication/276732451\_An\_O ptimized\_Round\_Robin\_CPU\_Scheduling\_Algorithm\_with\_Dy namic\_Time\_Quantum](https://www.researchgate.net/publication/276732451_An_Optimized_Round_Robin_CPU_Scheduling_Algorithm_with_Dynamic_Time_Quantum)

* Yosef Berhanu, Abebe Alemu and Manish Kumar Mishra. **“Dynamic Time Quantum based Round Robin CPU Scheduling Algorithm”.** *International Journal of Computer Applications* 167(13):48-55, June 2017.

Link:

[https://www.ijcaonline.org/archives/volume167/number13/ berhanu-2017-ijca-914569.pdf](https://www.ijcaonline.org/archives/volume167/number13/berhanu-2017-ijca-914569.pdf)

* Sohrawordi, Ehasn Ali, Palash Uddin and Mahabub Hossain. (2019); **“A MODIFIED**

**ROUND ROBIN CPU SCHEDULING**

[40.2907&rep=rep1&type=pdf](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.640.2907&rep=rep1&type=pdf)

* **A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and Its Performance Analysis** H. S. Behera, Rakesh Mohanty, Debashree Nayak [https://arxiv.org/ftp/arxiv/papers/1103/1103.](https://arxiv.org/ftp/arxiv/papers/1103/1103.3831.pdf)

[3831.pdf](https://arxiv.org/ftp/arxiv/papers/1103/1103.3831.pdf)

* **A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average**

Abbas Noon, Ali Kalakech, Seifedine Kadry [https://arxiv.org/ftp/arxiv/papers/1111/1111.](https://arxiv.org/ftp/arxiv/papers/1111/1111.5348.pdf)

[5348.pdf](https://arxiv.org/ftp/arxiv/papers/1111/1111.5348.pdf)

* **An Enhanced Round Robin CPU Scheduling Algorithm**

Jayanti Khatri

[https://www.researchgate.net/profile/Jayanti \_Khatri/publication/305391498\_An\_Enhanc ed\_Round\_Robin\_CPU\_Scheduling\_Algorit hm/links/578dccef08ae254b1de87f6e/AnEnhanced-Round-Robin-CPU-Scheduling-](https://www.researchgate.net/profile/Jayanti_Khatri/publication/305391498_An_Enhanced_Round_Robin_CPU_Scheduling_Algorithm/links/578dccef08ae254b1de87f6e/An-Enhanced-Round-Robin-CPU-Scheduling-Algorithm.pdf)

[Algorithm.pdf](https://www.researchgate.net/profile/Jayanti_Khatri/publication/305391498_An_Enhanced_Round_Robin_CPU_Scheduling_Algorithm/links/578dccef08ae254b1de87f6e/An-Enhanced-Round-Robin-CPU-Scheduling-Algorithm.pdf)

* **Determining the Optimum Time**

**Quantum Value in Round Robin Process**

**Scheduling Method** Shahram Saeidi, Hakimeh Alemi Baktash [http://citeseerx.ist.psu.edu/viewdoc/downloa](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.859.2163&rep=rep1&type=pdf)

[d?doi=10.1.1.859.2163&rep=rep1&type=pd](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.859.2163&rep=rep1&type=pdf)

[f](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.859.2163&rep=rep1&type=pdf)

* **Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running Processes**

Rami J. Matarneh

[https://www.researchgate.net/profile/Rami\_ Matarneh/publication/40832774\_Self-](https://www.researchgate.net/profile/Rami_Matarneh/publication/40832774_Self-Adjustment_Time_Quantum_in_Round_Robin_Algorithm_Depending_on_Burst_Time_of_the_Now_Running_Processes/links/56bf6c5b08aeedba0562dda8.pdf)

[Adjustment\_Time\_Quantum\_in\_Round\_Ro bin\_Algorithm\_Depending\_on\_Burst\_Time \_of\_the\_Now\_Running\_Processes/links/56b](https://www.researchgate.net/profile/Rami_Matarneh/publication/40832774_Self-Adjustment_Time_Quantum_in_Round_Robin_Algorithm_Depending_on_Burst_Time_of_the_Now_Running_Processes/links/56bf6c5b08aeedba0562dda8.pdf)

[f6c5b08aeedba0562dda8.pdf](https://www.researchgate.net/profile/Rami_Matarneh/publication/40832774_Self-Adjustment_Time_Quantum_in_Round_Robin_Algorithm_Depending_on_Burst_Time_of_the_Now_Running_Processes/links/56bf6c5b08aeedba0562dda8.pdf)

* Mahmoud, Shams & Ali, Abd & Falah Hasan, Hasan &

Abed, Zainab. (2018). **Advanced Dynamic Round**

**Robin Algorithm for CPU Scheduling in Real Time Systems.** Journal of Advanced Research in Dynamical and Control Systems. 10. 2314.

[https://www.researchgate.net/publication/329427270\_Adva nced\_Dynamic\_Round\_Robin\_Algorithm\_for\_CPU\_Schedulin g\_in\_Real\_Time\_Systems](https://www.researchgate.net/publication/329427270_Advanced_Dynamic_Round_Robin_Algorithm_for_CPU_Scheduling_in_Real_Time_Systems)

* Manish Kumar Mishra1 and Dr.Faizur Rashid2, **AN**

**IMPROVED ROUND ROBIN CPU SCHEDULING**

**ALGORITHM WITH VARYING TIME QUANTUM**,

International Journal of Computer Science,

Engineering and Applications (IJCSEA) Vol.4, No.4, August 2014

[https://www.researchgate.net/profile/Faizur\_Rashid/publica tion/273011366\_An\_Improved\_Round\_Robin\_CPU\_Scheduli](https://www.researchgate.net/profile/Faizur_Rashid/publication/273011366_An_Improved_Round_Robin_CPU_Scheduling_Algorithm_with_Varying_Time_Quantum/links/5e3bd92d92851c7f7f201c5a/An-Improved-Round-Robin-CPU-Scheduling-Algorithm-with-Varying-Time-Quantum.pdf)

[ng\_Algorithm\_with\_Varying\_Time\_Quantum/links/5e3bd92d](https://www.researchgate.net/profile/Faizur_Rashid/publication/273011366_An_Improved_Round_Robin_CPU_Scheduling_Algorithm_with_Varying_Time_Quantum/links/5e3bd92d92851c7f7f201c5a/An-Improved-Round-Robin-CPU-Scheduling-Algorithm-with-Varying-Time-Quantum.pdf)

[92851c7f7f201c5a/An-Improved-Round-Robin-CPU-](https://www.researchgate.net/profile/Faizur_Rashid/publication/273011366_An_Improved_Round_Robin_CPU_Scheduling_Algorithm_with_Varying_Time_Quantum/links/5e3bd92d92851c7f7f201c5a/An-Improved-Round-Robin-CPU-Scheduling-Algorithm-with-Varying-Time-Quantum.pdf)

[Scheduling-Algorithm-with-Varying-Time-Quantum.pdf](https://www.researchgate.net/profile/Faizur_Rashid/publication/273011366_An_Improved_Round_Robin_CPU_Scheduling_Algorithm_with_Varying_Time_Quantum/links/5e3bd92d92851c7f7f201c5a/An-Improved-Round-Robin-CPU-Scheduling-Algorithm-with-Varying-Time-Quantum.pdf)

* Abdulrazaq Abdulrahim, Saleh E Abdullahi, Junaidu B. Sahalu, **A New Improved Round Robin (NIRR)**

**CPU Scheduling Algorithm.** International Journal of

Computer Applications (0975 – 8887) Volume 90 – No 4, March 2014

[https://eduwavepool.unizwa.edu.om/lmsdatapool/00008828](https://eduwavepool.unizwa.edu.om/lmsdatapool/00008828/LearningObjects/paper%201.pdf)

[/LearningObjects/paper%201.pdf](https://eduwavepool.unizwa.edu.om/lmsdatapool/00008828/LearningObjects/paper%201.pdf)

* Jayanti Khatri, **An Improved Dynamic Round Robin CPU Scheduling Algorithm Based on Variant Time Quantum.** IOSR Journal of Computer Engineering

(IOSR-JCE) e-ISSN: 2278-0661, p-ISSN: 2278-8727,

Volume 18, Issue 6, Ver. IV (Nov.-Dec. 2016), PP 3540

[https://www.researchgate.net/profile/Jayanti\_Khatri/publica tion/311417385\_An\_Improved\_Dynamic\_Round\_Robin\_CPU](https://www.researchgate.net/profile/Jayanti_Khatri/publication/311417385_An_Improved_Dynamic_Round_Robin_CPU_Scheduling_Algorithm_Based_on_Variant_Time_Quantum/links/584bbb4a08aed95c24fbf7cb/An-Improved-Dynamic-Round-Robin-CPU-Scheduling-Algorithm-Based-on-Variant-Time-Quantum.pdf)

[\_Scheduling\_Algorithm\_Based\_on\_Variant\_Time\_Quantum/li nks/584bbb4a08aed95c24fbf7cb/An-Improved-Dynamic-](https://www.researchgate.net/profile/Jayanti_Khatri/publication/311417385_An_Improved_Dynamic_Round_Robin_CPU_Scheduling_Algorithm_Based_on_Variant_Time_Quantum/links/584bbb4a08aed95c24fbf7cb/An-Improved-Dynamic-Round-Robin-CPU-Scheduling-Algorithm-Based-on-Variant-Time-Quantum.pdf)

[Round-Robin-CPU-Scheduling-Algorithm-Based-on-Variant-](https://www.researchgate.net/profile/Jayanti_Khatri/publication/311417385_An_Improved_Dynamic_Round_Robin_CPU_Scheduling_Algorithm_Based_on_Variant_Time_Quantum/links/584bbb4a08aed95c24fbf7cb/An-Improved-Dynamic-Round-Robin-CPU-Scheduling-Algorithm-Based-on-Variant-Time-Quantum.pdf)

[Time-Quantum.pdf](https://www.researchgate.net/profile/Jayanti_Khatri/publication/311417385_An_Improved_Dynamic_Round_Robin_CPU_Scheduling_Algorithm_Based_on_Variant_Time_Quantum/links/584bbb4a08aed95c24fbf7cb/An-Improved-Dynamic-Round-Robin-CPU-Scheduling-Algorithm-Based-on-Variant-Time-Quantum.pdf)

* M. U. Farooq, A. Shakoor and A. B. Siddique**, "An Efficient Dynamic Round Robin algorithm for CPU scheduling**," 2017 International Conference on Communication, Computing and Digital Systems (CCODE), Islamabad, 2017, pp. 244-248, doi:

10.1109/C-CODE.2017.7918936.

* Round Robin Scheduling Algorithm based on Dynamic time quantum by Rashmi Dhruv[https://www.ijeat.org/wp-](https://www.ijeat.org/wp-content/uploads/papers/v8i6/F8070088619.pdf)

[content/uploads/papers/v8i6/F8070088619.pdf](https://www.ijeat.org/wp-content/uploads/papers/v8i6/F8070088619.pdf)

* International Journal of Computer Science,

Engineering and Information Technology (IJCSEIT),

Vol. 5,No.1, February 2015

<https://arxiv.org/ftp/arxiv/papers/1605/1605.00362.pdf>

# • A NEW PROPOSED TWO PROCESSOR BASED CPU SCHEDULING

ALGORITHM WITH VARYING TIME QUANTUM FOR REAL TIME

SYSTEMS [http://www.rroij.com/open-access/a-new-proposed-twoprocessor-based-cpu-scheduling-algorithm-with-varyingtime-quantum-for-real-time-systems-81-87.php?aid=37048](http://www.rroij.com/open-access/a-new-proposed-two-processor-based-cpu-scheduling-algorithm-with-varying-time-quantum-for-real-time-systems-81-87.php?aid=37048)

* Computer Science and Software Engineering

Smart Optimized Round Robin (SORR) CPU Scheduling

Algorithm

[http://ijarcsse.com/Before\_August\_2017/docs/papers/Volum e\_5/7\_July2015/V5I7-0337.pdf](http://ijarcsse.com/Before_August_2017/docs/papers/Volume_5/7_July2015/V5I7-0337.pdf)

* [https://iopscience.iop.org/article/10.1088/1757-](https://iopscience.iop.org/article/10.1088/1757-899X/263/4/042001/pdf)

[899X/263/4/042001/pdf](https://iopscience.iop.org/article/10.1088/1757-899X/263/4/042001/pdf)

* [https://www.irejournals.com/formatedpaper/17002](https://www.irejournals.com/formatedpaper/1700273.pdf)

[73.pdf](https://www.irejournals.com/formatedpaper/1700273.pdf)