Predictive Modelling Project

Problem - 1

The comp-activ database comprises activity measures of computer systems. Data was gathered from a Sun Sparcstation 20/712 with 128 Mbytes of memory, operating in a multi-user university department. Users engaged in diverse tasks, such as internet access, file editing, and CPU-intensive programs.

Being an aspiring data scientist, you aim to establish a linear equation for predicting 'usr' (the percentage of time CPUs operate in user mode). Your goal is to analyze various system attributes to understand their influence on the system's 'usr' mode.

Data Description:

lread - Reads (transfers per second) between system memory and user memory

lwrite - writes (transfers per second) between system memory and user memory

scall - Number of system calls of all types per second

sread - Number of system read calls per second .

swrite - Number of system write calls per second .

fork - Number of system fork calls per second.

exec - Number of system exec calls per second.

rchar - Number of characters transferred per second by system read calls

wchar - Number of characters transfreed per second by system write calls

pgout - Number of page out requests per second

ppgout - Number of pages, paged out per second

pgfree - Number of pages per second placed on the free list.

pgscan - Number of pages checked if they can be freed per second

atch - Number of page attaches (satisfying a page fault by reclaiming a page in memory) per second

pgin - Number of page-in requests per second

ppgin - Number of pages paged in per second

pflt - Number of page faults caused by protection errors (copy-on-writes).

vflt - Number of page faults caused by address translation .

runqsz - Process run queue size (The number of kernel threads in memory that are waiting for a CPU to run.

Typically, this value should be less than 2. Consistently higher values mean that the system might be CPU-bound.)

freemem - Number of memory pages available to user processes

freeswap - Number of disk blocks available for page swapping.

usr - Portion of time (%) that cpus run in user mode

Define the problem and perform exploratory Data Analysis?

Shape of Data Frame as below:

Number of Rows: 8192
Number of Columns: 22

Check the Data Types of the columns for the Dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8192 entries, 0 to 8191
Data columns (total 22 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   lread     8192 non-null   int64
 1   lwrite    8192 non-null   int64
 2   scall     8192 non-null   int64
 3   sread     8192 non-null   int64
 4   swrite    8192 non-null   int64
 5   fork      8192 non-null   float64
 6   exec      8192 non-null   float64
 7   rchar     8088 non-null   float64
 8   wchar     8177 non-null   float64
 9   pgout     8192 non-null   float64
 10  ppgout    8192 non-null   float64
 11  pgfree    8192 non-null   float64
 12  pgscan    8192 non-null   float64
 13  atch      8192 non-null   float64
 14  pgin      8192 non-null   float64
 15  ppgin     8192 non-null   float64
 16  pflt      8192 non-null   float64
 17  vflt      8192 non-null   float64
 18  runqsz    8192 non-null   object
 19  freemem   8192 non-null   int64
 20  freeswap  8192 non-null   int64
 21  usr       8192 non-null   int64
dtypes: float64(13), int64(8), object(1)
memory usage: 1.4+ MB
```

Check the Statistical Summary of Data

	count	mean	std	min	25%	50%	75%	max
lread	8192.0	1.955969e+01	53.353799	0.0	2.0	7.0	20.000	1845.00
lwrite	8192.0	1.310620e+01	29.891726	0.0	0.0	1.0	10.000	575.00
scall	8192.0	2.306318e+03	1633.617322	109.0	1012.0	2051.5	3317.250	12493.00
sread	8192.0	2.104800e+02	198.980146	6.0	86.0	166.0	279.000	5318.00
swrite	8192.0	1.500582e+02	160.478980	7.0	63.0	117.0	185.000	5456.00
fork	8192.0	1.884554e+00	2.479493	0.0	0.4	0.8	2.200	20.12
exec	8192.0	2.791998e+00	5.212456	0.0	0.2	1.2	2.800	59.56
rchar	8088.0	1.973857e+05	239837.493526	278.0	34091.5	125473.5	267828.750	2526649.00
wchar	8177.0	9.590299e+04	140841.707911	1498.0	22916.0	46619.0	106101.000	1801623.00
pgout	8192.0	2.285317e+00	5.307038	0.0	0.0	0.0	2.400	81.44
ppgout	8192.0	5.977229e+00	15.214590	0.0	0.0	0.0	4.200	184.20
pgfree	8192.0	1.191971e+01	32.363520	0.0	0.0	0.0	5.000	523.00
pgscan	8192.0	2.152685e+01	71.141340	0.0	0.0	0.0	0.000	1237.00
atch	8192.0	1.127505e+00	5.708347	0.0	0.0	0.0	0.600	211.58
pgin	8192.0	8.277960e+00	13.874978	0.0	0.6	2.8	9.765	141.20
ppgin	8192.0	1.238859e+01	22.281318	0.0	0.6	3.8	13.800	292.61
pflt	8192.0	1.097938e+02	114.419221	0.0	25.0	63.8	159.600	899.80
vflt	8192.0	1.853158e+02	191.000603	0.2	45.4	120.4	251.800	1365.00
freemem	8192.0	1.763456e+03	2482.104511	55.0	231.0	579.0	2002.250	12027.00
freeswap	8192.0	1.328126e+06	422019.426957	2.0	1042623.5	1289289.5	1730379.500	2243187.00
usr	8192.0	8.396887e+01	18.401905	0.0	81.0	89.0	94.000	99.00

Data Frame printing rows with Head (Prints top 5 rows) function as below:

	lread	lwrite	scall	sread	swrite	fork	exec	rchar	wchar	pgout	...	pgscan	atch	pgin	ppgin	pflt	vflt	runqsz	freemem	freeswap	usr
0	1	0	2147	79	68	0.2	0.2	40671.0	53995.0	0.0	...	0.0	0.0	1.6	2.6	16.00	26.40	CPU_Bound	4670	1730946	95
1	0	0	170	18	21	0.2	0.2	448.0	8385.0	0.0	...	0.0	0.0	0.0	0.0	15.63	16.83	Not_CPU_Bound	7278	1869002	97
2	15	3	2162	159	119	2.0	2.4	NaN	31950.0	0.0	...	0.0	1.2	6.0	9.4	150.20	220.20	Not_CPU_Bound	702	1021237	87
3	0	0	160	12	16	0.2	0.2	NaN	8670.0	0.0	...	0.0	0.0	0.2	0.2	15.60	16.80	Not_CPU_Bound	7248	1863704	98
4	5	1	330	39	38	0.4	0.4	NaN	12185.0	0.0	...	0.0	0.0	1.0	1.2	37.80	47.60	Not_CPU_Bound	633	1760253	90

5 rows × 22 columns

Data Frame printing rows with Tail (Prints last 5 rows) function as below:

	lread	lwrite	scall	sread	swrite	fork	exec	rchar	wchar	pgout	...	pgscan	atch	pgin	ppgin	pflt	vflt	runqsz	freemem	freeswap	usr
8187	16	12	3009	360	244	1.6	5.81	405250.0	85282.0	8.02	...	55.11	0.6	35.87	47.90	139.28	270.74	CPU_Bound	387	986647	80
8188	4	0	1596	170	146	2.4	1.80	89489.0	41764.0	3.80	...	0.20	0.8	3.80	4.40	122.40	212.60	Not_CPU_Bound	263	1055742	90
8189	16	5	3116	289	190	0.6	0.60	325948.0	52640.0	0.40	...	0.00	0.4	28.40	45.20	60.20	219.80	Not_CPU_Bound	400	969106	87
8190	32	45	5180	254	179	1.2	1.20	62571.0	29505.0	1.40	...	18.04	0.4	23.05	24.25	93.19	202.81	CPU_Bound	141	1022458	83
8191	2	0	985	55	46	1.6	4.80	111111.0	22256.0	0.00	...	0.00	0.2	3.40	6.20	91.80	110.00	CPU_Bound	659	1756514	94

5 rows × 22 columns

Check no duplicate values in Data Frame.

There are no duplicate values in data frame.

Find out the Missing Value

```
lread       0
lwrite      0
scall       0
sread       0
swrite      0
fork        0
exec        0
rchar     104
wchar      15
pgout       0
ppgout      0
pgfree      0
pgscan      0
atch        0
pgin        0
ppgin       0
pflt        0
vflt        0
runqsz      0
freemem     0
freeswap    0
usr         0
dtype: int64
```

There 2 columns (Rchar, wchar) in Dataset. Wherein we have find the Null value.

Treat Null Value Using Median Function

```
[ ]  wm = ca['wchar'].median()
     rm = ca['rchar'].median()
     ca['wchar'] = ca['wchar'].fillna(wm)
     ca['rchar'] = ca['rchar'].fillna(rm)
```

Check Again the Null Value in Dataset, Now There is no Null value found after treating the Null Value using Median Function.

```
lread       0
lwrite      0
scall       0
sread       0
swrite      0
fork        0
exec        0
rchar       0
wchar       0
pgout       0
ppgout      0
pgfree      0
pgscan      0
atch        0
pgin        0
ppgin       0
pflt        0
vflt        0
runqsz      0
freemem     0
freeswap    0
usr         0
dtype: int64
```

Categories the Datatypes

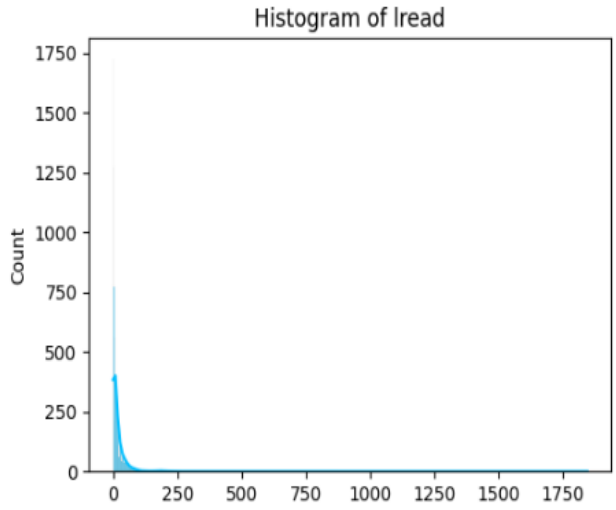
Numaric Data Types:

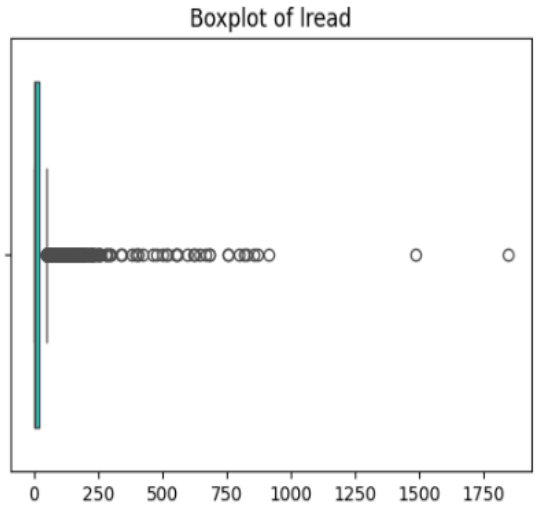
['lread', 'lwrite', 'scall', 'sread', 'swrite', 'fork', 'exec', 'rchar', 'wchar', 'pgout', 'ppgout', 'pgfree', 'pgscan', 'atch', 'pgin', 'ppgin', 'pflt', 'vflt', 'freemem', 'freeswap', 'usr']

Categorical Data Types: ['runqsz']

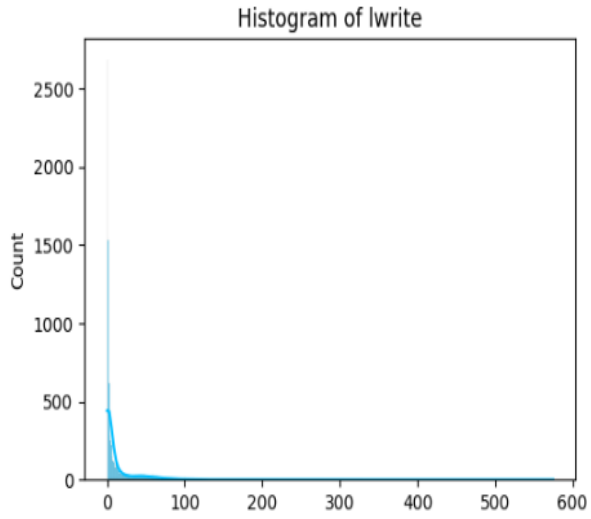
Univariate Analysis:

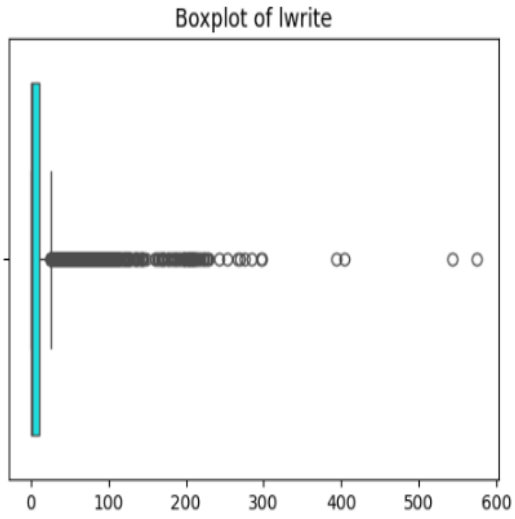
```
 lread Description
count    8192.000000
mean       19.559692
std        53.353799
min         0.000000
25%         2.000000
50%         7.000000
75%        20.000000
max      1845.000000
Name: lread, dtype: float64
Skewness = 13.897852242774922
```



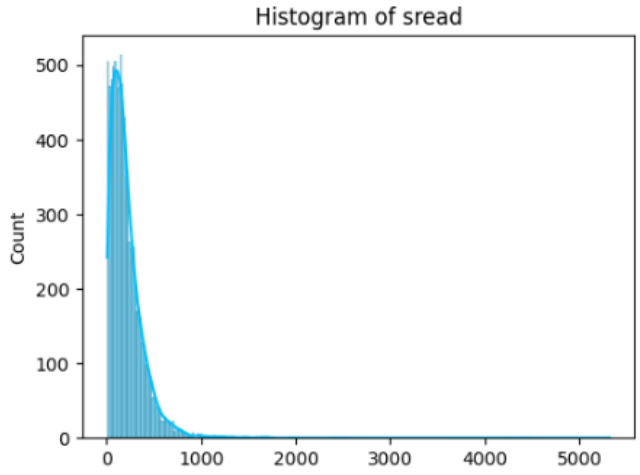


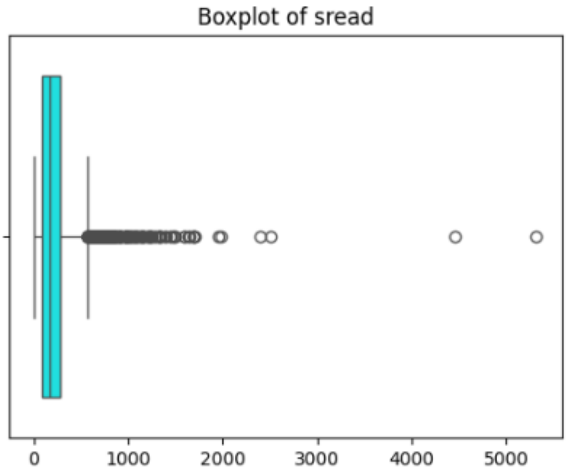

```
 lwrite Description
count    8192.000000
mean       13.106201
std        29.891726
min         0.000000
25%         0.000000
50%         1.000000
75%        10.000000
max       575.000000
Name: lwrite, dtype: float64
Skewness = 5.27764452621306
```



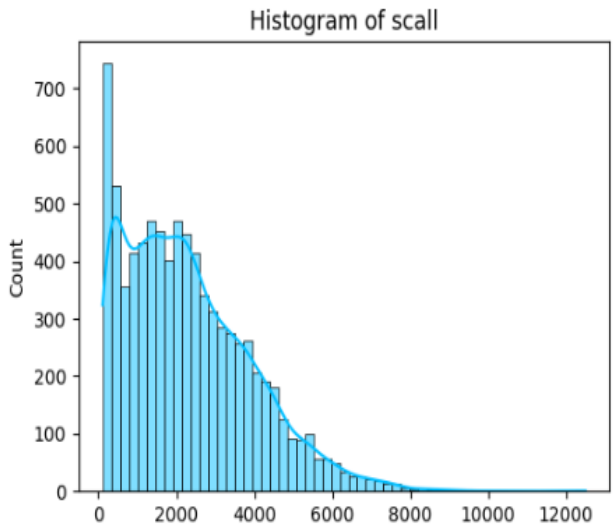


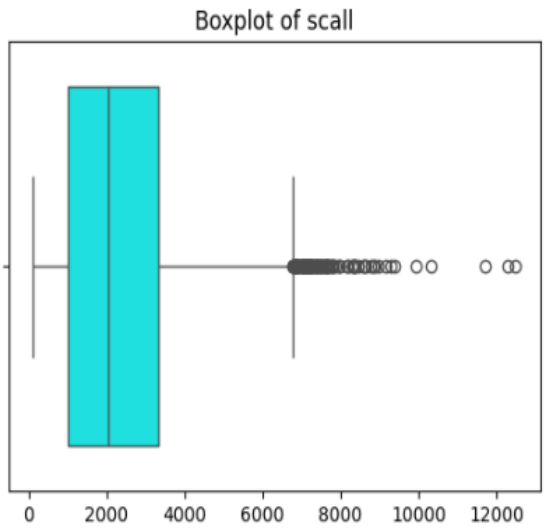
```
 sread Description
count    8192.000000
mean      210.479980
std       198.980146
min         6.000000
25%        86.000000
50%       166.000000
75%       279.000000
max      5318.000000
Name: sread, dtype: float64
Skewness = 5.459465962452425
```



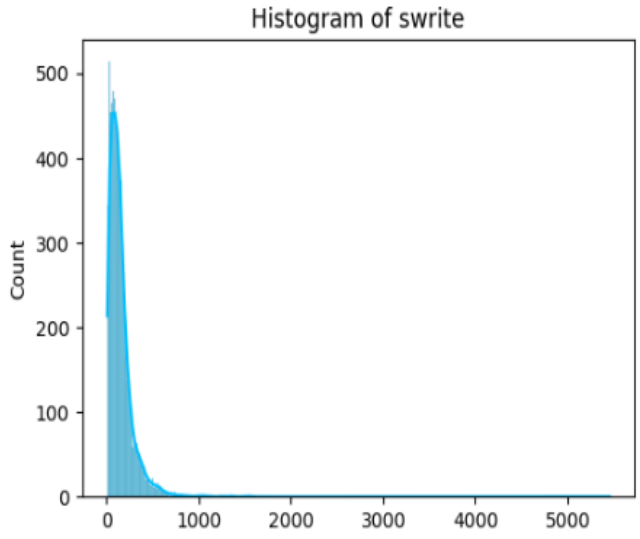


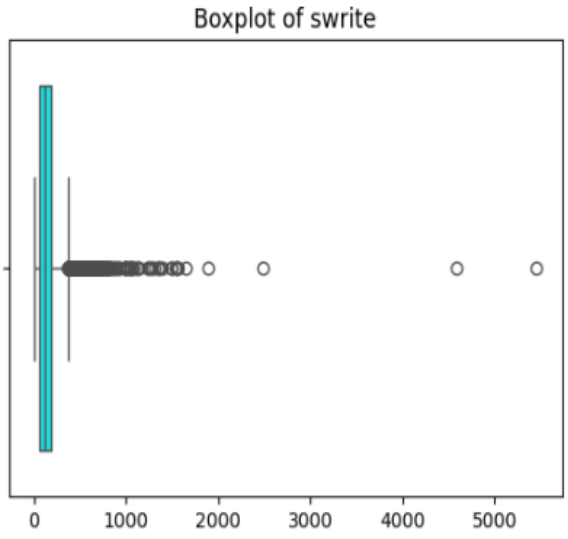

```
 scall Description
count     8192.000000
mean      2306.318237
std       1633.617322
min        109.000000
25%       1012.000000
50%       2051.500000
75%       3317.250000
max      12493.000000
Name: scall, dtype: float64
Skewness = 0.9025312213201333
```



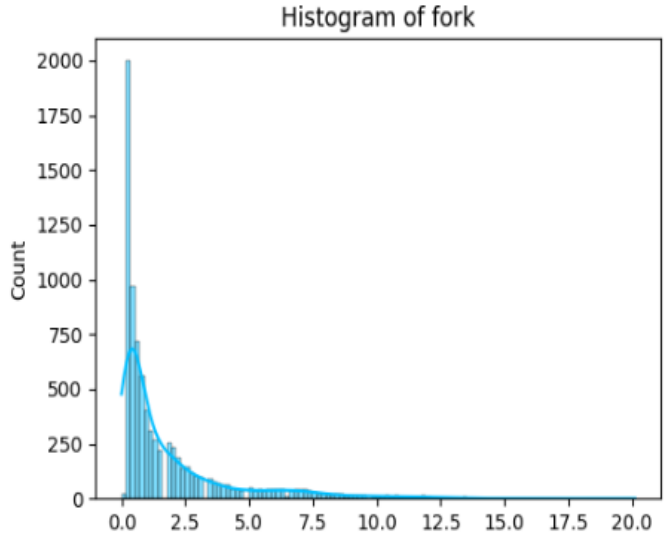


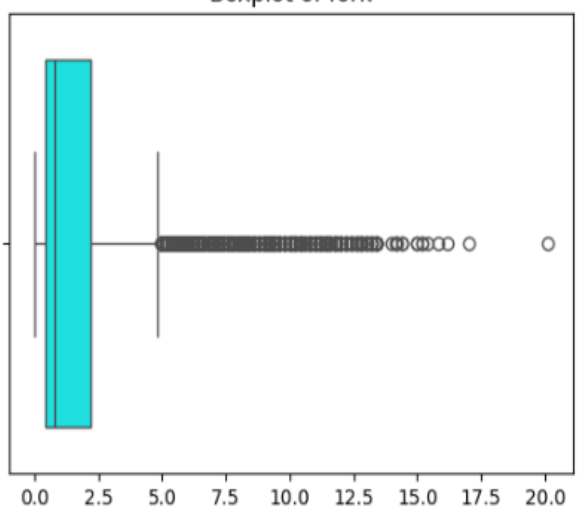
```
 swrite Description
count    8192.000000
mean      150.058228
std       160.478980
min         7.000000
25%        63.000000
50%       117.000000
75%       185.000000
max      5456.000000
Name: swrite, dtype: float64
Skewness = 9.605843698195871
```



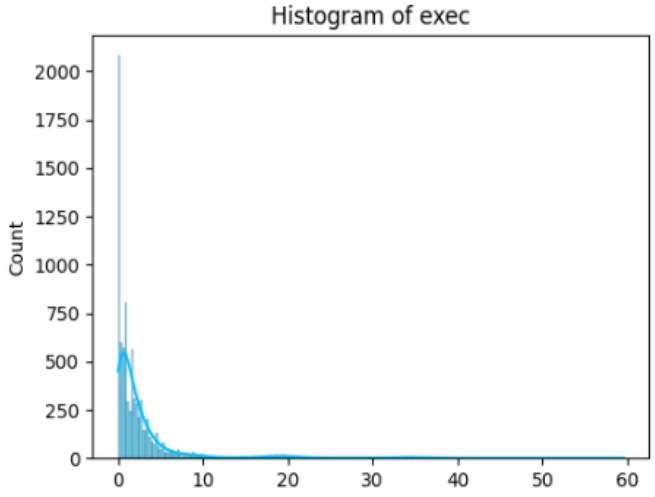


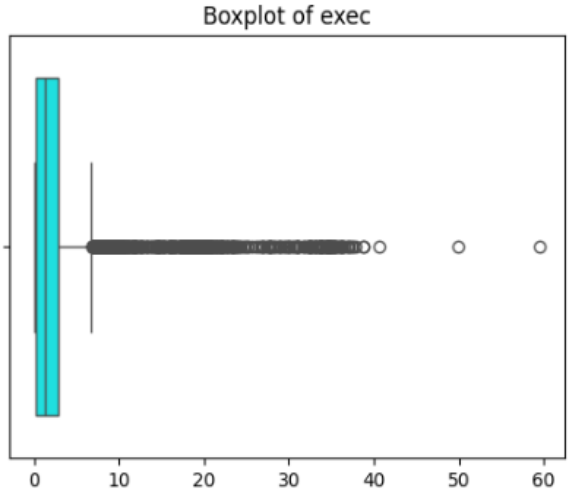

```
 fork Description
count    8192.000000
mean        1.884554
std         2.479493
min         0.000000
25%         0.400000
50%         0.800000
75%         2.200000
max        20.120000
Name: fork, dtype: float64
Skewness = 2.2496891391571325
```



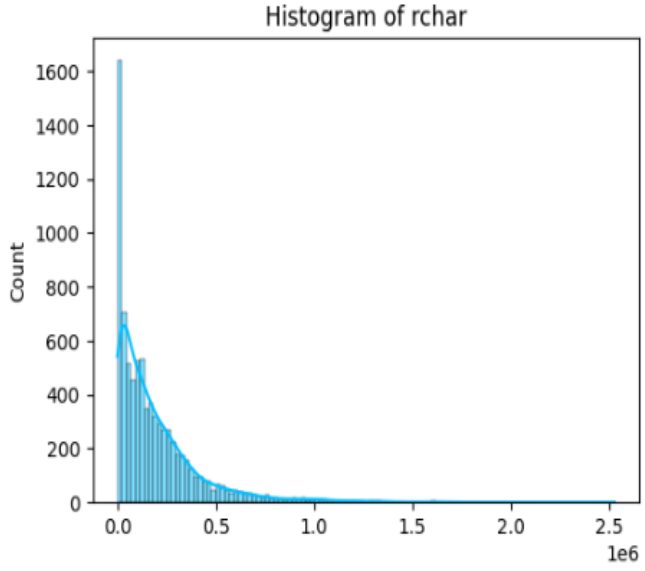


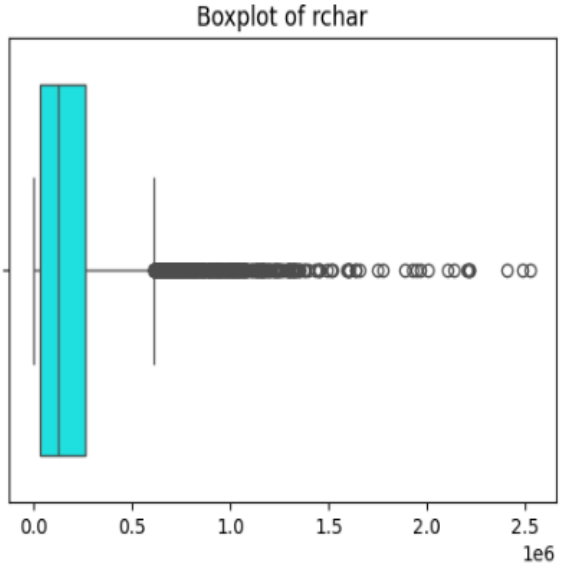
```
 exec Description
count    8192.000000
mean        2.791998
std         5.212456
min         0.000000
25%         0.200000
50%         1.200000
75%         2.800000
max        59.560000
Name: exec, dtype: float64
Skewness = 4.069237707552533
```



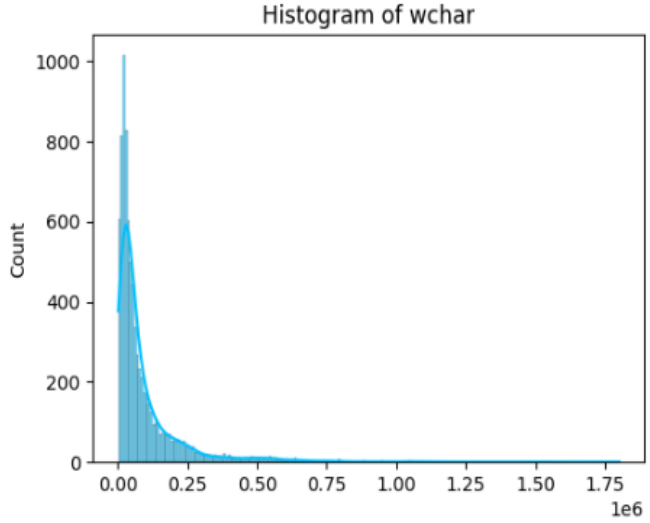


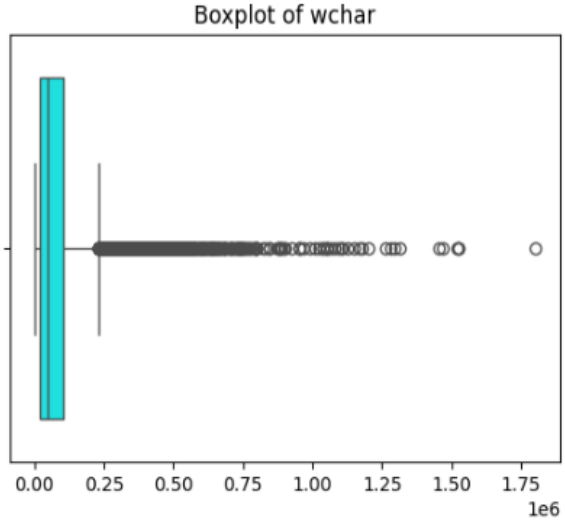

```
 rchar Description
count    8.192000e+03
mean     1.964728e+05
std      2.384460e+05
min      2.780000e+02
25%      3.486050e+04
50%      1.254735e+05
75%      2.653948e+05
max      2.526649e+06
Name: rchar, dtype: float64
Skewness = 2.8785581933662114
```



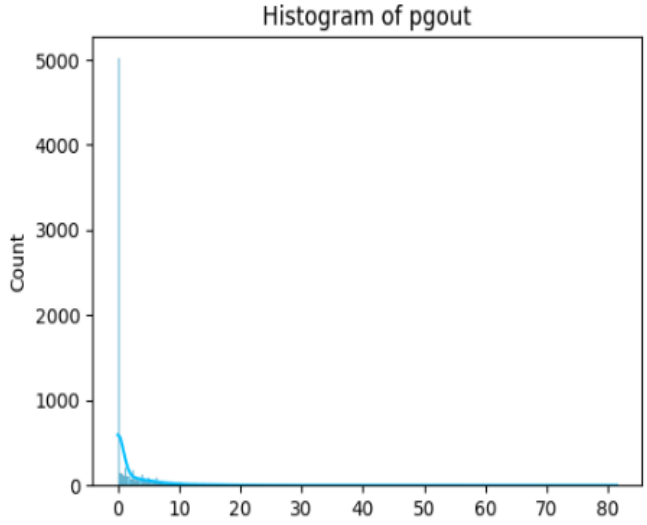


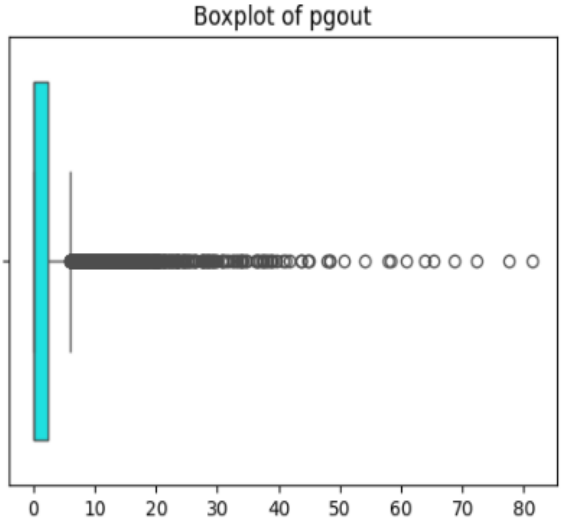
```
 wchar Description
count    8.192000e+03
mean     9.581275e+04
std      1.407285e+05
min      1.498000e+03
25%      2.297775e+04
50%      4.661900e+04
75%      1.060370e+05
max      1.801623e+06
Name: wchar, dtype: float64
Skewness = 3.851730992818844
```



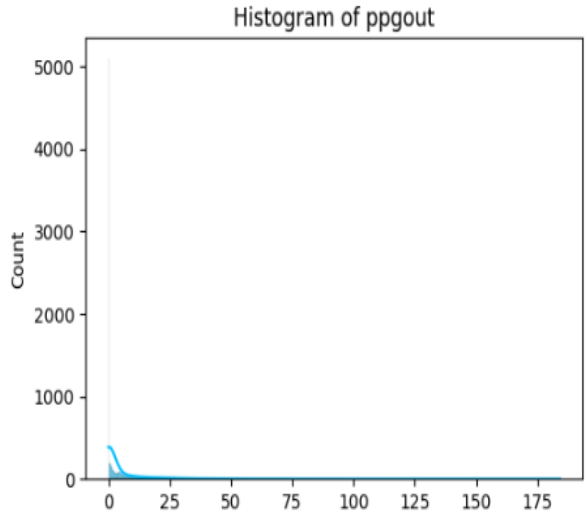


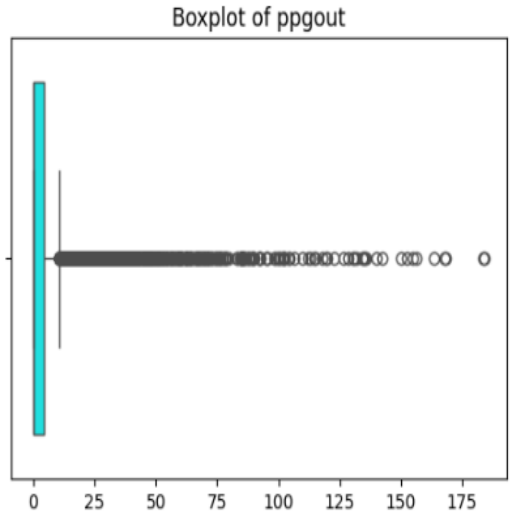

```
 pgout Description
count    8192.000000
mean        2.285317
std         5.307038
min         0.000000
25%         0.000000
50%         0.000000
75%         2.400000
max        81.440000
Name: pgout, dtype: float64
Skewness = 5.0669841185950535
```



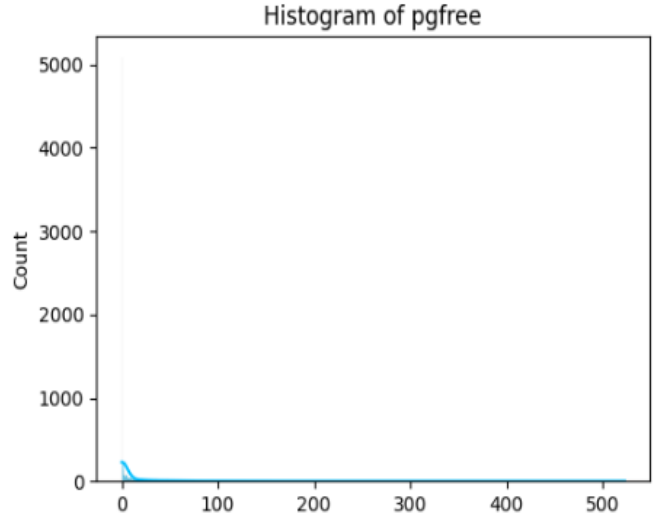


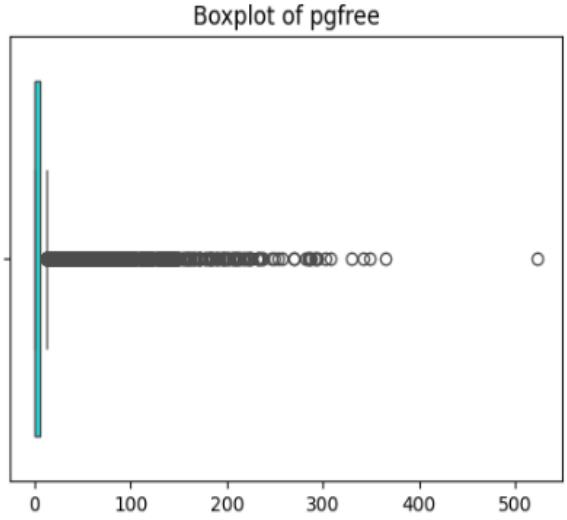
```
 ppgout Description
count    8192.000000
mean        5.977229
std        15.214590
min         0.000000
25%         0.000000
50%         0.000000
75%         4.200000
max       184.200000
Name: ppgout, dtype: float64
Skewness = 4.680441654574661
```



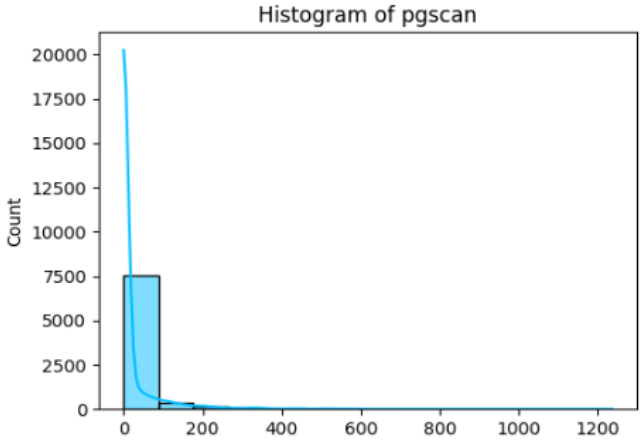


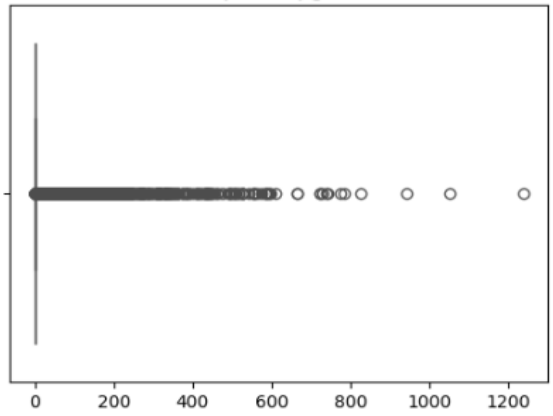

```
 pgfree Description
count    8192.000000
mean       11.919712
std        32.363520
min         0.000000
25%         0.000000
50%         0.000000
75%         5.000000
max       523.000000
Name: pgfree, dtype: float64
Skewness = 4.768191252103855
```



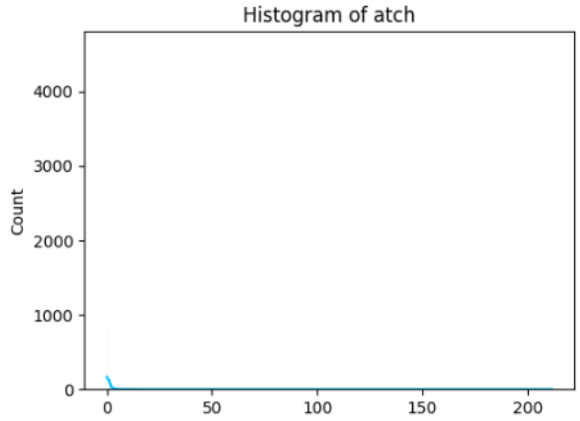


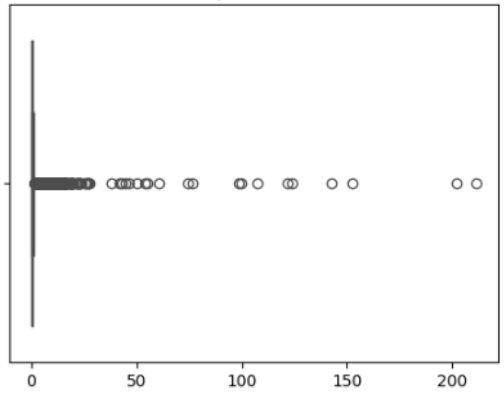
```
 pgscan Description
count    8192.000000
mean       21.526849
std        71.141340
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max      1237.000000
Name: pgscan, dtype: float64
Skewness = 5.813415144064877
```



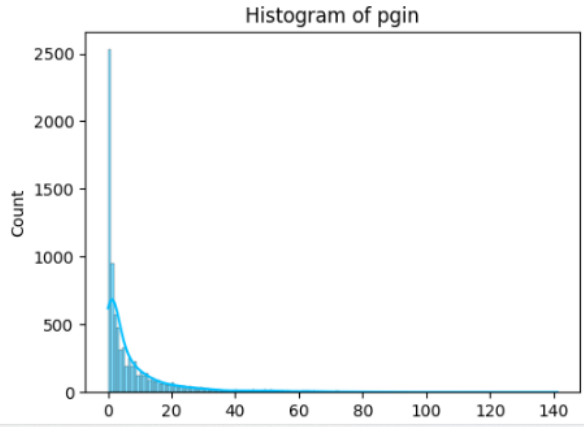


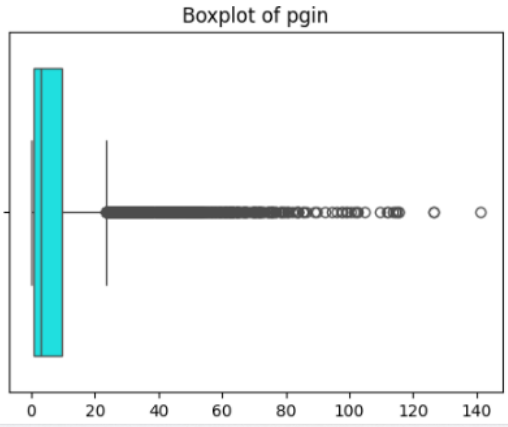

```
 atch Description
count    8192.000000
mean        1.127505
std         5.708347
min         0.000000
25%         0.000000
50%         0.000000
75%         0.600000
max       211.580000
Name: atch, dtype: float64
Skewness = 21.542019683247847
```



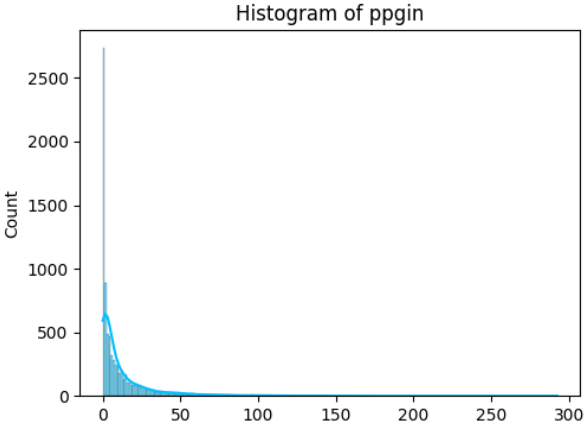


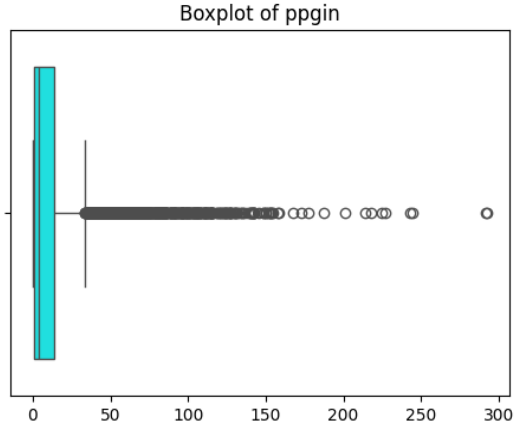
```
 pgin Description
count    8192.000000
mean        8.277960
std        13.874978
min         0.000000
25%         0.600000
50%         2.800000
75%         9.765000
max       141.200000
Name: pgin, dtype: float64
Skewness = 3.2424124762557356
```





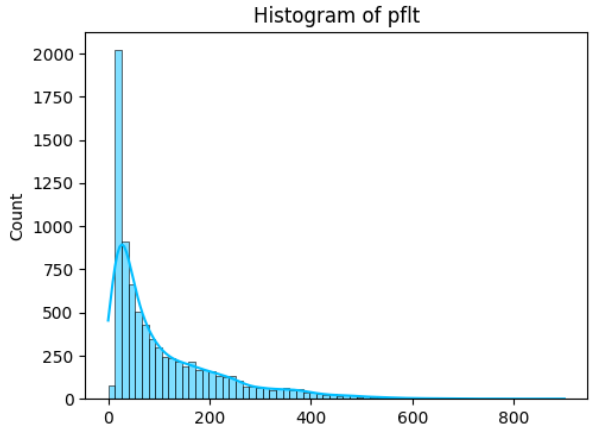

```
ppgin Description
count    8192.000000
mean       12.388586
std        22.281318
min         0.000000
25%         0.600000
50%         3.800000
75%        13.800000
max       292.610000
Name: ppgin, dtype: float64
Skewness = 3.902764914157577
```

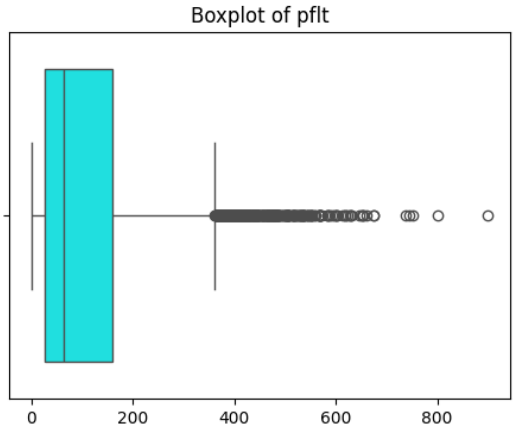




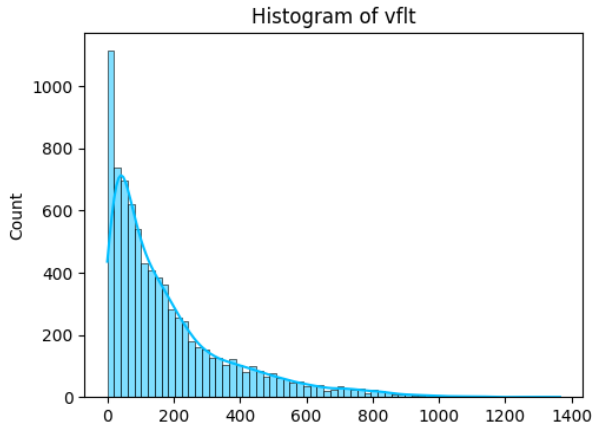

```
 pflt Description
count    8192.000000
mean      109.793799
std       114.419221
min         0.000000
25%        25.000000
```

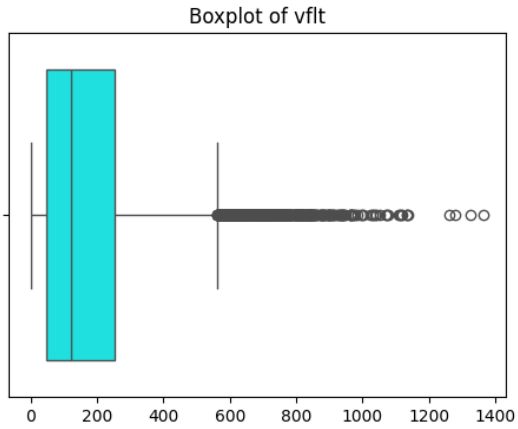
```
50%         63.800000
75%        159.600000
max        899.800000
Name: pflt, dtype: float64
Skewness = 1.7202841192012033
```



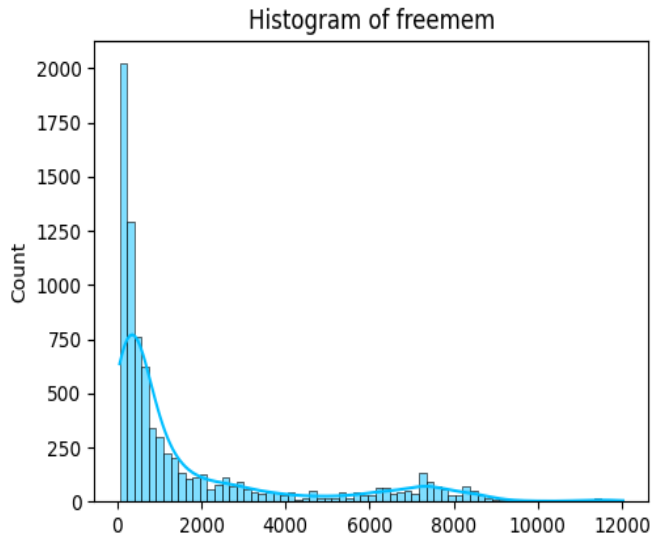


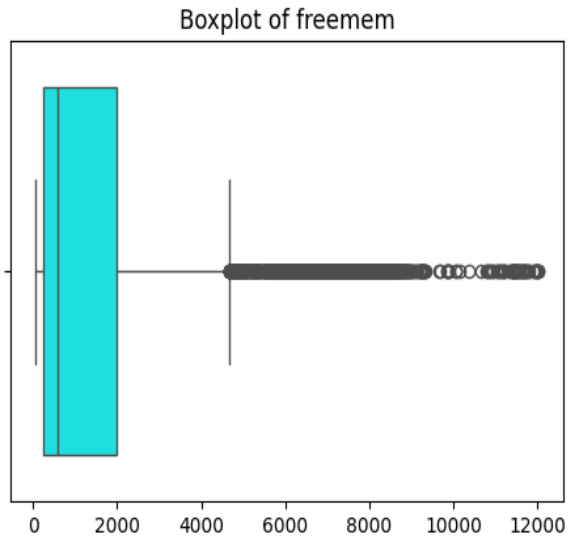

```
 vflt Description
count    8192.000000
mean      185.315796
std       191.000603
min         0.200000
25%        45.400000
50%       120.400000
75%       251.800000
max      1365.000000
Name: vflt, dtype: float64
Skewness = 1.7373265929727528
```



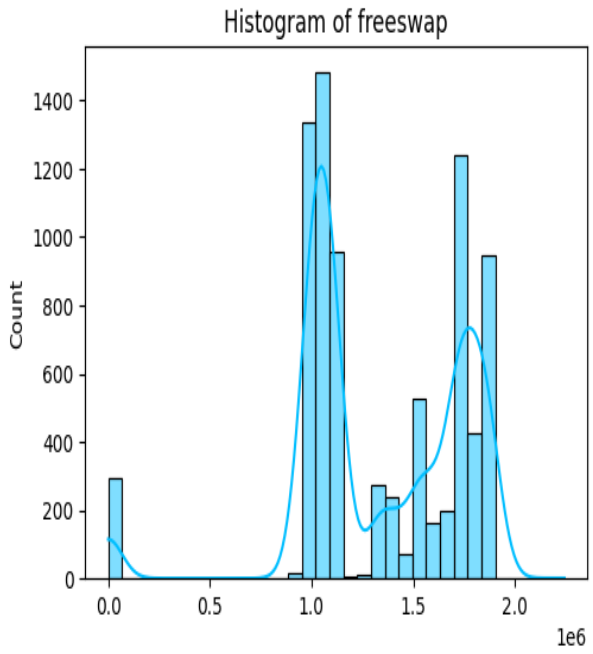


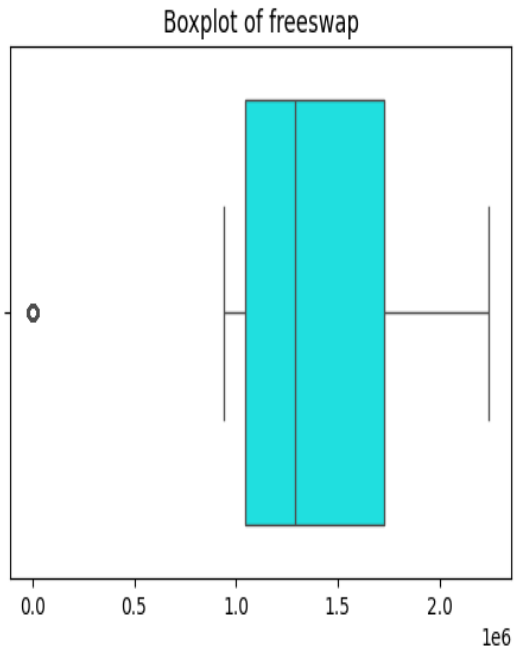

```
 freemem Description
count     8192.000000
mean      1763.456299
std       2482.104511
min         55.000000
25%        231.000000
50%        579.000000
75%       2002.250000
max      12027.000000
Name: freemem, dtype: float64
Skewness = 1.8075546533224125
```



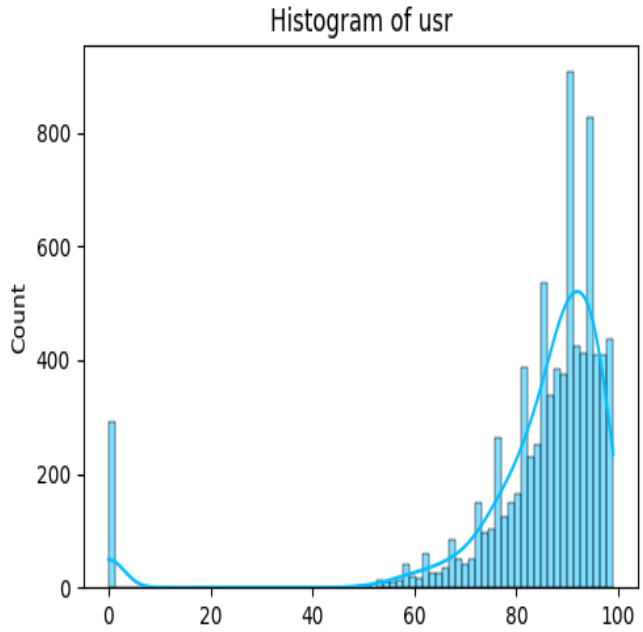


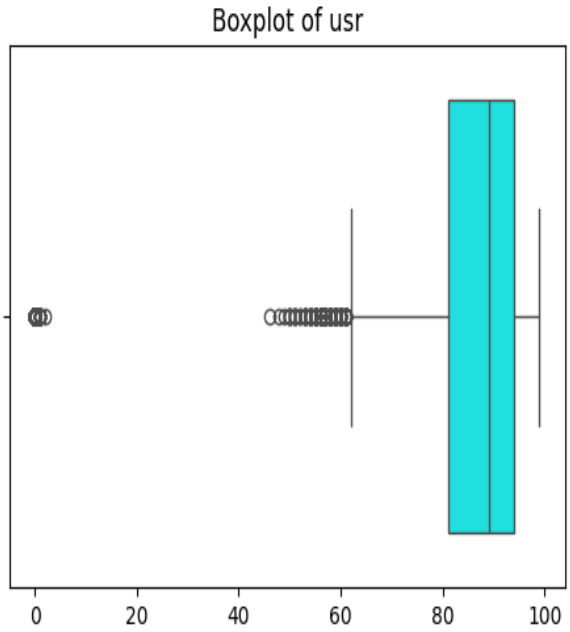

```
 freeswap Description
count    8.192000e+03
mean     1.328126e+06
std      4.220194e+05
min      2.000000e+00
25%      1.042624e+06
50%      1.289290e+06
75%      1.730380e+06
max      2.243187e+06
Name: freeswap, dtype: float64
Skewness = -0.7916644438525977
```



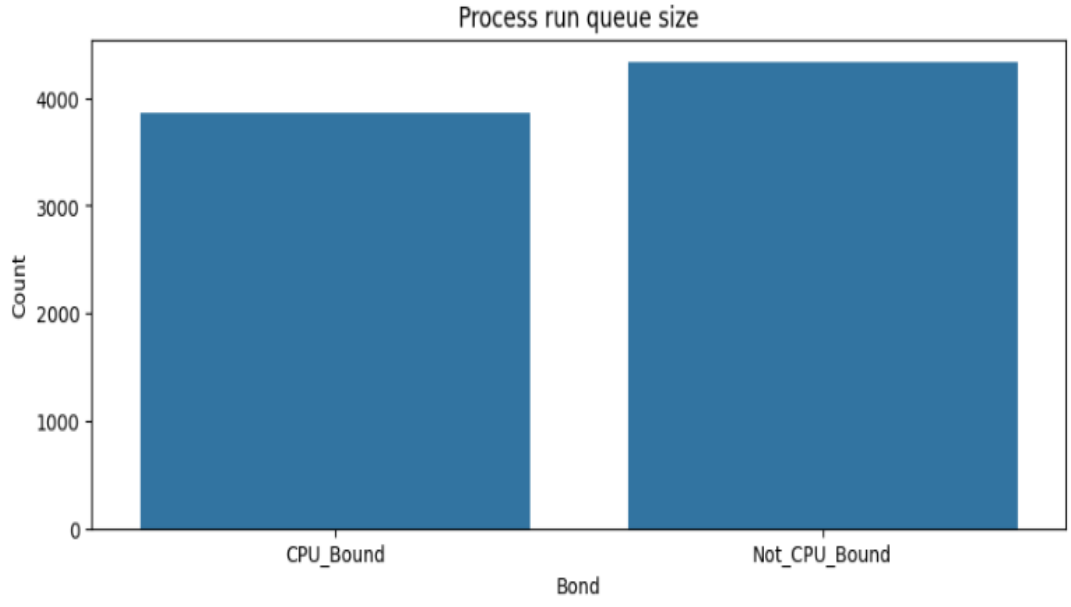


```
usr Description
count    8192.000000
mean       83.968872
std        18.401905
min         0.000000
25%        81.000000
50%        89.000000
75%        94.000000
max        99.000000
Name: usr, dtype: float64
Skewness = -3.4167496030437094
```



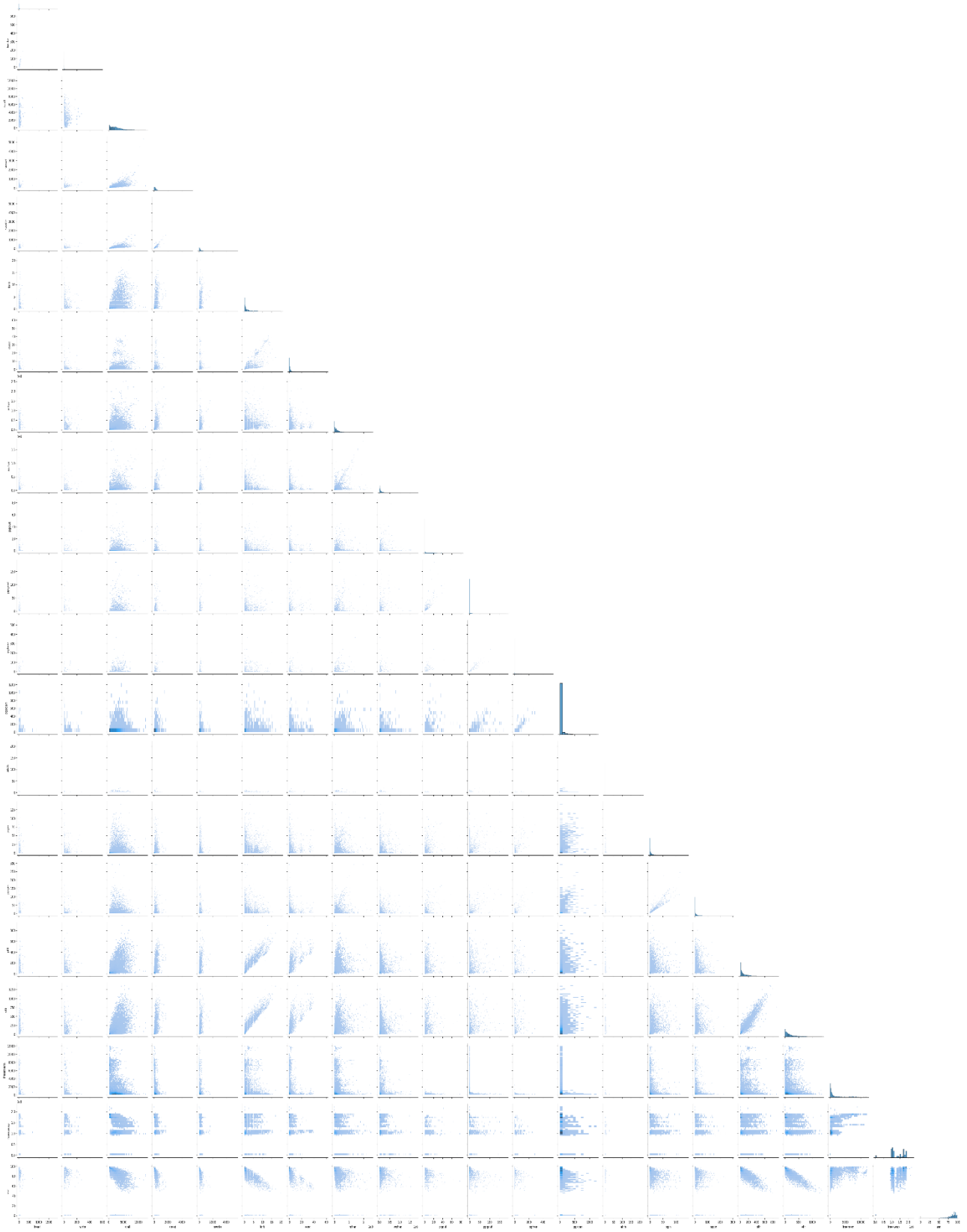


Count Plot for Categorical Variable

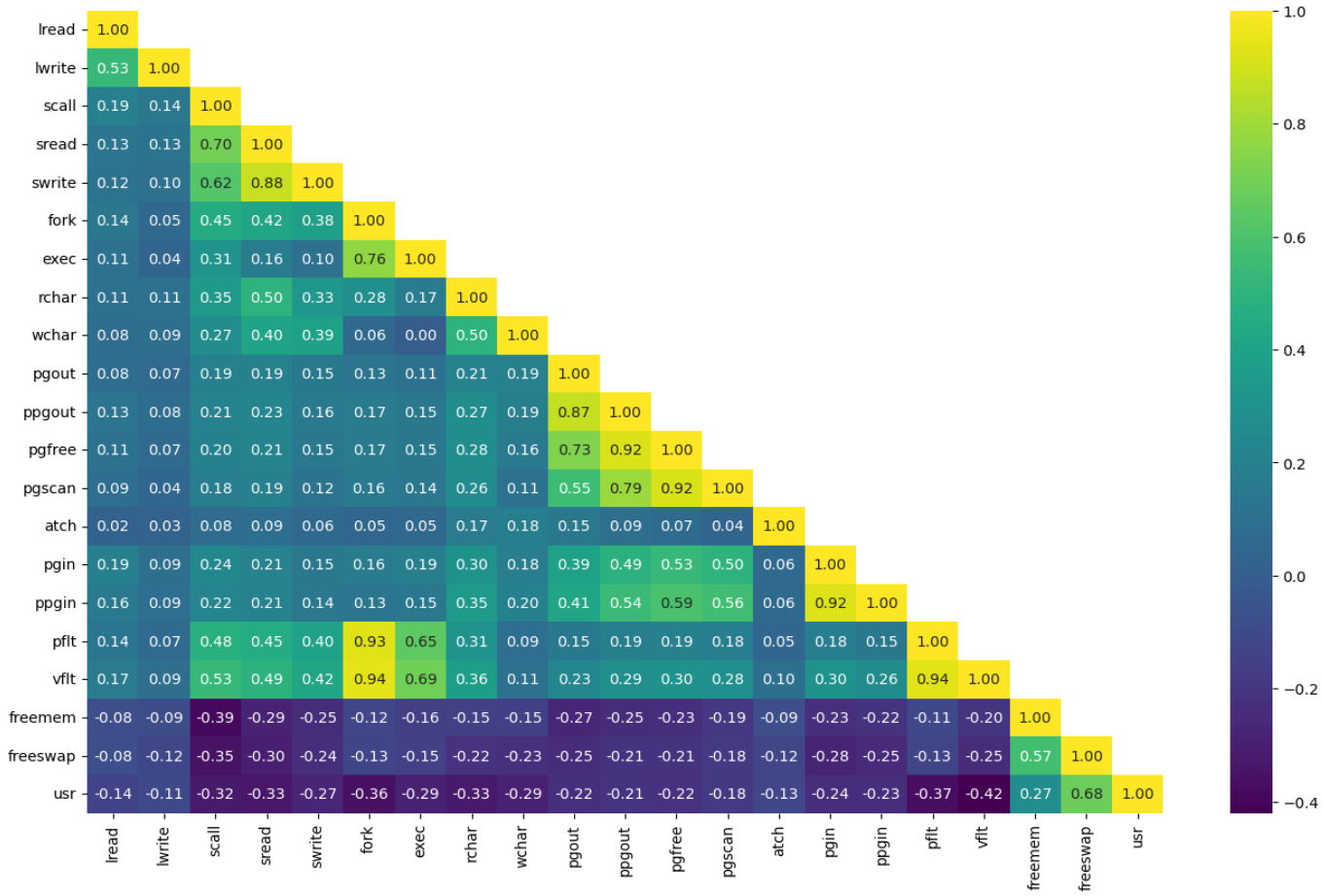


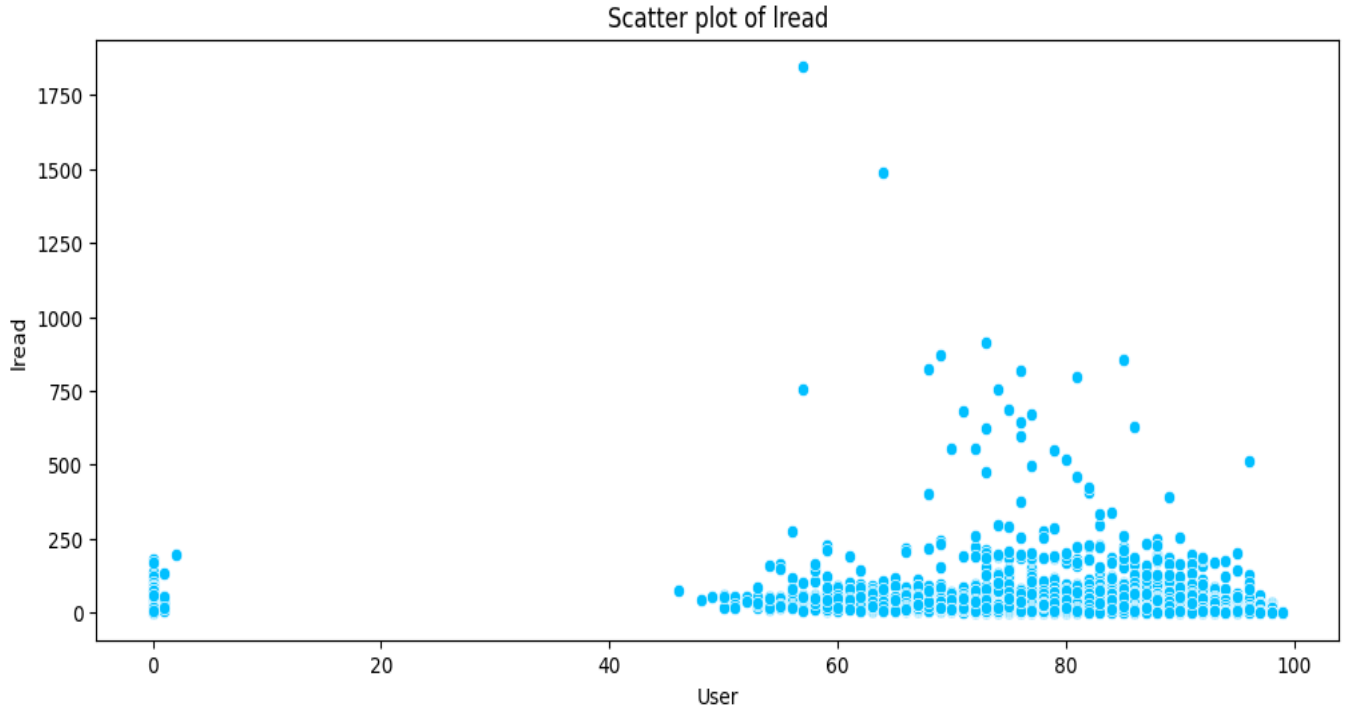
Multivariate Analysis

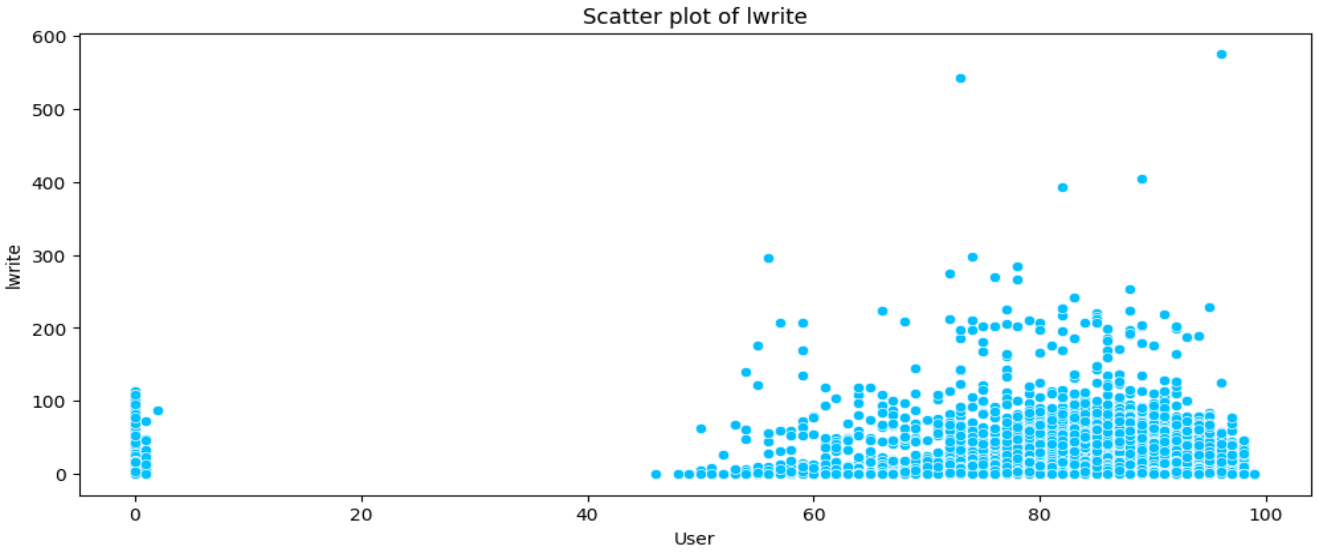
Pair Plot -

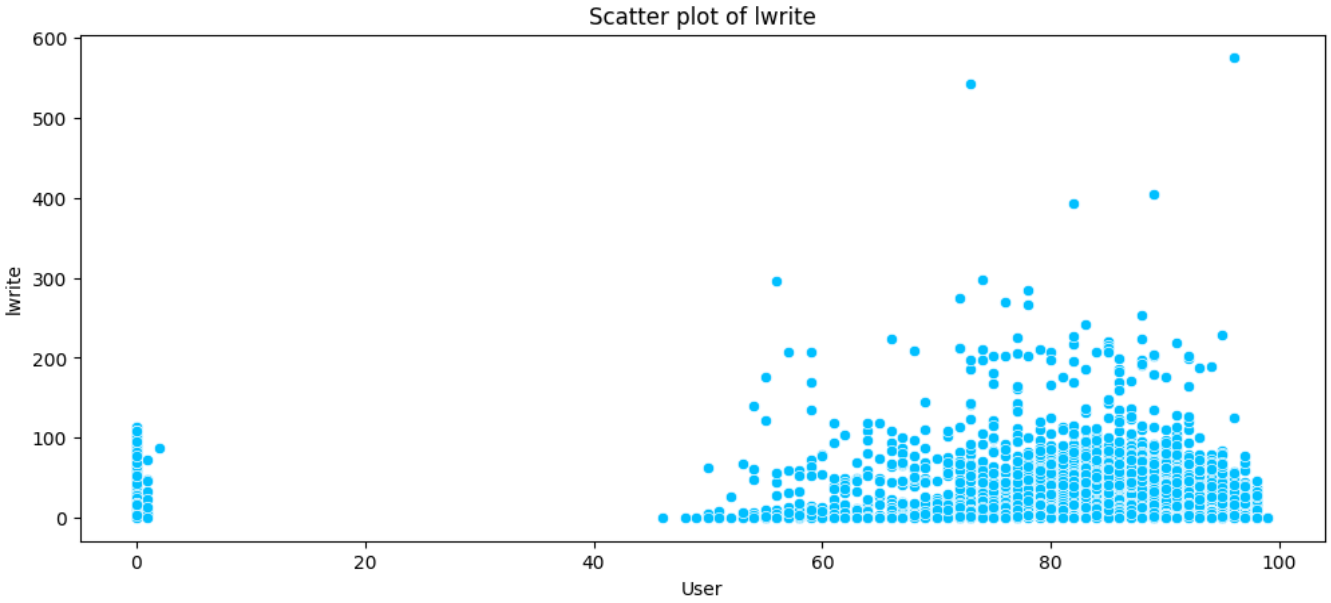


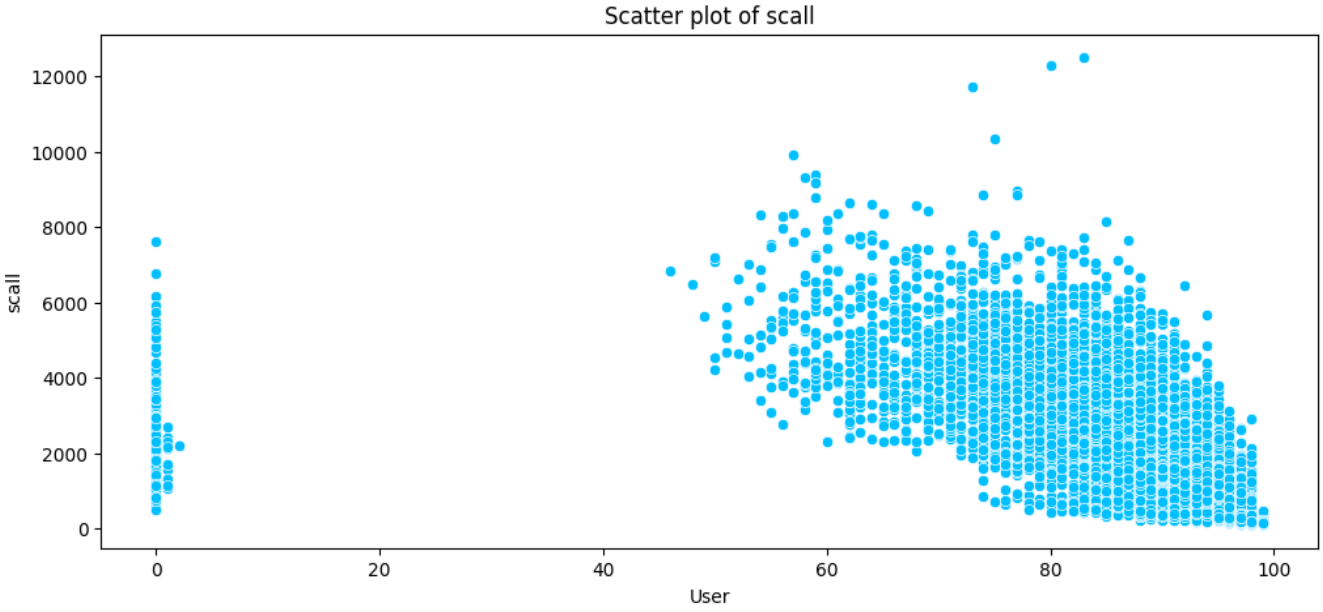
Heatmap for to check the high correlation between the variables.

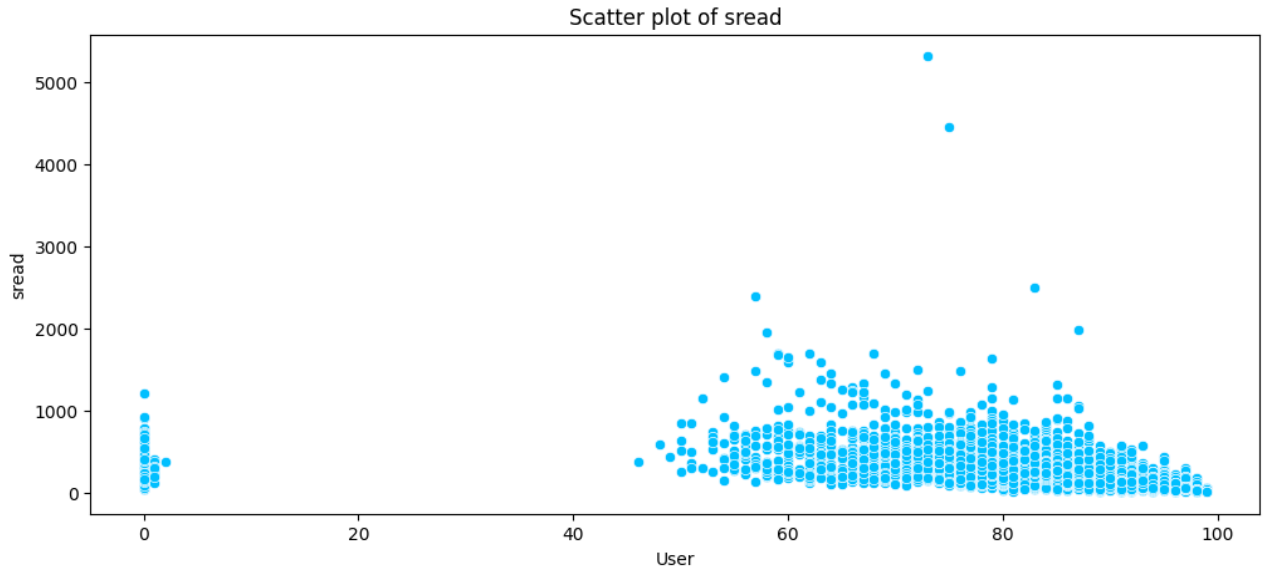


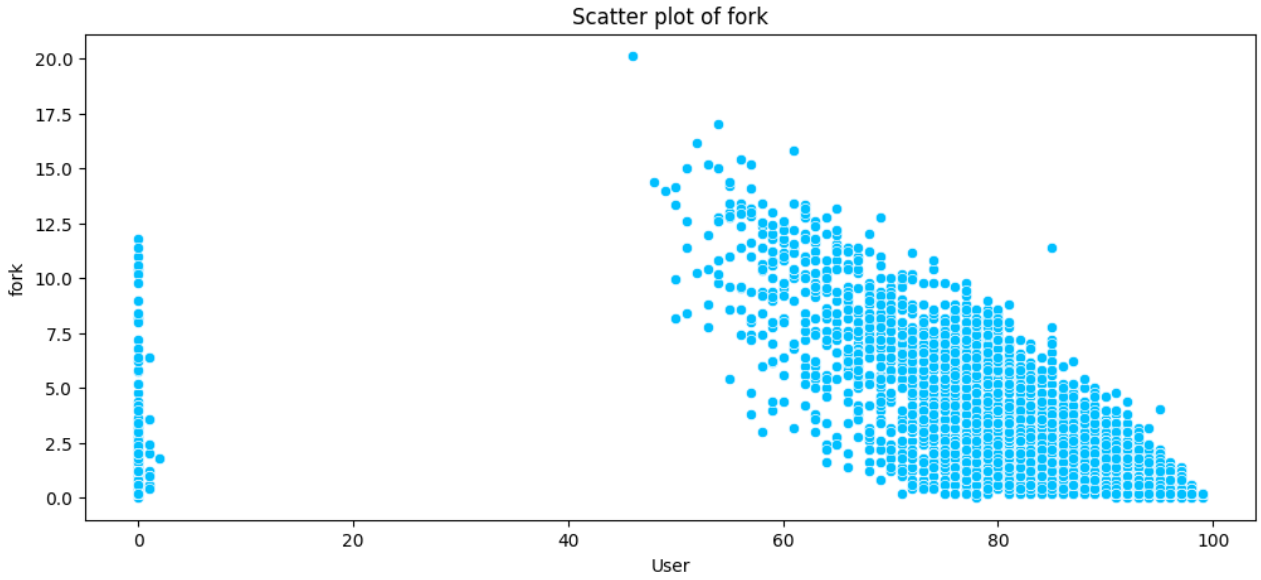


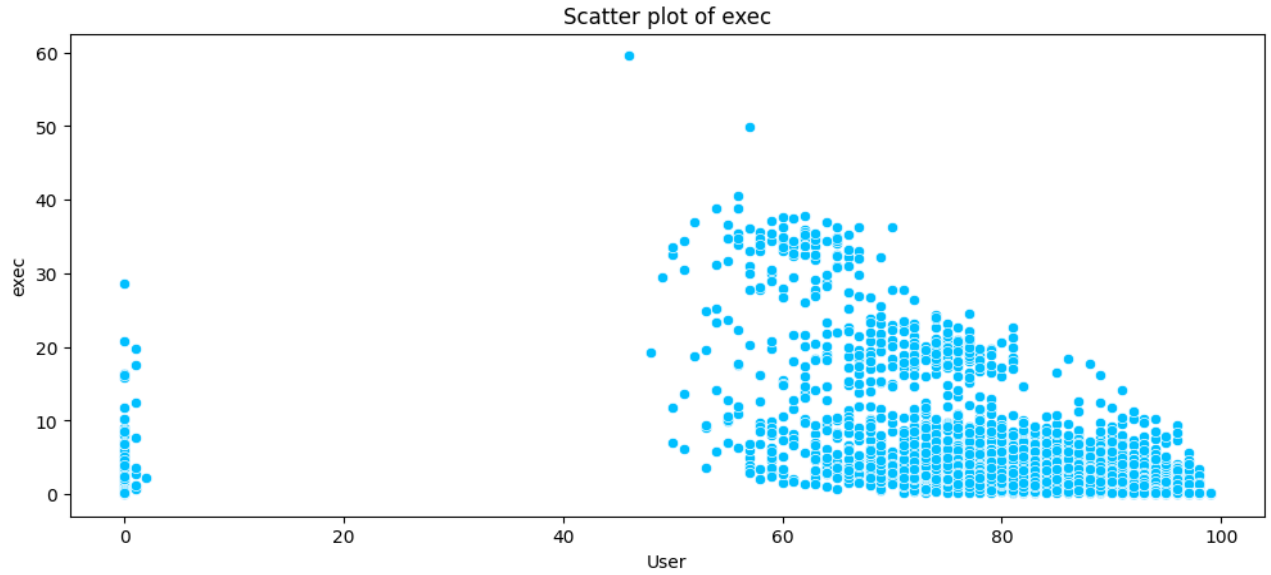

Scatter plot of lwrite
lwrite
User
0
100
200
300
400
500
600
0
20
40
60
80
100

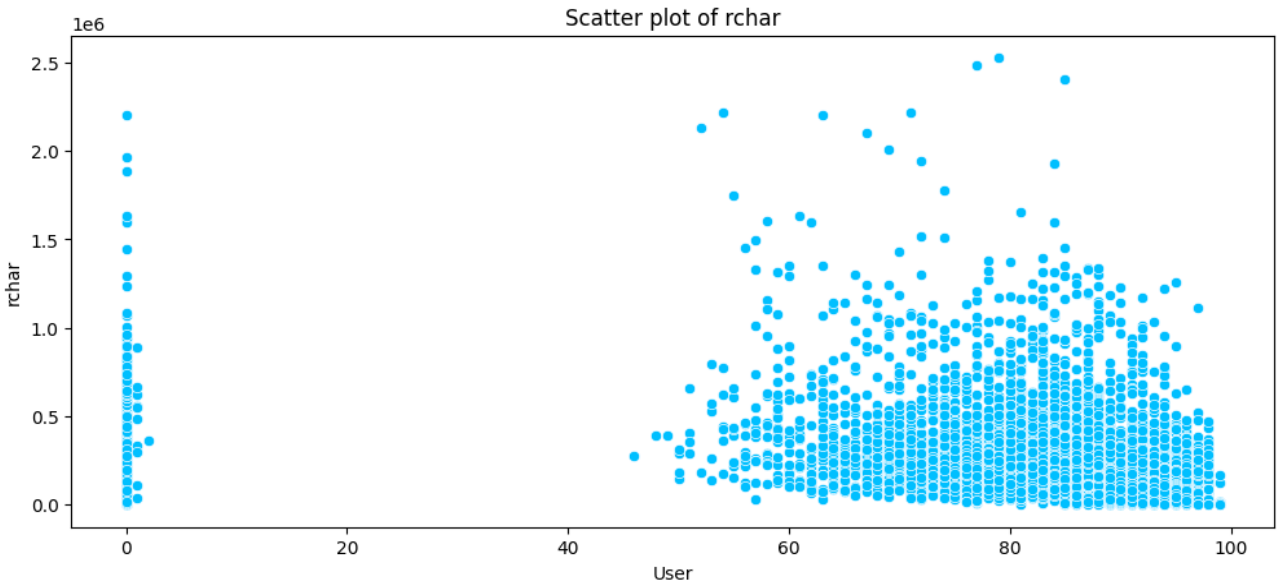

Scatter plot of lwrite
lwrite
User
0
100
200
300
400
500
600
0
20
40
60
80
100

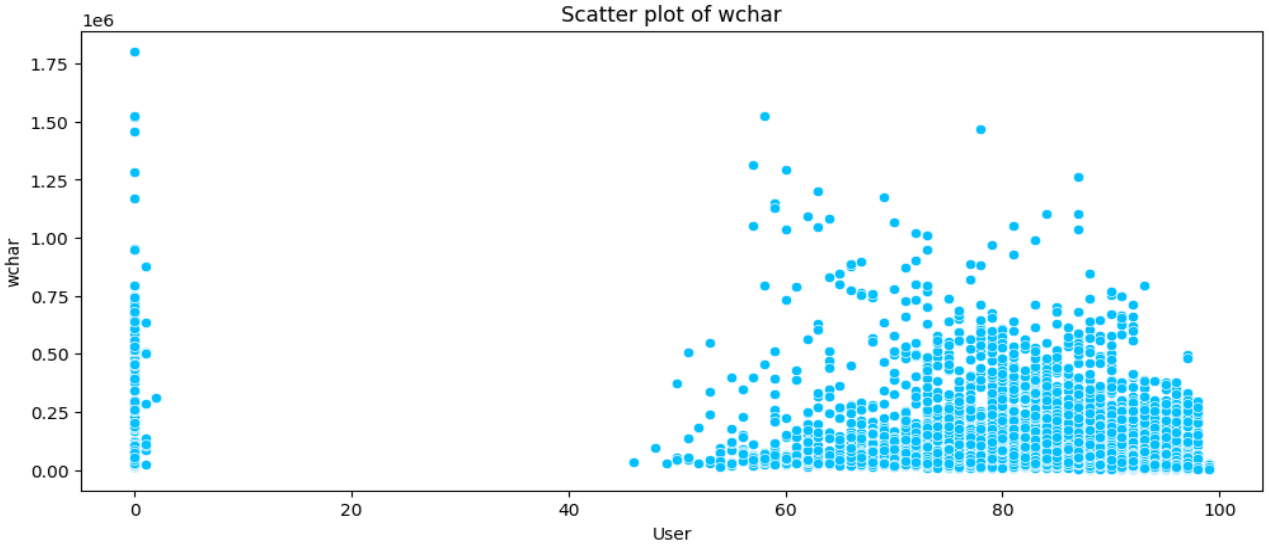

Scatter plot of scall
scall
User
0
2000
4000
6000
8000
10000
12000
0
20
40
60
80
100

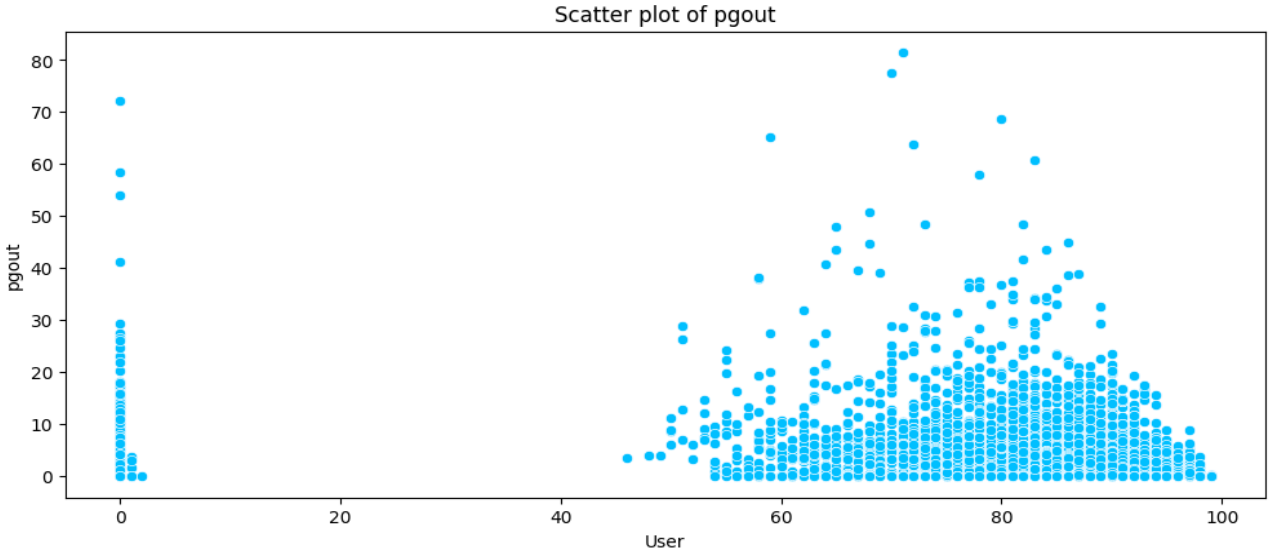

Scatter plot of sread
sread
User
0
1000
2000
3000
4000
5000
0
20
40
60
80
100

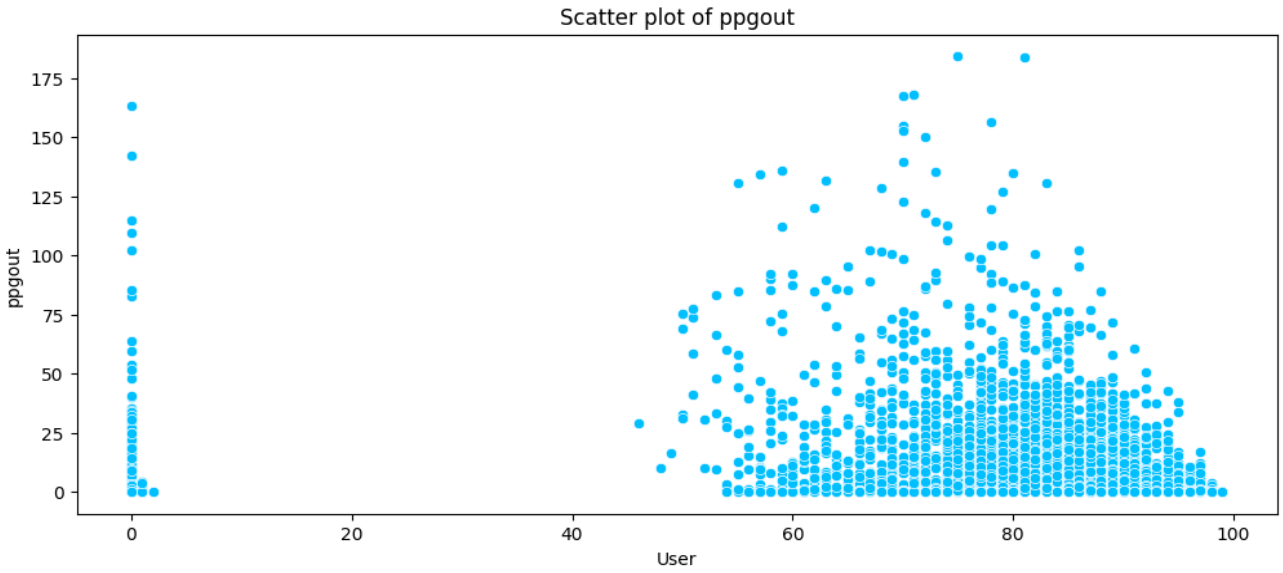

Scatter plot of fork
fork
User
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0
20
40
60
80
100

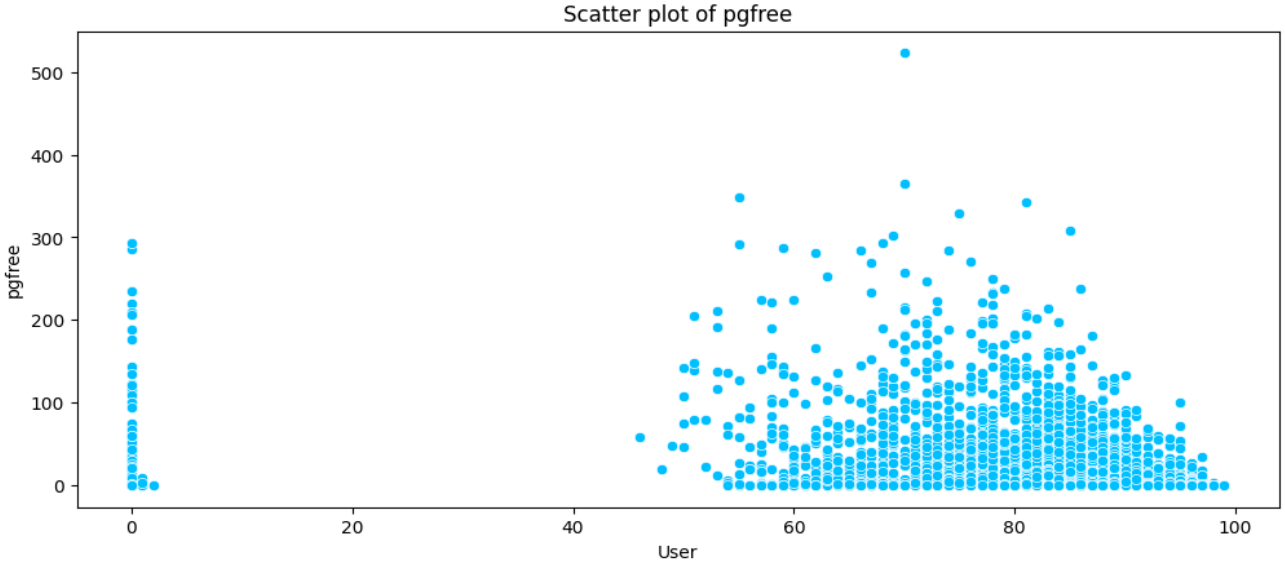

Scatter plot of exec
exec
User
0
10
20
30
40
50
60
0
20
40
60
80
100

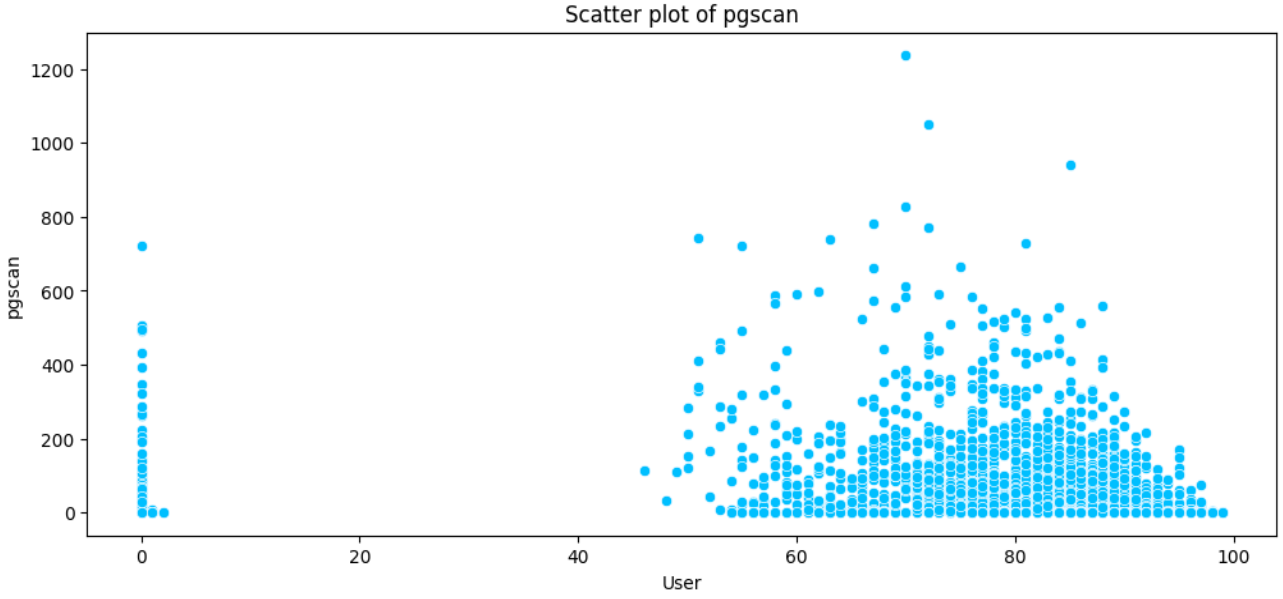

Scatter plot of rchar
1e6
2.5
2.0
1.5
1.0
0.5
0.0
rchar
0
20
40
60
80
100
User

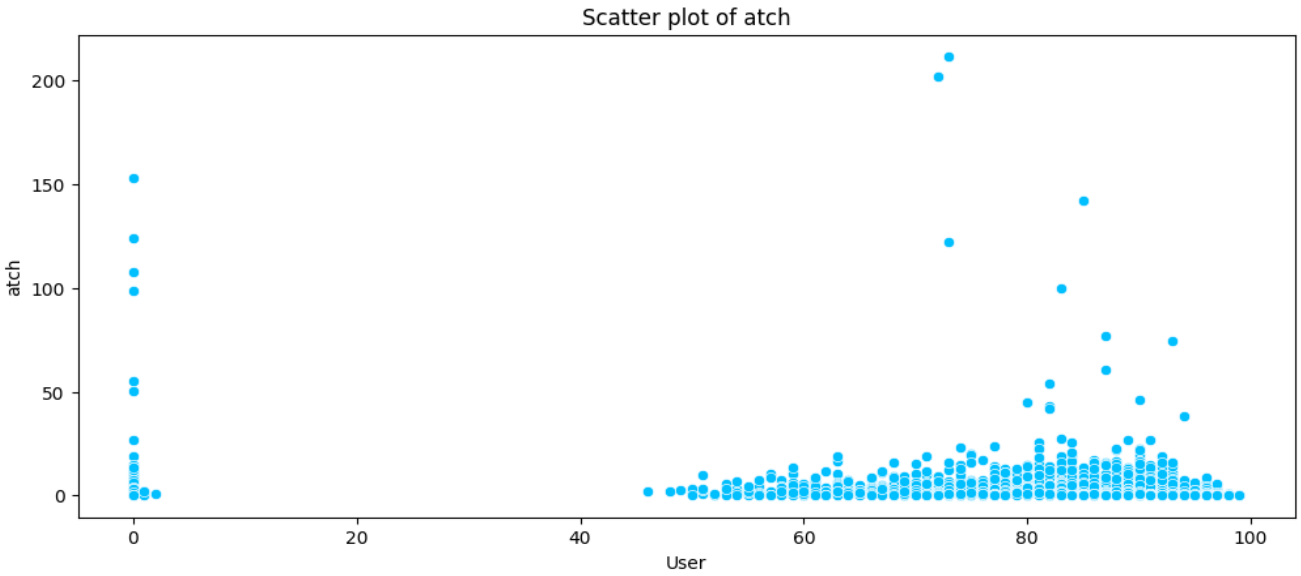

Scatter plot of wchar
1e6
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
wchar
0
20
40
60
80
100
User

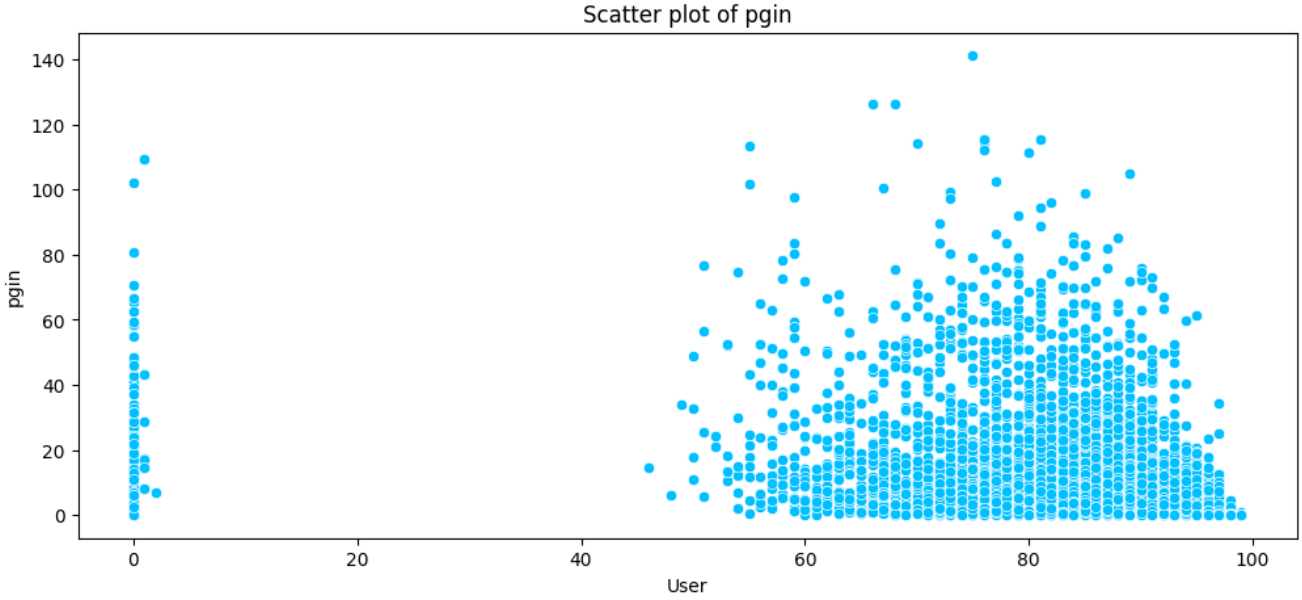

Scatter plot of pgout
80
70
60
50
40
30
20
10
0
pgout
0
20
40
60
80
100
User

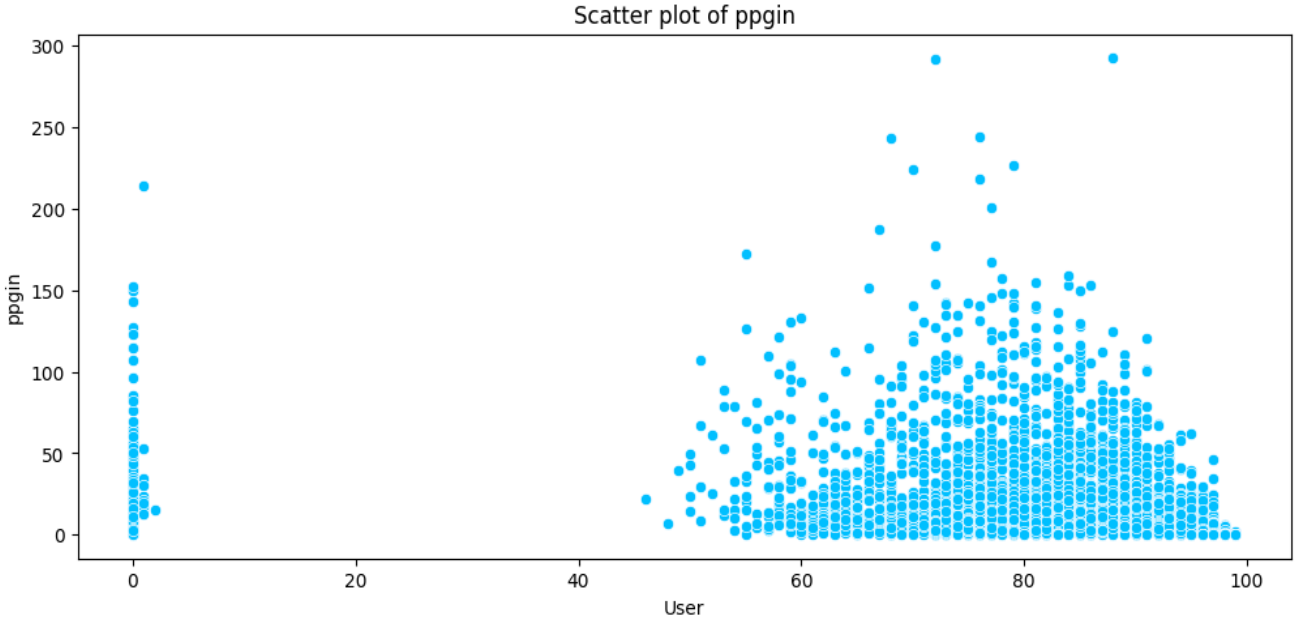

Scatter plot of ppgout
ppgout
User
0
25
50
75
100
125
150
175
0
20
40
60
80
100

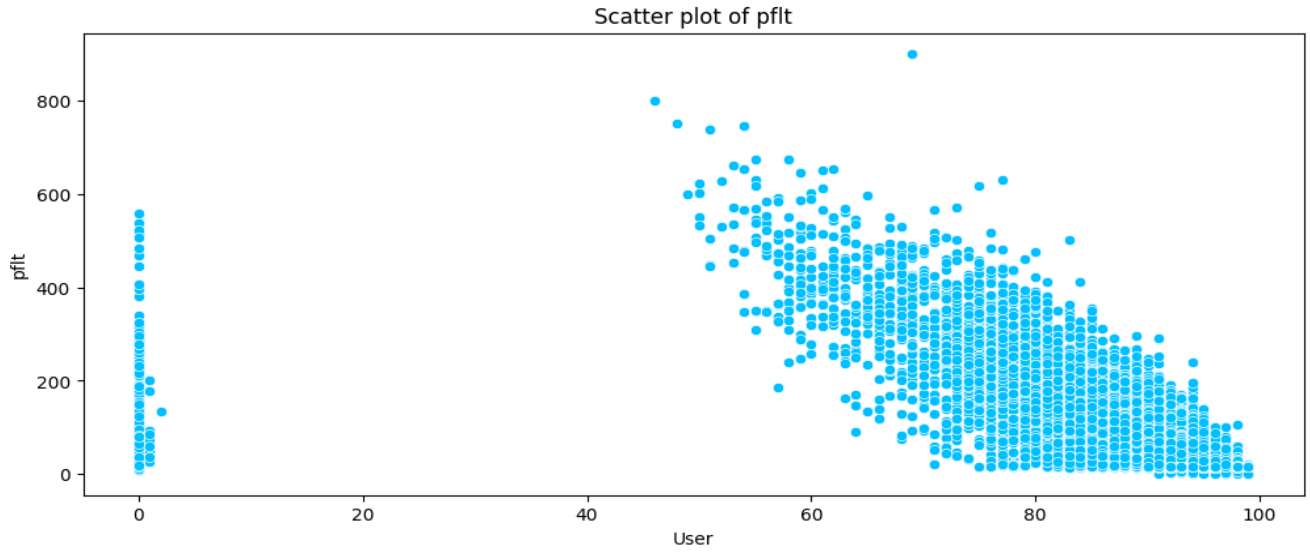

Scatter plot of pgfree
pgfree
User
0
100
200
300
400
500
0
20
40
60
80
100

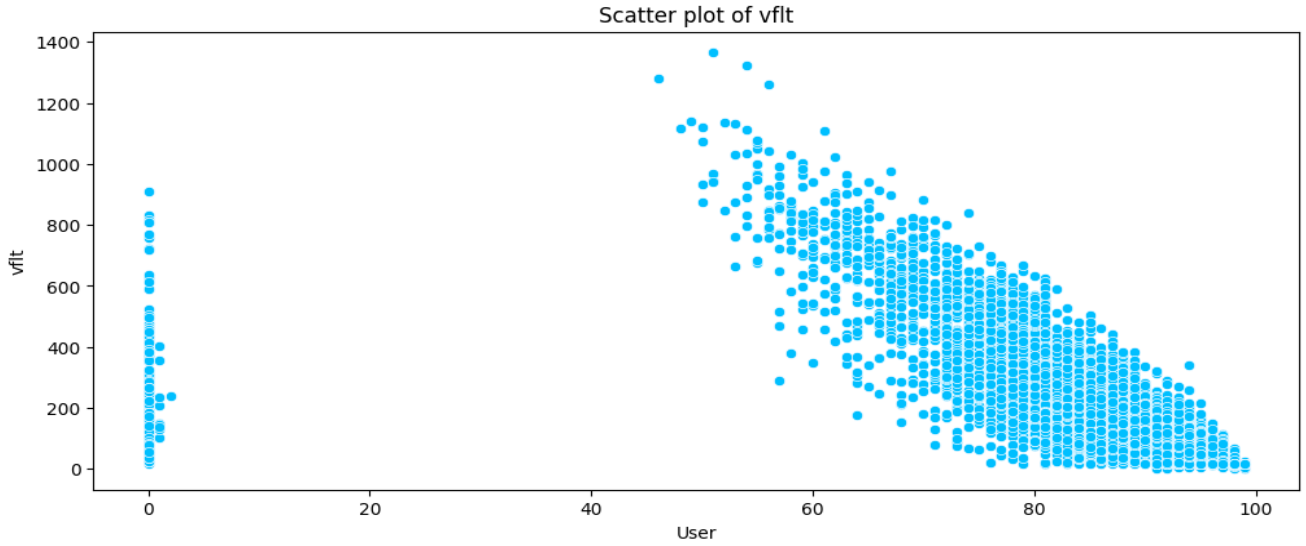

Scatter plot of pgscan
pgscan
User
0
200
400
600
800
1000
1200
0
20
40
60
80
100

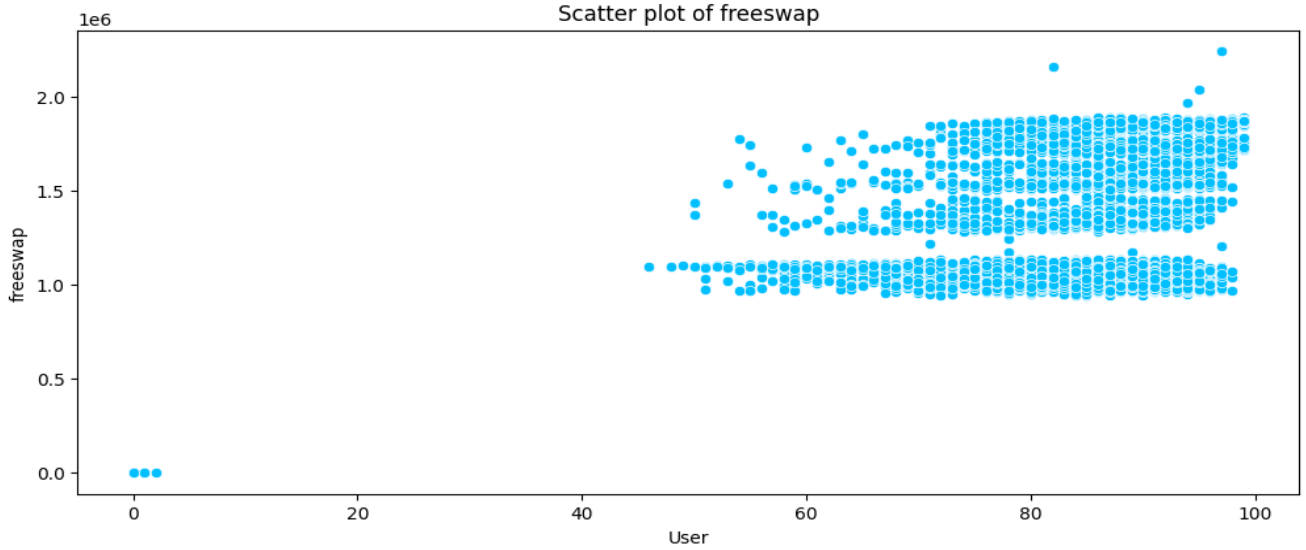

Scatter plot of atch
atch
User
0
50
100
150
200
0
20
40
60
80
100

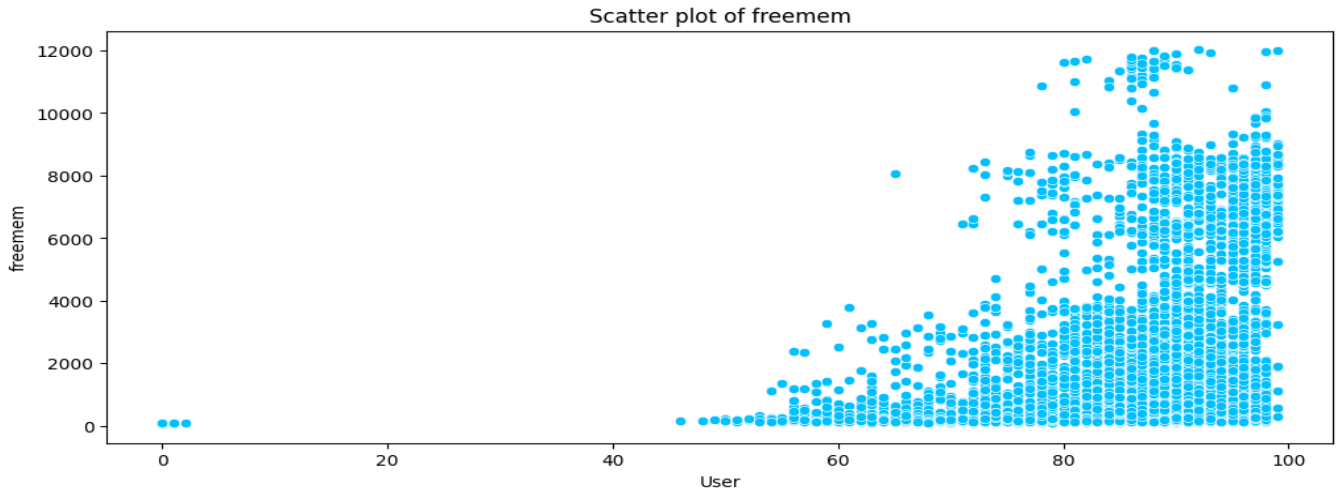

Scatter plot of pgin
pgin
User
0
20
40
60
80
100
120
140
0
20
40
60
80
100

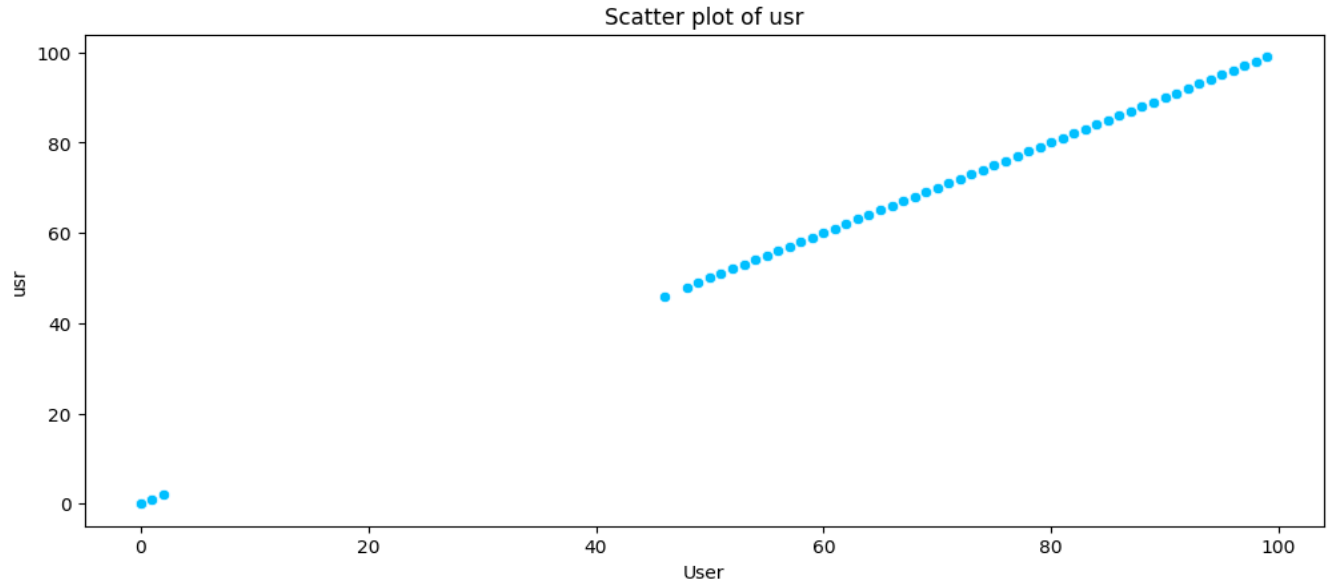

Scatter plot of ppgin
ppgin
User
0
50
100
150
200
250
300
0
20
40
60
80
100

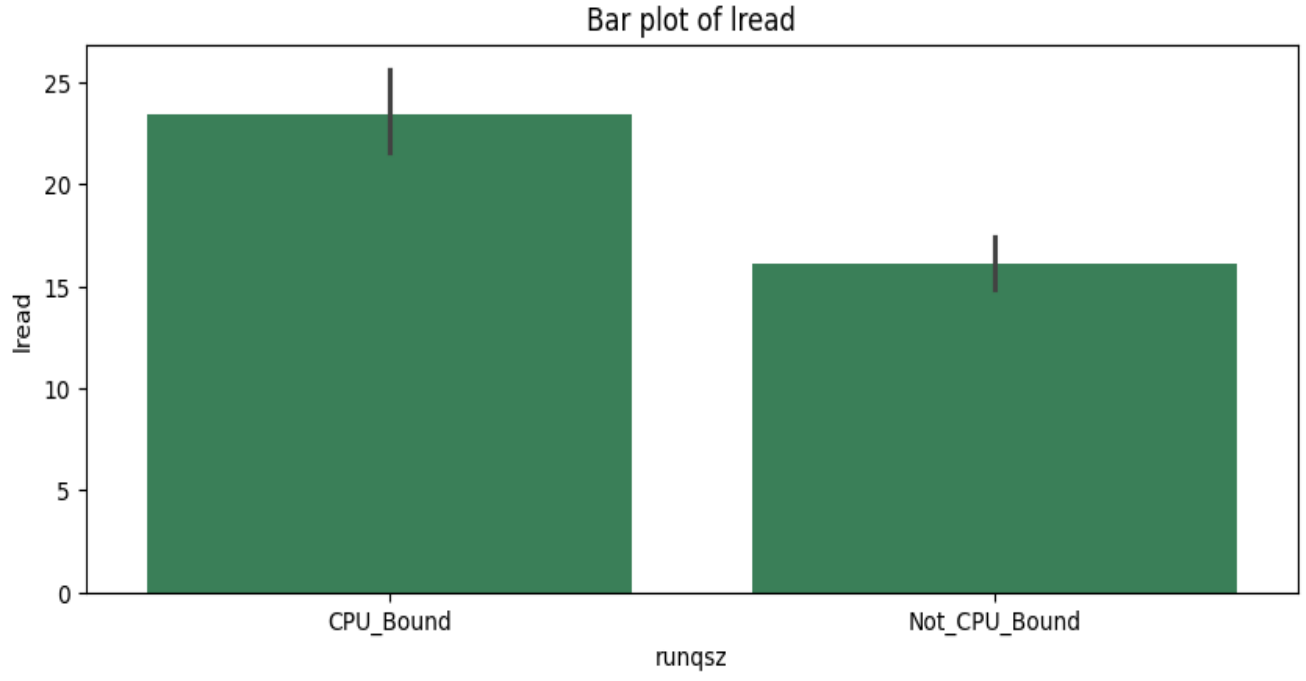

Scatter plot of pflt
pflt
User
0
200
400
600
800
0
20
40
60
80
100

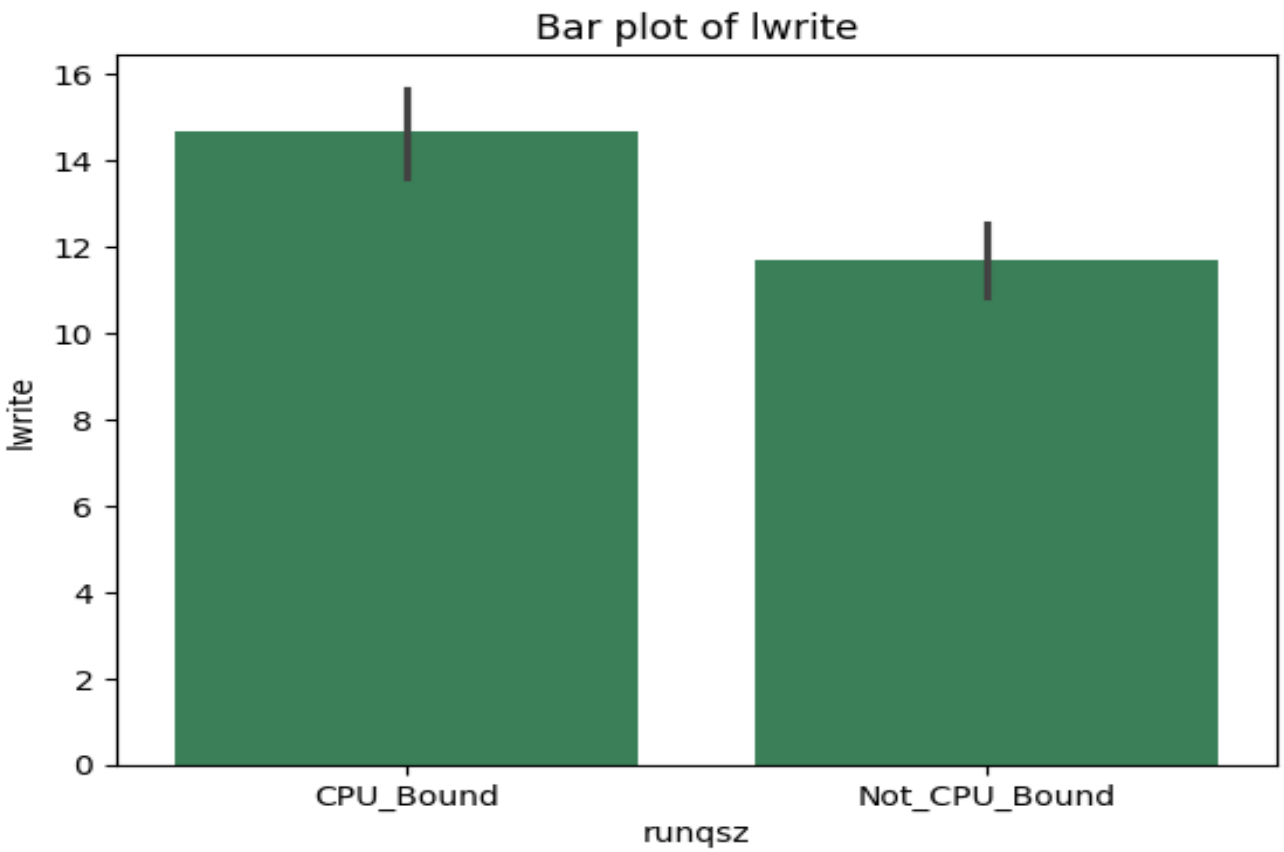

Scatter plot of vflt
vflt
User
0
200
400
600
800
1000
1200
1400
0
20
40
60
80
100

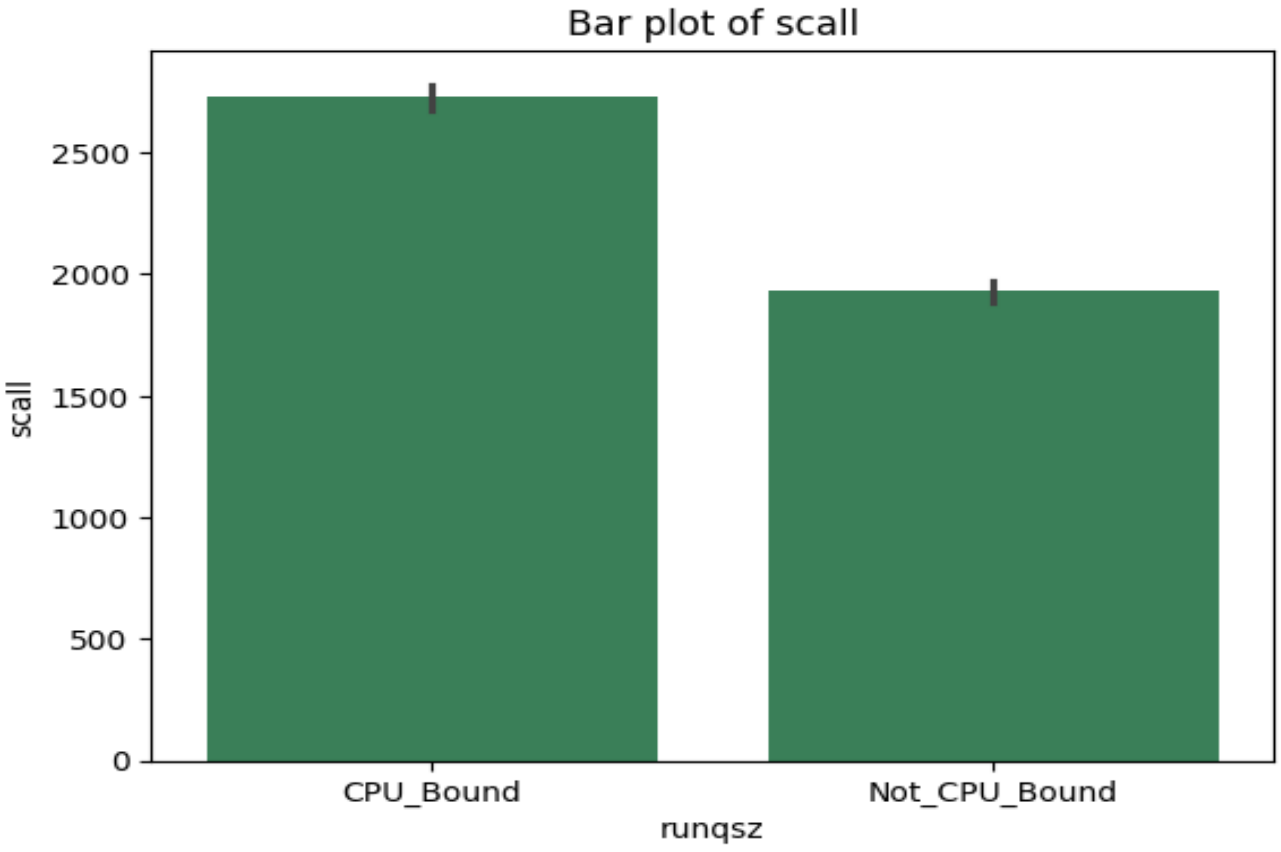

Scatter plot of freeswap
1e6
freeswap
User
0.0
0.5
1.0
1.5
2.0
0
20
40
60
80
100

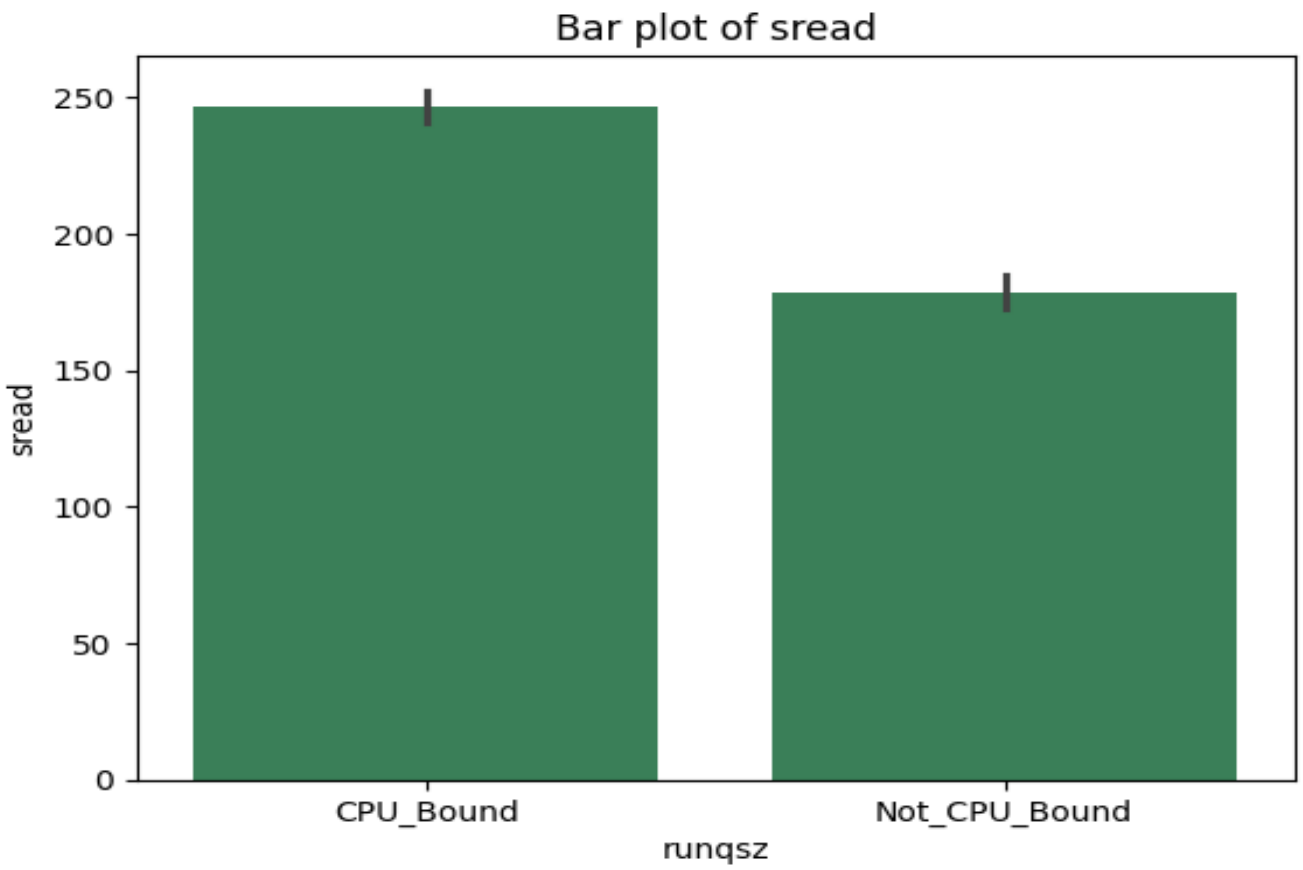

Scatter plot of freemem
freemem
User
0
2000
4000
6000
8000
10000
12000
0
20
40
60
80
100

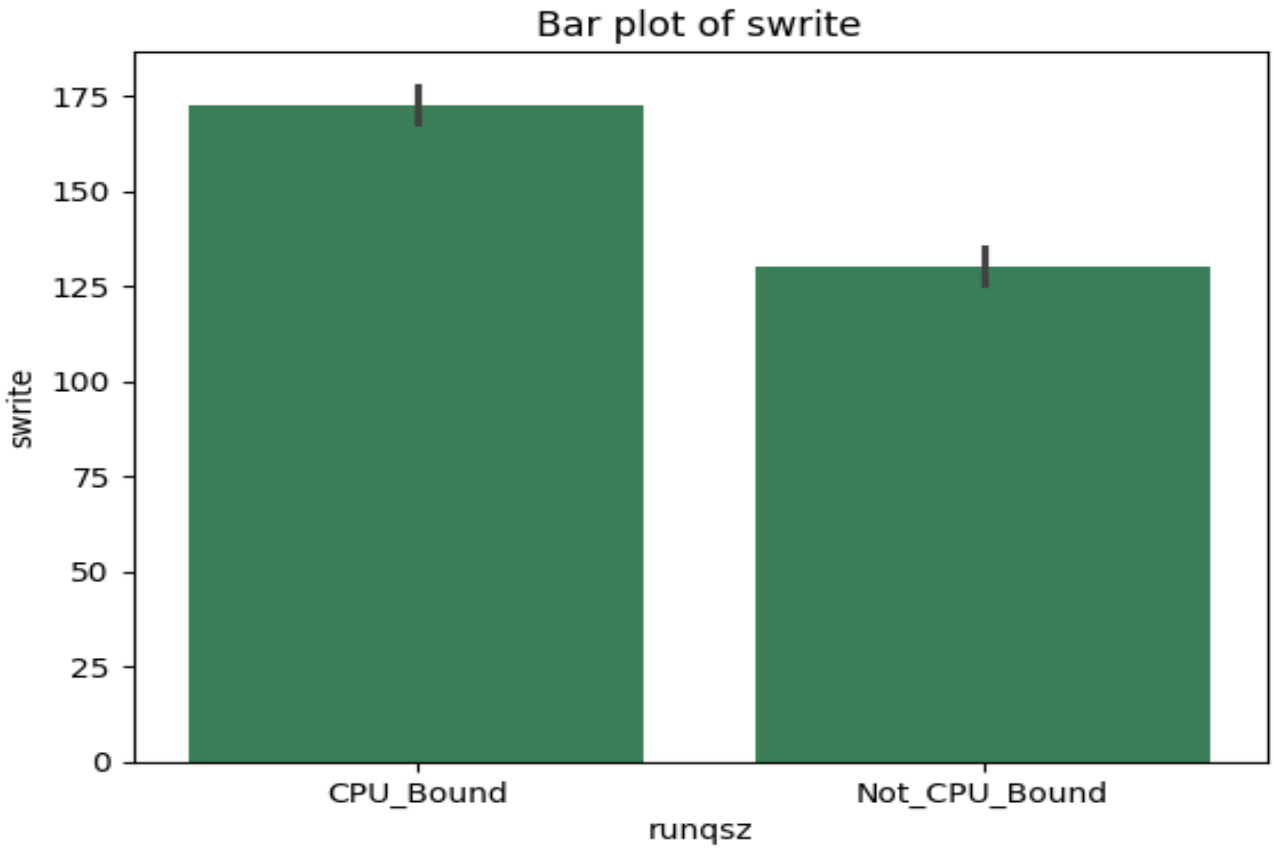

Scatter plot of usr
usr
User
0
20
40
60
80
100
0
20
40
60
80
100

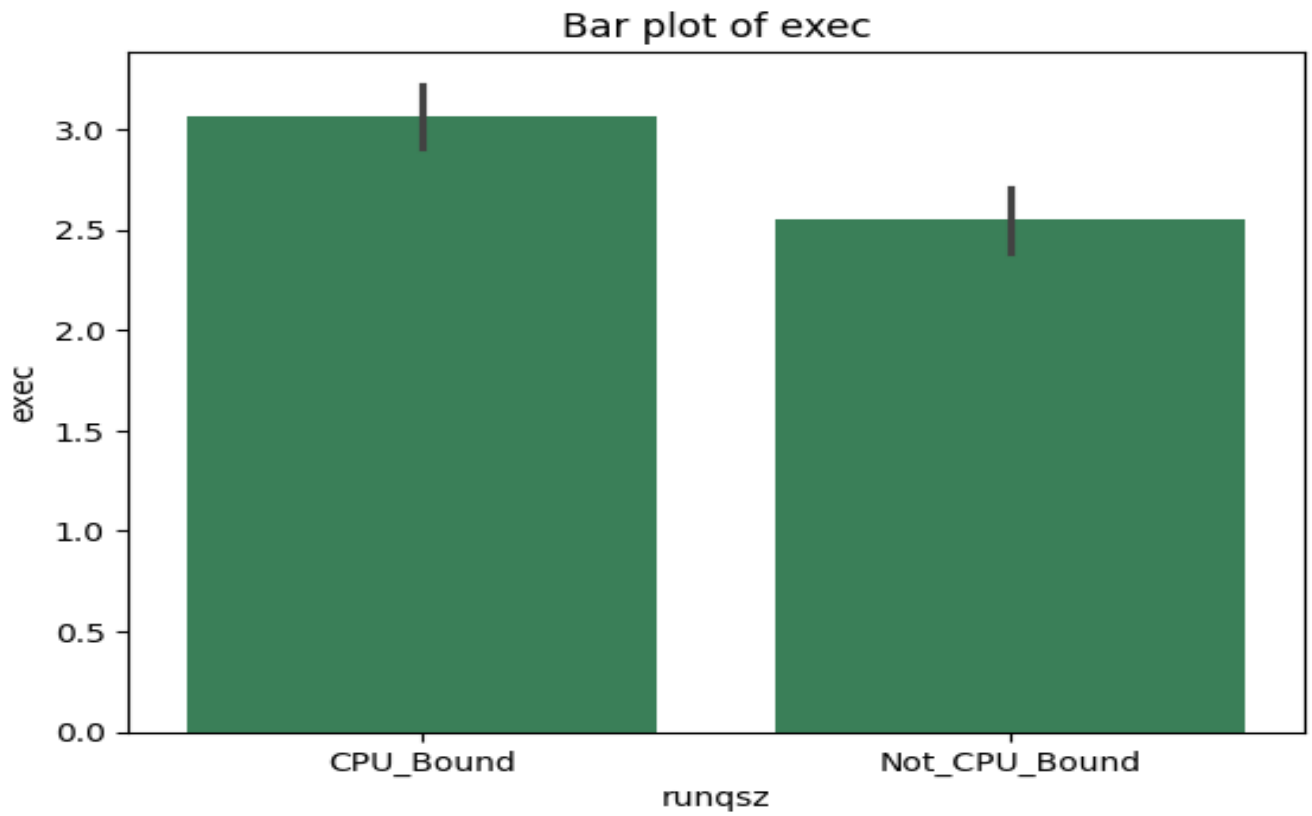

Bar plot of lread
lread
runqsz
0
5
10
15
20
25
CPU_Bound
Not_CPU_Bound

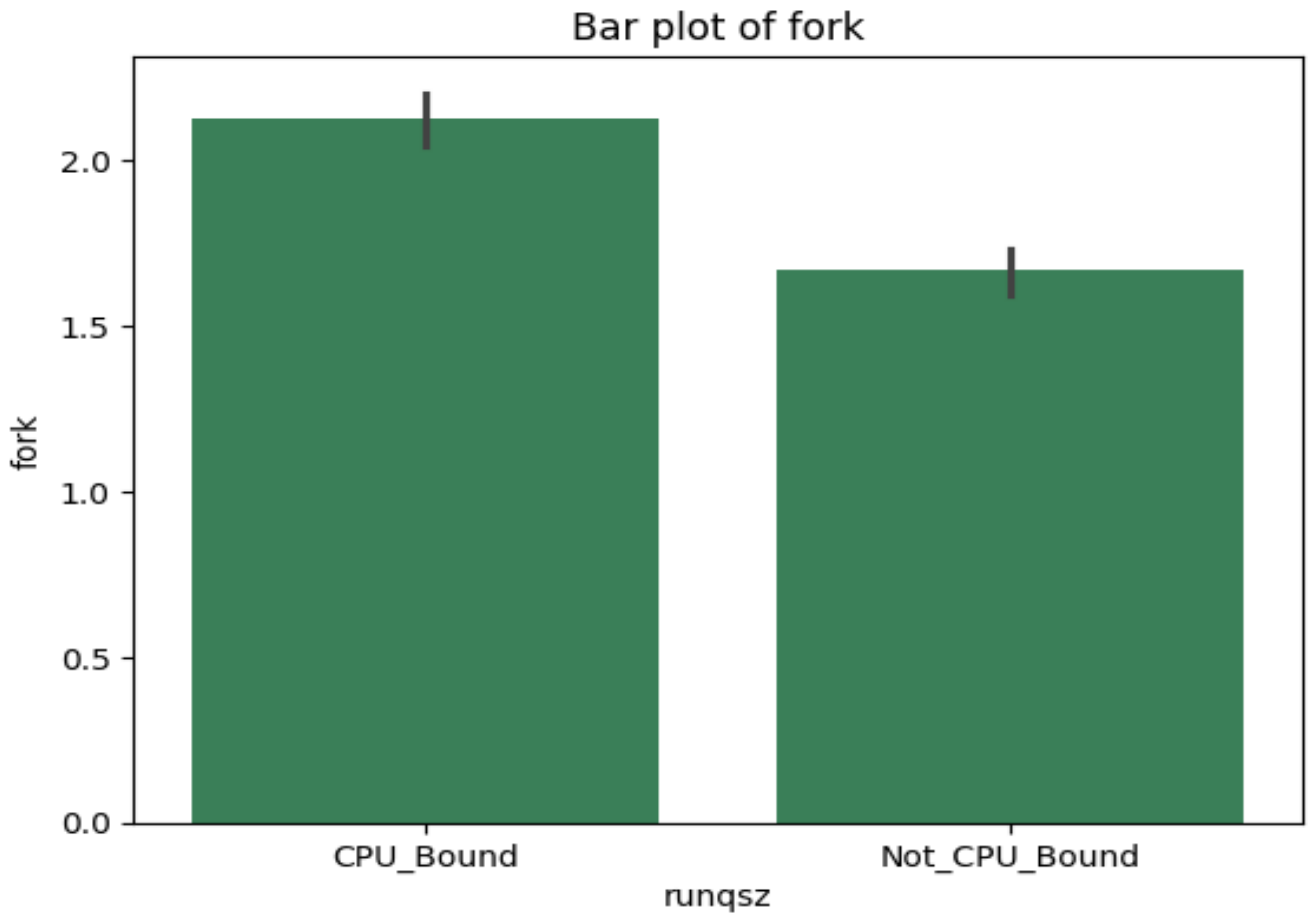

Bar plot of lwrite
lwrite
16
14
12
10
8
6
4
2
0
CPU_Bound
Not_CPU_Bound
runqsz

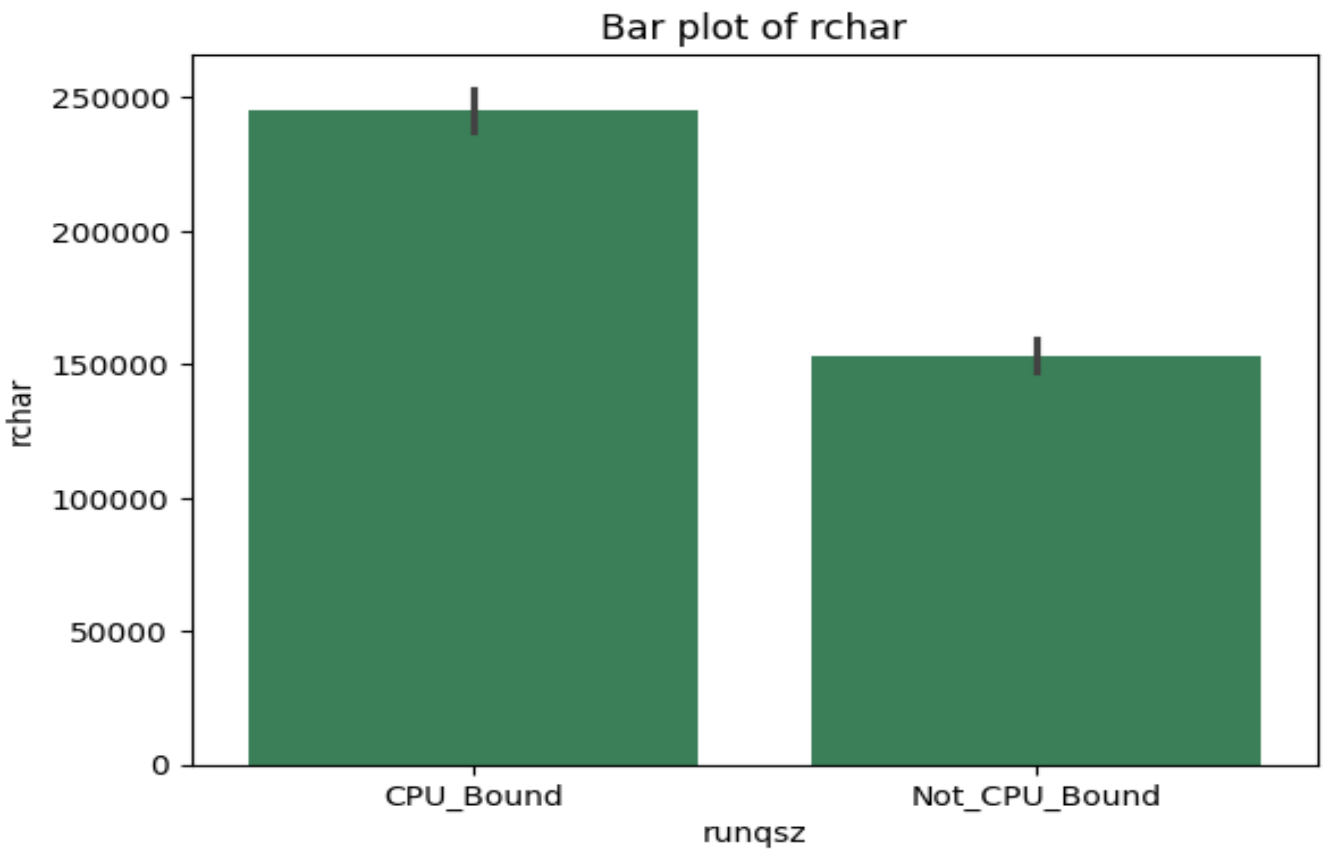

Bar plot of scall
scall
2500
2000
1500
1000
500
0
CPU_Bound
Not_CPU_Bound
runqsz

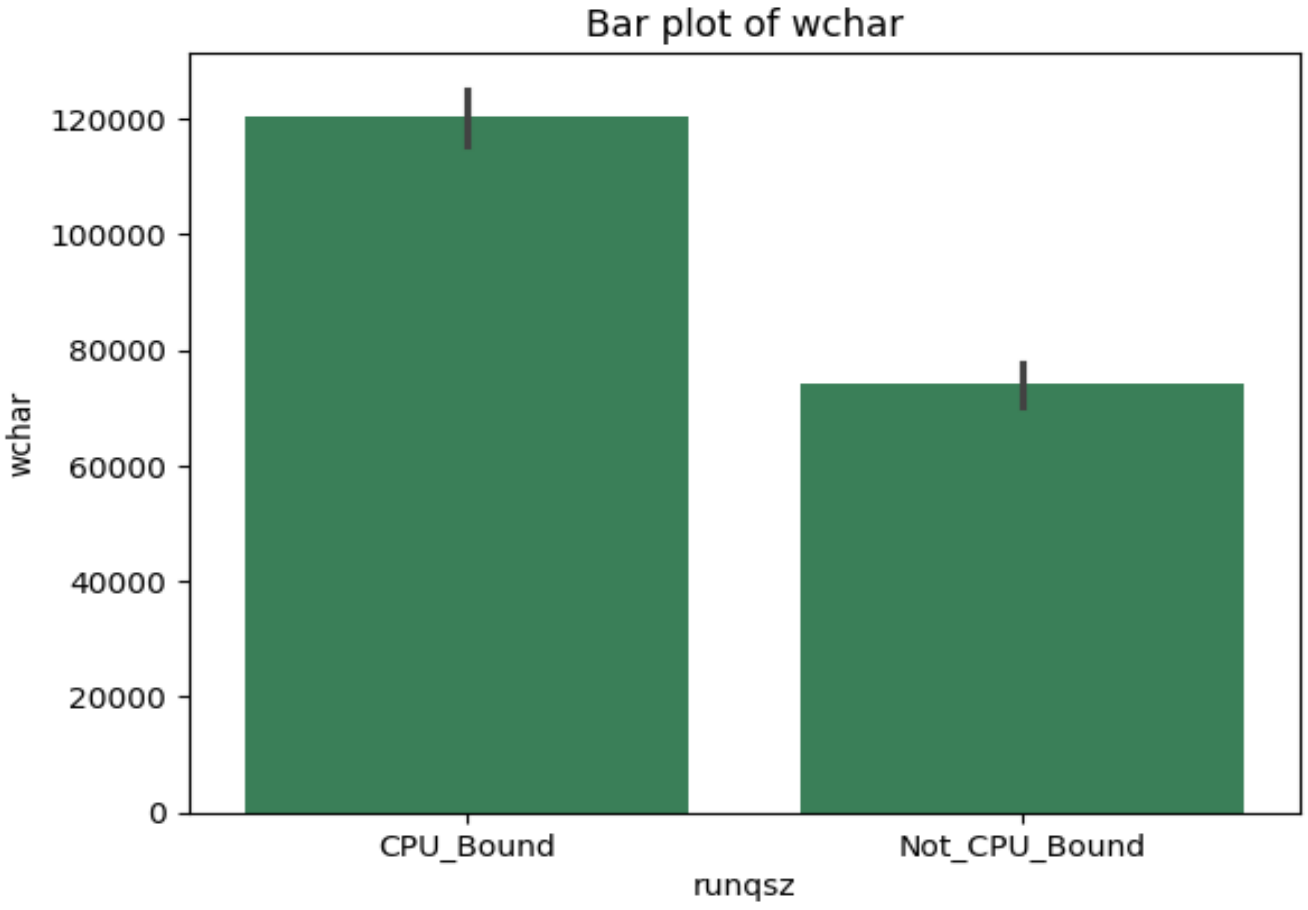

Bar plot of sread
250
200
150
100
50
0
sread
CPU_Bound
Not_CPU_Bound
runqsz

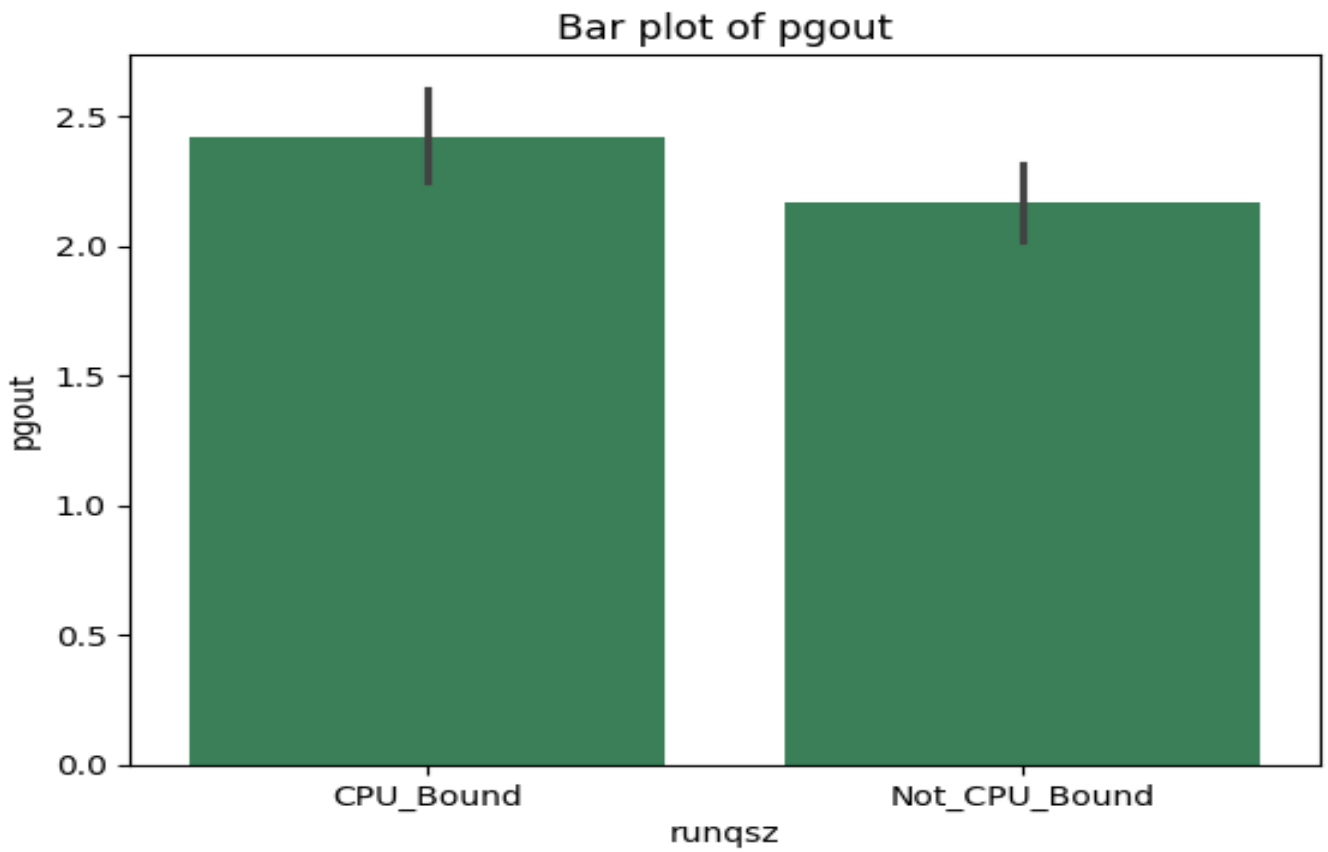

Bar plot of swrite
175
150
125
100
75
50
25
0
swrite
CPU_Bound
Not_CPU_Bound
runqsz

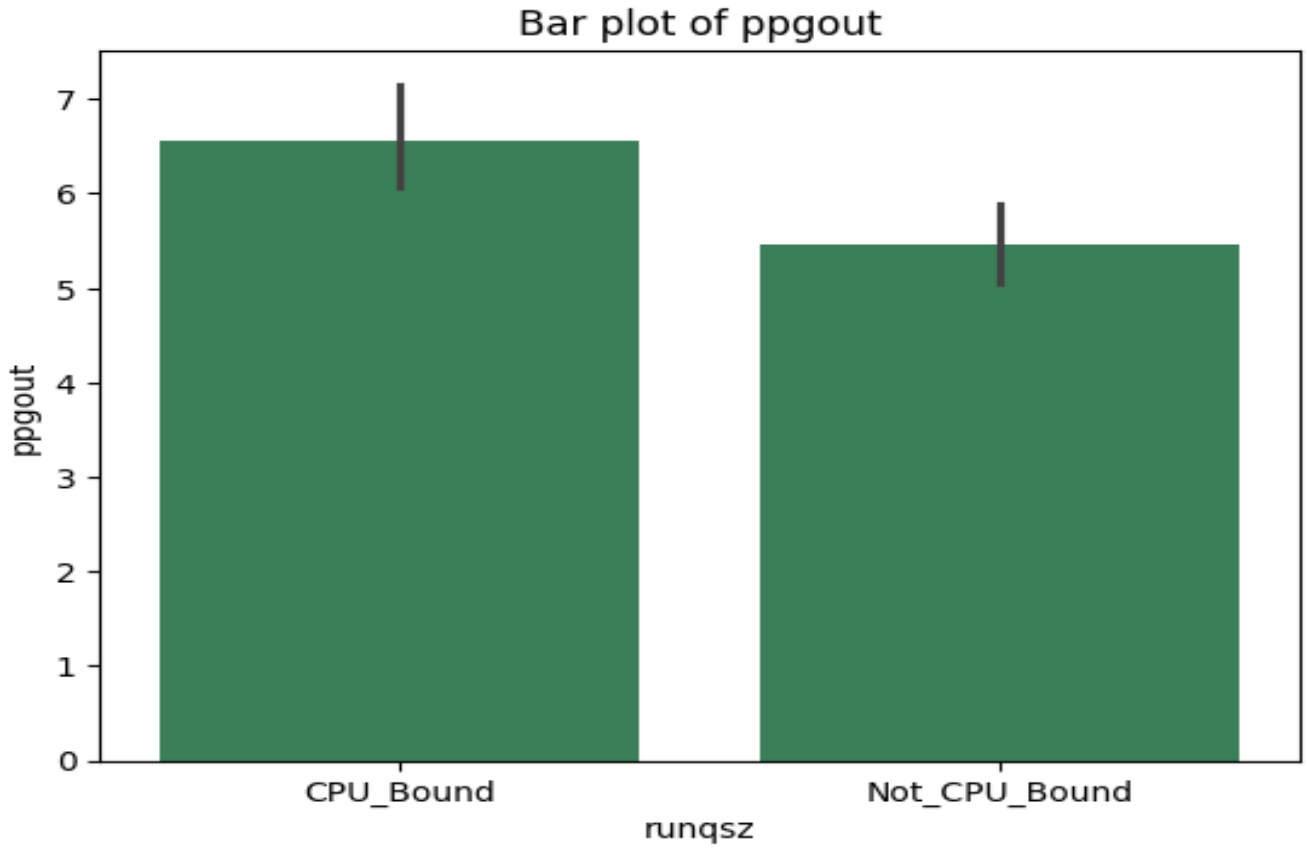

Bar plot of exec
3.0
2.5
2.0
1.5
1.0
0.5
0.0
exec
CPU_Bound
Not_CPU_Bound
runqsz

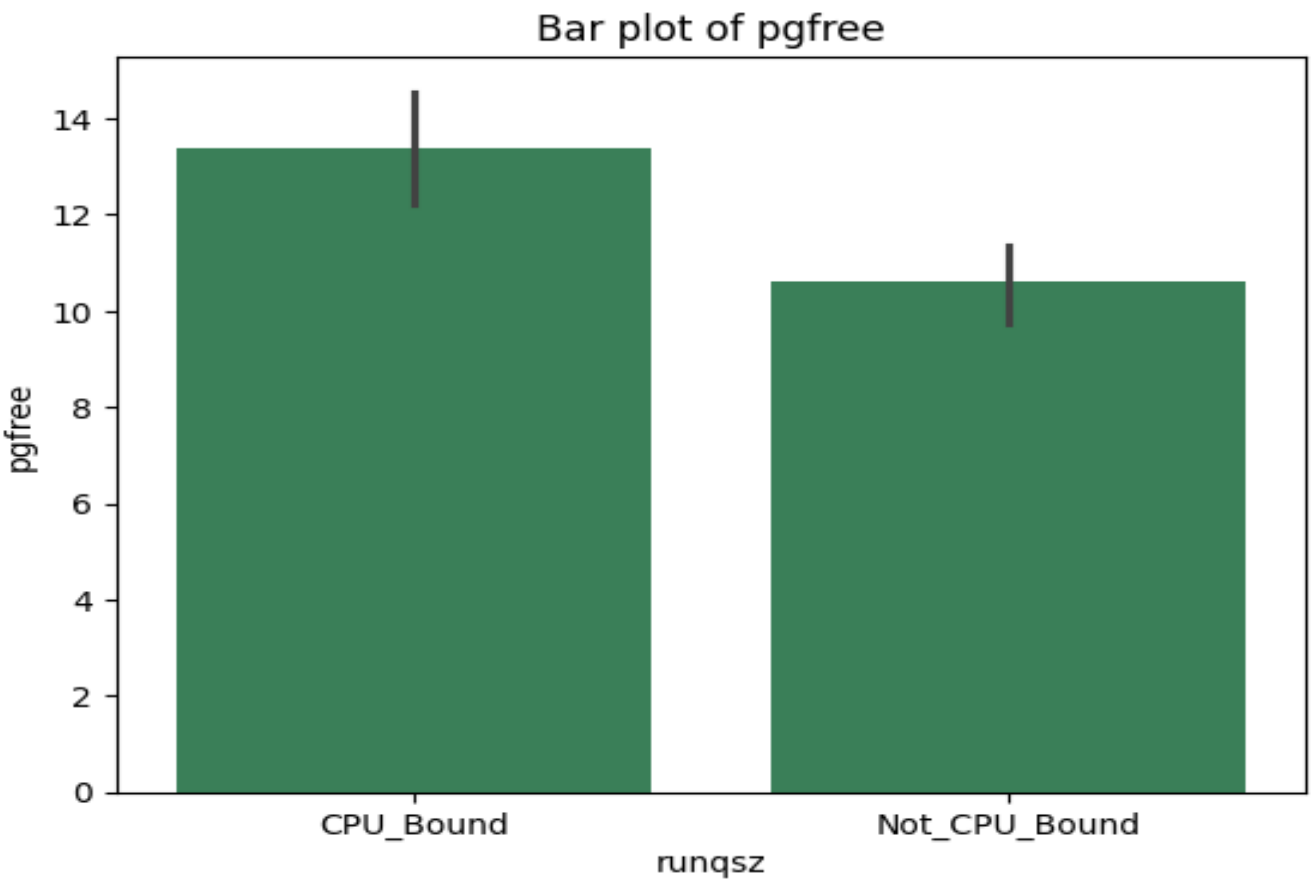

Bar plot of fork
2.0
1.5
1.0
0.5
0.0
fork
CPU_Bound
Not_CPU_Bound
runqsz

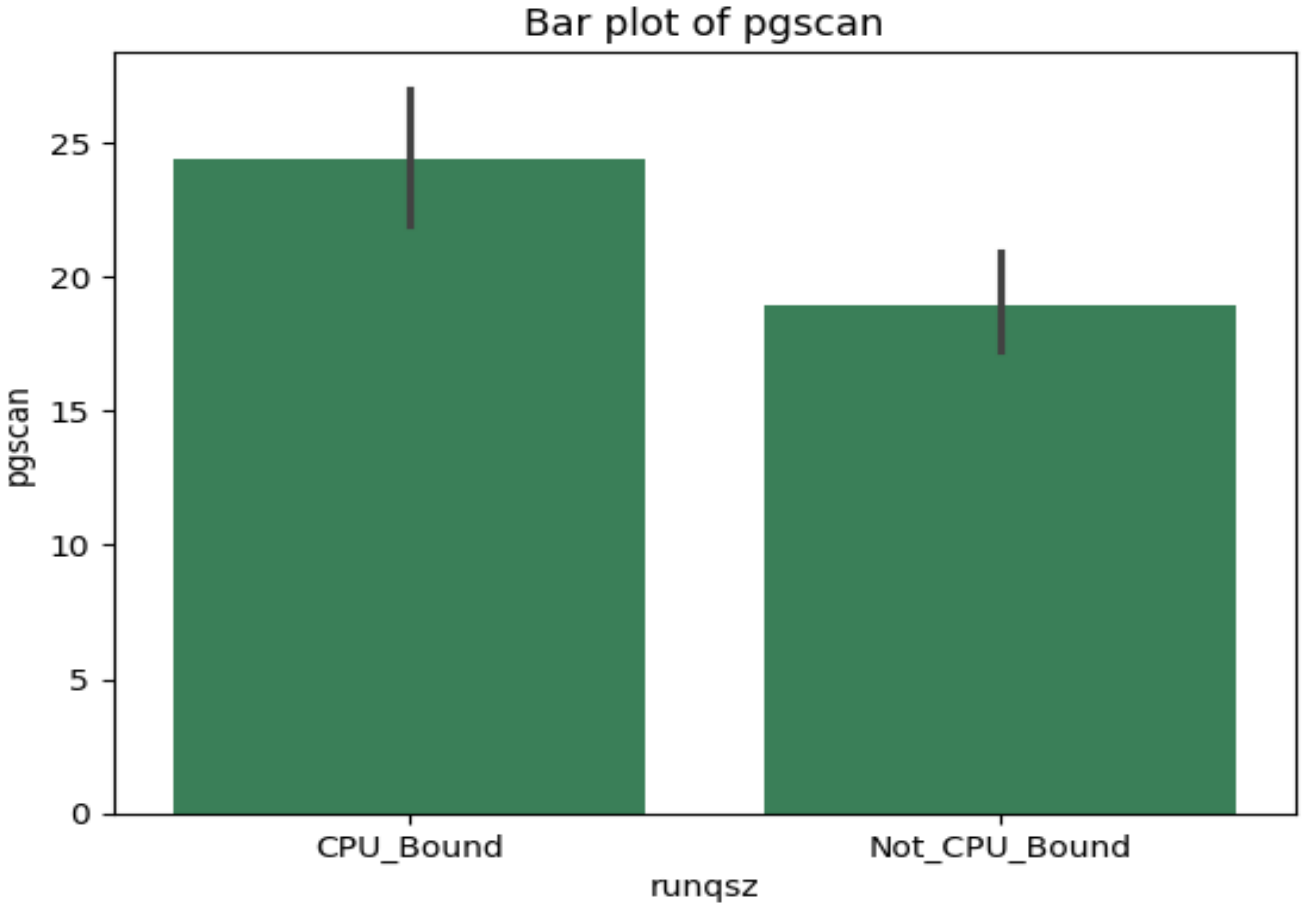

Bar plot of rchar
250000
200000
150000
100000
50000
0
rchar
CPU_Bound
Not_CPU_Bound
runqsz

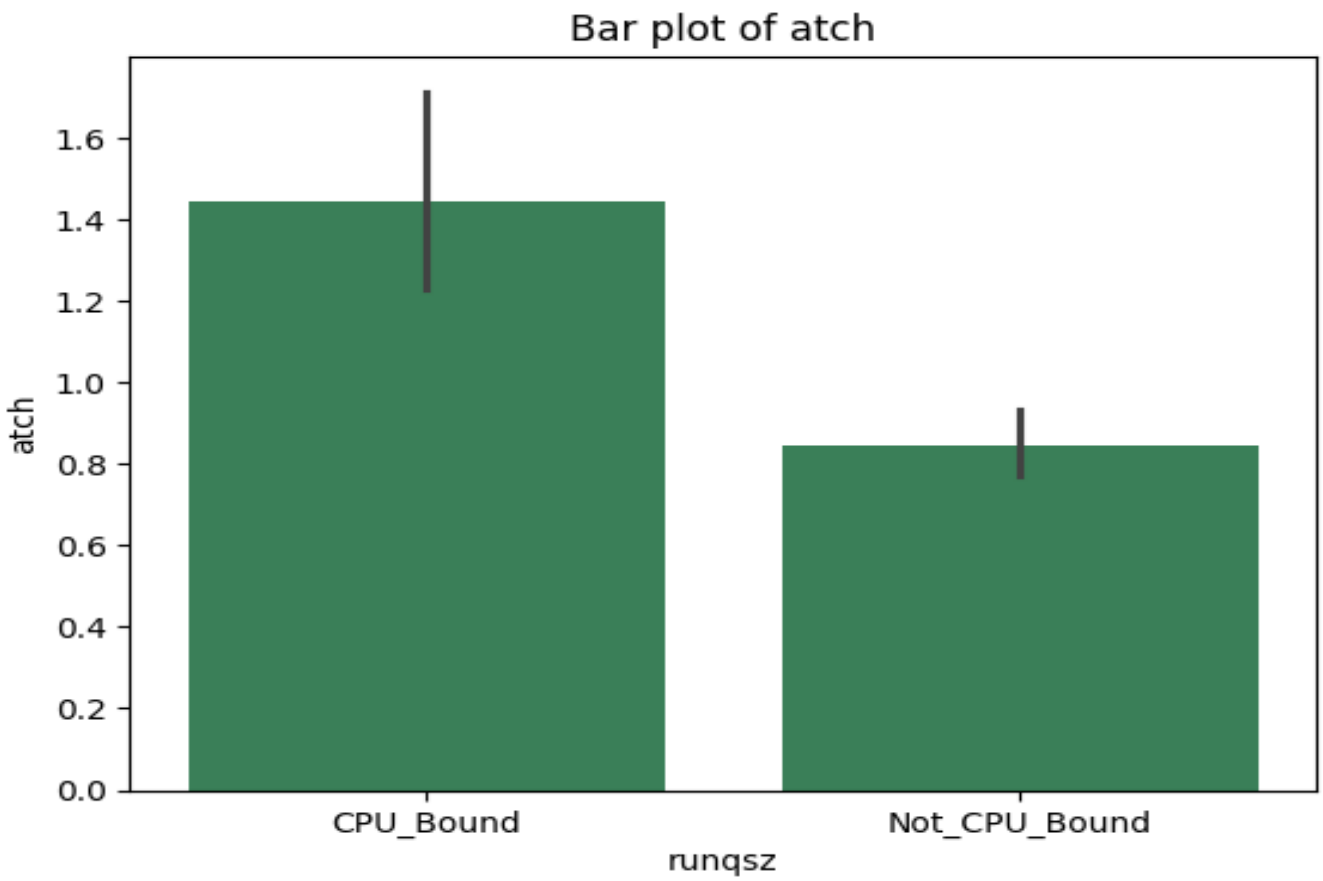

Bar plot of wchar
120000
100000
80000
60000
40000
20000
0
wchar
CPU_Bound
Not_CPU_Bound
runqsz

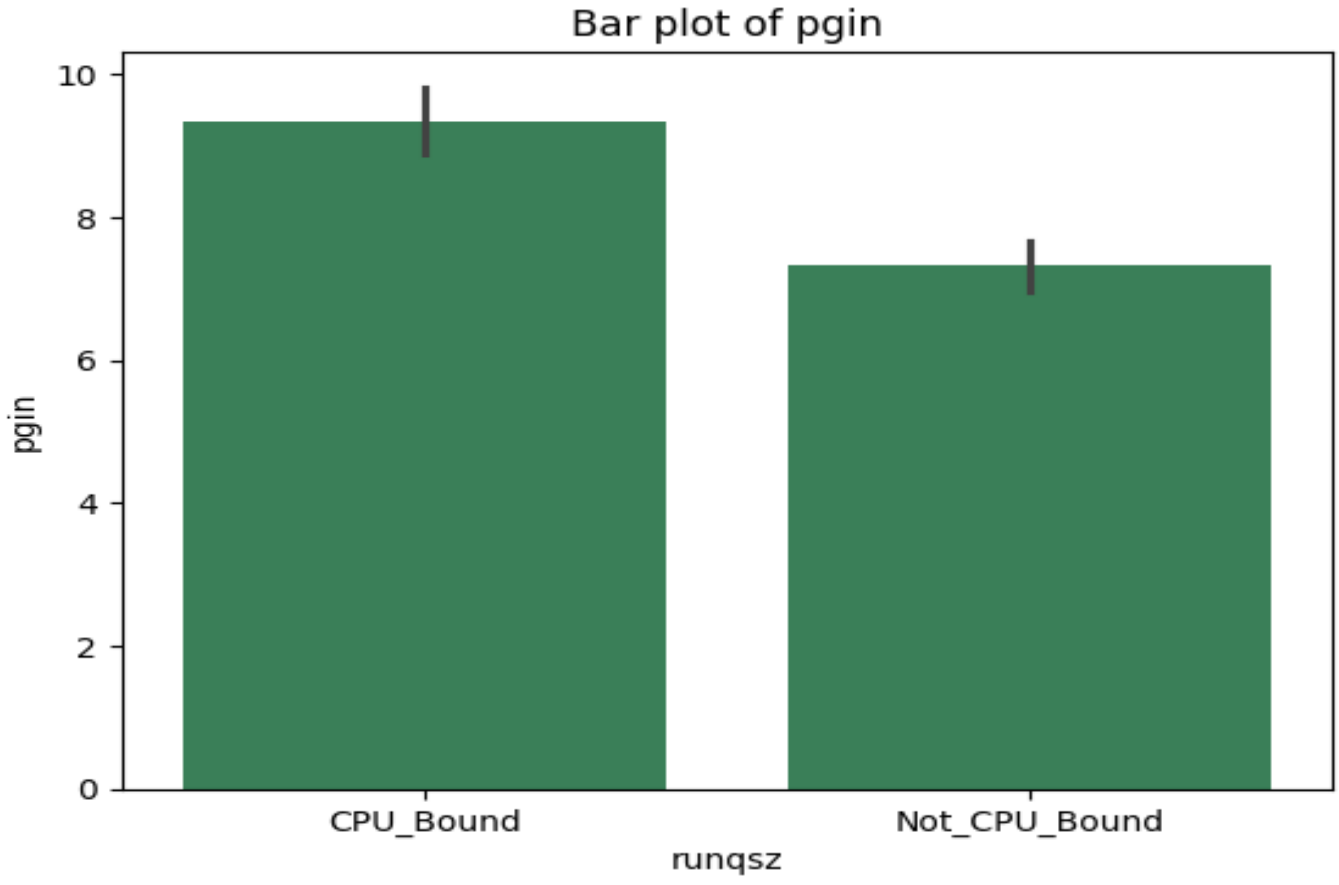

Bar plot of pgout
2.5
2.0
1.5
1.0
0.5
0.0
pgout
CPU_Bound
Not_CPU_Bound
runqsz

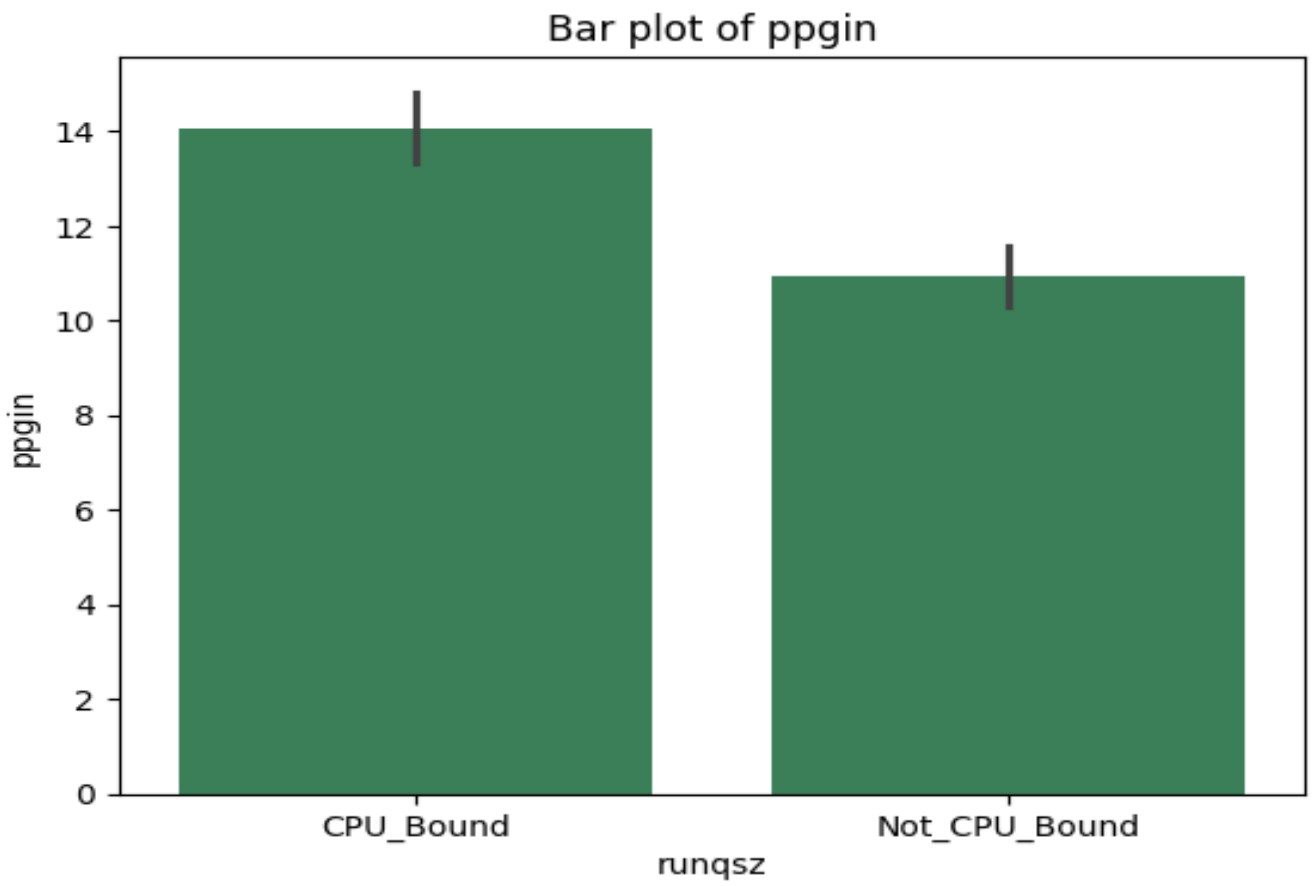

Bar plot of ppgout
7
6
5
4
3
2
1
0
ppgout
CPU_Bound
Not_CPU_Bound
runqsz

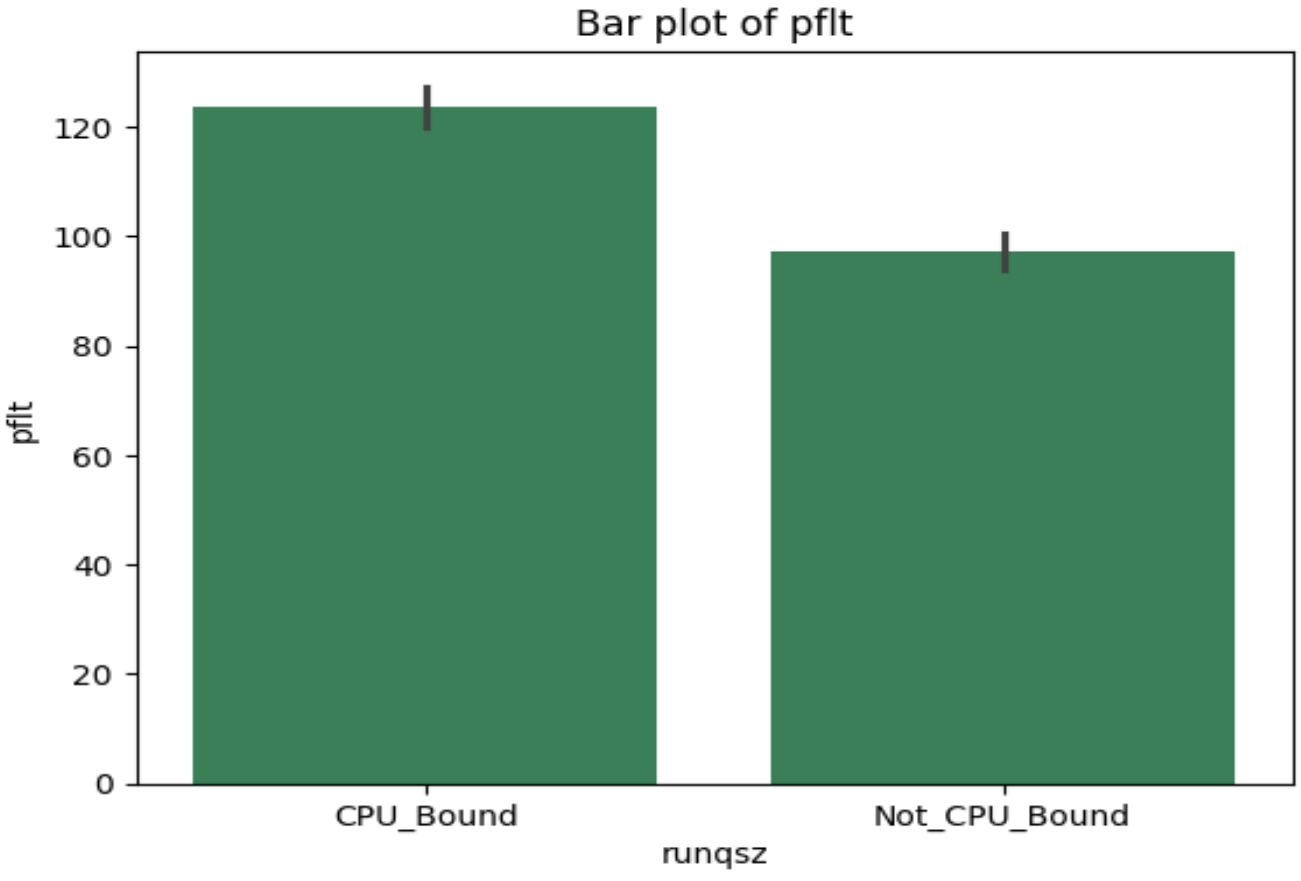

Bar plot of pgfree
pgfree
0
2
4
6
8
10
12
14
CPU_Bound
Not_CPU_Bound
runqsz

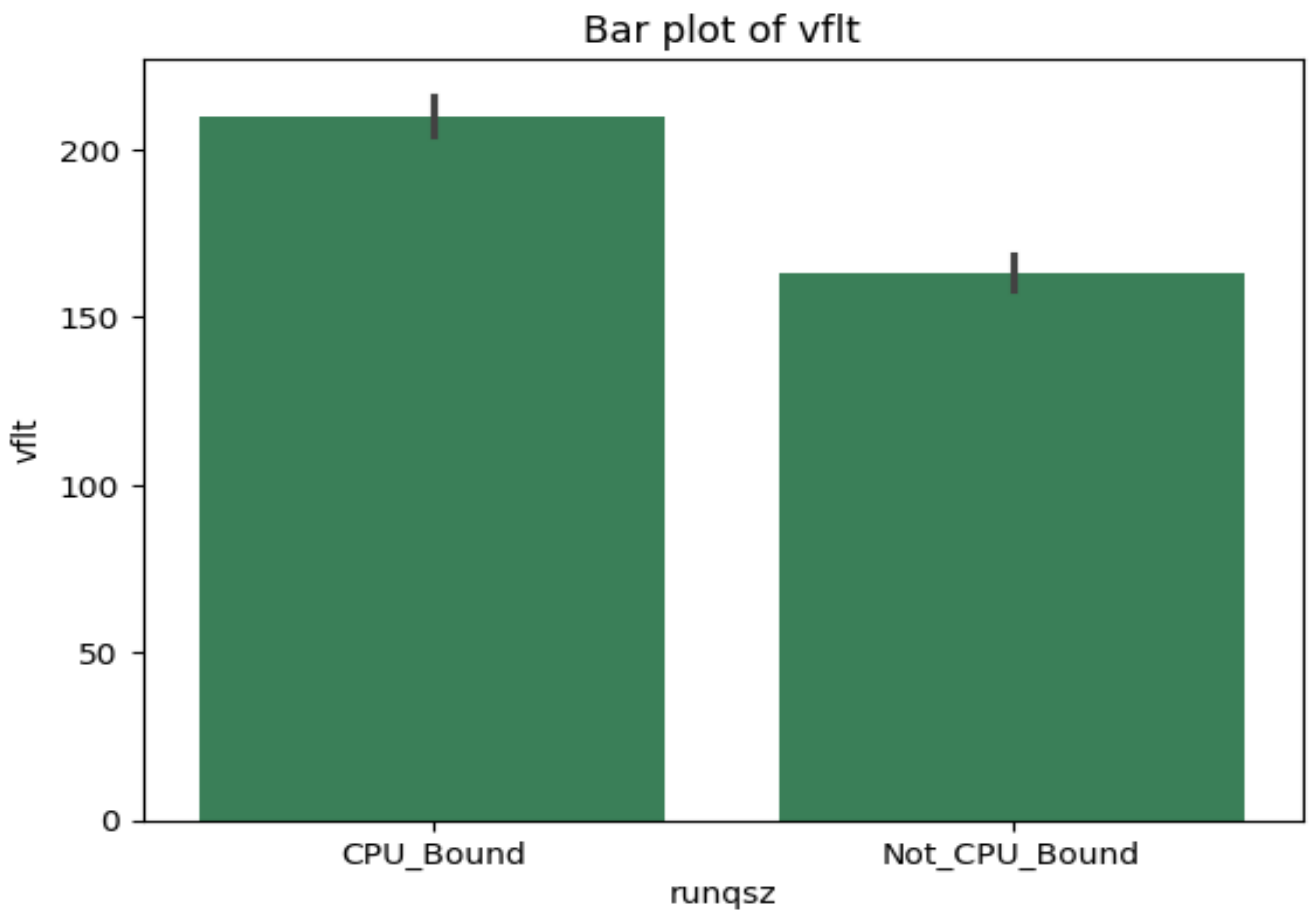

Bar plot of pgscan
pgscan
0
5
10
15
20
25
CPU_Bound
Not_CPU_Bound
runqsz

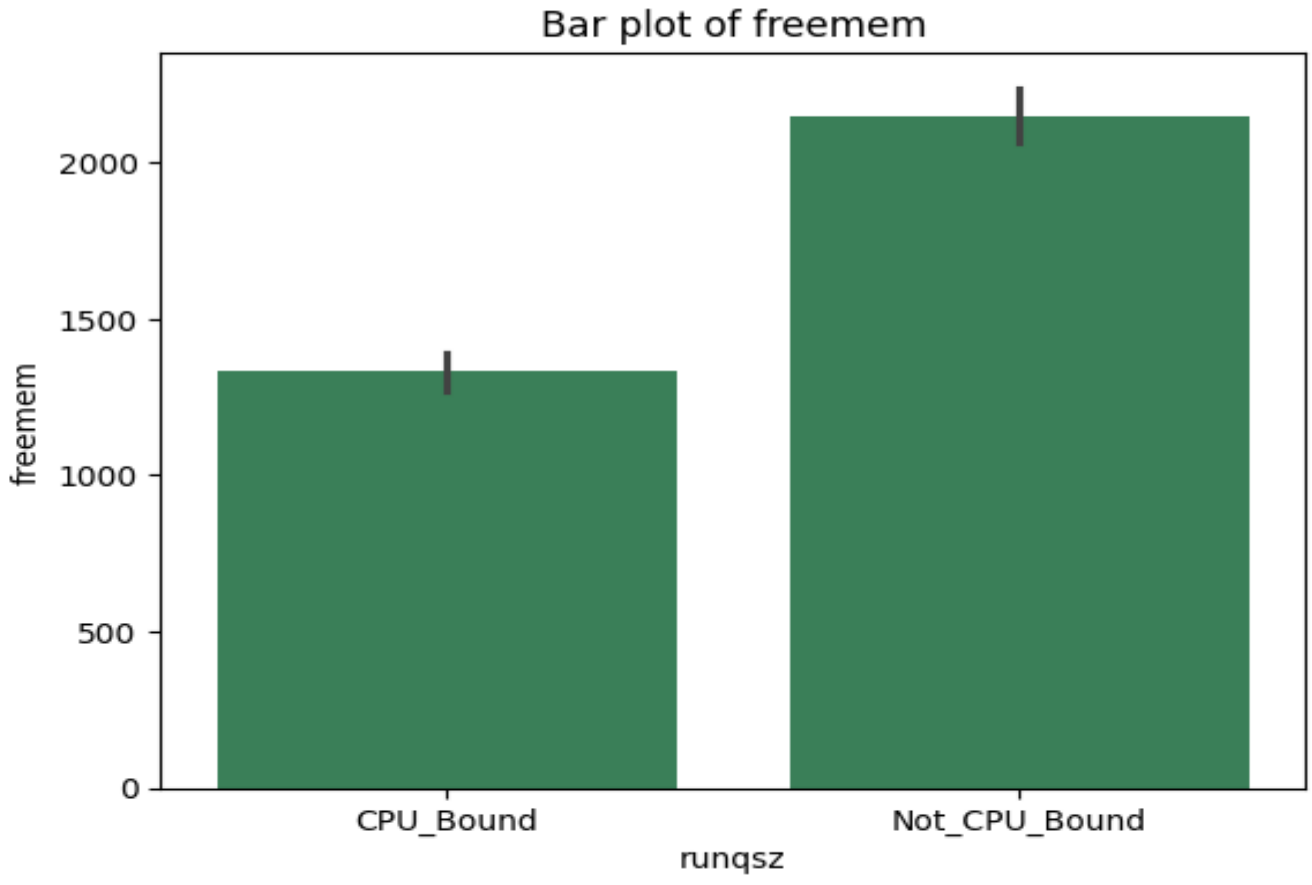

Bar plot of atch
atch
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
CPU_Bound
Not_CPU_Bound
runqsz

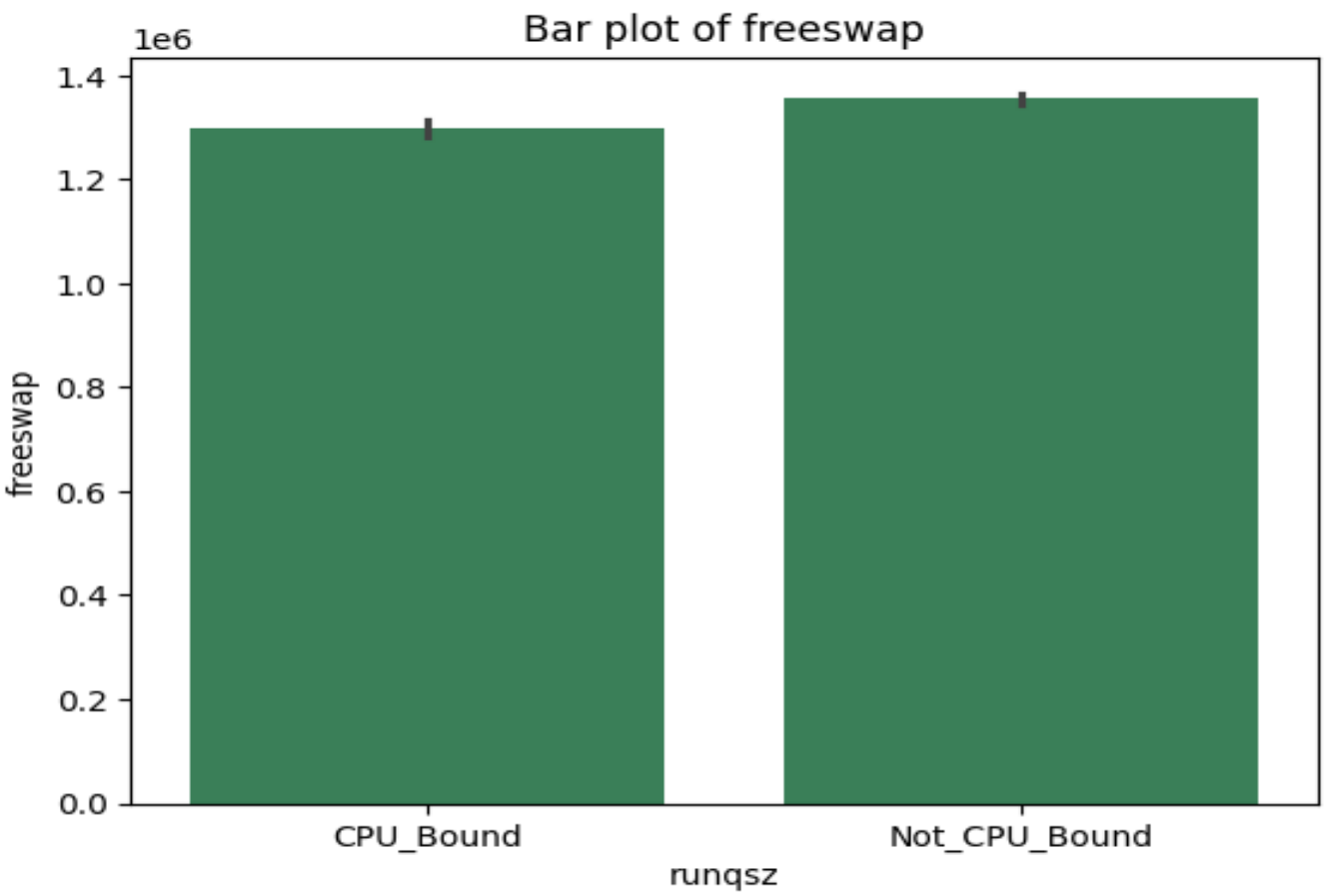

Bar plot of pgin
pgin
10
8
6
4
2
0
CPU_Bound
Not_CPU_Bound
runqsz

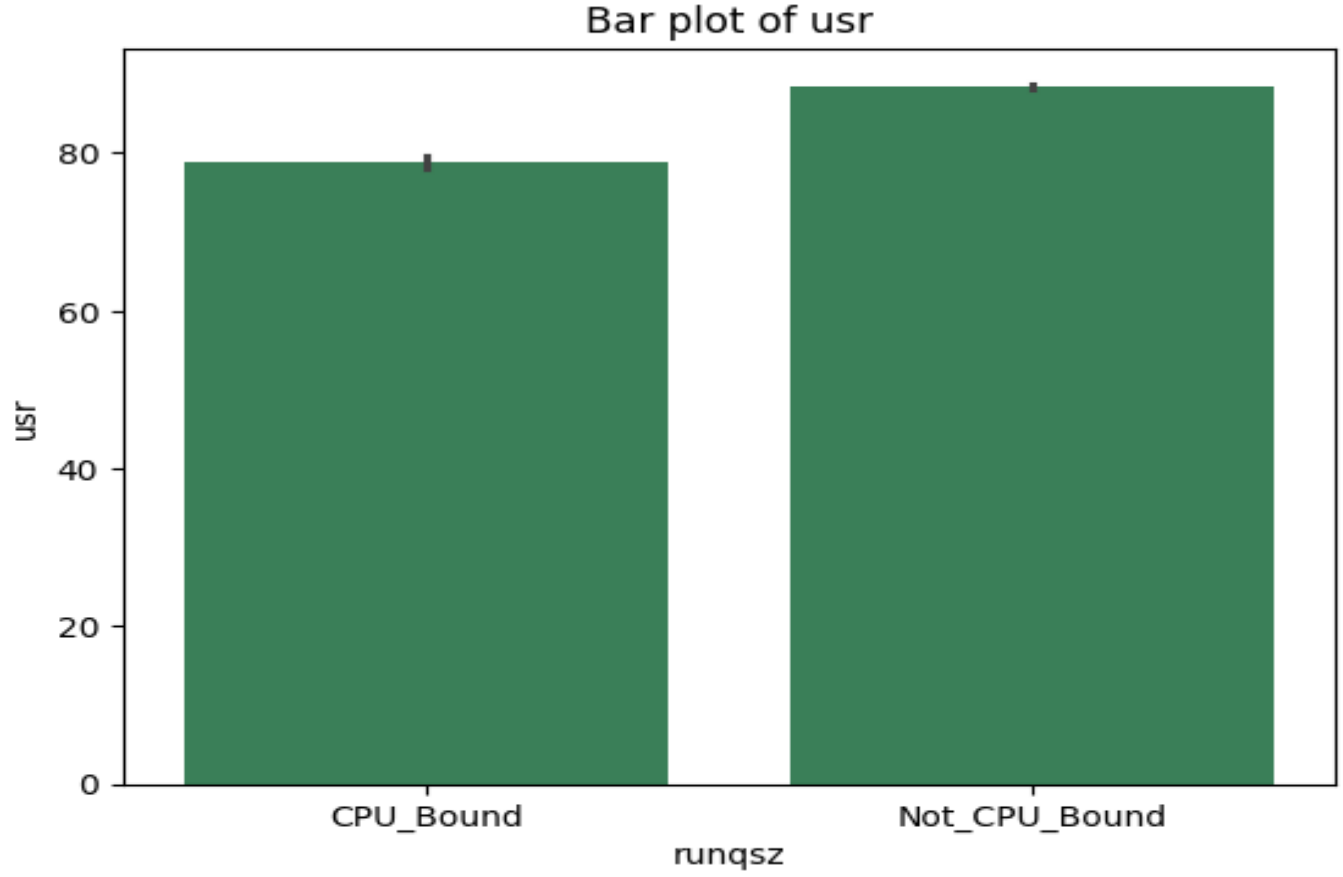

Bar plot of ppgin
ppgin
0
2
4
6
8
10
12
14
CPU_Bound
Not_CPU_Bound
runqsz

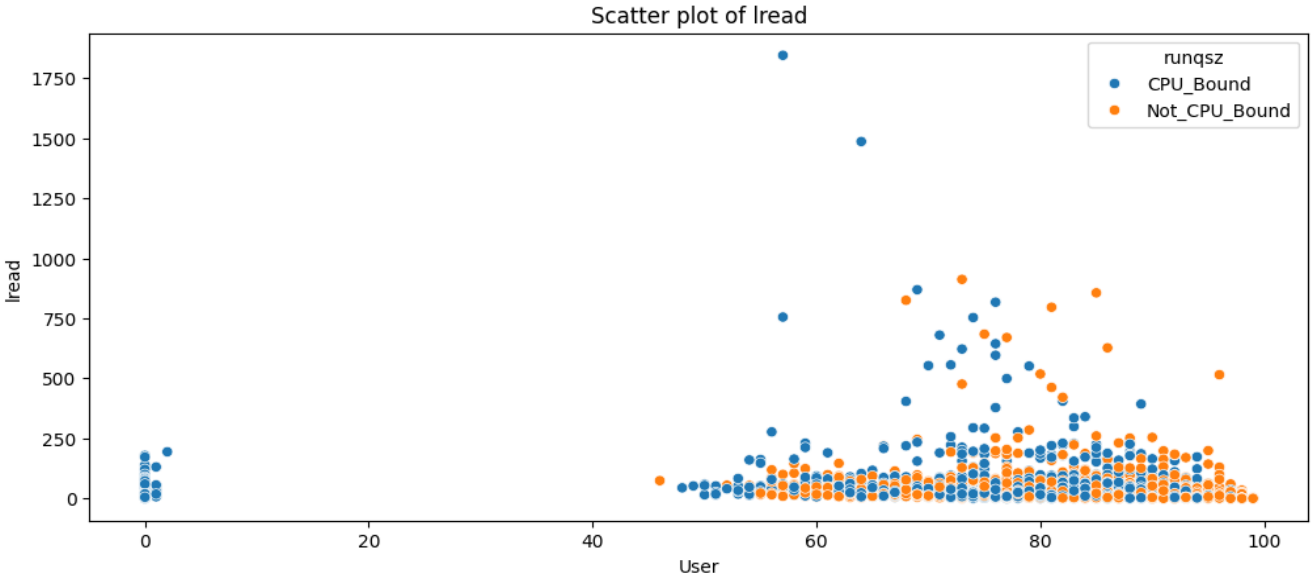

Bar plot of pflt
pflt
0
20
40
60
80
100
120
CPU_Bound
Not_CPU_Bound
runqsz

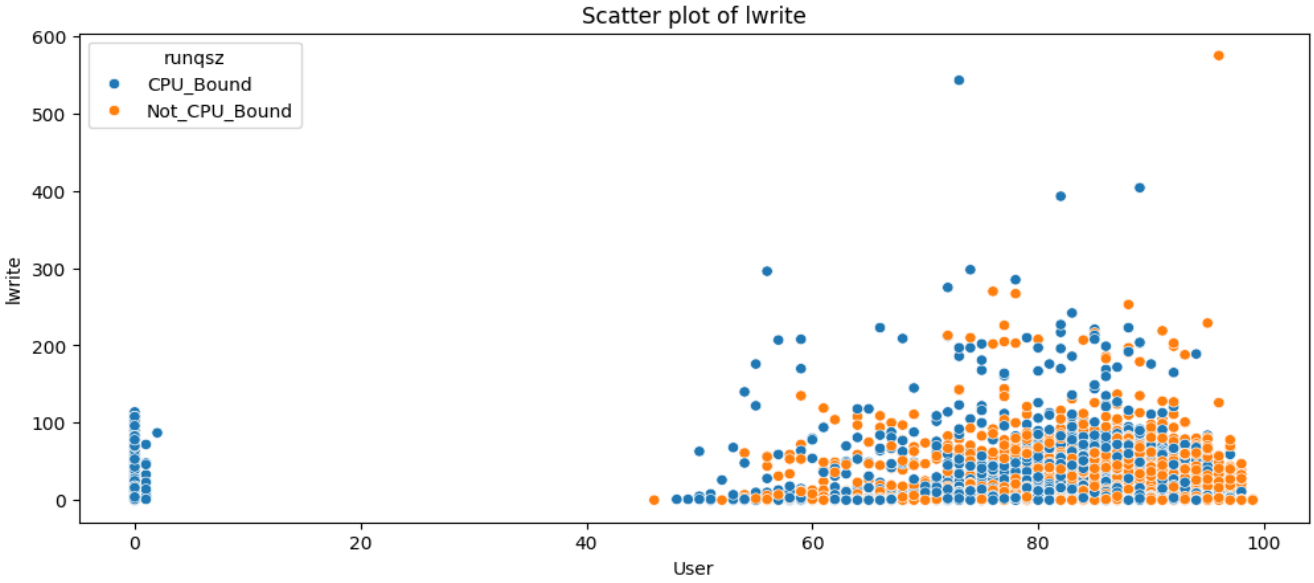

Bar plot of vflt
200
150
100
50
0
vflt
CPU_Bound
Not_CPU_Bound
runqsz

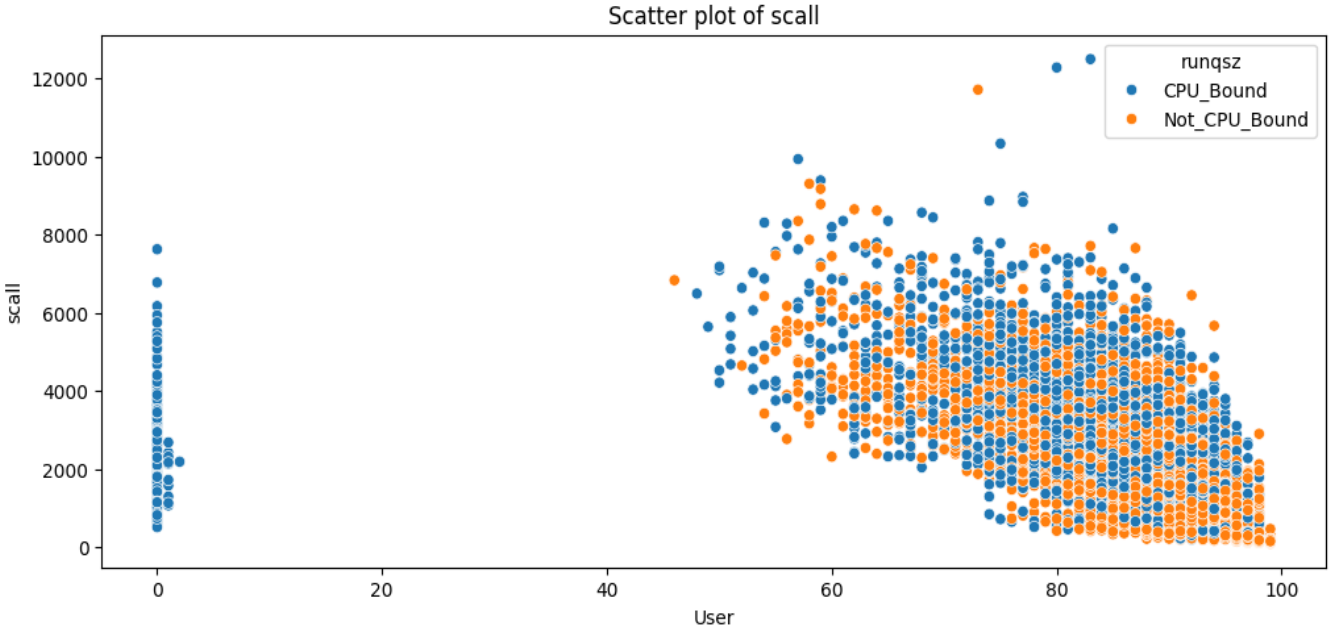

Bar plot of freemem
2000
1500
1000
500
0
freemem
CPU_Bound
Not_CPU_Bound
runqsz

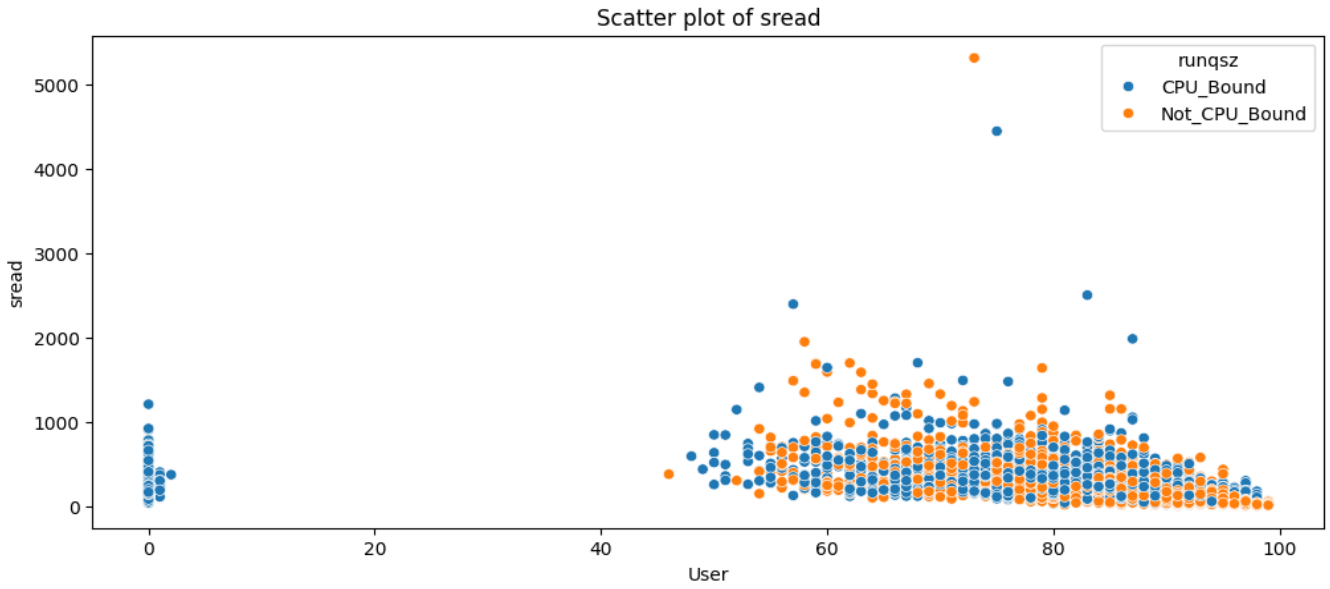

Bar plot of freeswap
1e6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
freeswap
CPU_Bound
Not_CPU_Bound
runqsz

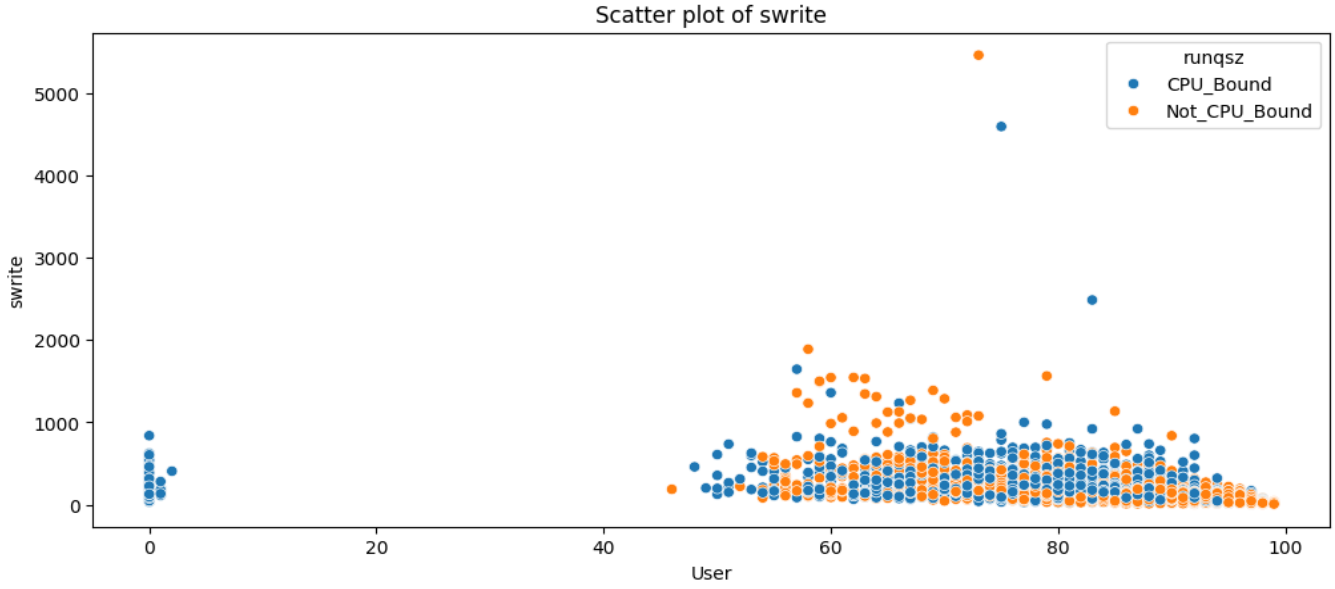

Bar plot of usr
80
60
40
20
0
usr
CPU_Bound
Not_CPU_Bound
runqsz

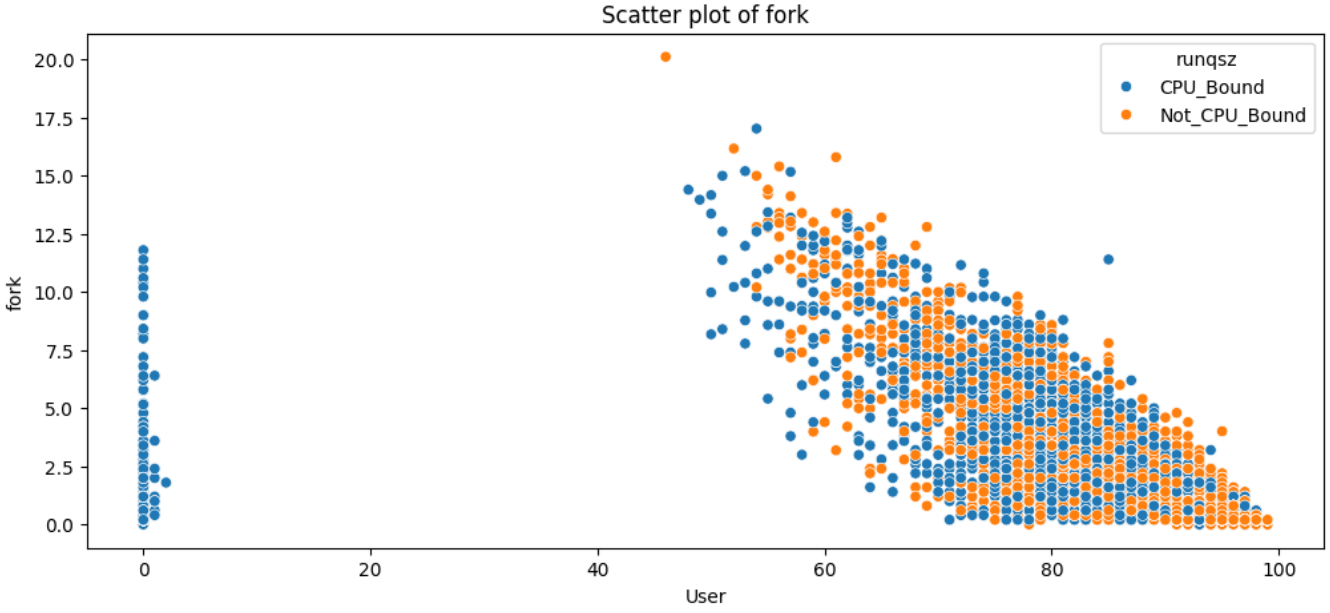

Scatter plot of lread
runqsz
CPU_Bound
Not_CPU_Bound
lread
User
0
250
500
750
1000
1250
1500
1750
0
20
40
60
80
100

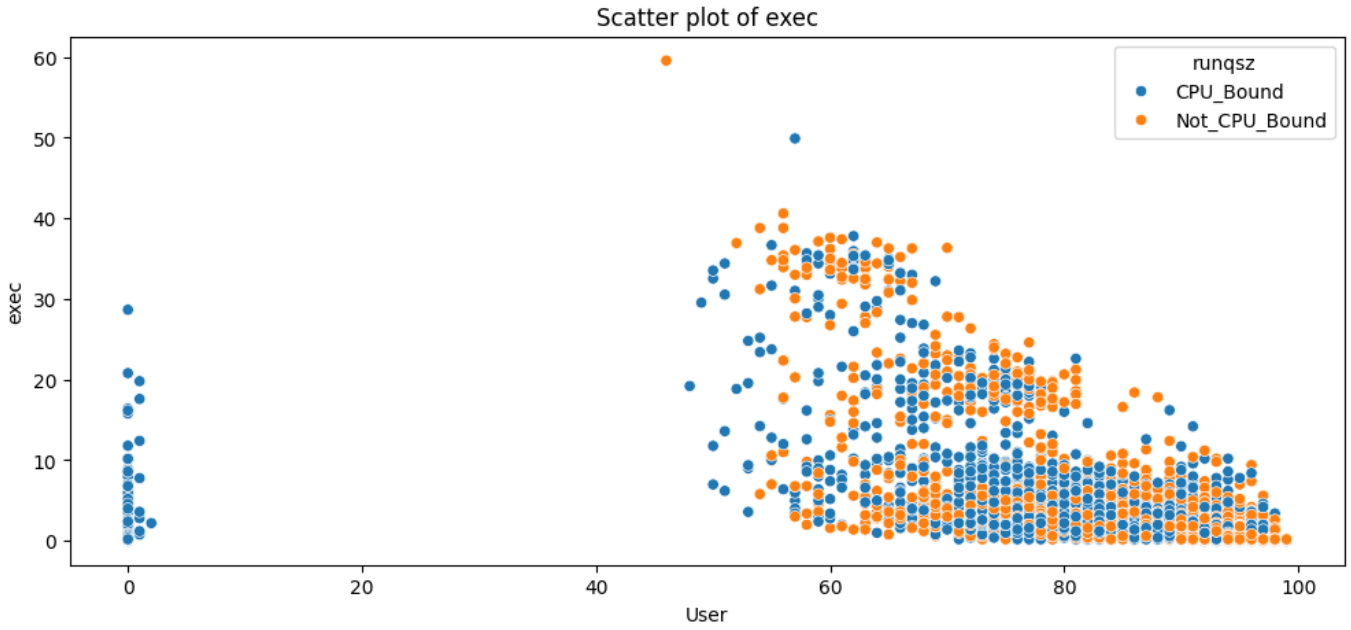

Scatter plot of lwrite
runqsz
CPU_Bound
Not_CPU_Bound
lwrite
User
0
100
200
300
400
500
600
0
20
40
60
80
100

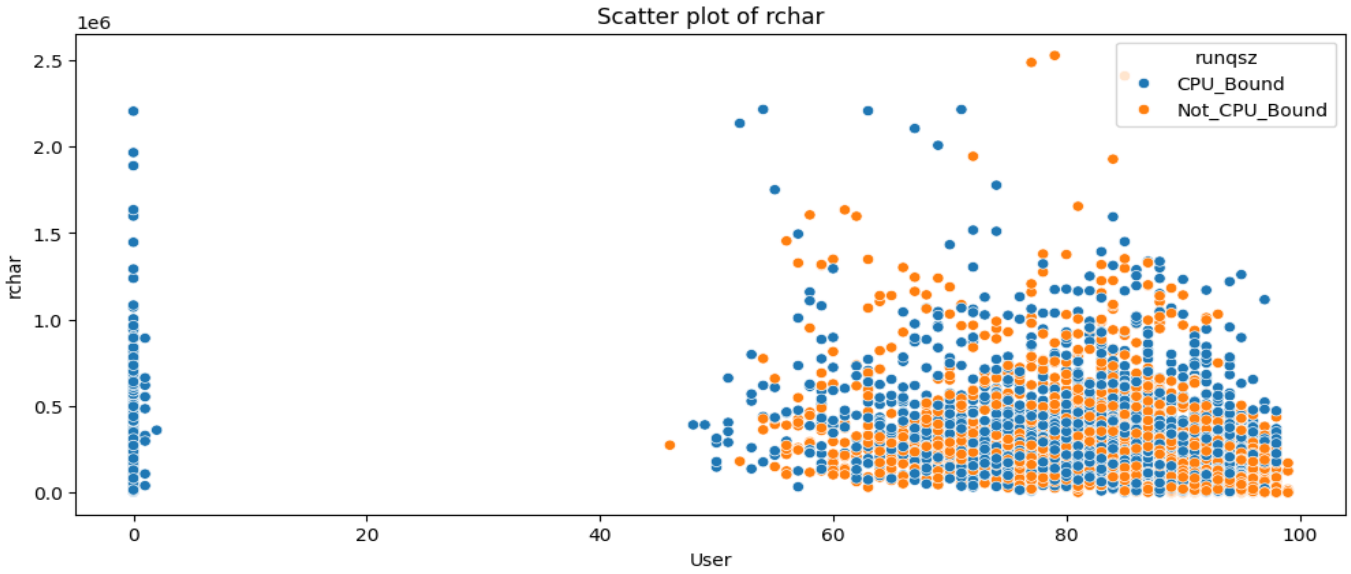

Scatter plot of scall
runqsz
CPU_Bound
Not_CPU_Bound
scall
User
0
2000
4000
6000
8000
10000
12000
0
20
40
60
80
100

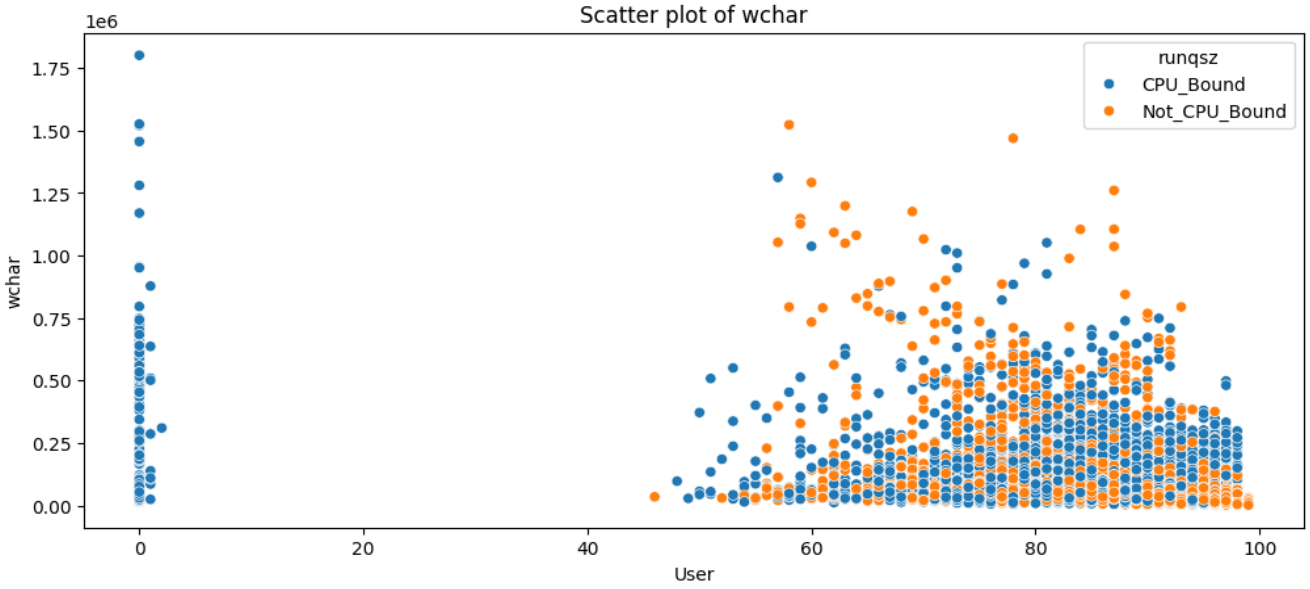

Scatter plot of sread
runqsz
CPU_Bound
Not_CPU_Bound
sread
User
0
1000
2000
3000
4000
5000
0
20
40
60
80
100

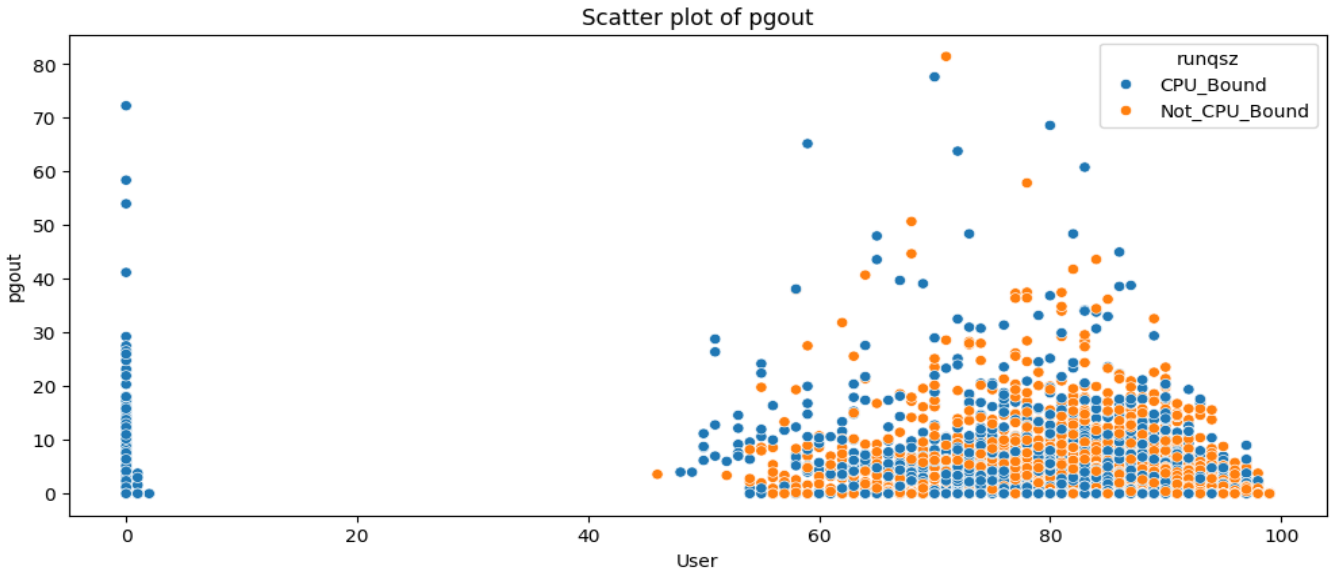

Scatter plot of swrite
runqsz
CPU_Bound
Not_CPU_Bound
swrite
User
0
1000
2000
3000
4000
5000
0
20
40
60
80
100

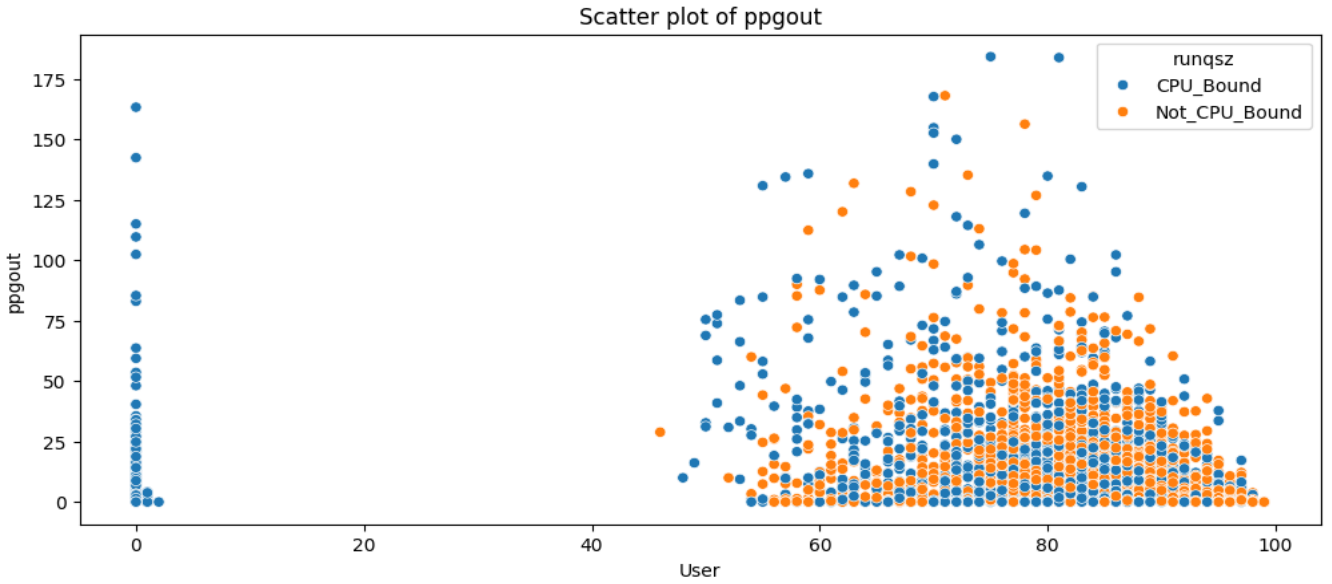

Scatter plot of fork
runqsz
CPU_Bound
Not_CPU_Bound
fork
User
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0
20
40
60
80
100

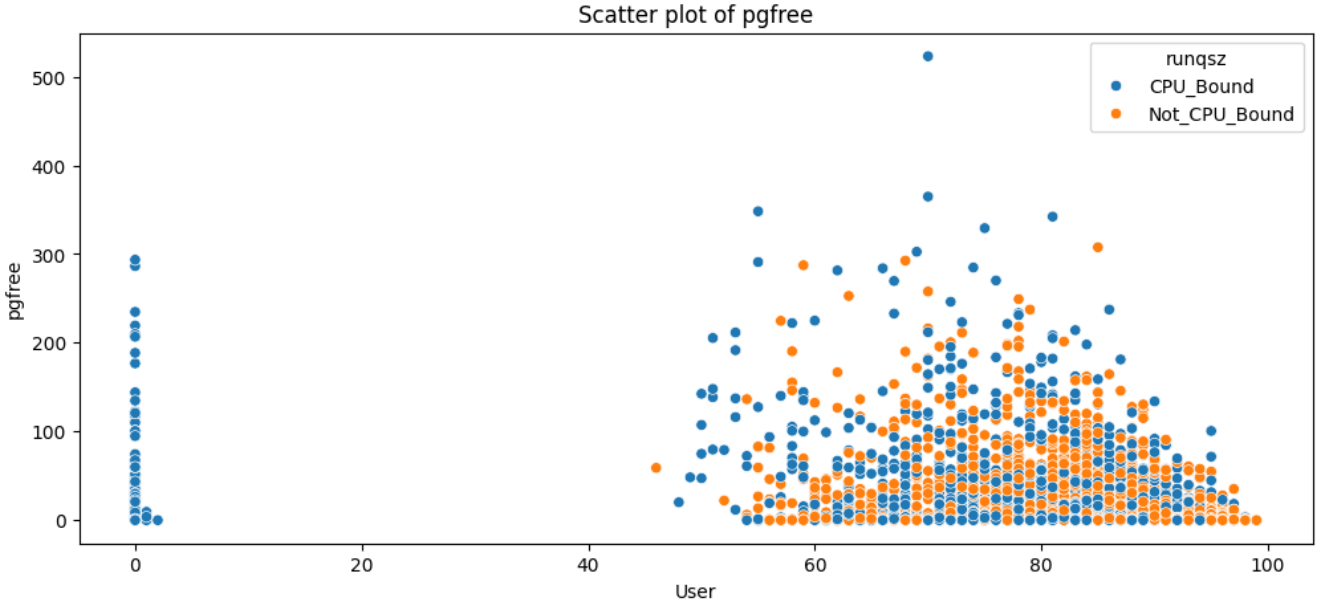

Scatter plot of exec
runqsz
CPU_Bound
Not_CPU_Bound
exec
User
0
10
20
30
40
50
60
0
20
40
60
80
100

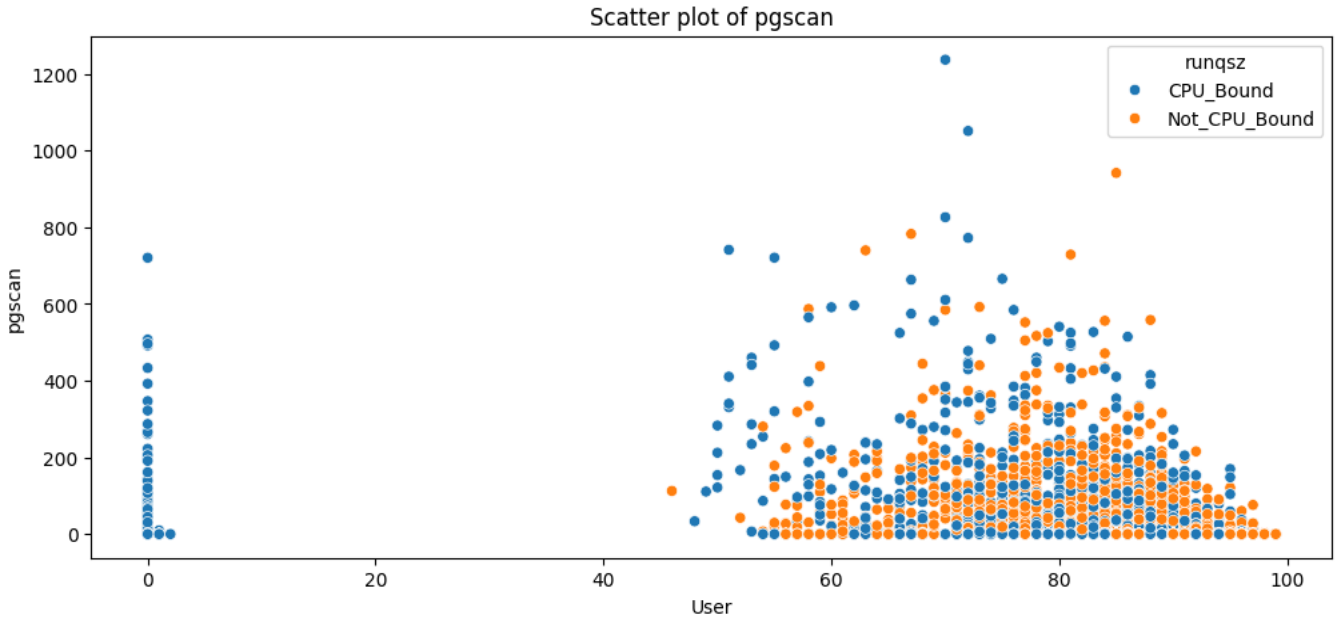

Scatter plot of rchar
1e6
runqsz
CPU_Bound
Not_CPU_Bound
rchar
User
0.0
0.5
1.0
1.5
2.0
2.5
0
20
40
60
80
100

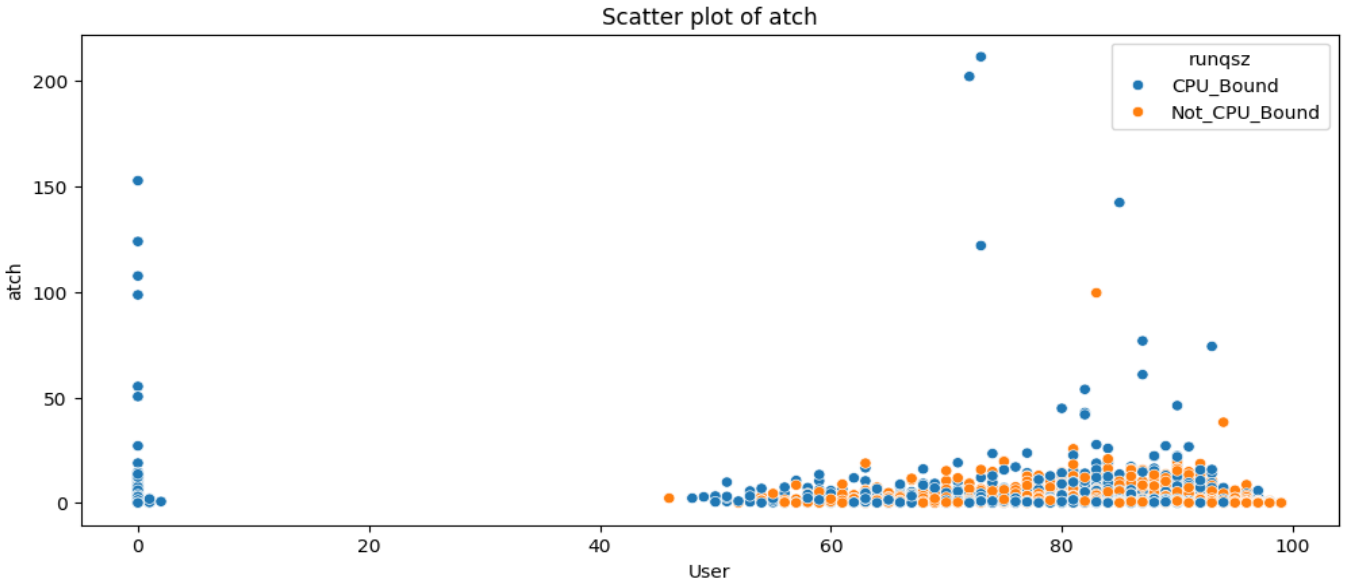

Scatter plot of wchar
1e6
runqsz
CPU_Bound
Not_CPU_Bound
wchar
User
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
0
20
40
60
80
100

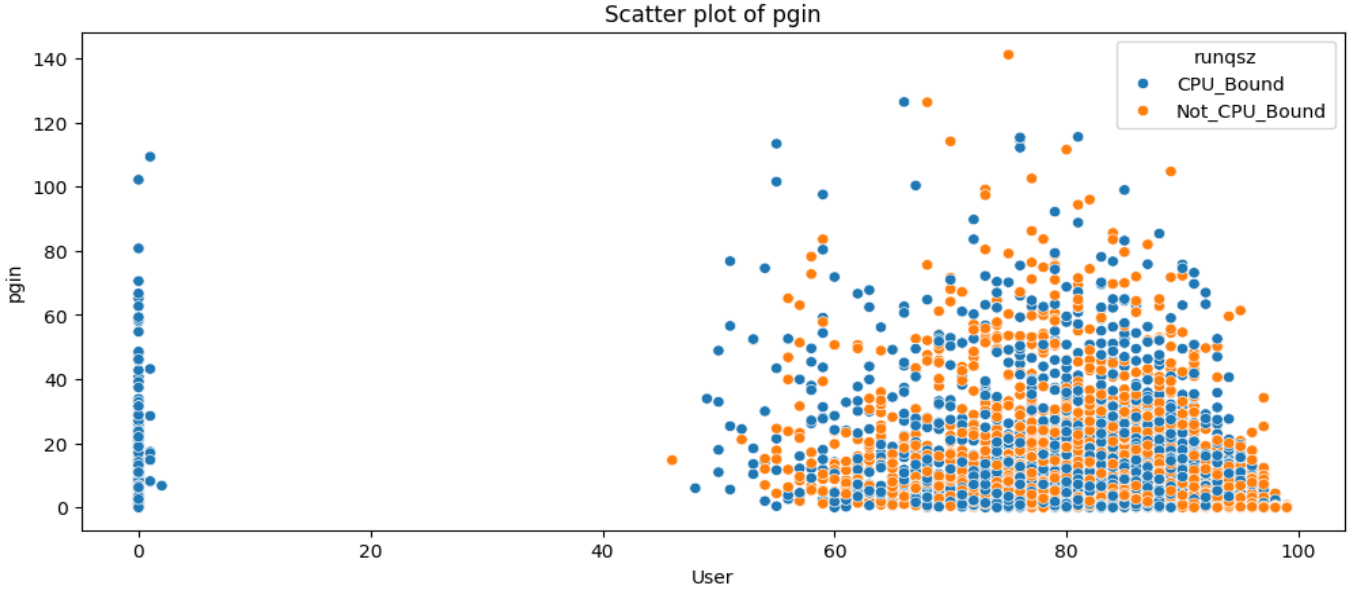

Scatter plot of pgout
runqsz
CPU_Bound
Not_CPU_Bound
pgout
User
0
10
20
30
40
50
60
70
80
0
20
40
60
80
100

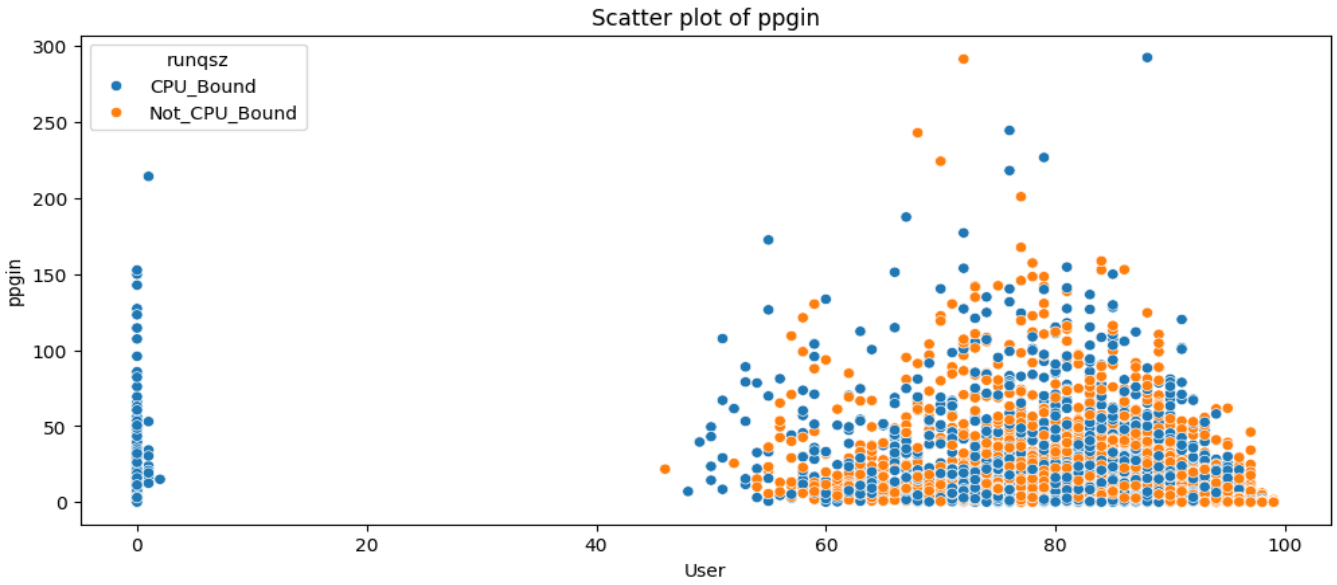

Scatter plot of ppgout
runqsz
CPU_Bound
Not_CPU_Bound
ppgout
User
0
25
50
75
100
125
150
175
0
20
40
60
80
100

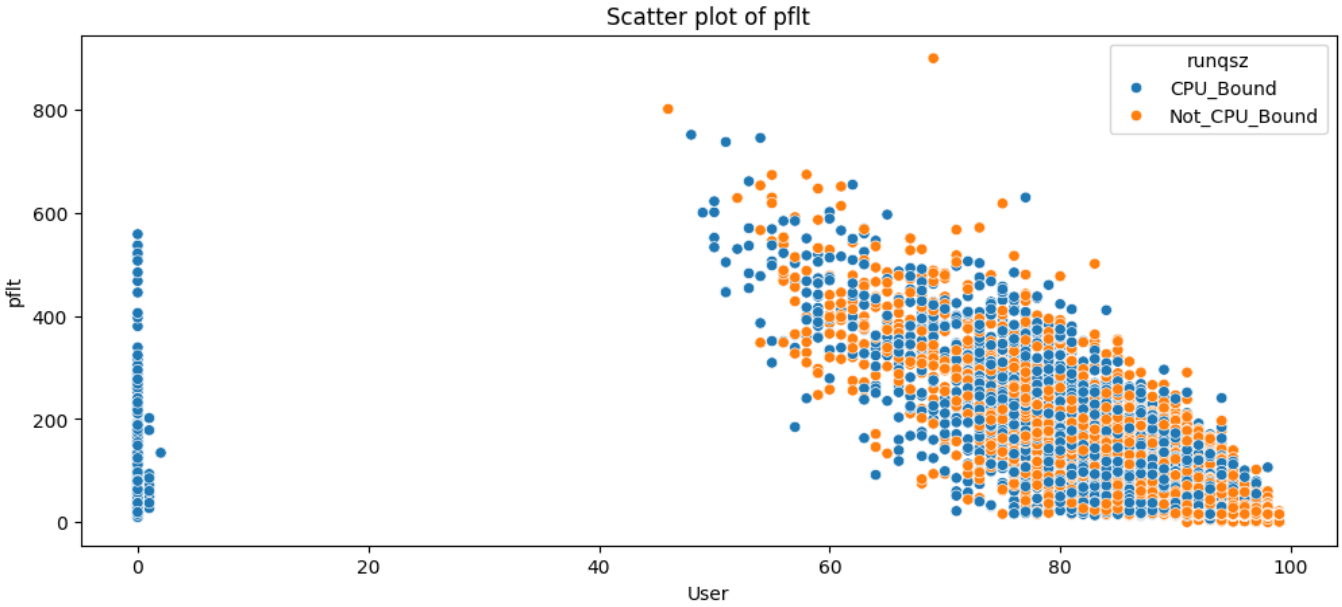

Scatter plot of pgfree
runqsz
CPU_Bound
Not_CPU_Bound
pgfree
User
0
100
200
300
400
500
0
20
40
60
80
100

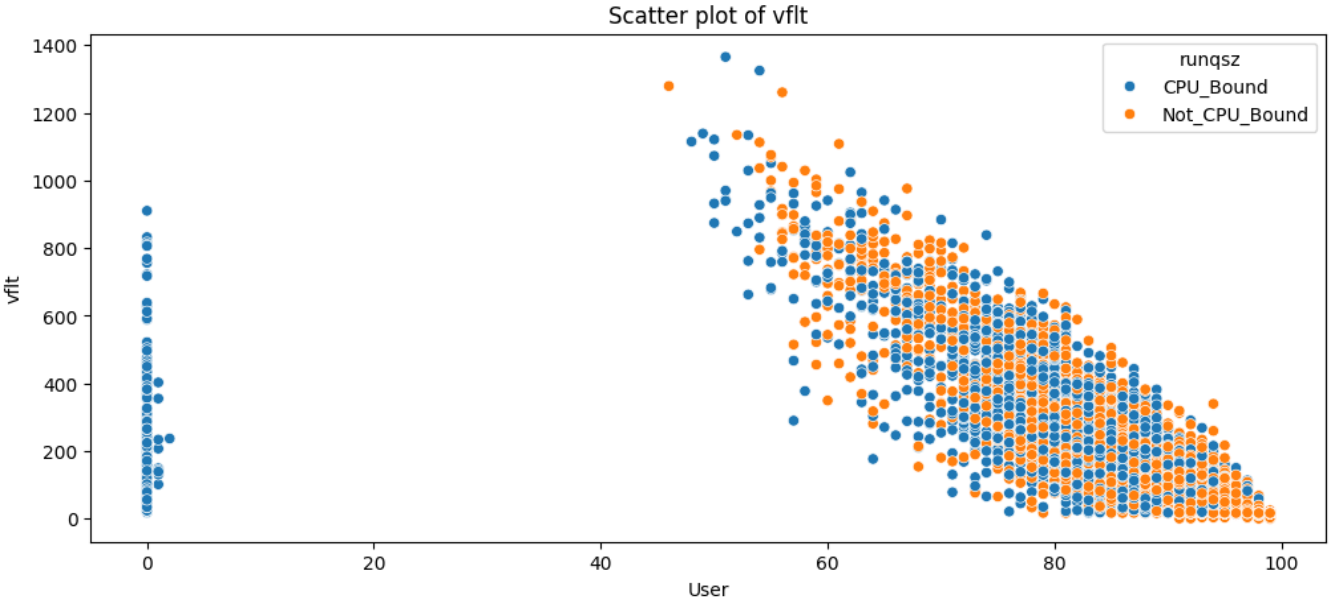

Scatter plot of pgscan
runqsz
CPU_Bound
Not_CPU_Bound
pgscan
User
0
200
400
600
800
1000
1200
0
20
40
60
80
100

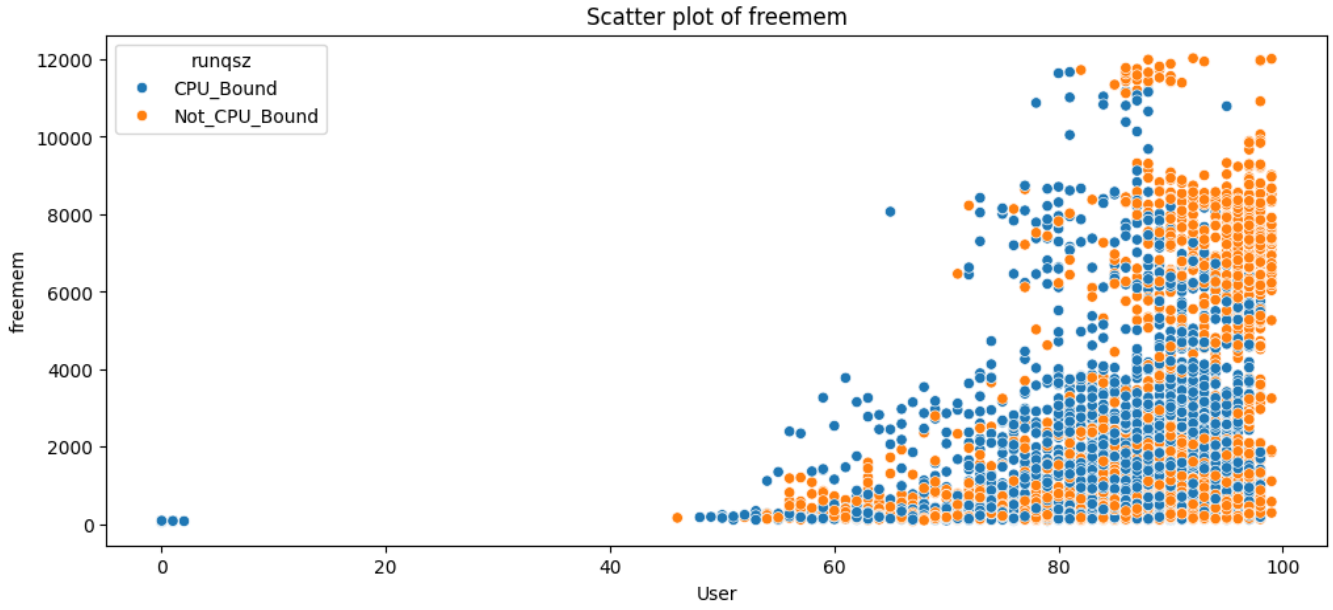

Scatter plot of atch
runqsz
CPU_Bound
Not_CPU_Bound
atch
User
0
50
100
150
200
0
20
40
60
80
100

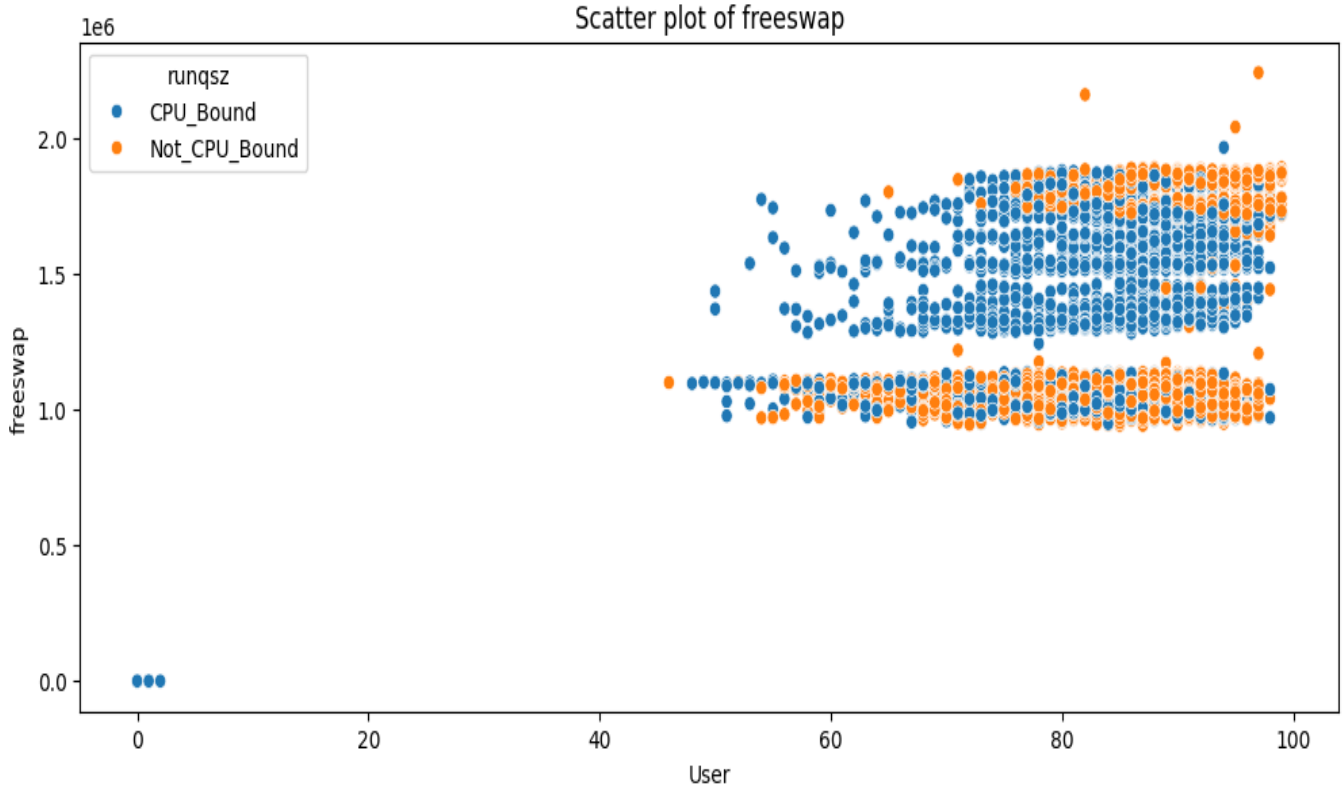

Scatter plot of pgin
runqsz
CPU_Bound
Not_CPU_Bound
pgin
User
0
20
40
60
80
100
120
140
0
20
40
60
80
100

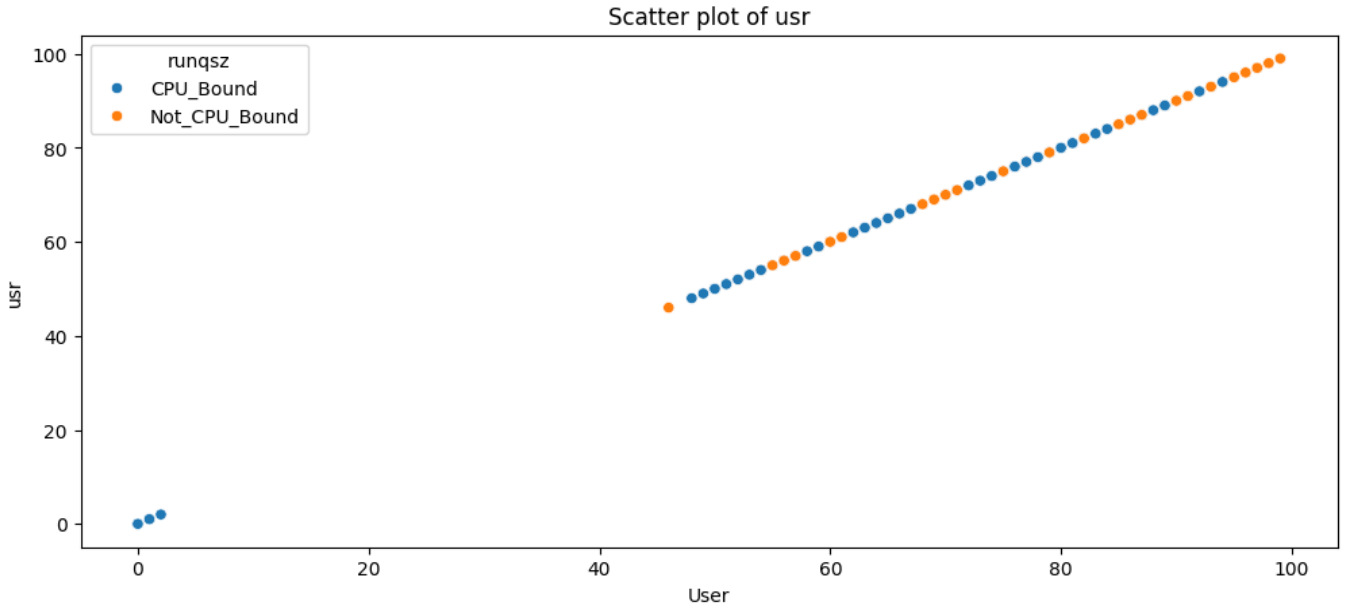

Scatter plot of ppgin
runqsz
CPU_Bound
Not_CPU_Bound
ppgin
User
0
50
100
150
200
250
300
0
20
40
60
80
100


Scatter plot of pflt
runqsz
CPU_Bound
Not_CPU_Bound
pflt
User
0
200
400
600
800
0
20
40
60
80
100

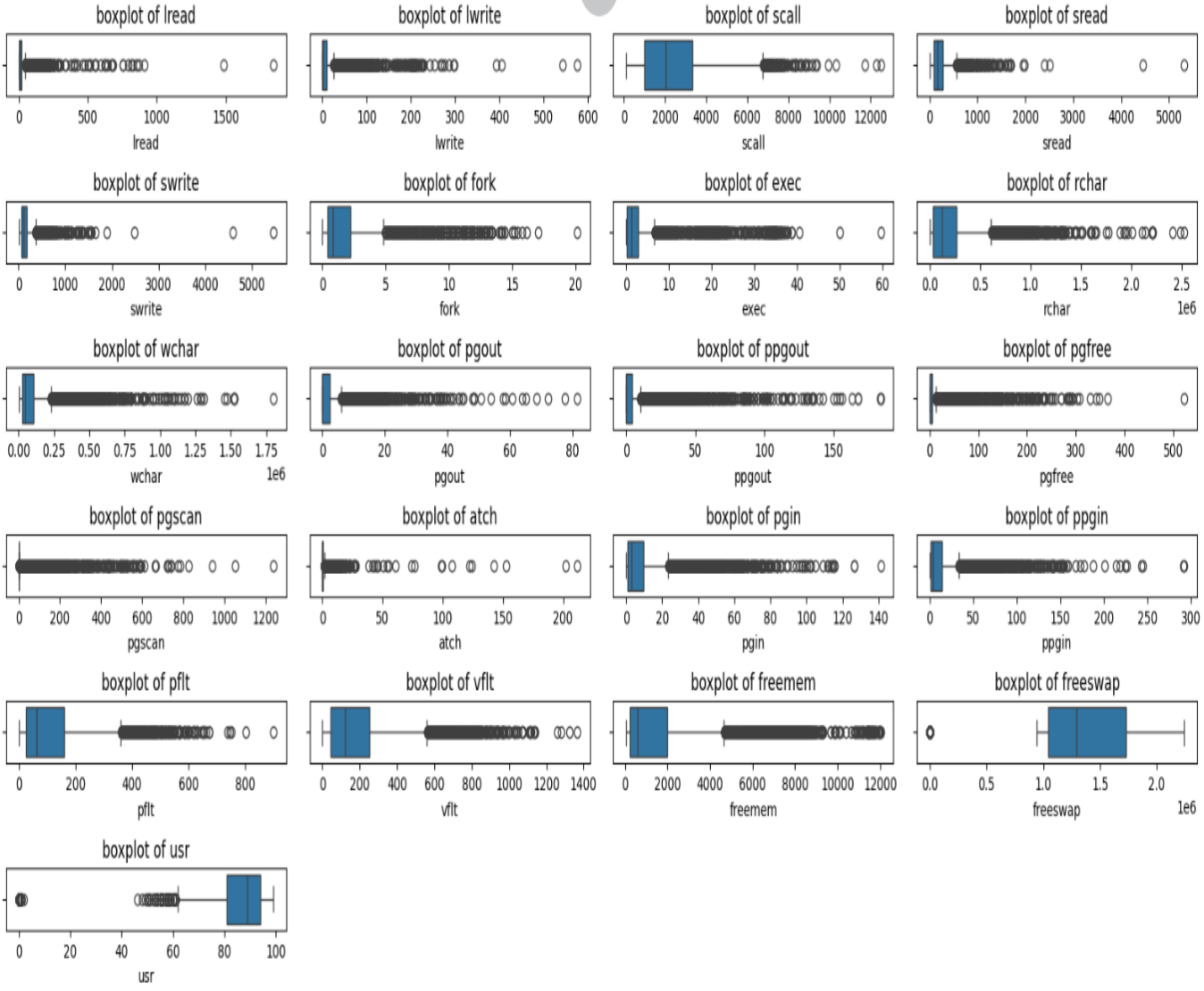

Scatter plot of vflt
runqsz
CPU_Bound
Not_CPU_Bound
vflt
User
0
200
400
600
800
1000
1200
1400
0
20
40
60
80
100


Scatter plot of freemem
runqsz
CPU_Bound
Not_CPU_Bound
freemem
User
0
2000
4000
6000
8000
10000
12000
0
20
40
60
80
100


Scatter plot of freeswap
1e6
runqsz
CPU_Bound
Not_CPU_Bound
freeswap
User
0.0
0.5
1.0
1.5
2.0
0
20
40
60
80
100

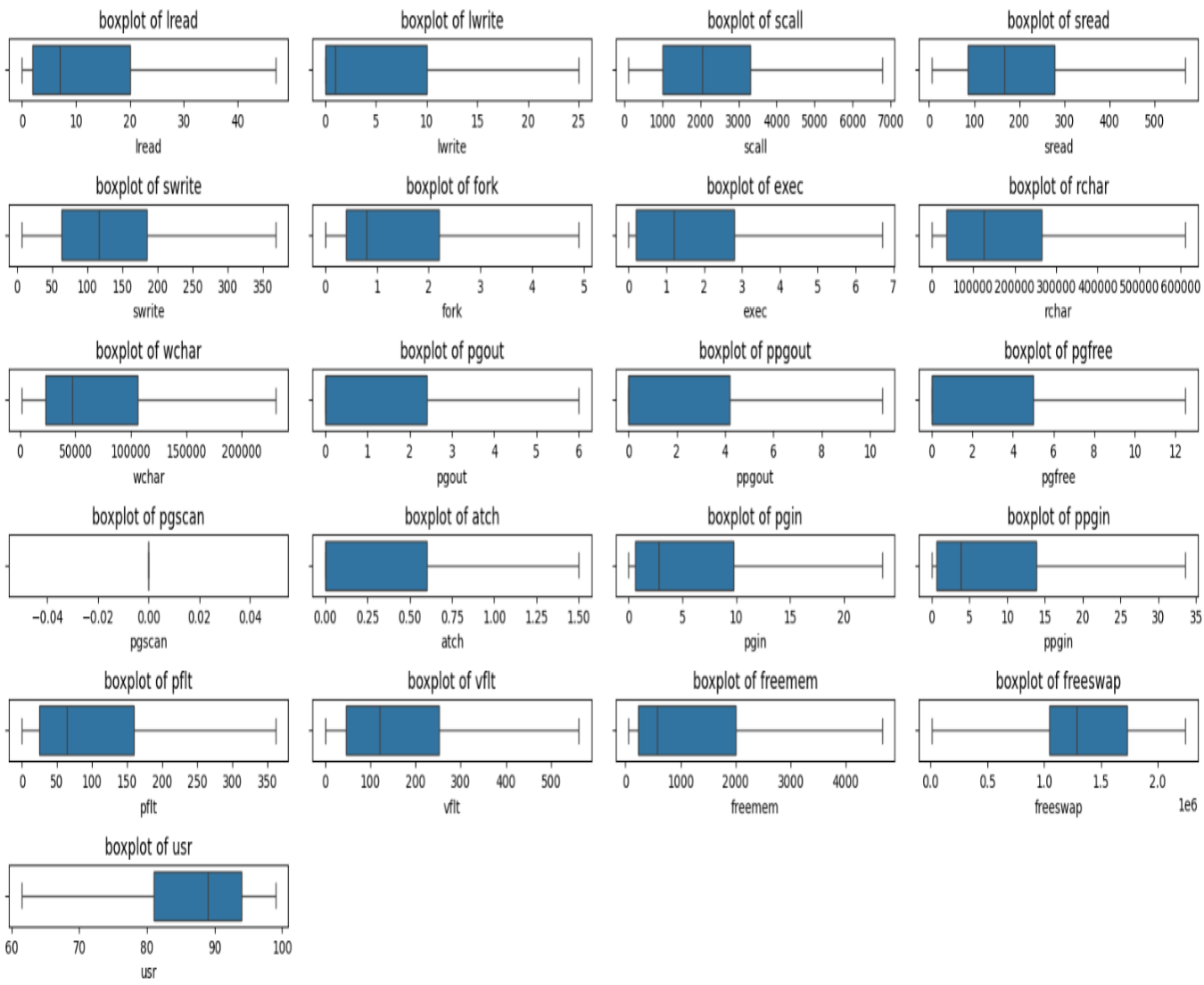


Find out Outlier Using Boxplot



Outlier Treatment: We are going to treat outliers by IQR Method (IQR - Interquartile Range).

Check the Outlier after Treatment:



ENCODING:

Linear regression model requires only numerical values, but the data set have one object variable, we can encode the object as numerical variable.

In data set there is a column ‘runqsz’ as object data type.

Now Converting the columns as numerical by using the Label encoding method and replacing the ‘Cpu_bound’ as 0 and ‘Not_cpu_bound’ as 1.

Train-Test Split:

Let us create the x and y variable data with respect to 'usr' column as the target variable. Now x having every data except the target variable and y having only the target variable.

Using stats model api as SM to intercept the X variable.

Using sklearn to split the data into x_tain and y_train.

Split X and y into training and test set in 70:30 ratio.

Now x_train data having the follows,

	lread	lwrite	scall	sread	swrite	fork	exec	rchar	wchar	pgout	...	pgfree	pgscan	atch	pgin	ppgin	pflt	vflt	freemem	freeswap	runqsz_Not_CPU_Bound
0	1	0	2147	79	68	0.2	0.2	40671.0	53995.0	0.0	...	0.0	0.0	0.0	1.6	2.6	16.00	26.40	4670	1730946	0
1	0	0	170	18	21	0.2	0.2	448.0	8385.0	0.0	...	0.0	0.0	0.0	0.0	0.0	15.63	16.83	7278	1869002	1
2	15	3	2162	159	119	2.0	2.4	125473.5	31950.0	0.0	...	0.0	0.0	1.2	6.0	9.4	150.20	220.20	702	1021237	1
3	0	0	160	12	16	0.2	0.2	125473.5	8670.0	0.0	...	0.0	0.0	0.0	0.2	0.2	15.60	16.80	7248	1863704	1
4	5	1	330	39	38	0.4	0.4	125473.5	12185.0	0.0	...	0.0	0.0	0.0	1.0	1.2	37.80	47.60	633	1760253	1

5 rows × 21 columns

Now y_train data having the follows:

	usr
694	91
5535	94
4244	0
2472	83
7052	94

```
Count of Rows & Columns of the Training Set for the Independent Variables: (5734, 21)
Count of Rows & Columns of the Training Set for the Dependent Variable: (5734, 1)
Count of Rows & Columns of the Test Set for the Independent Variables: (2458, 21)
Count of Rows & Columns of the Test Set for the Dependent Variable: (2458, 1)
```

After Train Test Split there are 5734 rows and 21 columns for independent varables (X_train).

5734 Rows and 1 Column for dependent variables i.e for Y_train.

For X_test variable (independent variable) there are 2458 Rows and 21 Columns.

For Y_test variable contains 2458 Rows and 1 Column present.

Linear Regression using statsmodel(OLS):

==========

OLS Regression Results

==========

Dep. Variable: usr R-squared: 0.643

Model: OLS Adj. R-squared: 0.642

Method: Least Squares F-statistic: 489.6

Date: Sun, 19 May 2024 Prob (F-statistic): 0.00

Time: 12:28:31 Log-Likelihood: -21788.

No. Observations: 5734 AIC: 4.362e+04

Df Residuals: 5712 BIC: 4.377e+04

Df Model: 21

Covariance Type: nonrobust

==========

==========

	coef	std err	t	P>|t|	[0.025	0.975]
const	44.6380	0.746	59.831	0.000	43.175	46.101
lread	-0.0199	0.003	-6.214	0.000	-0.026	-0.014
lwrite	0.0048	0.006	0.795	0.427	-0.007	0.017
scall	0.0010	0.000	7.451	0.000	0.001	0.001
sread	-0.0005	0.002	-0.257	0.797	-0.004	0.003

swrite	-0.0020	0.002	-1.018	0.309	-0.006	0.002
fork	-1.7222	0.244	-7.052	0.000	-2.201	-1.244
exec	-0.0896	0.048	-1.879	0.060	-0.183	0.004
rchar	-4.062e-06	8.29e-07	-4.898	0.000	-5.69e-06	-2.44e-06
wchar	-1.164e-05	1.28e-06	-9.118	0.000	-1.41e-05	-9.14e-06
pgout	-0.1739	0.064	-2.717	0.007	-0.299	-0.048
ppgout	0.0989	0.037	2.701	0.007	0.027	0.171
pgfree	-0.0703	0.020	-3.508	0.000	-0.110	-0.031
pgscan	0.0086	0.006	1.362	0.173	-0.004	0.021
atch	-0.0786	0.027	-2.949	0.003	-0.131	-0.026
pgin	0.0913	0.029	3.103	0.002	0.034	0.149
ppgin	-0.0594	0.019	-3.128	0.002	-0.097	-0.022
pflt	-0.0415	0.004	-9.697	0.000	-0.050	-0.033
vflt	0.0223	0.003	6.665	0.000	0.016	0.029
freemem	-0.0016	7.53e-05	-21.489	0.000	-0.002	-0.001
freeswap	3.219e-05	4.54e-07	70.985	0.000	3.13e-05	3.31e-05
runqsz_Not_CPU_Bound	7.7908	0.303	25.693	0.000	7.196	8.385

==========

Omnibus:	1507.319	Durbin-Watson:	2.057
Prob(Omnibus):	0.000	Jarque-Bera (JB):	4768.238
Skew:	-1.333	Prob(JB):	0.00
Kurtosis:	6.585	Cond. No.	7.48e+06

==========

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 7.48e+06. This might indicate that there are

strong multicollinearity or other numerical problems.

```
VIF values:

const                        27.191591
lread                         1.472618
lwrite                        1.405898
scall                         2.414301
sread                         6.836403
swrite                        5.320692
fork                         18.210503
exec                          3.059950
rchar                         1.974726
wchar                         1.553348
pgout                         5.776005
ppgout                       15.906900
pgfree                       20.437584
pgscan                        9.237017
atch                          1.087328
pgin                          8.075699
ppgin                         8.672927
pflt                         11.834374
vflt                         20.233207
freemem                       1.677241
freeswap                      1.761193
runqsz_Not_CPU_Bound          1.118922
dtype: float64
```

The R-square value tells that the model can explain. 64.3% of the variance in the training set.

Adjusted R-square also nearly to the R-square, 64.2%. Let's build another model.

And from the VIF factors we see that many independent variables have high VIF values present. So, we remove variables 1 by 1 to make sure that it doesn't affect the model.

At first, we remove 'lwrite', 'sread', 'swrite', 'exec', 'ppgout', 'pgfree', 'pgscan'

variables from the dataset.

After removing 'vflt' variable looks there are still variables to be removed in order to treat multicollinearity.

And if we look in R-squared and adj.R-squared values doesn't changed a bit.

OLS Regression Results

Dep. Variable:	usr	R-squared:	0.641
Model:	OLS	Adj. R-squared:	0.641
Method:	Least Squares	F-statistic:	730.8
Date:	Sun, 19 May 2024	Prob (F-statistic):	0.00
Time:	12:28:37	Log-Likelihood:	-21799.
No. Observations:	5734	AIC:	4.363e+04
Df Residuals:	5719	BIC:	4.373e+04
Df Model:	14		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	44.7128	0.731	61.129	0.000	43.279	46.147
lread	-0.0177	0.003	-6.496	0.000	-0.023	-0.012
scall	0.0009	0.000	7.894	0.000	0.001	0.001
fork	-1.8746	0.201	-9.323	0.000	-2.269	-1.480
rchar	-4.125e-06	7.64e-07	-5.398	0.000	-5.62e-06	-2.63e-06
wchar	-1.2e-05	1.23e-06	-9.781	0.000	-1.44e-05	-9.59e-06
pgout	-0.1517	0.031	-4.918	0.000	-0.212	-0.091
atch	-0.0752	0.026	-2.839	0.005	-0.127	-0.023
pgin	0.0889	0.029	3.078	0.002	0.032	0.145
ppgin	-0.0672	0.018	-3.747	0.000	-0.102	-0.032
pflt	-0.0393	0.004	-9.428	0.000	-0.047	-0.031
vflt	0.0204	0.003	6.359	0.000	0.014	0.027
freemem	-0.0016	7.51e-05	-21.211	0.000	-0.002	-0.001
freeswap	3.215e-05	4.48e-07	71.686	0.000	3.13e-05	3.3e-05
runqsz_Not_CPU_Bound	7.7429	0.303	25.553	0.000	7.149	8.337

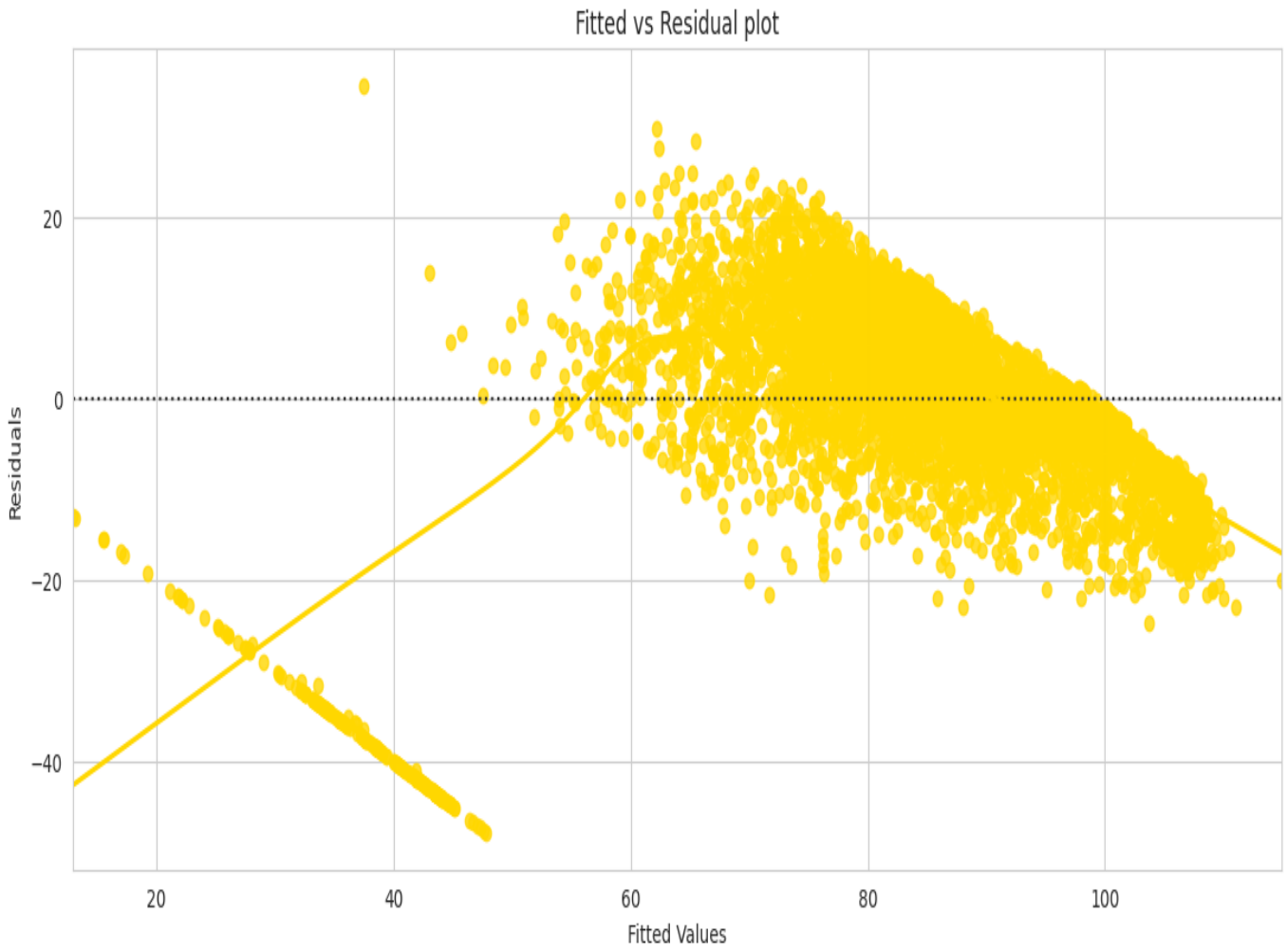
Omnibus:	1504.712	Durbin-Watson:	2.055
Prob(Omnibus):	0.000	Jarque-Bera (JB):	4761.600
Skew:	-1.331	Prob(JB):	0.00
Kurtosis:	6.584	Cond. No.	7.32e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.32e+06. This might indicate that there are strong multicollinearity or other numerical problems.

Linearity and Independence of predictors:

	Actual Values	Fitted Values	Residuals
0	91	84.074798	6.925202
1	94	83.563682	10.436318
2	0	42.752314	-42.752314
3	83	72.464990	10.535010
4	94	102.999957	-8.999957

The above chart shows that actual values and their fitted values and residuals present.



On plotting the graph for fitted values and their residuals we see that it is not linear to be honest.

Test for Homoscedasticity:

```
0.001280969596375458
```

Since p-value < 0.05 we can say that the residuals are heterocedastic.

Linear modelling using Sklearn:

The coefficients of the last generated variables are –

```
The coefficient for const is 0.0
The coefficient for lread is -0.017733425704424426
The coefficient for lwrite is 0.0009079606369232773
The coefficient for scall is -1.874638615005988
The coefficient for fork is -4.124879311433795e-06
The coefficient for rchar is -1.1995767441437544e-05
The coefficient for wchar is -0.1517129303659833
The coefficient for pgout is -0.07518760225280417
The coefficient for pgin is 0.08888485548896106
The coefficient for ppgin is -0.06715978466055525
The coefficient for pflt is -0.039296445244128 42
The coefficient for vflt is 0.020436028186492955
The coefficient for freemem is -0.0015925647476032238
The coefficient for freeswap is 3.214560026027066e-05
The coefficient for runqsz_Not_CPU_Bound is 7.742925313121714
```

R-Squared:

The R-Squared value for the derived variables for model evaluation comes to the value of around 0.641 from 0.643.

Adj.R-Squared:

The R-Squared value for the derived variables for model evaluation comes to the value of around 0.641 from 0.642.

RMSE – Root Mean Squared Error value:

10.834253985833387

RMSE on Testing data:

11.626325960446641

Business Insights & Recommendations

- Comment on the Linear Regression equation from the final model and impact of relevant variables (atleast 2) as per the equation

usr = 44.71280566622133 + -0.017733425704110392 * (lread) + 0.0090079606369170691 * (scall) + -1.8746386150004923 * (fork) + -4.124879311750328e-06 * (rchar) + -1.1995767440711308e-05 * (wchar) + -0.15171293036603106 * (pgout) + -0.07518760225284281 * (atch) + 0.08888485548893942 * (pgin) + -0.06715978466055815 * (ppgin) + -0.039296445244142185 * (pflt) + 0.020436028186510784 * (vflt) + -0.0015925647476030444 * (freemem) + 3.2145600260152854e-05 * (freeswap) + 7.742925313121747 * (runqsz_Not_CPU_Bound)

Observations:

The intercept term is 44.71280566622133. This represents the baseline value of "usr" when all other predictor variables are zero.

One-unit increase in "lread" is associated with a decrease of approximately 0.0177 units in "usr."

A one-unit increase in "scall" is associated with an increase of approximately 0.0009 units in "usr."

A one-unit increase in "pgin" is associated with an increase of approximately 0.08888 units in "usr."

A one-unit increase in "pgin " is associated with an increase of approximately 0.0888 units in "usr.

A one-unit increase in "vflt" is associated with an increase of approximately 0.02043 units in "usr.

Here are so many negative co-efficient are present in linear equation.

End of Project 1

Problem - 2

In your role as a statistician at the Republic of Indonesia Ministry of Health, you have been entrusted with a dataset containing information from a Contraceptive Prevalence Survey. This dataset encompasses data from 1473 married females who were either not pregnant or were uncertain of their pregnancy status during the survey.

Your task involves predicting whether these women opt for a contraceptive method of choice. This prediction will be based on a comprehensive analysis of their demographic and socio-economic attributes.

Data Description -

- Wife's age (numerical)
- Wife's education (categorical) 1=uneducated, 2, 3, 4=tertiary
- Husband's education (categorical) 1=uneducated, 2, 3, 4=tertiary
- Number of children ever born (numerical)
- Wife's religion (binary) Non-Scientology, Scientology
- Wife's now working? (binary) Yes, No
- Husband's occupation (categorical) 1, 2, 3, 4(random)
- Standard-of-living index (categorical) 1=very low, 2, 3, 4=high
- Media exposure (binary) Good, Not good
- Contraceptive method used (class attribute) No, Yes

Summery –

The Given dataset contains data of 1473 females collected from a Contraceptive Prevalence Survey. And the not sure if they were pregnant or not at the time of the survey.

The Model need predict do/don't they use a contraceptive method of choice based on their demographic and socio-economic characteristics.

Importing the required libraries regarding logistic regression, LDA and CART.

Reading the excel file.

As per the summary the excel □le contains data of data of 1473 females collected from a Contraceptive Prevalence Survey.

Check shape, Data types, Statistical Summary:

```
Count of Rows : 1473
Count of Columns : 10
```

To understand the Statistics/Description of data like min, 25%. 50%, 75%, max(), std, mean, count.

	count	mean	std	min	25%	50%	75%	max
Wife_age	1402.0	32.606277	8.274927	16.0	26.0	32.0	39.0	49.0
No_of_children_born	1452.0	3.254132	2.365212	0.0	1.0	3.0	4.0	16.0
Husband_Occupation	1473.0	2.137814	0.864857	1.0	1.0	2.0	3.0	4.0

To check Data Type, Column Name, Count of values:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1473 entries, 0 to 1472
Data columns (total 10 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Wife_age                   1402 non-null   float64
 1   Wife_ education            1473 non-null   object
 2   Husband_education          1473 non-null   object
 3   No_of_children_born        1452 non-null   float64
 4   Wife_religion              1473 non-null   object
 5   Wife_Working               1473 non-null   object
 6   Husband_Occupation         1473 non-null   int64
 7   Standard_of_living_index   1473 non-null   object
 8   Media_exposure             1473 non-null   object
 9   Contraceptive_method_used  1473 non-null   object
dtypes: float64(2), int64(1), object(7)
memory usage: 115.2+ KB
```

To get Top 5 & Bottom 5 Records

	Wife_age	Wife_ education	Husband_education	No_of_children_born	Wife_religion	Wife_Working	Husband_Occupation	Standard_of_living_index	Media_exposure	Contrac
0	24.0	Primary	Secondary	3.0	Scientology	No	2	High	Exposed	
1	45.0	Uneducated	Secondary	10.0	Scientology	No	3	Very High	Exposed	
2	43.0	Primary	Secondary	7.0	Scientology	No	3	Very High	Exposed	
3	42.0	Secondary	Primary	9.0	Scientology	No	3	High	Exposed	
4	36.0	Secondary	Secondary	8.0	Scientology	No	3	Low	Exposed	

	Wife_age	Wife_ education	Husband_education	No_of_children_born	Wife_religion	Wife_Working	Husband_Occupation	Standard_of_living_index	Media_exposure	Cont
1468	33.0	Tertiary	Tertiary	NaN	Scientology	Yes	2	Very High	Exposed	
1469	33.0	Tertiary	Tertiary	NaN	Scientology	No	1	Very High	Exposed	
1470	39.0	Secondary	Secondary	NaN	Scientology	Yes	1	Very High	Exposed	
1471	33.0	Secondary	Secondary	NaN	Scientology	Yes	2	Low	Exposed	
1472	17.0	Secondary	Secondary	1.0	Scientology	No	2	Very High	Exposed	

Check the Duplicate Value:

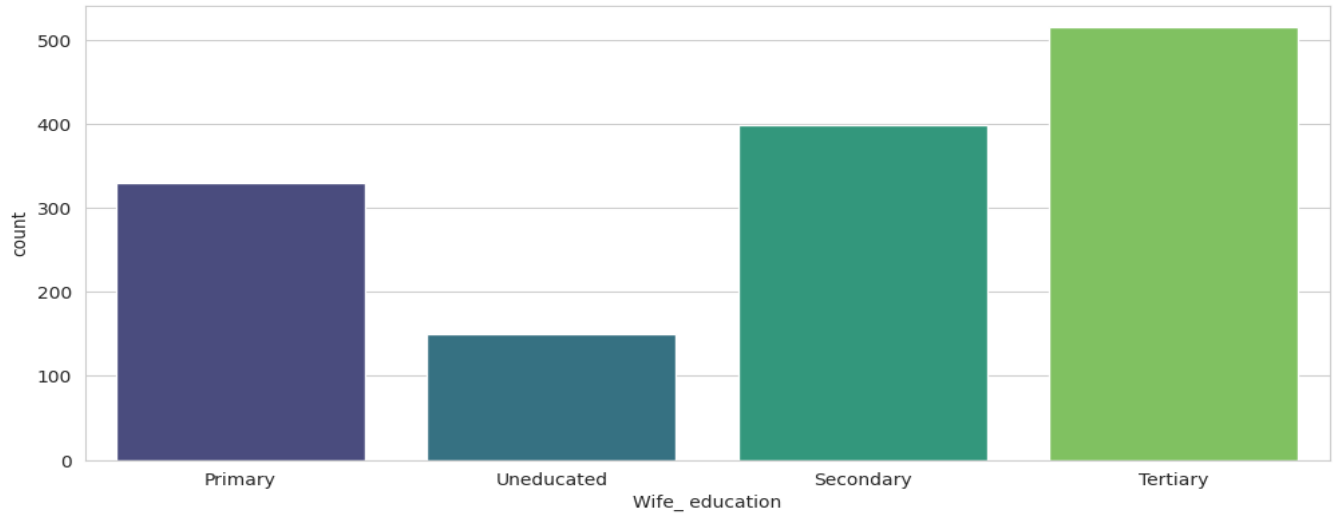
```
Number of Duplicate Rows 80
```

After deleting the duplicate data, we have checked again and found that there is no duplicate in dataset.

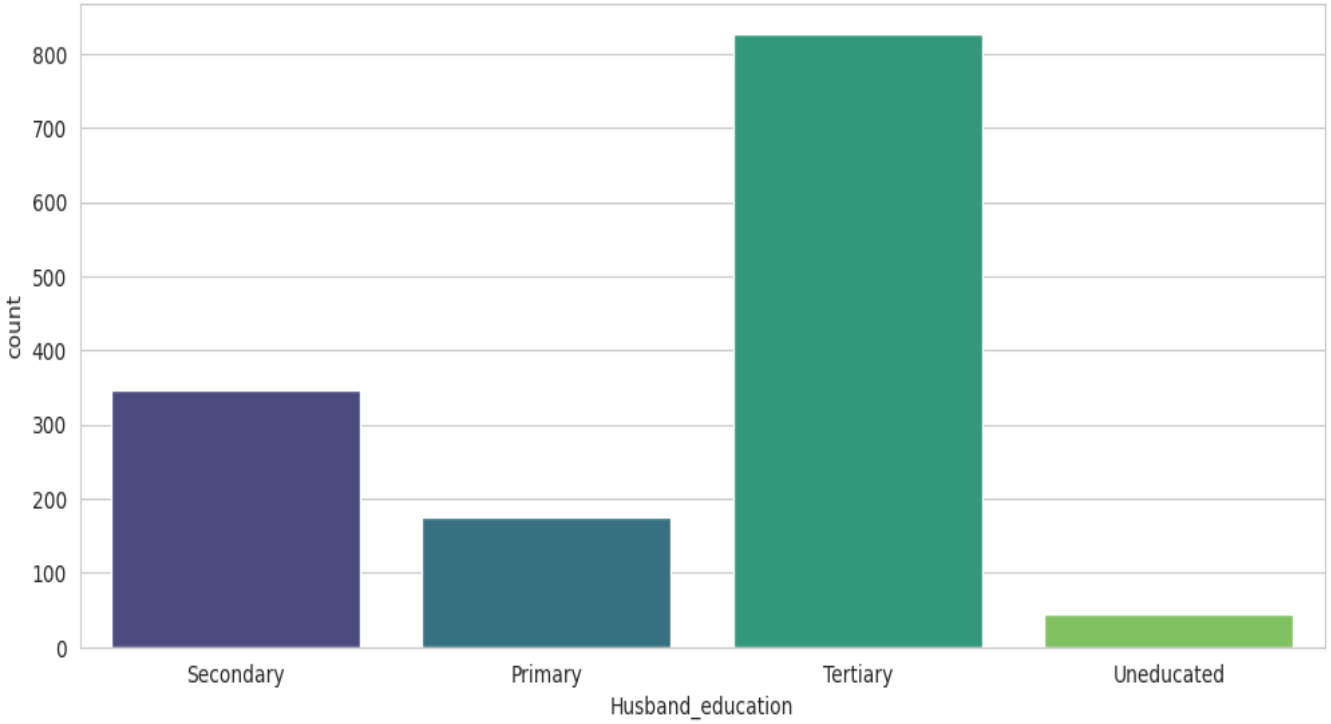
```
Number of duplicate rows 0
```

Univariate Analysis:

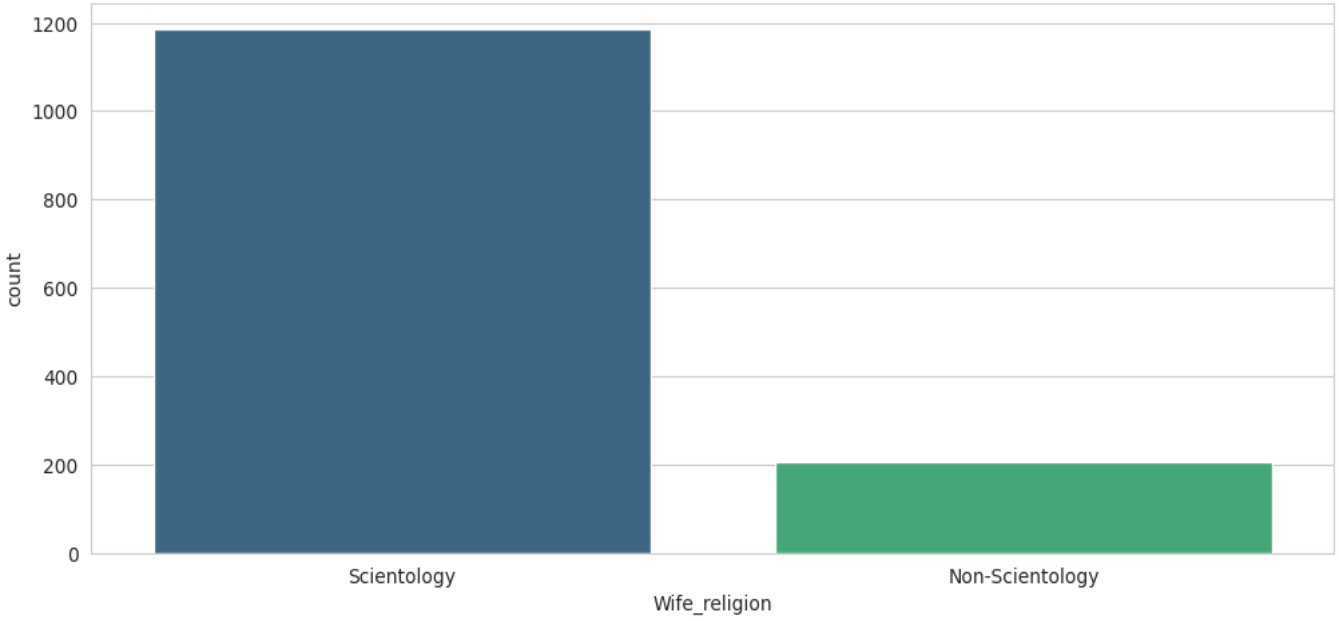
Wife_ education
Tertiary 515
Secondary 398
Primary 330
Uneducated 150
Name: count, dtype: int64



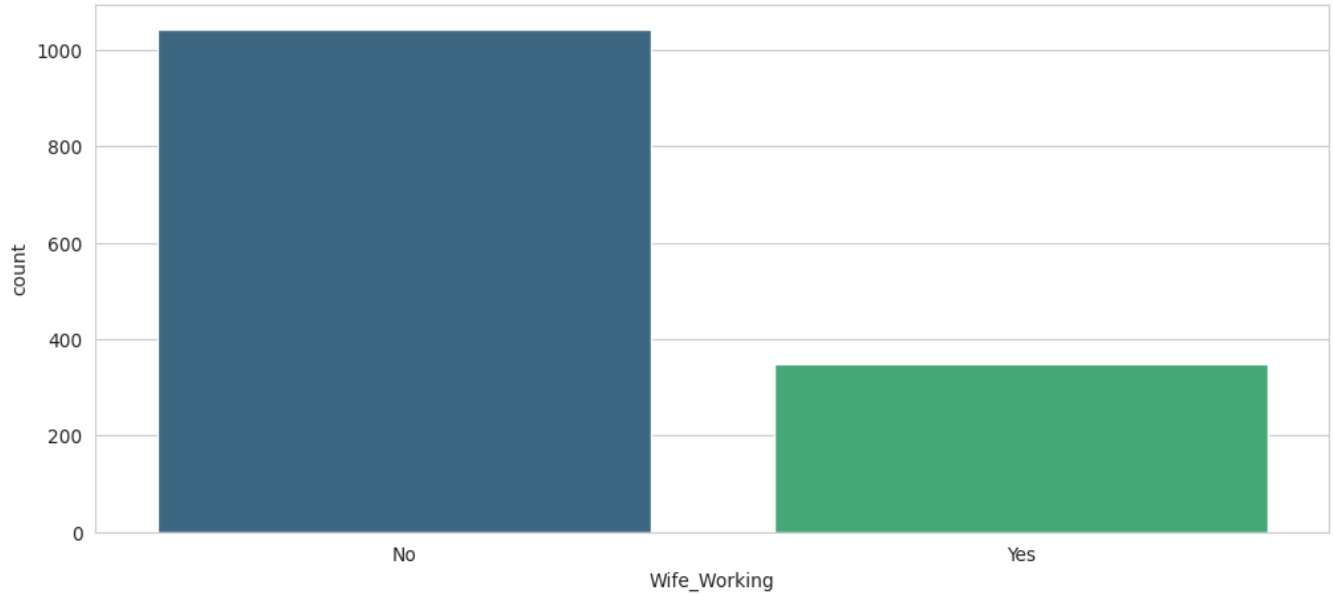
Husband_education
Tertiary 827
Secondary 347
Primary 175
Uneducated 44
Name: count, dtype: int64



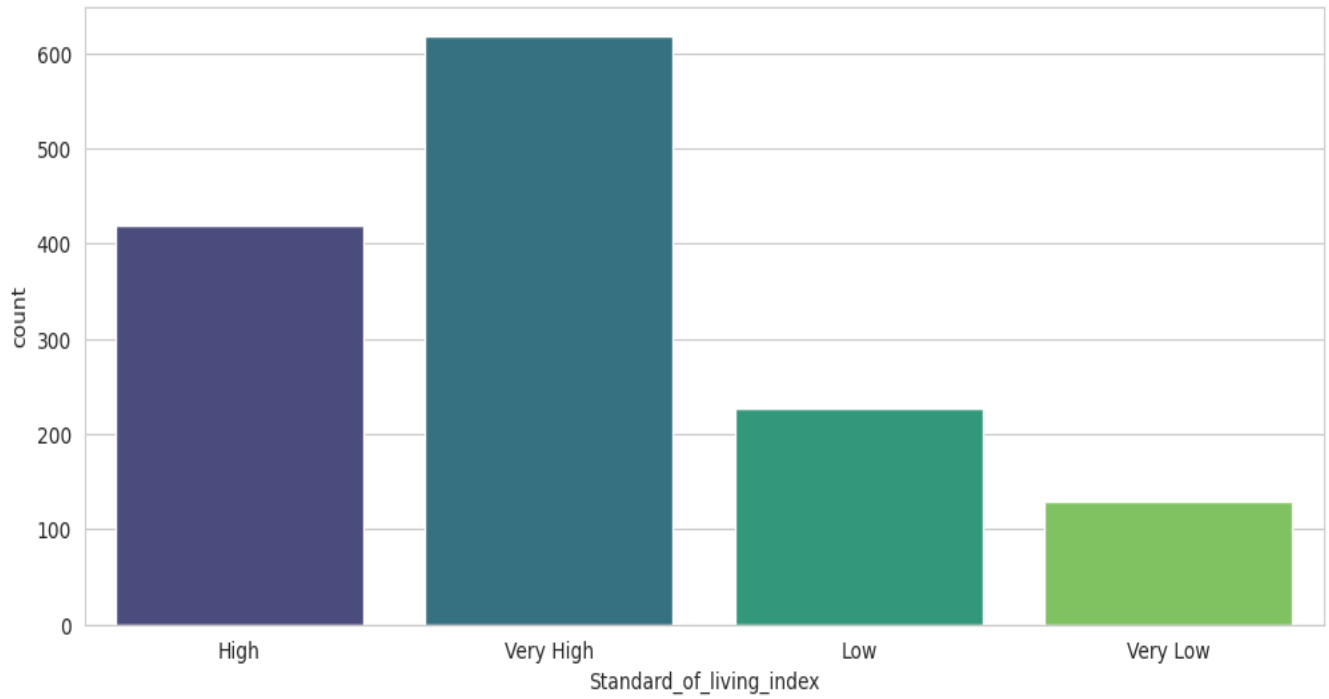
Wife_religion
Scientology 1186
Non-Scientology 207
Name: count, dtype: int64



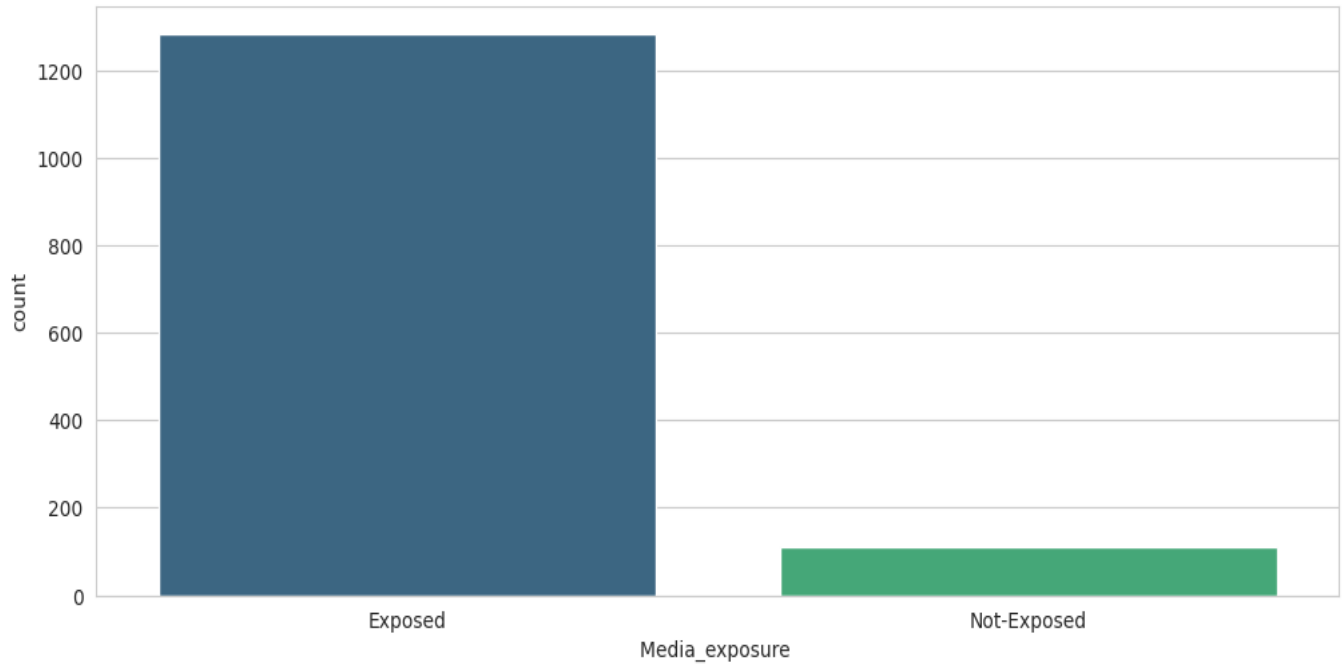
Wife_Working
No 1043
Yes 350
Name: count, dtype: int64



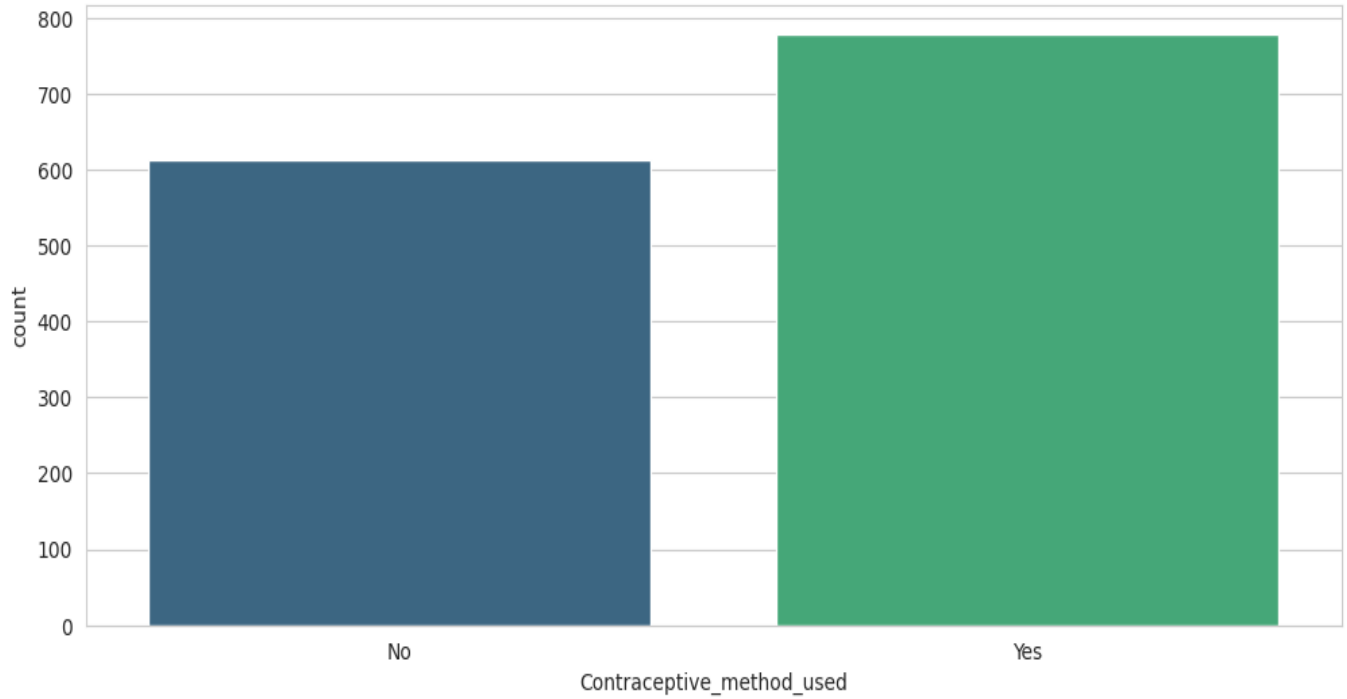
Standard_of_living_index
Very High 618
High 419
Low 227
Very Low 129
Name: count, dtype: int64

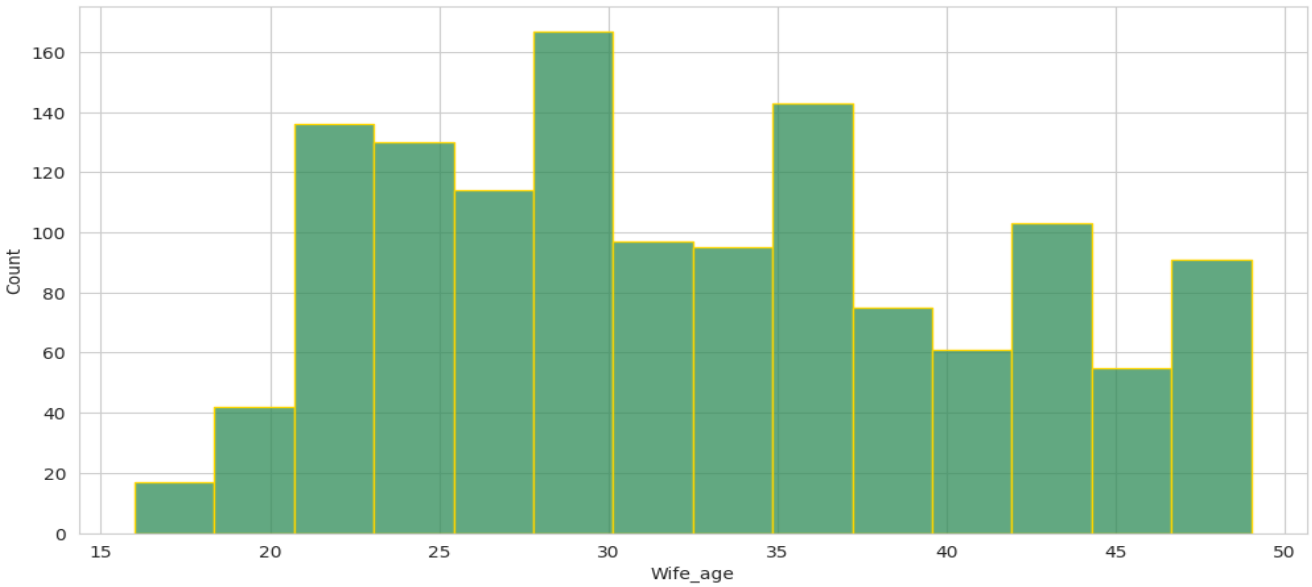


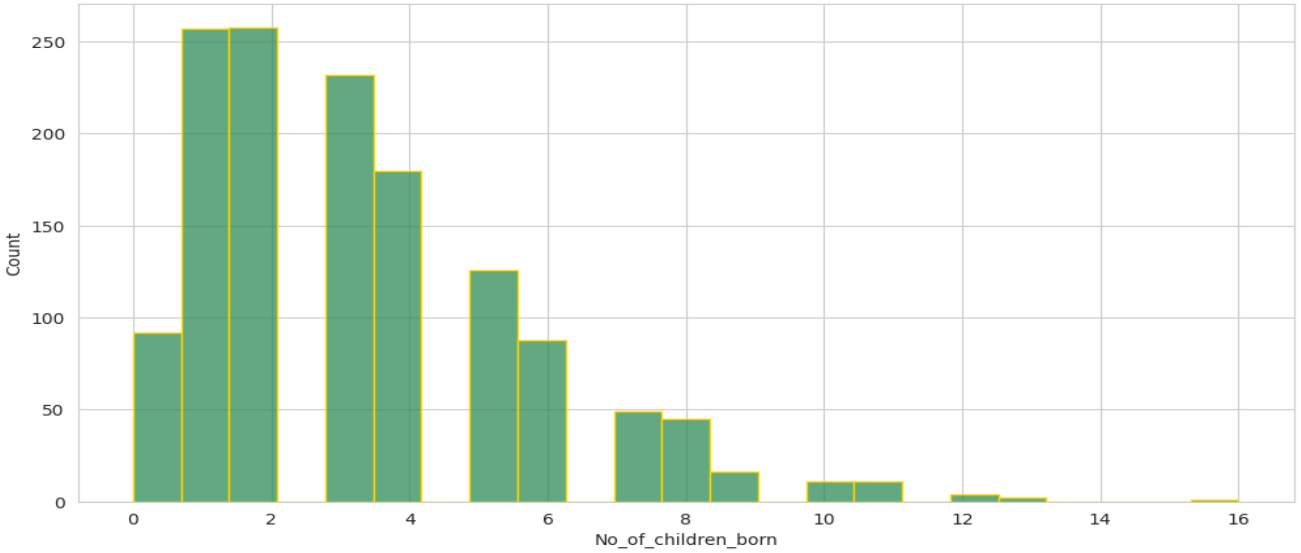
Media_exposure
Exposed 1284
Not-Exposed 109
Name: count, dtype: int64

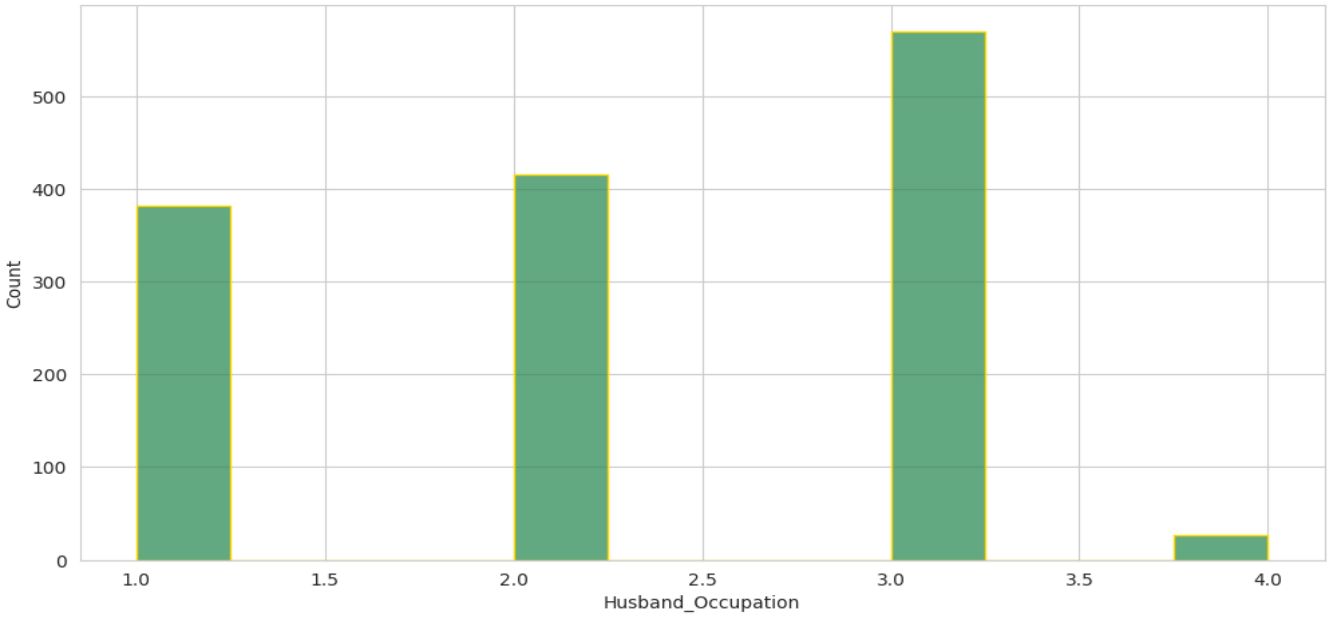


Contraceptive_method_used
Yes 779
No 614
Name: count, dtype: int64

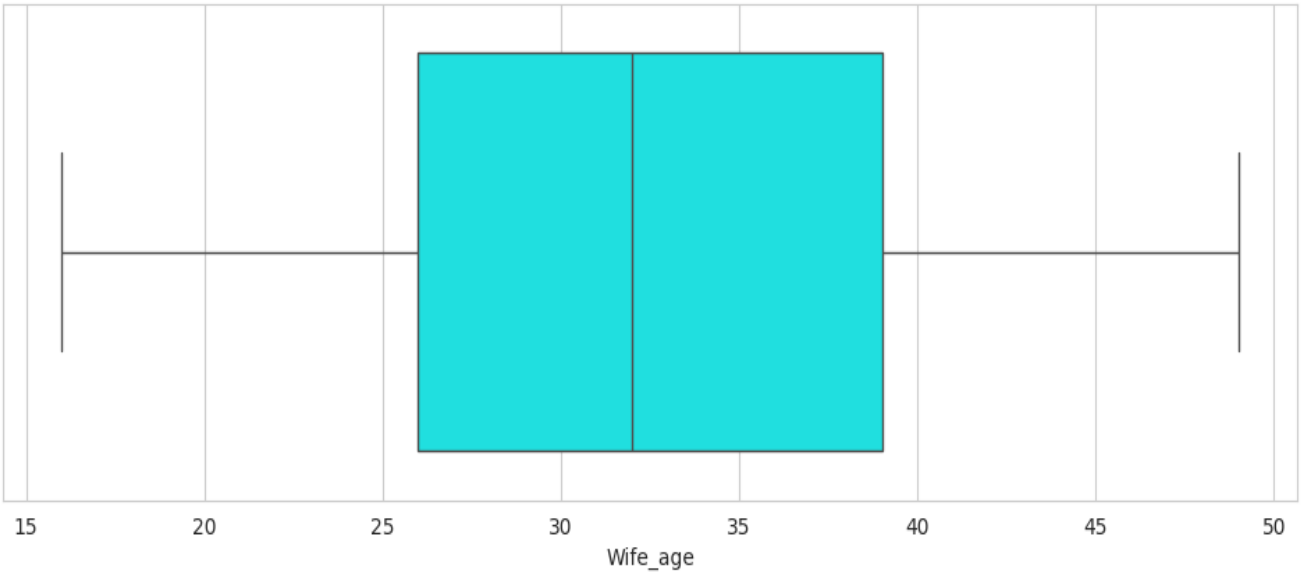


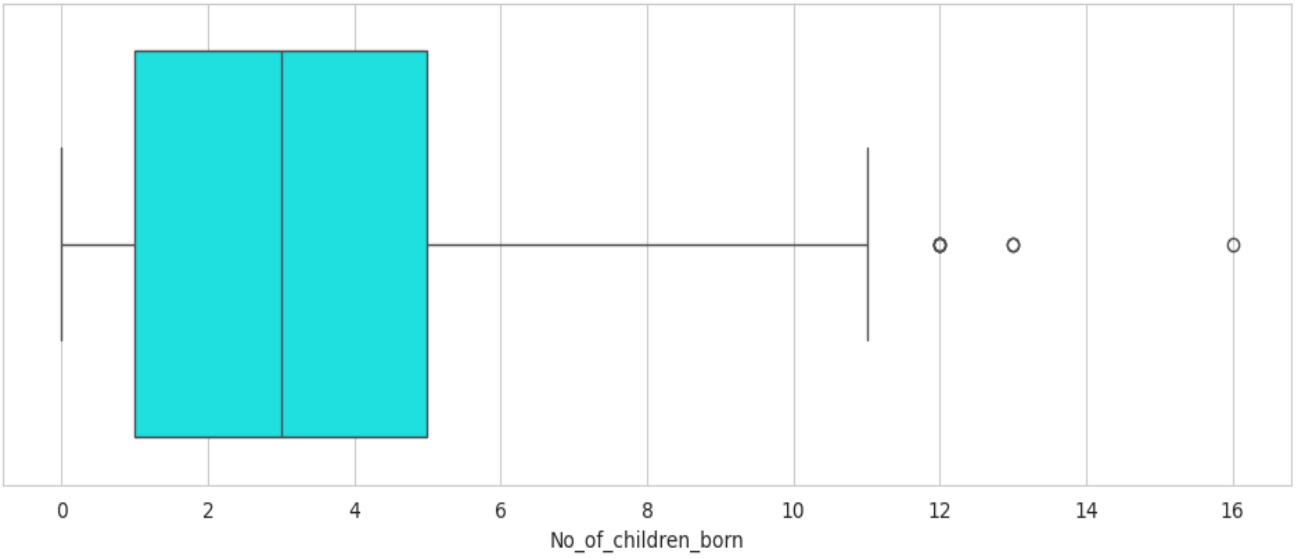

Count
Wife_age

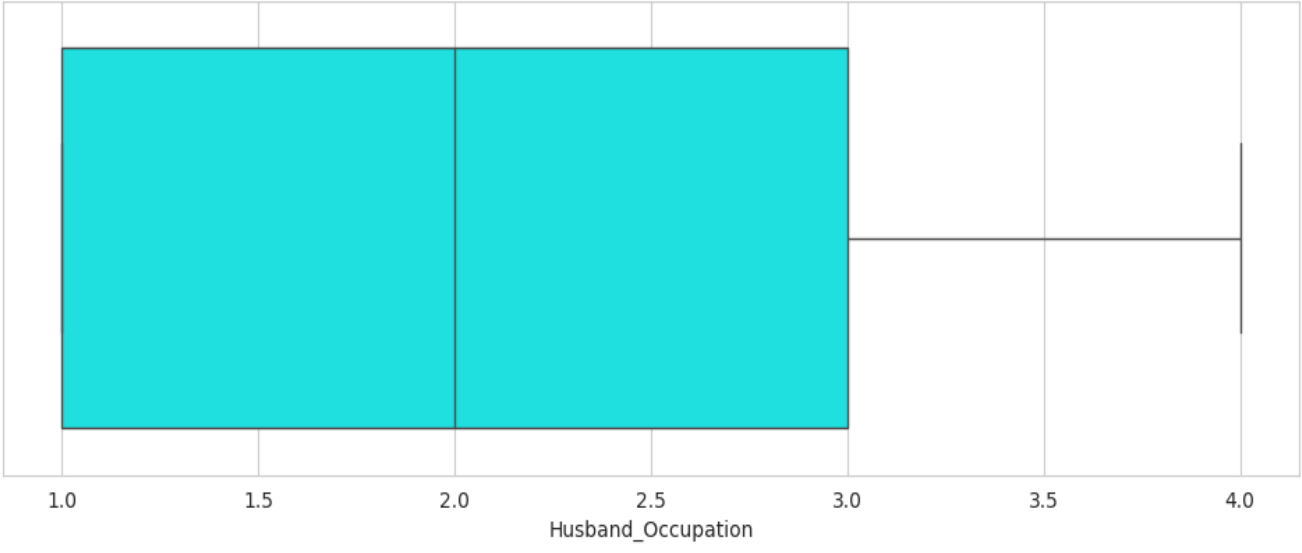

Count
No_of_children_born


Count
Husband_Occupation

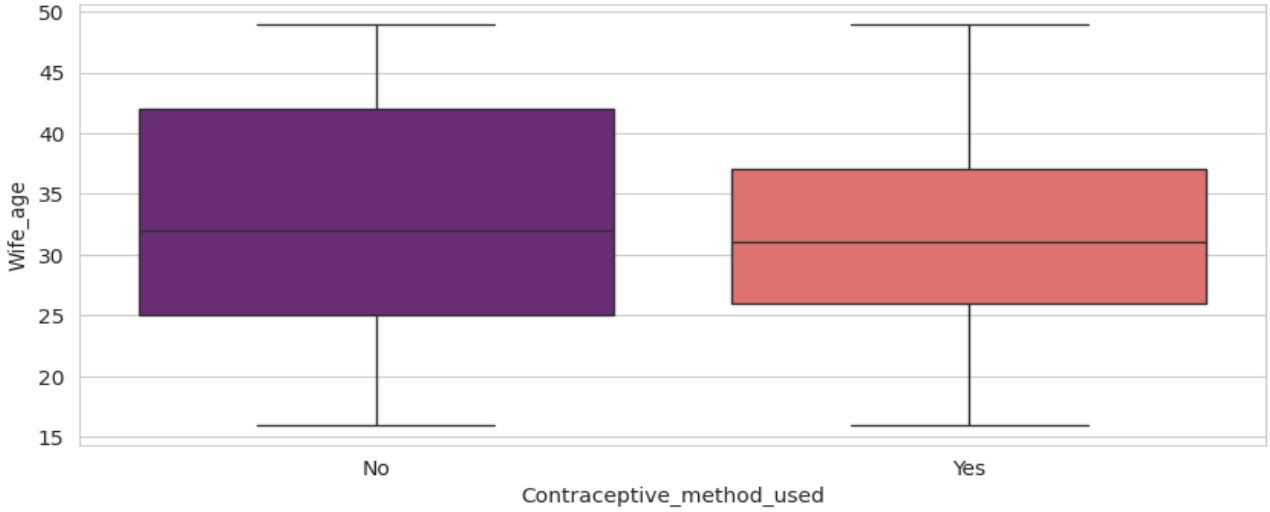
Boxplot -

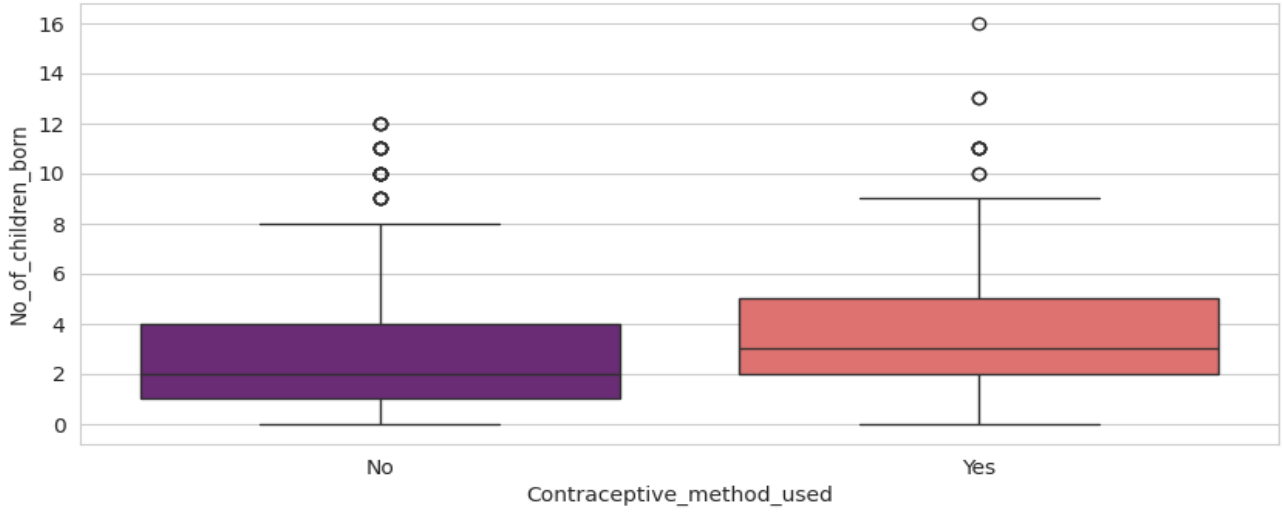

15
20
25
30
35
40
45
50
Wife_age

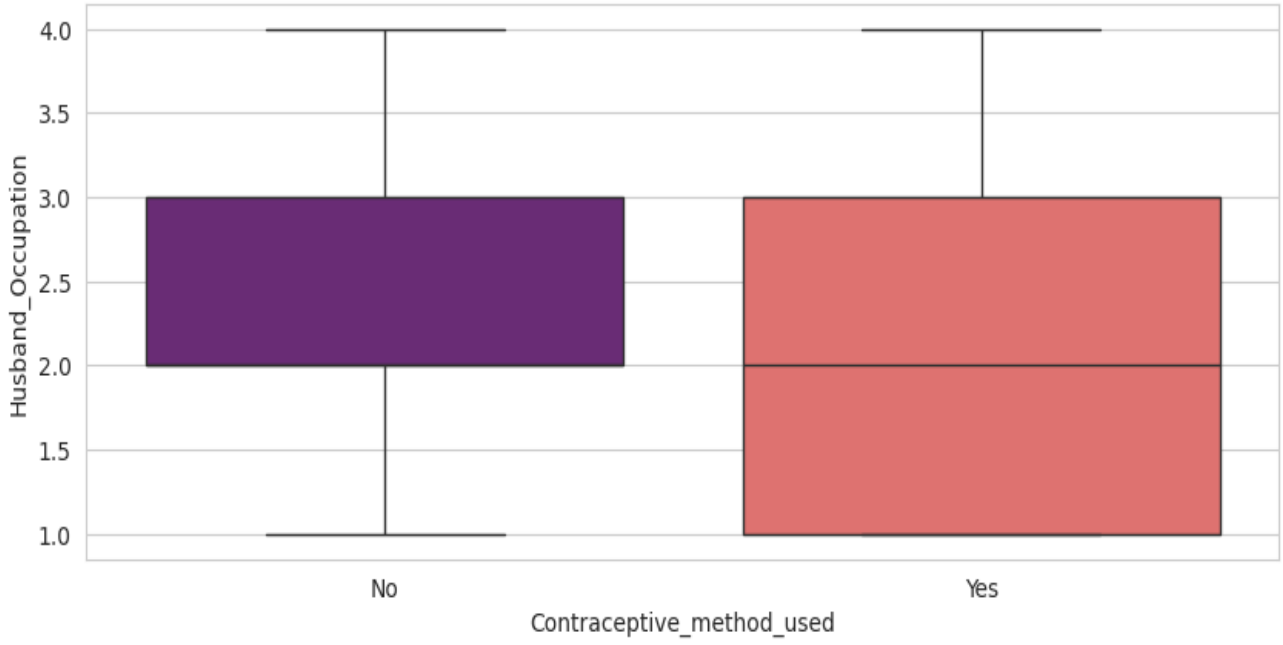

0
2
4
6
8
10
12
14
16
No_of_children_born


1.0
1.5
2.0
2.5
3.0
3.5
4.0
Husband_Occupation

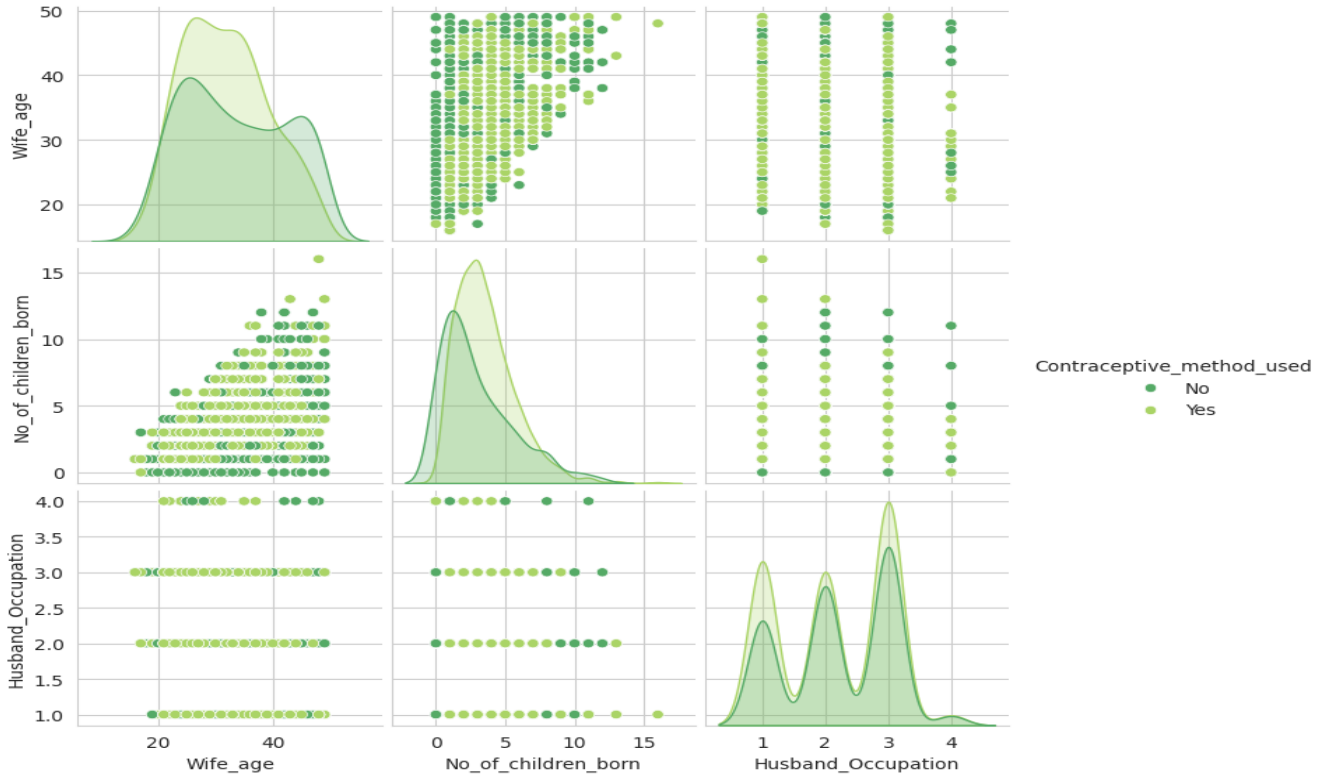
Multivariate Analysis:



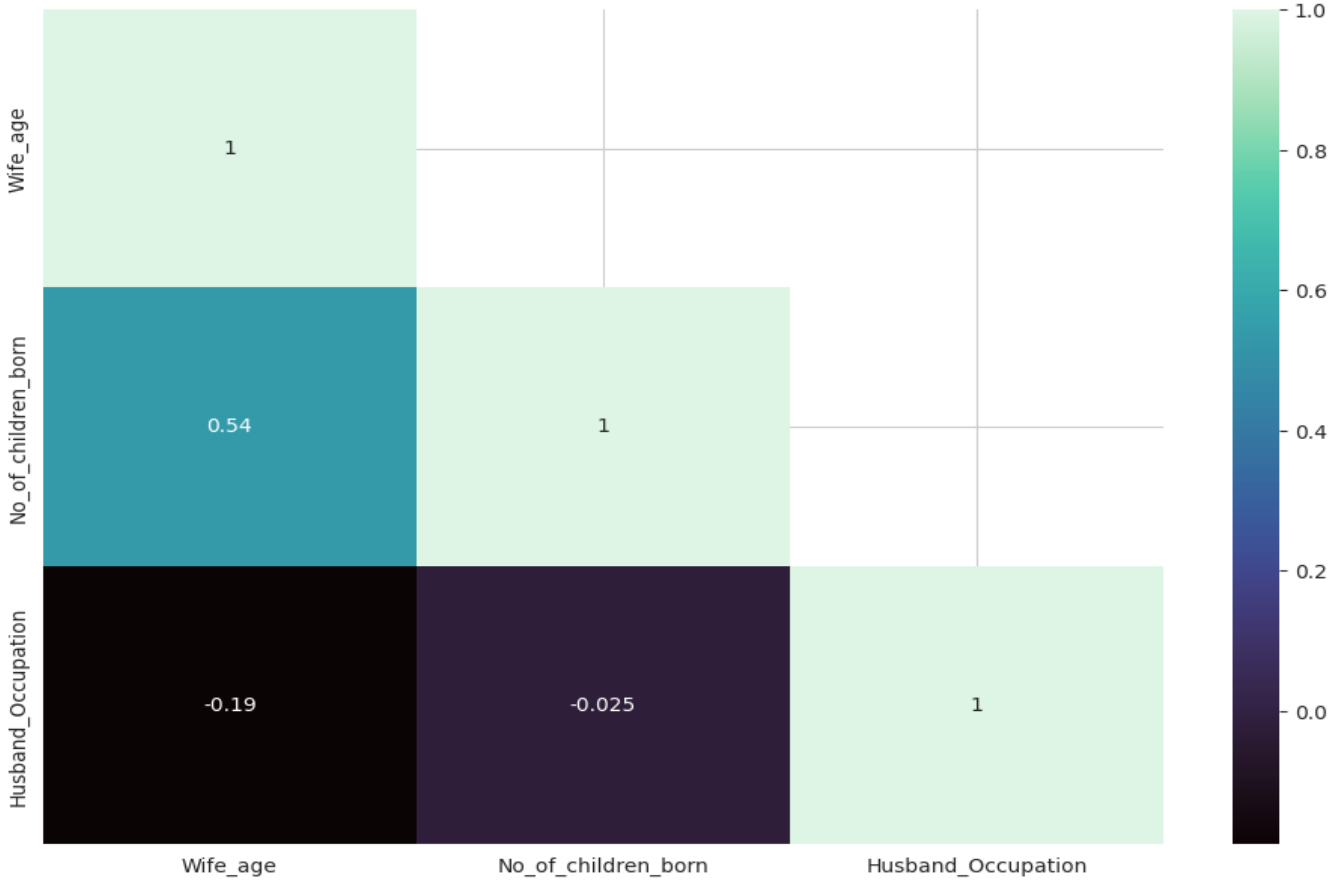




Pari plot -



Heatmap-



Data Pre-processing:

Find out the Missing Value-

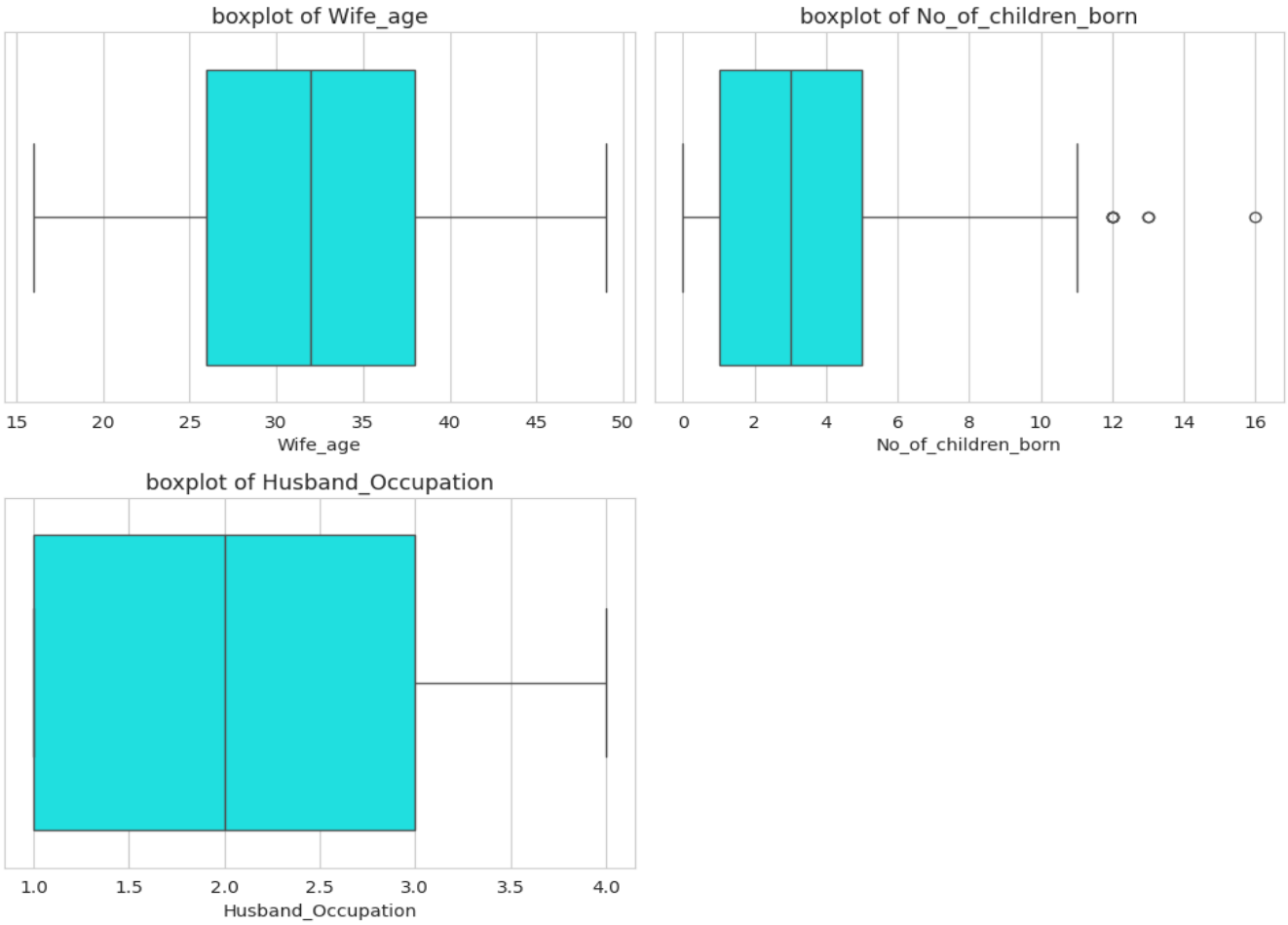
```
Wife_age                    67
Wife_ education              0
Husband_education            0
No_of_children_born         21
Wife_religion                0
Wife_Working                 0
Husband_Occupation           0
Standard_of_living_index     0
Media_exposure               0
Contraceptive_method_used    0
dtype: int64
```

Treatment the Missing value using Median Function.

Check the Missing value after Treatment.

```
Wife_age                    0
Wife_ education             0
Husband_education           0
No_of_children_born         0
Wife_religion               0
Wife_Working                0
Husband_Occupation          0
Standard_of_living_index    0
Media_exposure              0
Contraceptive_method_used   0
dtype: int64
```

Find out the Outlier-



In this case we are not treating the outlier.

Converting All the Objects to Categorical Codes-

```
df['Wife_ education'] =  np.where(df['Wife_ education'] == 'Uneducated', 1, df['Wife_ education'])
df['Wife_ education'] =  np.where(df['Wife_ education'] == 'Primary', 2, df['Wife_ education'])
df['Wife_ education'] =  np.where(df['Wife_ education'] == 'Secondary', 3, df['Wife_ education'])
df['Wife_ education'] =  np.where(df['Wife_ education'] == 'Tertiary', 4, df['Wife_ education'])
```

```
df['Husband_education'] =  np.where(df['Husband_education'] == 'Uneducated', 1, df['Husband_education'])
df['Husband_education'] =  np.where(df['Husband_education'] == 'Primary', 2, df['Husband_education'])
df['Husband_education'] =  np.where(df['Husband_education'] == 'Secondary', 3, df['Husband_education'])
df['Husband_education'] =  np.where(df['Husband_education'] == 'Tertiary', 4, df['Husband_education'])
```

```
df['Wife_religion'] = np.where(df['Wife_religion'] == 'Non-Scientology', 0, df['Wife_religion'])
df['Wife_religion'] = np.where(df['Wife_religion'] == 'Scientology', 1, df['Wife_religion'])
```

+ Code + Text

```
df['Wife_Working'] = np.where(df['Wife_Working']=='Yes', 0, df['Wife_Working'])
df['Wife_Working'] = np.where(df['Wife_Working']=='No', 1, df['Wife_Working'])
```

```
df['Standard_of_living_index'] = np.where(df['Standard_of_living_index'] == 'Very Low', 1, df['Standard_of_living_index'])
df['Standard_of_living_index'] = np.where(df['Standard_of_living_index'] == 'Low', 2, df['Standard_of_living_index'])
df['Standard_of_living_index'] = np.where(df['Standard_of_living_index'] == 'High', 3, df['Standard_of_living_index'])
df['Standard_of_living_index'] = np.where(df['Standard_of_living_index'] == 'Very High', 4, df['Standard_of_living_index'])
```

```
df['Media_exposure '] =  np.where(df['Media_exposure '] == 'Not-Exposed', 0, df['Media_exposure '])
df['Media_exposure '] =  np.where(df['Media_exposure '] == 'Exposed', 1, df['Media_exposure '])
```

```
df['Contraceptive_method_used'] = np.where(df['Contraceptive_method_used'] =='No', 0, df['Contraceptive_method_used'])
df['Contraceptive_method_used'] = np.where(df['Contraceptive_method_used'] =='Yes', 1, df['Contraceptive_method_used'])
```

Encode the Data –

```
df_dummy = pd.get_dummies(df, drop_first=True)
df_dummy.head()
```

	Wife_age	No_of_children_born	Husband_Occupation	Wife_ education_2	Wife_ education_3	Wife_ education_4	Husband_education_2	Husband_education_3	Husband_education_4	Wife_r
0	24.0	3.0	2	True	False	False	False	True	False	
1	45.0	10.0	3	False	False	False	False	True	False	
2	43.0	7.0	3	True	False	False	False	True	False	
3	42.0	9.0	3	False	True	False	True	False	False	
4	36.0	8.0	3	False	True	False	False	True	False	

Change the data type using astype(int) function. After Encoding the data, dataset look like.

```
<class 'pandas.core.frame.DataFrame'>
Index: 1393 entries, 0 to 1472
Data columns (total 16 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Wife_age                    1393 non-null   float64
 1   No_of_children_born         1393 non-null   float64
 2   Husband_Occupation          1393 non-null   int64
 3   Wife_ education_2           1393 non-null   int64
 4   Wife_ education_3           1393 non-null   int64
 5   Wife_ education_4           1393 non-null   int64
 6   Husband_education_2         1393 non-null   int64
 7   Husband_education_3         1393 non-null   int64
 8   Husband_education_4         1393 non-null   int64
 9   Wife_religion_1             1393 non-null   int64
 10  Wife_Working_1              1393 non-null   int64
 11  Standard_of_living_index_2  1393 non-null   int64
 12  Standard_of_living_index_3  1393 non-null   int64
 13  Standard_of_living_index_4  1393 non-null   int64
 14  Media_exposure _1           1393 non-null   int64
 15  Contraceptive_method_used_1 1393 non-null   int64
dtypes: float64(2), int64(14)
memory usage: 185.0 KB
```

Train-test split:

Here we are taking Y variable as ‘Contraceptive Method Used’ and we are splitting the variable to Train Test split.

```
Contraceptive_method_used_1
1    0.558974
0    0.441026
Name: proportion, dtype: float64
```

56% of people comes under the category that they have chosen contraceptive method used.

And 44% of women or people has not taken the contraceptive method used.

Model Building and Compare the Performance of the Models

- Build a Logistic Regression model

- Build a Linear Discriminant Analysis model

- Build a CART model

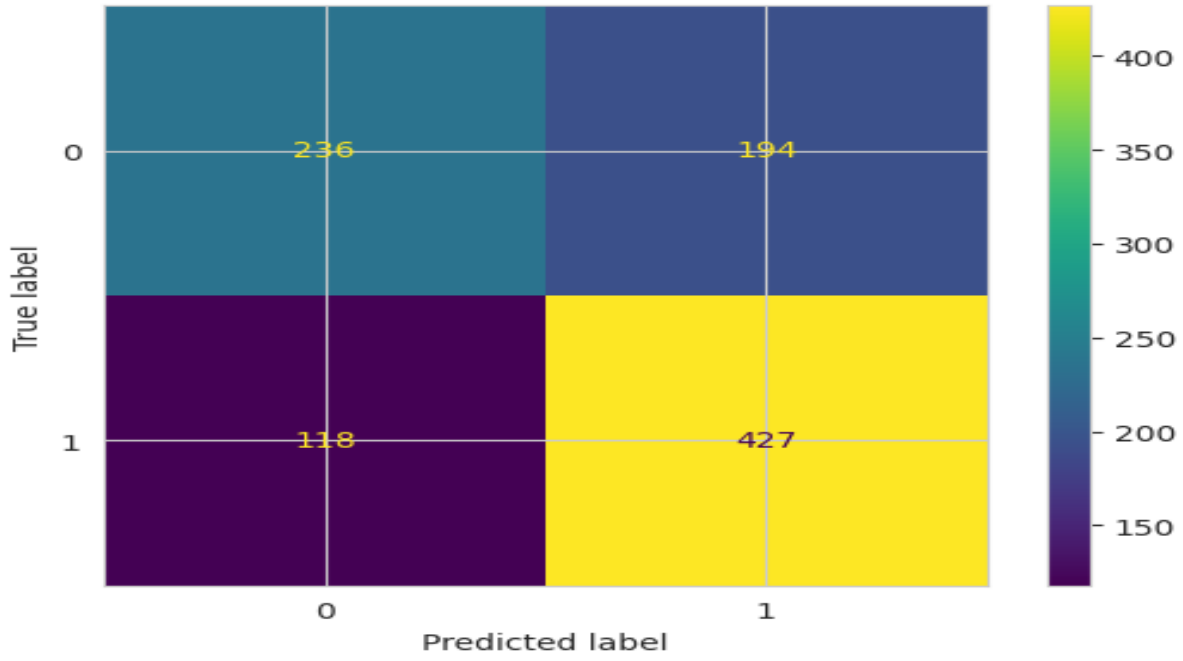
- Prune the CART model by finding the best hyperparameters using Grid Search

- Check the performance of the models across train and test set using different metrics

- Compare the performance of all the models built and choose the best one with proper rationale

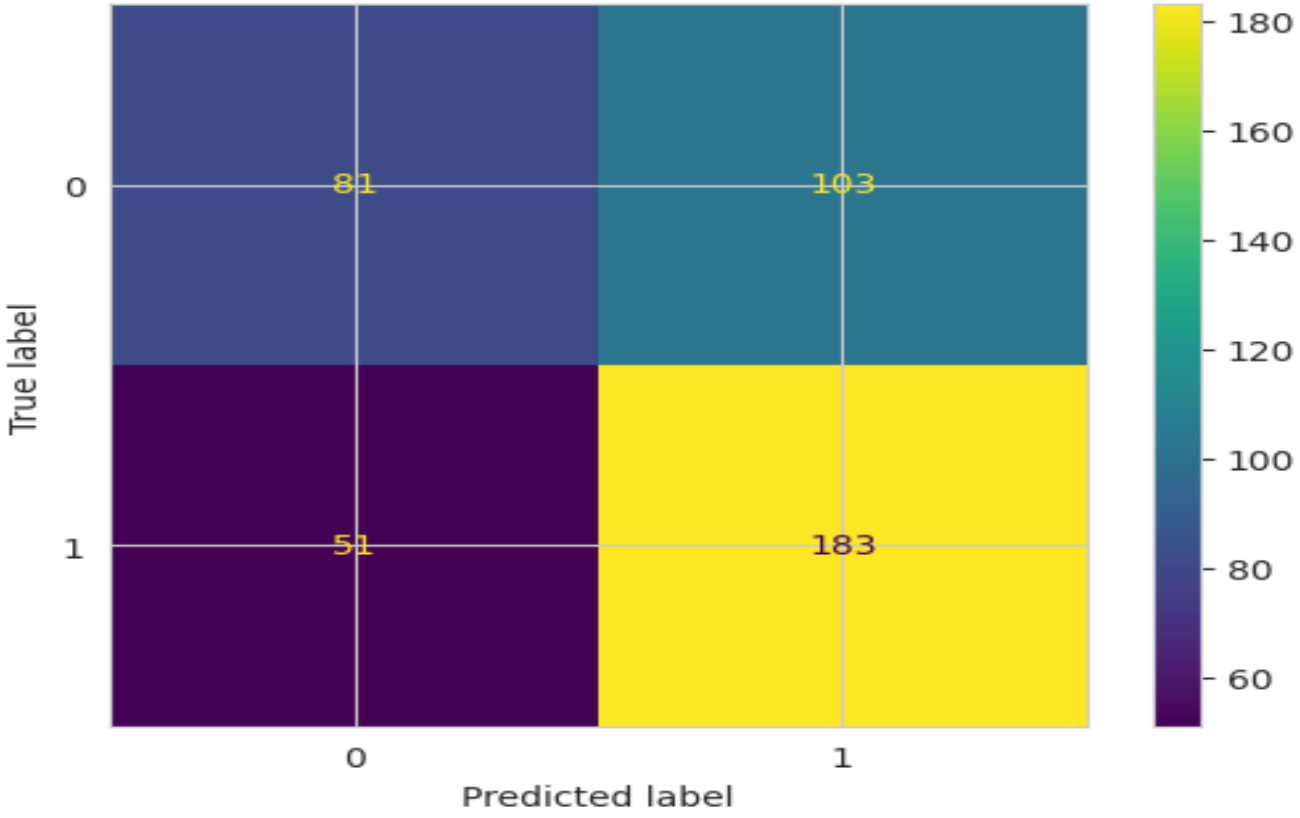
Build a Logistic Regression Model:
X-Train or Y-Train Accuracy. Accuracy of trained data is 68.
X -Test or Y-Test Accuracy. Accuracy of test data is 63.

Confusion Matrix for the training data



	precision	recall	f1-score	support
0	0.67	0.55	0.60	430
1	0.69	0.78	0.73	545
accuracy			0.68	975
macro avg	0.68	0.67	0.67	975
weighted avg	0.68	0.68	0.67	975

Confusion Matrix for test data



	precision	recall	f1-score	support
0	0.61	0.44	0.51	184
1	0.64	0.78	0.70	234
accuracy			0.63	418
macro avg	0.63	0.61	0.61	418
weighted avg	0.63	0.63	0.62	418

CONCLUSION -

For No Contraceptive_method_used (Label 0)

Precision (61%) – 61% of women predicted are not opting for Contraceptive_method.

Recall (44%) – 44% of women predicted are not opting for Contraceptive_method. 44% have been predicted correctly.

- **For Contraceptive_method_used (Label 1)**

Precision (64%) – 64% of women predicted are opting for Contraceptive_method.

Recall (78%) – 78% of women predicted are opting for Contraceptive_method, 78% of women have been predicted correctly.

Overall Accuracy of the model – 63 % of total predictions are correct.

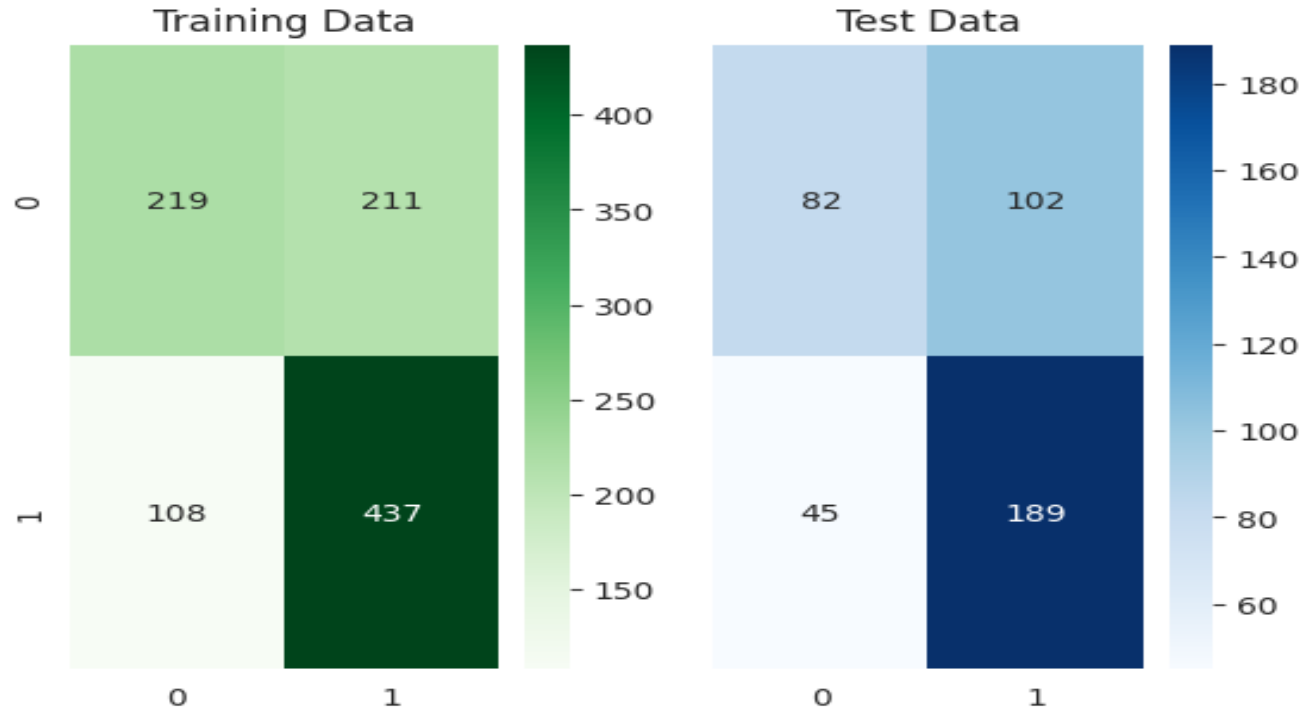
Build a Linear Discriminant Analysis Model:

```
<class 'pandas.core.frame.DataFrame'>
Index: 1393 entries, 0 to 1472
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Wife_age                  1393 non-null   float64
 1   Wife_ education           1393 non-null   object
 2   Husband_education         1393 non-null   object
 3   No_of_children_born       1393 non-null   float64
 4   Wife_religion             1393 non-null   object
 5   Wife_Working              1393 non-null   object
 6   Husband_Occupation        1393 non-null   int64
 7   Standard_of_living_index  1393 non-null   object
 8   Media_exposure            1393 non-null   object
 9   Contraceptive_method_used 1393 non-null   object
dtypes: float64(2), int64(1), object(7)
memory usage: 152.0+ KB
```

Train and Test Split: The procedure is same as the above Logistic regression for splitting the Train and test data. Need to import the LDA (Linear Discriminant analysis) from the sklearn library and the results is as follows.

```
Number of rows and columns of the training set for the independent variables: (975, 9)
Number of rows and columns of the training set for the dependent variable: (975,)
Number of rows and columns of the test set for the independent variables: (418, 9)
Number of rows and columns of the test set for the dependent variable: (418,)
```

Training Data and Test Data Confusion Matrix Comparison



Training Data and Test Data Classification Report Comparison

Classification Report of the training data:

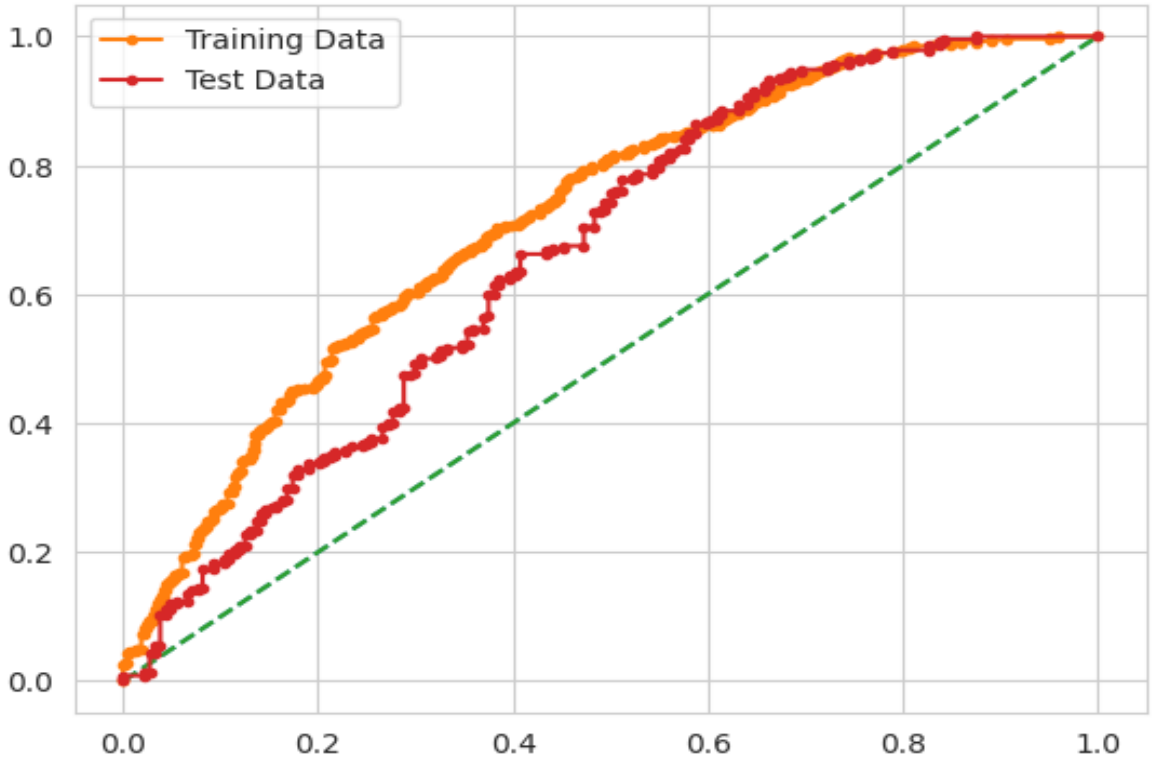
	precision	recall	f1-score	support
0	0.67	0.51	0.58	430
1	0.67	0.80	0.73	545
accuracy			0.67	975
macro avg	0.67	0.66	0.66	975
weighted avg	0.67	0.67	0.66	975

Classification Report of the test data:

	precision	recall	f1-score	support
0	0.65	0.45	0.53	184
1	0.65	0.81	0.72	234
accuracy			0.65	418
macro avg	0.65	0.63	0.62	418
weighted avg	0.65	0.65	0.64	418

AUC and ROC for the Training Data:

```
AUC for the Training Data: 0.719
AUC for the Test Data: 0.664
```



Classification Report of the test data:

```
Classification Report of the test data:

              precision    recall  f1-score   support

           0       0.63      0.46      0.53       184
           1       0.65      0.79      0.71       234

    accuracy                           0.64       418
   macro avg       0.64      0.62      0.62       418
weighted avg       0.64      0.64      0.63       418
```

Observations:

- **Women who did not prefer Contraceptive method (label : No)**

  Precision (63%) : 63% of women who did not chosen contraceptive method are truly predicted when compared out of all the women are predicted.

  Recall (46%): out of all women who didn't choose contraceptive method, 46% of women are correctly predicted.

- **Women who prefer Contraceptive method (label : Yes)**

  Precision (65%): 65% of women who did not chosen contraceptive method are truly predicted when compared out of all the women are predicted.

Recall (79%): out of all women who didn't choose contraceptive method, 79% of women are correctly predicted.

Overall accuracy of the model – 64% of total predictions are correct.

Build a CART Model

The Same procedure as the above Logistic regression and the LDA, Train and test data need to be splitted, and before that the necessary libraries need to be imported.

In Cart, the decision tree is the most important.

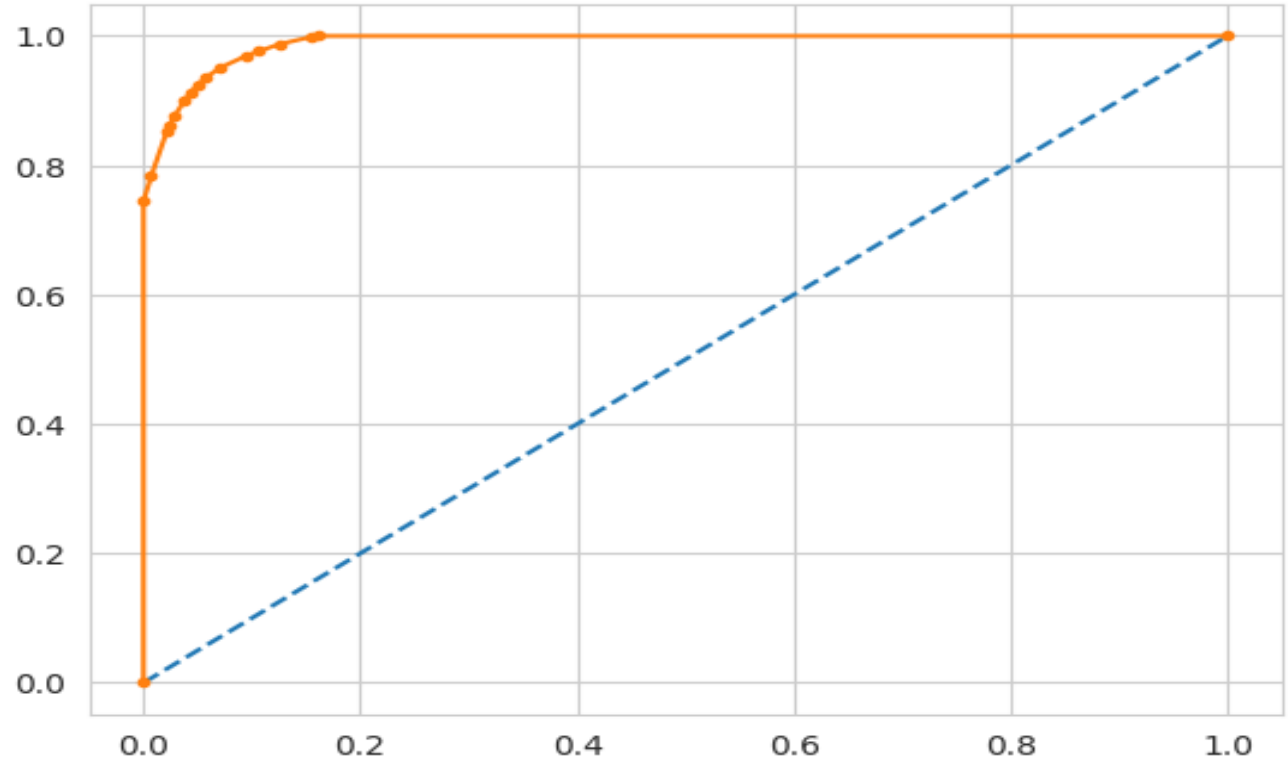
Decision Tree:

- Fit the train and test data into decision tree. We need to create in new word document and saved in Project folder.
- Now we can copy and paste the code in http://webgraphviz.com/. For checking the decision tree we can delete the existing codes and paste it there.
- The tree will be little messy as the data contains vast information on or classifications, so we will reduce the max.leaf, max.depth of the tree and the min. sample size.
- Here "GINI", a decision tree classifier plays the important role. And creating a new word document with reduced branches as 30, leaf is 10 and depth is 7 and saved the document in project folder.
- Now decision tree is looking better than before.

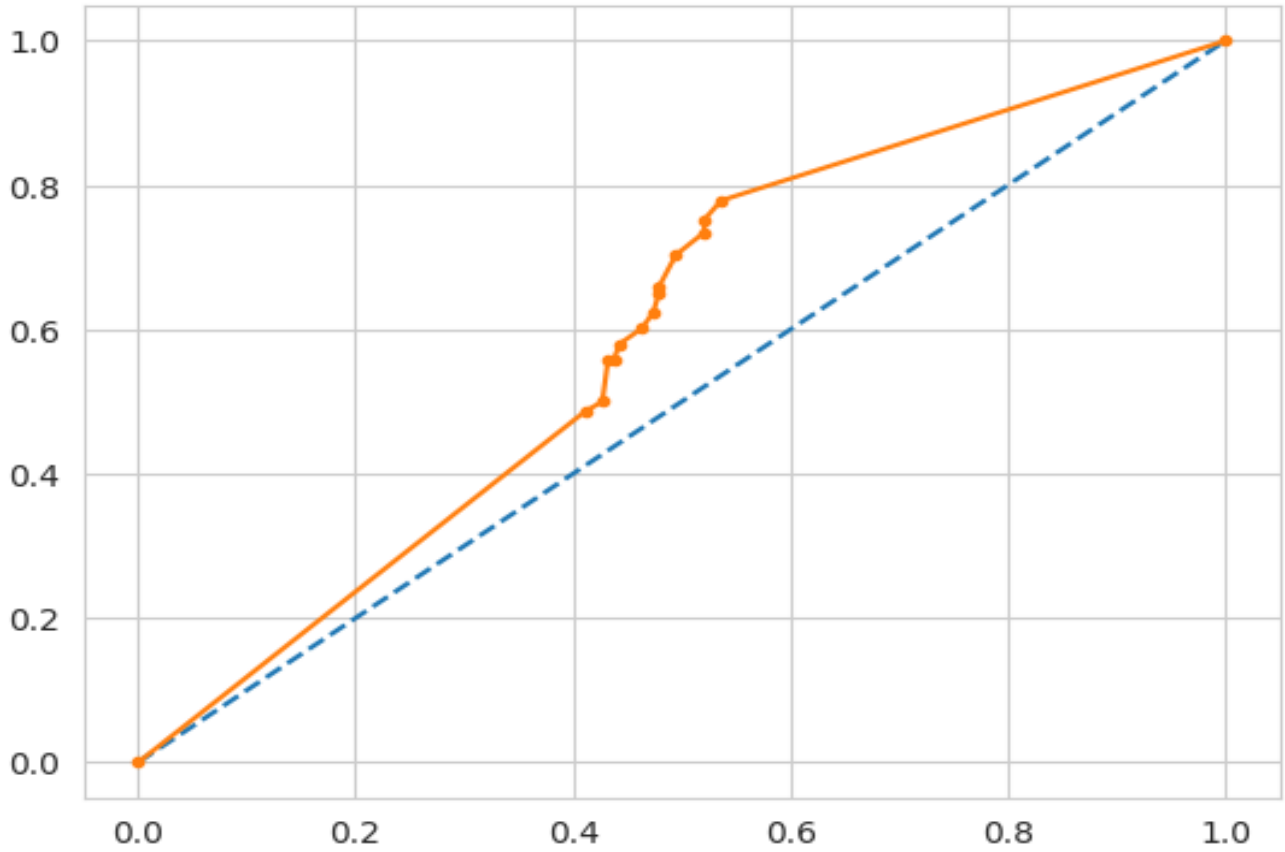
Now Let us check the feature Importance, where Feature importance refers to techniques that assign a score to input features based on how useful they are at predicting a target variable.

	Imp
Wife_age	0.302816
No_of_children_born	0.245536
Wife_ education	0.101109
Husband_Occupation	0.101109
Standard_of_living_index	0.094737
Husband_education	0.052621
Wife_Working	0.047123
Wife_religion	0.037422
Media_exposure	0.017527

AUC and ROC for the training data



AUC and ROC for the test data



Confusion matrix for train data

array([[392, 30],
[27, 526]])

Train Data Accuracy:

0.9415384615384615

Classification Report -

	precision	recall	f1-score	support
0	0.94	0.93	0.93	422
1	0.95	0.95	0.95	553
accuracy			0.94	975
macro avg	0.94	0.94	0.94	975
weighted avg	0.94	0.94	0.94	975

Confusion Matrix for test data

```
array([[ 97,  95],
       [ 67, 159]])
```

Test Data Accuracy

0.6124401913875598

Classification Report -

	precision	recall	f1-score	support
0	0.59	0.51	0.54	192
1	0.63	0.70	0.66	226
accuracy			0.61	418
macro avg	0.61	0.60	0.60	418
weighted avg	0.61	0.61	0.61	418

Conclusion -

Accuracy on the Training Data: 94%

Accuracy on the Test Data: 61%.