Introduction to the project:

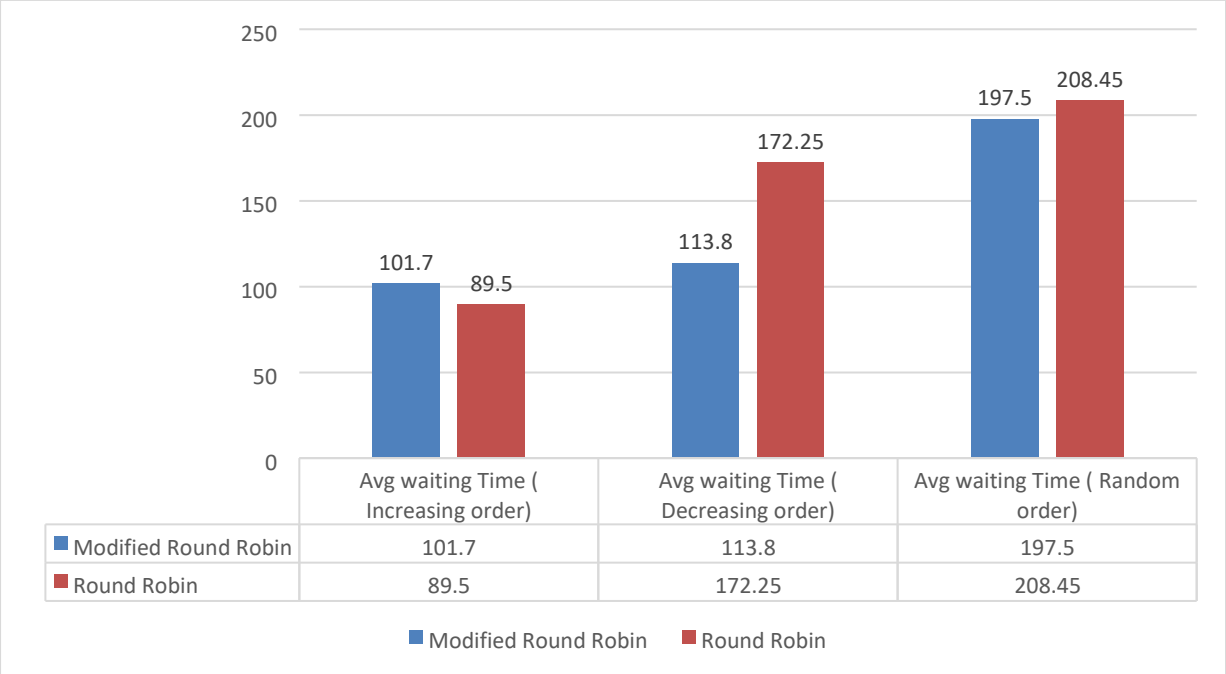
Scheduling is the central concept used in Operating Systems. It helps in choosing the processes for execution in an efficient sequence. Round Robin (RR) is one of the most commonly used CPU scheduling algorithms. But there is a degradation of performance with respect to context switching, which is an overhead cost to the system. The performance of a system depends on the choice of an optimal time quantum, so as to reduce context switching. In this paper, we have proposed a new variant of Round Robin which is better than the traditional Round Robin scheduling by reducing context switching, average waiting time and average turnaround time

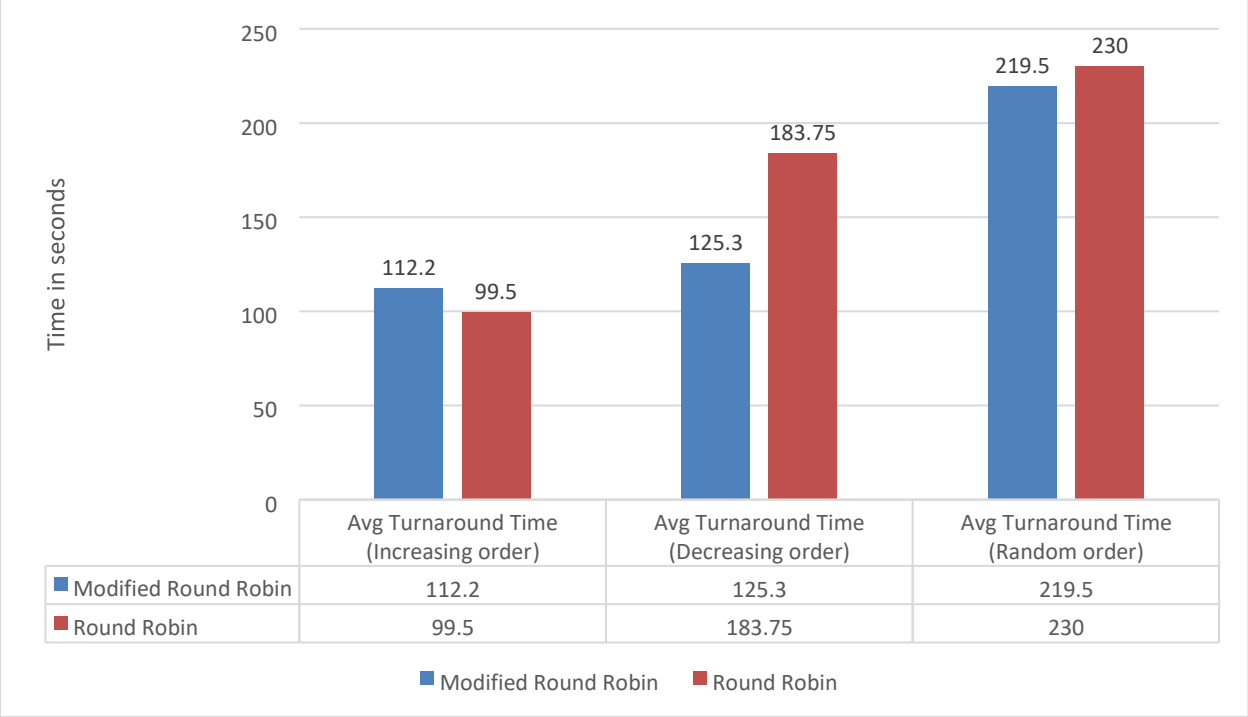
Objectives of the project:

- **To reduce context switching.**
- **To reduce average waiting time.**
- **To reduce average turnaround time.**
- **Predicting Burst Time.**

Graphical representation:

Comparison Round robin and modified round robin algorithm:





Knn machine learning algorithm predict burst time:

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
data = pd.read_csv('processes_datasets.csv')
data.head()
```

Out[2]:

	JobID	SubmitTime	WaitTime	RunTime	NProcs	AverageCPUTimeUsed	Used Memory	ReqNProcs	ReqTime:	ReqMemory	...	JobStructureParams	UsedNetwork
0	1	1136070024	203761	138467	1	138371	98652	1	259200	-1	...	-1	-1
1	2	1136070690	0	11	1	4	35848	1	259200	-1	...	-1	-1
2	3	1136071207	117	201203	1	0	0	1	259200	-1	...	-1	-1
3	4	1136071267	4406	196985	1	0	0	1	259200	-1	...	-1	-1
4	5	1136071269	202516	19520	1	18731	522268	1	259200	-1	...	-1	-1

5 rows × 29 columns

In [3]:
```
data.drop(['JobStructure','JobStructureParams','UsedNetwork','UsedLocalDiskSpace','UsedResources','ReqPlatform','ReqNetwork',
           'ReqLocalDiskSpace','ReqResources','VOID','ProjectID'],axis=1,inplace=True)
```

In [4]:
```
data.drop(['UserID','QueueID','GroupID','ExecutableID','OrigSiteID','LastRunSiteID'],axis=1,inplace=True)
```

In [5]:
```
data.head()
```

Out[5]:

	JobID	SubmitTime	WaitTime	RunTime	NProcs	AverageCPUTimeUsed	Used Memory	ReqNProcs	ReqTime:	ReqMemory	Status	PartitionID
0	1	1136070024	203761	138467	1	138371	98652	1	259200	-1	1	1
1	2	1136070690	0	11	1	4	35848	1	259200	-1	1	1
2	3	1136071207	117	201203	1	0	0	1	259200	-1	1	1
3	4	1136071267	4406	196985	1	0	0	1	259200	-1	1	1
4	5	1136071269	202516	19520	1	18731	522268	1	259200	-1	1	1

In [6]:
```
X = data.drop('RunTime ',axis=1)
y = data['RunTime ']
X.head()
```

Out[6]:

	JobID	SubmitTime	WaitTime	NProcs	AverageCPUTimeUsed	Used Memory	ReqNProcs	ReqTime:	ReqMemory	Status	PartitionID
0	1	1136070024	203761	1	138371	98652	1	259200	-1	1	1
1	2	1136070690	0	1	4	35848	1	259200	-1	1	1
2	3	1136071207	117	1	0	0	1	259200	-1	1	1
3	4	1136071267	4406	1	0	0	1	259200	-1	1	1
4	5	1136071269	202516	1	18731	522268	1	259200	-1	1	1

In [7]:
```
from sklearn import preprocessing
```

In [8]:
```
min_max_scaler = preprocessing.MinMaxScaler()
```

In [9]:
```
X_minmax = min_max_scaler.fit_transform(X)
```

In [10]:
```
from sklearn.model_selection import train_test_split
```

In [11]:
```
X_train, X_test, y_train, y_test = train_test_split(X_minmax,y
```

```
In [8]: min_max_scaler = preprocessing.MinMaxScaler()

In [9]: X_minmax = min_max_scaler.fit_transform(X)

In [10]: from sklearn.model_selection import train_test_split

In [11]: X_train, X_test, y_train, y_test = train_test_split(X_minmax,y
                                                    , test_size=0.30,random_state=50)

In [12]: from sklearn.neighbors import KNeighborsClassifier

In [13]: knn = KNeighborsClassifier(n_neighbors=4)

In [14]: knn.fit(X_train, y_train)
Out[14]: KNeighborsClassifier(n_neighbors=4)

In [15]: y_pred_knn = knn.predict(X_test)
         y_pred_knn
Out[15]: array([20976, 39986,    14, ..., 16301,  9947, 62371], dtype=int64)

In [16]: from sklearn.metrics import  accuracy_score, r2_score

In [17]: r2_score(y_pred_knn, y_test)
Out[17]: 0.8644730548750641

In [ ]:
```

****