Importing required libraries

In [147…

```
# Libraries to help with reading and manipulating data
import pandas as pd
import numpy as np

# libaries to help with data visualization
import matplotlib.pyplot as plt
import seaborn as sns

# to scale the data using z-score
from sklearn.preprocessing import StandardScaler

# to perform Linear Regression
import statsmodels.api as sm
import statsmodels.stats.api as sms
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.compat import lzip
import scipy.stats as stats
from scipy.stats import shapiro
import pylab

# To check model performance
from sklearn.metrics import mean_absolute_error, mean_squared_error

# to suppress warnings
import warnings

warnings.filterwarnings("ignore")
```

Problem Statement:

Context:

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 Dictionary

System measures used:

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

Understanding the structure of data

In [148…

```
df_lr = pd.read_excel('compactiv.xlsx')
```

In [149…

```
df_lr.head() # Returns first 5 rows
```

Out[149…

	lread	lwrite	scall	sread	swrite	fork	exec	rchar	wchar	pgout	...	pgscan	atc
0	1	0	2147	79	68	0.2	0.2	40671.0	53995.0	0.0	...	0.0	0.
1	0	0	170	18	21	0.2	0.2	448.0	8385.0	0.0	...	0.0	0.
2	15	3	2162	159	119	2.0	2.4	NaN	31950.0	0.0	...	0.0	1.
3	0	0	160	12	16	0.2	0.2	NaN	8670.0	0.0	...	0.0	0.
4	5	1	330	39	38	0.4	0.4	NaN	12185.0	0.0	...	0.0	0.

5 rows × 22 columns

Number of rows and columns in the dataset

In [150…

```
# checking shape of the data

rows = str(df_lr.shape[0])
columns = str(df_lr.shape[1])

print(f"There are \033[1m" + rows + "\033[0m rows and \033[1m" + columns + "\033[0m
```

```
There are 8192 rows and 22 columns in the dataset.
```

Datatypes of the different columns in the dataset

In [151…

```
df_lr.info() # Concise summary of 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
```

There are 22 columns in the dataset. Out of which 13 have float data type, 8 have integer data type and 1 have object data type.

Finding missing values in the dataset

In [152…

```
df_lr.isna().sum()  # Count NaN values in all columns of dataset
```

Out[152…

```
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
```

rchar and wchar columns have NaN values in 104 and 15 rows.

Treating missing values in the dataset

In [153…

```
# Use of fillna method to treat missing values in rchar and wchar columns

df_lr['rchar'] = df_lr['rchar'].fillna(df_lr['rchar'].median()) # Replace NaN value
df_lr['wchar'] = df_lr['wchar'].fillna(df_lr['wchar'].median()) # Replace NaN value
```

Median is used for treating the missing values for rchar and wchar columns as distribution is skewed.

In [154…

```
df_lr.isna().sum()  # Count NaN values in all columns of dataset
```

Out[154…

```
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
```

We can see from above list that there are no NaN values in rchar and wchar columns.

Checking for Duplicates

In [155…

```
df_lr.duplicated().sum()
```

Out[155…

```
0
```

There are no duplicate rows in the dataset.

Checking Summary Statistic

In [156…

```
df_lr.describe(include='all').T
```

Out[156…

	count	unique	top	freq	mean	std	min
lread	8192.0	NaN	NaN	NaN	19.559692	53.353799	0.0
lwrite	8192.0	NaN	NaN	NaN	13.106201	29.891726	0.0
scall	8192.0	NaN	NaN	NaN	2306.318237	1633.617322	109.0
sread	8192.0	NaN	NaN	NaN	210.47998	198.980146	6.0
swrite	8192.0	NaN	NaN	NaN	150.058228	160.47898	7.0
fork	8192.0	NaN	NaN	NaN	1.884554	2.479493	0.0
exec	8192.0	NaN	NaN	NaN	2.791998	5.212456	0.0
rchar	8192.0	NaN	NaN	NaN	196472.780151	238446.012054	278.0
wchar	8192.0	NaN	NaN	NaN	95812.751099	140728.464118	1498.0
pgout	8192.0	NaN	NaN	NaN	2.285317	5.307038	0.0
ppgout	8192.0	NaN	NaN	NaN	5.977229	15.21459	0.0
pgfree	8192.0	NaN	NaN	NaN	11.919712	32.36352	0.0
pgscan	8192.0	NaN	NaN	NaN	21.526849	71.14134	0.0
atch	8192.0	NaN	NaN	NaN	1.127505	5.708347	0.0
pgin	8192.0	NaN	NaN	NaN	8.27796	13.874978	0.0
ppgin	8192.0	NaN	NaN	NaN	12.388586	22.281318	0.0
pflt	8192.0	NaN	NaN	NaN	109.793799	114.419221	0.0
vflt	8192.0	NaN	NaN	NaN	185.315796	191.000603	0.2
runqsz	8192	2	Not_CPU_Bound	4331	NaN	NaN	NaN
freemem	8192.0	NaN	NaN	NaN	1763.456299	2482.104511	55.0
freeswap	8192.0	NaN	NaN	NaN	1328125.959839	422019.426957	2.0
usr	8192.0	NaN	NaN	NaN	83.968872	18.401905	0.0

Observations and Insights:

1. Most runqsz is Not_CPU_Bound.
2. Median is 0 for pgout, ppgout, pgfree, pgscan and atch columns.

Categorial variables in the dataset

In [157…

```
df_lr['runqsz'].value_counts().sort_values() # Frequency of each distinct value in
```

Out[157…

```
runqsz
CPU_Bound        3861
Not_CPU_Bound    4331
Name: count, dtype: int64
```

There are 2 runqsz with Not_CPU_Bound having the maximum count.

Exploratory Data Analysis (EDA)

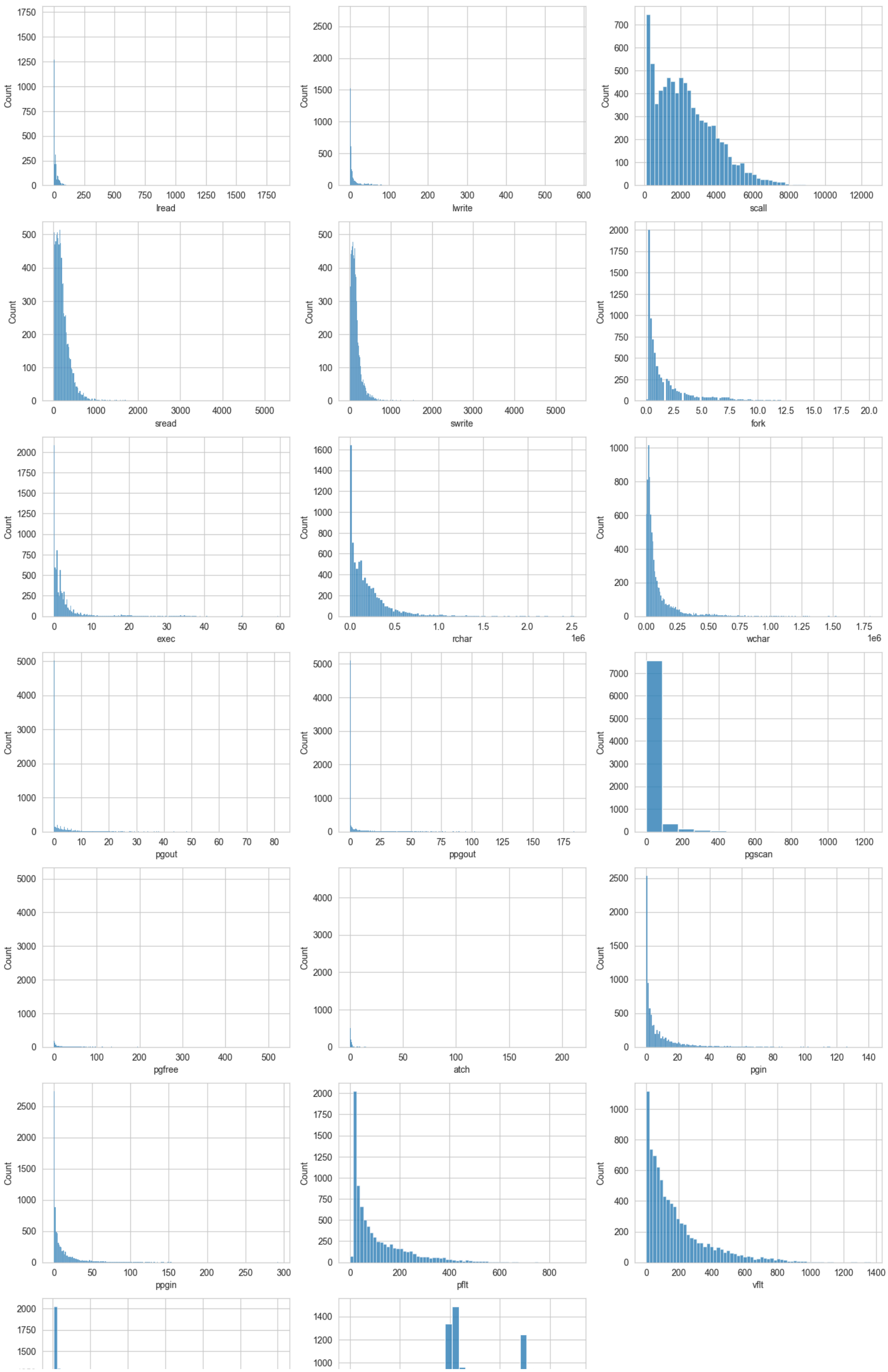
Univariate analysis

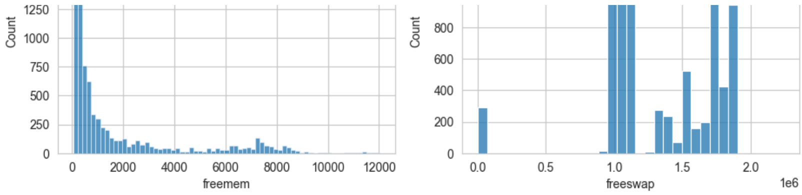
In [158…

```
# Hist Plots for lread, lwrite, scall, sread, swrite, fork, exec, rchar, wchar, pgo
# ppgin, pflt, vflt, freemem, freeswap

fig, axes = plt.subplots(7,3, figsize=(17, 30))

sns.histplot(ax=axes[0, 0], data=df_lr, x='lread')
sns.histplot(ax=axes[0, 1], data=df_lr, x='lwrite')
sns.histplot(ax=axes[0, 2], data=df_lr, x='scall')
sns.histplot(ax=axes[1, 0], data=df_lr, x='sread')
sns.histplot(ax=axes[1, 1], data=df_lr, x='swrite')
sns.histplot(ax=axes[1, 2], data=df_lr, x='fork')
sns.histplot(ax=axes[2, 0], data=df_lr, x='exec')
sns.histplot(ax=axes[2, 1], data=df_lr, x='rchar')
sns.histplot(ax=axes[2, 2], data=df_lr, x='wchar')
sns.histplot(ax=axes[3, 0], data=df_lr, x='pgout')
sns.histplot(ax=axes[3, 1], data=df_lr, x='ppgout')
sns.histplot(ax=axes[3, 2], data=df_lr, x='pgscan')
sns.histplot(ax=axes[4, 0], data=df_lr, x='pgfree')
sns.histplot(ax=axes[4, 1], data=df_lr, x='atch')
sns.histplot(ax=axes[4, 2], data=df_lr, x='pgin')
sns.histplot(ax=axes[5, 0], data=df_lr, x='ppgin')
sns.histplot(ax=axes[5, 1], data=df_lr, x='pflt')
sns.histplot(ax=axes[5, 2], data=df_lr, x='vflt')
sns.histplot(ax=axes[6, 0], data=df_lr, x='freemem')
sns.histplot(ax=axes[6, 1], data=df_lr, x='freeswap')
axes[6,2].axis("off")


axes[0, 0].set(xlabel='lread')
axes[0, 1].set(xlabel='lwrite')
axes[0, 2].set(xlabel='scall')
axes[1, 0].set(xlabel='sread')
axes[1, 1].set(xlabel='swrite')
axes[1, 2].set(xlabel='fork')
axes[2, 0].set(xlabel='exec')
axes[2, 1].set(xlabel='rchar')
axes[2, 2].set(xlabel='wchar')
axes[3, 0].set(xlabel='pgout')
axes[3, 1].set(xlabel='ppgout')
axes[3, 2].set(xlabel='pgscan')
axes[4, 0].set(xlabel='pgfree')
axes[4, 1].set(xlabel='atch')
axes[4, 2].set(xlabel='pgin')
```

```
axes[5, 0].set(xlabel='ppgin')
axes[5, 1].set(xlabel='pflt')
axes[5, 2].set(xlabel='vflt')
axes[6, 0].set(xlabel='freemem')
axes[6, 1].set(xlabel='freeswap')

plt.suptitle('Fig 1: Hist Plots: lread, lwrite, scall, sread, swrite, fork, exec, r

plt.show()
```

Fig 1: Hist Plots: lread, lwrite, scall, sread, swrite, fork, exec, rchar, wchar, pgout, ppgout, pgscan, pgfree, atch, pgin, ppgin, pflt, vflt, freemem, freeswap





Observations and Insights:

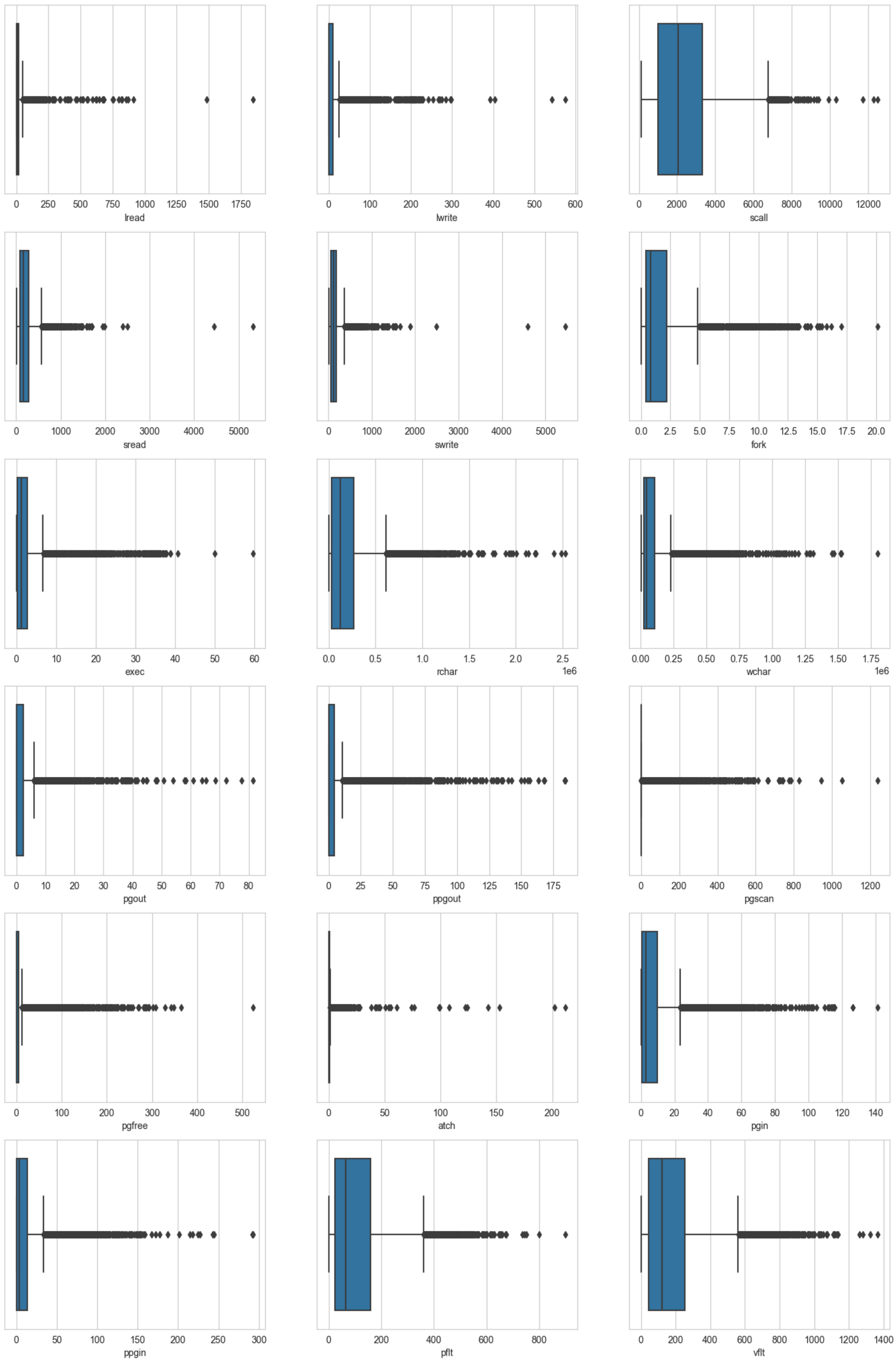
1. No distribution (lread, lwrite, scall, sread, swrite, fork, exec, rchar, wchar, pgout, ppgout, pgscan, pgfree, atch, pgin, ppgin, pflt, vflt, freemem, freeswap) is evenly distributed (symmetric).
2. Except freeswap remaining all distributions are Positively Skewed (mean is more than the mode).

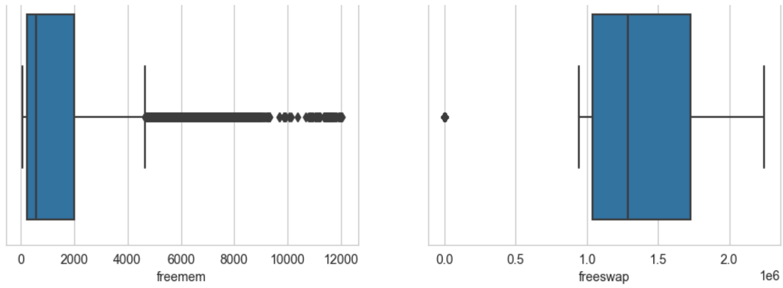
In [159…

```
# Box Plots for lread, lwrite, scall, sread, swrite, fork, exec, rchar, wchar, pgou
# ppgin, pflt, vflt, freemem, freeswap, usr, runqsz_Not_CPU_Bound

fig, axes = plt.subplots(7,3, figsize=(17, 30))

sns.boxplot(ax=axes[0, 0], data=df_lr, x='lread')
sns.boxplot(ax=axes[0, 1], data=df_lr, x='lwrite')
sns.boxplot(ax=axes[0, 2], data=df_lr, x='scall')
sns.boxplot(ax=axes[1, 0], data=df_lr, x='sread')
sns.boxplot(ax=axes[1, 1], data=df_lr, x='swrite')
sns.boxplot(ax=axes[1, 2], data=df_lr, x='fork')
sns.boxplot(ax=axes[2, 0], data=df_lr, x='exec')
sns.boxplot(ax=axes[2, 1], data=df_lr, x='rchar')
sns.boxplot(ax=axes[2, 2], data=df_lr, x='wchar')
sns.boxplot(ax=axes[3, 0], data=df_lr, x='pgout')
sns.boxplot(ax=axes[3, 1], data=df_lr, x='ppgout')
sns.boxplot(ax=axes[3, 2], data=df_lr, x='pgscan')
sns.boxplot(ax=axes[4, 0], data=df_lr, x='pgfree')
sns.boxplot(ax=axes[4, 1], data=df_lr, x='atch')
sns.boxplot(ax=axes[4, 2], data=df_lr, x='pgin')
sns.boxplot(ax=axes[5, 0], data=df_lr, x='ppgin')
sns.boxplot(ax=axes[5, 1], data=df_lr, x='pflt')
sns.boxplot(ax=axes[5, 2], data=df_lr, x='vflt')
sns.boxplot(ax=axes[6, 0], data=df_lr, x='freemem')
sns.boxplot(ax=axes[6, 1], data=df_lr, x='freeswap')
axes[6,2].axis("off")

axes[0, 0].set(xlabel='lread')
axes[0, 1].set(xlabel='lwrite')
axes[0, 2].set(xlabel='scall')
axes[1, 0].set(xlabel='sread')
axes[1, 1].set(xlabel='swrite')
axes[1, 2].set(xlabel='fork')
axes[2, 0].set(xlabel='exec')
axes[2, 1].set(xlabel='rchar')
axes[2, 2].set(xlabel='wchar')
axes[3, 0].set(xlabel='pgout')
axes[3, 1].set(xlabel='ppgout')
axes[3, 2].set(xlabel='pgscan')
```

```
axes[4, 0].set(xlabel='pgfree')
axes[4, 1].set(xlabel='atch')
axes[4, 2].set(xlabel='pgin')
axes[5, 0].set(xlabel='ppgin')
axes[5, 1].set(xlabel='pflt')
axes[5, 2].set(xlabel='vflt')
axes[6, 0].set(xlabel='freemem')
axes[6, 1].set(xlabel='freeswap')

plt.suptitle('Fig 2: Box Plots: lread, lwrite, scall, sread, swrite, fork, exec, rc

plt.show()
```

Fig 2: Box Plots: lread, lwrite, scall, sread, swrite, fork, exec, rchar, wchar, pgout, ppgout, pgscan, pgfree, atch, pgin, ppgin, pflt, vflt, freemem, freeswap





Observations and Insights:

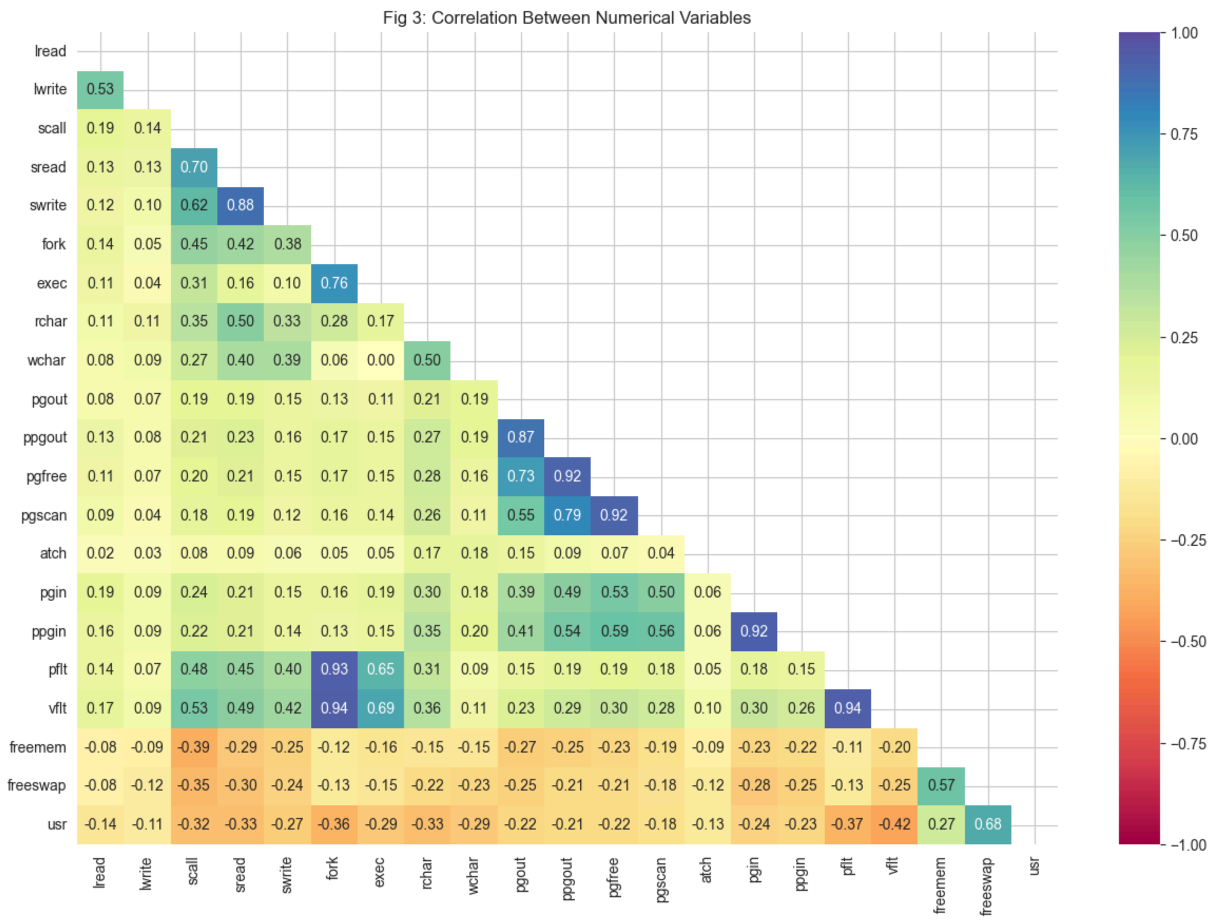
1. All numerical columns are having outliers.

Multivariate Analysis

Correlation Plot

In [160…

```
# Heatmap to plot correlation between all numerical variables in the dataset

df_lr_corr = df_lr.drop('runqsz', axis=1)
corr = df_lr_corr.corr(method='pearson')
mask = np.triu(np.ones_like(corr, dtype=bool))

plt.figure(figsize=(15, 10))
sns.heatmap(df_lr_corr.corr(), annot=True, vmin=-1, vmax=1, fmt=".2f", cmap="Spectr
plt.title('Fig 3: Correlation Between Numerical Variables')
plt.show()
```



Observations and Insights:

1. There is moderate correlation between lread and lwrite.
2. There is moderate correlation between scall and sread.
3. There is moderate correlation between scall and swrite.
4. There is strong correlation between sread and swrite.
5. There is moderate correlation between sread and rchar.
6. There is moderate correlation between rchar and wchar.
7. There is strong correlation between exec and fork.
8. There is moderate correlation between exec and pfit.
9. There is moderate correlation between exec and vfit.
10. There is strong correlation between fork and pfit.
11. There is strong correlation between fork and vfit.
12. There is strong correlation between pgout and ppgout.
13. There is strong correlation between ppgout and pgfree.
14. There is strong correlation between pgfree and pgscan.
15. There is moderate correlation between pgout and pgfree.
16. There is moderate correlation between pgout and pgscan.
17. There is moderate correlation between ppgout and pgscan.
18. There is moderate correlation between ppgout and pgin.
19. There is moderate correlation between pgscan and pgin.

20. There is moderate correlation between pgfree and pgin.
21. There is moderate correlation between ppgout and ppgin.
22. There is moderate correlation between pgscan and ppgin.
23. There is moderate correlation between pgfree and ppgin.
24. There is strong correlation between pgin and ppgin.
25. There is moderate correlation between freemem and freeswap.

Outlier Treatment

In [161…

```
# User Defined Function (UDF) to treat outliers
def treat_outlier(x):
    # taking 5,25,75 percentile of column
    q5=np.percentile(x,5)
    q25=np.percentile(x,25)
    q75=np.percentile(x,75)
    q95=np.percentile(x,95)
    #calculationg IQR range
    IQR=q75-q25
    #Calculating minimum threshold
    lower_bound=q25-(1.5*IQR)
    upper_bound=q75+(1.5*IQR)
    #Capping outliers
    return x.apply(lambda y: upper_bound if y > upper_bound else y).apply(lambda y:
```

In [162…

```
no_outlier = ['runqsz','usr'] # Removing runqsz_Not_CPU_Bound and usr columns
outlier_list = [x for x in df_lr.columns if x not in no_outlier] # Numerical column
```

In [163…

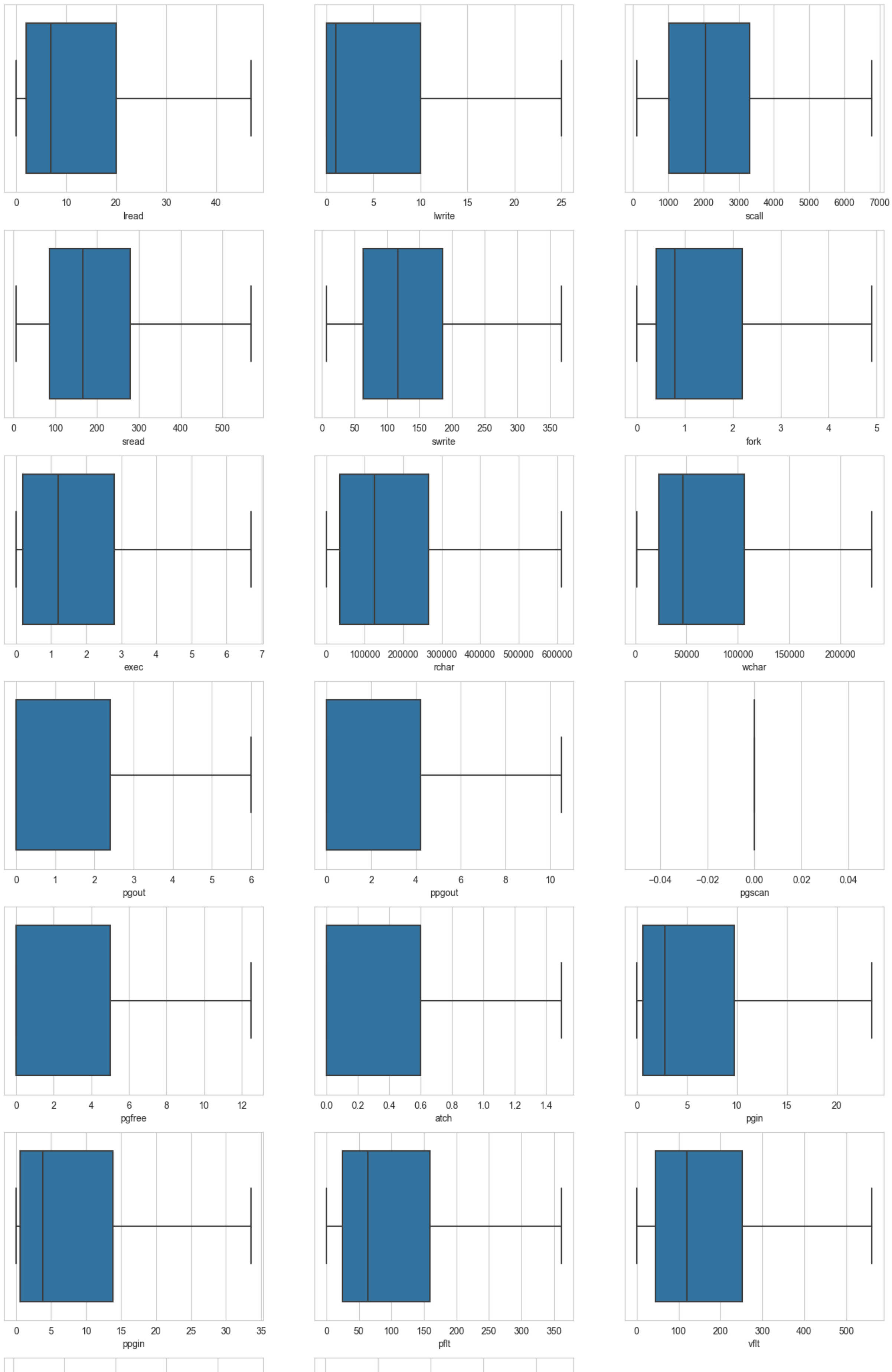
```
# Using for loop to iterate over numerical columns and calling treat_outlier UDF to
for i in df_lr[outlier_list]:
    df_lr[i]=treat_outlier(df_lr[i])
```

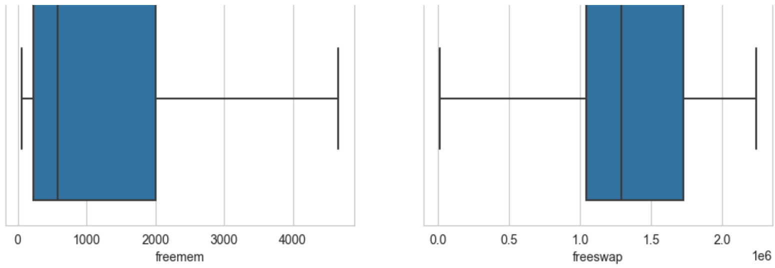
In [164…

```
# Box Plots for lread, lwrite, scall, sread, swrite, fork, exec, rchar, wchar, pgou
# ppgin, pflt, vflt, freemem, freeswap

fig, axes = plt.subplots(7,3, figsize=(17, 30))

sns.boxplot(ax=axes[0, 0], data=df_lr, x='lread')
sns.boxplot(ax=axes[0, 1], data=df_lr, x='lwrite')
sns.boxplot(ax=axes[0, 2], data=df_lr, x='scall')
sns.boxplot(ax=axes[1, 0], data=df_lr, x='sread')
sns.boxplot(ax=axes[1, 1], data=df_lr, x='swrite')
sns.boxplot(ax=axes[1, 2], data=df_lr, x='fork')
sns.boxplot(ax=axes[2, 0], data=df_lr, x='exec')
sns.boxplot(ax=axes[2, 1], data=df_lr, x='rchar')
sns.boxplot(ax=axes[2, 2], data=df_lr, x='wchar')
sns.boxplot(ax=axes[3, 0], data=df_lr, x='pgout')
sns.boxplot(ax=axes[3, 1], data=df_lr, x='ppgout')
sns.boxplot(ax=axes[3, 2], data=df_lr, x='pgscan')
sns.boxplot(ax=axes[4, 0], data=df_lr, x='pgfree')
sns.boxplot(ax=axes[4, 1], data=df_lr, x='atch')
sns.boxplot(ax=axes[4, 2], data=df_lr, x='pgin')
sns.boxplot(ax=axes[5, 0], data=df_lr, x='ppgin')
```

```
sns.boxplot(ax=axes[5, 1], data=df_lr, x='pflt')
sns.boxplot(ax=axes[5, 2], data=df_lr, x='vflt')
sns.boxplot(ax=axes[6, 0], data=df_lr, x='freemem')
sns.boxplot(ax=axes[6, 1], data=df_lr, x='freeswap')
axes[6,2].axis("off")

axes[0, 0].set(xlabel='lread')
axes[0, 1].set(xlabel='lwrite')
axes[0, 2].set(xlabel='scall')
axes[1, 0].set(xlabel='sread')
axes[1, 1].set(xlabel='swrite')
axes[1, 2].set(xlabel='fork')
axes[2, 0].set(xlabel='exec')
axes[2, 1].set(xlabel='rchar')
axes[2, 2].set(xlabel='wchar')
axes[3, 0].set(xlabel='pgout')
axes[3, 1].set(xlabel='ppgout')
axes[3, 2].set(xlabel='pgscan')
axes[4, 0].set(xlabel='pgfree')
axes[4, 1].set(xlabel='atch')
axes[4, 2].set(xlabel='pgin')
axes[5, 0].set(xlabel='ppgin')
axes[5, 1].set(xlabel='pflt')
axes[5, 2].set(xlabel='vflt')
axes[6, 0].set(xlabel='freemem')
axes[6, 1].set(xlabel='freeswap')

plt.suptitle('Fig 4: Box Plots: lread, lwrite, scall, sread, swrite, fork, exec, rc

plt.show()
```

Fig 4: Box Plots: lread, lwrite, scall, sread, swrite, fork, exec, rchar, wchar, pgout, ppgout, pgscan, pgfree, atch, pgin, ppgin, pflt, vflt, freemem, freeswap (after outliers treatment)





We can observe from above Box Plots that there are no outliers in the numerical columns (to be used for Linear Regression) after the treatment.

Feature Encoding

In [165…
```
df_lr = pd.get_dummies(df_lr, columns=['runqsz'],drop_first=True,dtype='int') # Enc
```

In [166…
```
print('runqsz_Not_CPU_Bound data type:',df_lr['runqsz_Not_CPU_Bound'].dtypes)
```

```
runqsz_Not_CPU_Bound data type: int32
```

In [167…
```
df_lr['runqsz_Not_CPU_Bound'].value_counts().sort_values()
```

Out[167…
```
runqsz_Not_CPU_Bound
0    3861
1    4331
Name: count, dtype: int64
```

0 - CPU_Bound, 1 - Not_CPU_Bound

Train-Test Split

In [168…
```
# Copy all the predictor variables into X dataframe
X = df_lr.drop('usr', axis=1)

# Copy target into the y dataframe.
y = df_lr[['usr']]
```

In [169…
```
# let's add the intercept to data
X = sm.add_constant(X)
```

Split X and y into train and test sets in a 70:30 ratio.

In [170…
```
# Split X and y into training and test set in 70:30 ratio
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30 , random_s
```

In [171…
```
X_train.head()
```

Out[171…

	const	lread	lwrite	scall	sread	swrite	fork	exec	rchar	wchar	...	pg
694	1.0	1.0	1.0	1345.0	223.0	192.0	0.6	0.6	198703.0	230625.875	...	1
5535	1.0	1.0	1.0	1429.0	87.0	67.0	0.2	0.2	7163.0	24842.000	...	[illegible]
4244	1.0	47.0	25.0	3273.0	225.0	180.0	0.6	0.4	83246.0	53705.000	...	[illegible]
2472	1.0	13.0	8.0	4349.0	300.0	191.0	2.8	3.0	96009.0	70467.000	...	[illegible]
7052	1.0	17.0	23.0	225.0	13.0	13.0	0.4	1.6	17132.0	12514.000	...	[illegible]

5 rows × 22 columns

In [172…

```
X_test.head()
```

Out[172…

	const	lread	lwrite	scall	sread	swrite	fork	exec	rchar	wchar	...	pg
3894	1.0	27.0	25.0	1252.0	53.0	118.0	0.2	0.2	26592.0	54394.000	...	
4276	1.0	1.0	0.0	996.0	85.0	55.0	0.4	0.4	16667.0	36431.000	...	
3414	1.0	9.0	7.0	1530.0	247.0	135.0	0.4	0.4	14513.0	61905.000	...	
4165	1.0	32.0	4.0	3243.0	182.0	140.0	4.9	5.6	337517.0	94832.000	...	
7385	1.0	16.0	3.0	5017.0	259.0	249.0	2.8	1.4	73537.0	230625.875	...	

5 rows × 22 columns

Linear Regression Model

In [173…

```
olsmod = sm.OLS(y_train, X_train)
olsres = olsmod.fit()
```

In [174…

```
# let's print the regression summary
print(olsres.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    usr   R-squared:                       0.620
Model:                            OLS   Adj. R-squared:                  0.619
Method:                 Least Squares   F-statistic:                     465.8
Date:                Sat, 16 Dec 2023   Prob (F-statistic):               0.00
Time:                        22:44:06   Log-Likelihood:                -21966.
No. Observations:                5734   AIC:                         4.397e+04
Df Residuals:                    5713   BIC:                         4.411e+04
Df Model:                          20
Covariance Type:            nonrobust
==================================================================================
====
                     coef    std err          t      P>|t|      [0.025      0.
975]
----------------------------------------------------------------------------------
----
const             45.7600      0.798     57.348      0.000      44.196      4
7.324
lread             -0.0750      0.023     -3.312      0.001      -0.119      -
0.031
lwrite             0.0396      0.033      1.197      0.232      -0.025
0.105
scall              0.0011      0.000      6.927      0.000       0.001
0.001
sread              0.0010      0.003      0.386      0.699      -0.004
0.006
swrite            -0.0054      0.004     -1.480      0.139      -0.012
0.002
fork              -0.9027      0.333     -2.713      0.007      -1.555      -
0.250
exec              -0.0537      0.130     -0.412      0.681      -0.309
0.202
rchar          -1.038e-05   1.23e-06     -8.439      0.000   -1.28e-05   -7.97
e-06
wchar          -6.736e-06   2.61e-06     -2.584      0.010   -1.18e-05   -1.63
e-06
pgout             -0.7158      0.227     -3.151      0.002      -1.161      -
0.270
ppgout            -0.0178      0.199     -0.089      0.929      -0.407
0.372
pgfree             0.1048      0.121      0.870      0.384      -0.131
0.341
pgscan          2.508e-14   6.04e-16     41.493      0.000    2.39e-14    2.63
e-14
atch               1.0793      0.361      2.994      0.003       0.373
1.786
pgin               0.2987      0.072      4.161      0.000       0.158
0.439
ppgin             -0.1704      0.050     -3.423      0.001      -0.268      -
0.073
pflt              -0.0548      0.005    -10.958      0.000      -0.065      -
0.045
vflt               0.0155      0.004      4.293      0.000       0.008
0.023
freemem           -0.0024      0.000    -18.443      0.000      -0.003      -
```

```
0.002
freeswap               3.266e-05    4.8e-07     68.079      0.000    3.17e-05    3.36
e-05
runqsz_Not_CPU_Bound      7.0117      0.318     22.046      0.000       6.388
7.635
==============================================================================
Omnibus:                     1424.388   Durbin-Watson:                   2.061
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4185.515
Skew:                          -1.287   Prob(JB):                         0.00
Kurtosis:                       6.301   Cond. No.                     2.92e+22
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
[2] The smallest eigenvalue is 1.34e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

The R-squared value tells us that our model can explain 62% of the variance in the training set.

Check for Multicollinearity

In [175…

```
# Check the VIF of the predictors

vif_series1 = pd.Series(
    [variance_inflation_factor(X_train.values, i) for i in range(X_train.shape[1])]
    index=X_train.columns,
)
print("VIF values: \n\n{}\n".format(vif_series1))
```

```
VIF values:

const                   29.229332
lread                    5.350560
lwrite                   4.328397
scall                    2.960609
sread                    6.420172
swrite                   5.597135
fork                    13.035359
exec                     3.241417
rchar                    2.133616
wchar                    1.584381
pgout                   11.360363
ppgout                  29.404223
pgfree                  16.496748
pgscan                        NaN
atch                     1.875901
pgin                    13.809339
ppgin                   13.951855
pflt                    12.001460
vflt                    15.971049
freemem                  1.961304
freeswap                 1.841239
runqsz_Not_CPU_Bound     1.156815
dtype: float64
```

The VIF values indicate that the features lread, lwrite, scall, sread, swrite, fork, exec, rchar, pgout, ppgout, pgfree, pgin, ppgin, pflt and vflt are correlated with one or more independent features.

Remove/drop multicollinear columns one by one and observe the effect on predictive model

In [176…

```
X_train1 = X_train.drop(["lread"], axis=1)
olsmod_1 = sm.OLS(y_train, X_train1)
olsres_1 = olsmod_1.fit()
print(
    "R-squared:",
    np.round(olsres_1.rsquared, 3),
    "\nAdjusted R-squared:",
    np.round(olsres_1.rsquared_adj, 3),
)
```

```
R-squared: 0.619
Adjusted R-squared: 0.618
```

- On dropping 'lread', adj. R-squared decreased by 0.001

In [177…

```
X_train2 = X_train.drop(["lwrite"], axis=1)
olsmod_2= sm.OLS(y_train, X_train2)
olsres_2 = olsmod_2.fit()
print(
    "R-squared:",
```

```
    np.round(olsres_2.rsquared, 3),
    "\nAdjusted R-squared:",
    np.round(olsres_2.rsquared_adj, 3),
)
```

```
R-squared: 0.62
Adjusted R-squared: 0.619
```

- On dropping 'lwrite', adj. R-squared do not decrease

In [178…

```
X_train3 = X_train.drop(["scall"], axis=1)
olsmod_3 = sm.OLS(y_train, X_train3)
olsres_3 = olsmod_3.fit()
print(
    "R-squared:",
    np.round(olsres_3.rsquared, 3),
    "\nAdjusted R-squared:",
    np.round(olsres_3.rsquared_adj, 3),
)
```

```
R-squared: 0.617
Adjusted R-squared: 0.615
```

- On dropping 'scall', adj. R-squared decrease by 0.004

In [179…

```
X_train4 = X_train.drop(["sread"], axis=1)
olsmod_4 = sm.OLS(y_train, X_train4)
olsres_4 = olsmod_4.fit()
print(
    "R-squared:",
    np.round(olsres_4.rsquared, 3),
    "\nAdjusted R-squared:",
    np.round(olsres_4.rsquared_adj, 3),
)
```

```
R-squared: 0.62
Adjusted R-squared: 0.619
```

- On dropping 'sread', adj. R-squared do not decrease

In [180…

```
X_train5 = X_train.drop(["swrite"], axis=1)
olsmod_5 = sm.OLS(y_train, X_train5)
olsres_5 = olsmod_5.fit()
print(
    "R-squared:",
    np.round(olsres_5.rsquared, 3),
    "\nAdjusted R-squared:",
    np.round(olsres_5.rsquared_adj, 3),
)
```

```
R-squared: 0.62
Adjusted R-squared: 0.618
```

- On dropping 'swrite', adj. R-squared decrease by 0.001

In [181...

```
X_train6 = X_train.drop(["fork"], axis=1)
olsmod_6 = sm.OLS(y_train, X_train6)
olsres_6 = olsmod_6.fit()
print(
    "R-squared:",
    np.round(olsres_6.rsquared, 3),
    "\nAdjusted R-squared:",
    np.round(olsres_6.rsquared_adj, 3),
)
```

```
R-squared: 0.619
Adjusted R-squared: 0.618
```

- On dropping 'fork', adj. R-squared decrease by 0.001

In [182...

```
X_train7 = X_train.drop(["exec"], axis=1)
olsmod_7 = sm.OLS(y_train, X_train7)
olsres_7 = olsmod_7.fit()
print(
    "R-squared:",
    np.round(olsres_7.rsquared, 3),
    "\nAdjusted R-squared:",
    np.round(olsres_7.rsquared_adj, 3),
)
```

```
R-squared: 0.62
Adjusted R-squared: 0.619
```

- On dropping 'exec', adj. R-squared do not decrease

In [183...

```
X_train8 = X_train.drop(["rchar"], axis=1)
olsmod_8 = sm.OLS(y_train, X_train8)
olsres_8 = olsmod_8.fit()
print(
    "R-squared:",
    np.round(olsres_8.rsquared, 3),
    "\nAdjusted R-squared:",
    np.round(olsres_8.rsquared_adj, 3),
)
```

```
R-squared: 0.615
Adjusted R-squared: 0.614
```

- On dropping 'rchar', adj. R-squared decrease by 0.005

In [184...

```
X_train9 = X_train.drop(["pgout"], axis=1)
olsmod_9 = sm.OLS(y_train, X_train9)
olsres_9 = olsmod_9.fit()
print(
    "R-squared:",
```

```
    np.round(olsres_9.rsquared, 3),
    "\nAdjusted R-squared:",
    np.round(olsres_9.rsquared_adj, 3),
)
```

```
R-squared: 0.619
Adjusted R-squared: 0.618
```

- On dropping 'pgout', adj. R-squared decrease by 0.001

In [185…

```
X_train10 = X_train.drop(["ppgout"], axis=1)
olsmod_10 = sm.OLS(y_train, X_train10)
olsres_10 = olsmod_10.fit()
print(
    "R-squared:",
    np.round(olsres_10.rsquared, 3),
    "\nAdjusted R-squared:",
    np.round(olsres_10.rsquared_adj, 3),
)
```

```
R-squared: 0.62
Adjusted R-squared: 0.619
```

- On dropping 'ppgout', adj. R-squared do not decrease

In [186…

```
X_train11 = X_train.drop(["pgfree"], axis=1)
olsmod_11 = sm.OLS(y_train, X_train11)
olsres_11 = olsmod_11.fit()
print(
    "R-squared:",
    np.round(olsres_11.rsquared, 3),
    "\nAdjusted R-squared:",
    np.round(olsres_11.rsquared_adj, 3),
)
```

```
R-squared: 0.62
Adjusted R-squared: 0.619
```

- On dropping 'pgfree', adj. R-squared do not decrease

In [187…

```
X_train12 = X_train.drop(["pgin"], axis=1)
olsmod_12 = sm.OLS(y_train, X_train12)
olsres_12 = olsmod_12.fit()
print(
    "R-squared:",
    np.round(olsres_12.rsquared, 3),
    "\nAdjusted R-squared:",
    np.round(olsres_12.rsquared_adj, 3),
)
```

```
R-squared: 0.619
Adjusted R-squared: 0.617
```

- On dropping 'pgin', adj. R-squared decrease by 0.002

In [188…

```
X_train13 = X_train.drop(["ppgin"], axis=1)
olsmod_13 = sm.OLS(y_train, X_train13)
olsres_13 = olsmod_13.fit()
print(
    "R-squared:",
    np.round(olsres_13.rsquared, 3),
    "\nAdjusted R-squared:",
    np.round(olsres_13.rsquared_adj, 3),
)
```

```
R-squared: 0.619
Adjusted R-squared: 0.618
```

- On dropping 'ppgin', adj. R-squared decrease by 0.001

In [189…

```
X_train14 = X_train.drop(["pflt"], axis=1)
olsmod_14 = sm.OLS(y_train, X_train14)
olsres_14 = olsmod_14.fit()
print(
    "R-squared:",
    np.round(olsres_14.rsquared, 3),
    "\nAdjusted R-squared:",
    np.round(olsres_14.rsquared_adj, 3),
)
```

```
R-squared: 0.612
Adjusted R-squared: 0.611
```

- On dropping 'pflt', adj. R-squared decrease by 0.008

In [190…

```
X_train15 = X_train.drop(["vflt"], axis=1)
olsmod_15 = sm.OLS(y_train, X_train15)
olsres_15 = olsmod_15.fit()
print(
    "R-squared:",
    np.round(olsres_15.rsquared, 3),
    "\nAdjusted R-squared:",
    np.round(olsres_15.rsquared_adj, 3),
)
```

```
R-squared: 0.619
Adjusted R-squared: 0.617
```

- On dropping 'vflt', adj. R-squared decrease by 0.002

Since there is no effect on adj. R-squared after dropping the 'ppgout' column, we can remove it from the training set.

In [191…

```
X_train = X_train.drop(["ppgout"], axis=1)
```

In [192…

```
olsmod_16 = sm.OLS(y_train, X_train)
olsres_16 = olsmod_16.fit()
print(olsres_16.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    usr   R-squared:                       0.620
Model:                            OLS   Adj. R-squared:                  0.619
Method:                 Least Squares   F-statistic:                     490.4
Date:                Sat, 16 Dec 2023   Prob (F-statistic):               0.00
Time:                        22:44:21   Log-Likelihood:                -21966.
No. Observations:                5734   AIC:                         4.397e+04
Df Residuals:                    5714   BIC:                         4.411e+04
Df Model:                          19
Covariance Type:            nonrobust
=====================================================================================
====
                           coef    std err          t      P>|t|      [0.025      0.
975]
-------------------------------------------------------------------------------------
----
const                   45.7660      0.795     57.565      0.000      44.207       4
7.325
lread                   -0.0751      0.023     -3.312      0.001      -0.119       -
0.031
lwrite                   0.0396      0.033      1.197      0.231      -0.025
0.105
scall                    0.0011      0.000      6.927      0.000       0.001
0.001
sread                    0.0010      0.003      0.386      0.700      -0.004
0.006
swrite                  -0.0054      0.004     -1.480      0.139      -0.012
0.002
fork                    -0.9020      0.333     -2.712      0.007      -1.554       -
0.250
exec                    -0.0540      0.130     -0.414      0.679      -0.309
0.201
rchar                -1.038e-05   1.23e-06     -8.440      0.000   -1.28e-05   -7.97
e-06
wchar                -6.747e-06    2.6e-06     -2.591      0.010   -1.19e-05   -1.64
e-06
pgout                   -0.7291      0.171     -4.259      0.000      -1.065       -
0.393
pgfree                   0.0963      0.074      1.306      0.191      -0.048
0.241
pgscan               -2.577e-14   5.97e-16    -43.197      0.000   -2.69e-14   -2.46
e-14
atch                     1.0798      0.360      2.995      0.003       0.373
1.786
pgin                     0.2989      0.072      4.169      0.000       0.158
0.440
ppgin                   -0.1706      0.050     -3.435      0.001      -0.268       -
0.073
pflt                    -0.0548      0.005    -10.959      0.000      -0.065       -
0.045
vflt                     0.0155      0.004      4.293      0.000       0.008
0.023
freemem                 -0.0024      0.000    -18.457      0.000      -0.003       -
0.002
freeswap              3.266e-05   4.79e-07     68.138      0.000    3.17e-05    3.36
```

```
e-05
runqsz_Not_CPU_Bound     7.0112      0.318     22.050      0.000       6.388
7.635
==============================================================================
Omnibus:                     1424.268   Durbin-Watson:                   2.061
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4184.829
Skew:                          -1.287   Prob(JB):                         0.00
Kurtosis:                       6.301   Cond. No.                     1.14e+22
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
[2] The smallest eigenvalue is 8.8e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

In [193…

```
vif_series2 = pd.Series(
    [variance_inflation_factor(X_train.values, i) for i in range(X_train.shape[1])]
    index=X_train.columns,
)
print("VIF values: \n\n{}\n".format(vif_series2))
```

```
VIF values:

const                   29.021961
lread                    5.350387
lwrite                   4.328325
scall                    2.960379
sread                    6.420135
swrite                   5.597025
fork                    13.027305
exec                     3.239231
rchar                    2.133614
wchar                    1.580894
pgout                    6.453978
pgfree                   6.172847
pgscan                        NaN
atch                     1.875553
pgin                    13.784007
ppgin                   13.898848
pflt                    12.001460
vflt                    15.966865
freemem                  1.959267
freeswap                 1.838167
runqsz_Not_CPU_Bound     1.156421
dtype: float64
```

Since there is no effect on adj. R-squared after dropping the 'vflt' column, we can remove it from the training set.

In [194…

```
X_train = X_train.drop(["vflt"], axis=1)
```

In [195…

```
olsmod_17 = sm.OLS(y_train, X_train)
olsres_17 = olsmod_17.fit()
```

```
print(olsres_17.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    usr   R-squared:                       0.619
Model:                            OLS   Adj. R-squared:                  0.617
Method:                 Least Squares   F-statistic:                     515.1
Date:                Sat, 16 Dec 2023   Prob (F-statistic):               0.00
Time:                        22:44:22   Log-Likelihood:                -21975.
No. Observations:                5734   AIC:                         4.399e+04
Df Residuals:                    5715   BIC:                         4.412e+04
Df Model:                          18
Covariance Type:            nonrobust
==================================================================================
====
                          coef    std err          t      P>|t|      [0.025      0.
975]
----------------------------------------------------------------------------------
----
const                  46.1566      0.791     58.352      0.000      44.606      4
7.707
lread                  -0.0699      0.023     -3.085      0.002      -0.114      -
0.025
lwrite                  0.0373      0.033      1.124      0.261      -0.028
0.102
scall                   0.0011      0.000      7.140      0.000       0.001
0.001
sread                   0.0019      0.003      0.749      0.454      -0.003
0.007
swrite                 -0.0056      0.004     -1.542      0.123      -0.013
0.002
fork                   -0.2240      0.293     -0.764      0.445      -0.799
0.351
exec                   -0.0347      0.130     -0.266      0.790      -0.290
0.221
rchar               -1.003e-05   1.23e-06     -8.155      0.000   -1.24e-05   -7.62
e-06
wchar               -8.064e-06   2.59e-06     -3.114      0.002   -1.31e-05   -2.99
e-06
pgout                  -0.7456      0.171     -4.350      0.000      -1.082      -
0.410
pgfree                  0.1159      0.074      1.573      0.116      -0.029
0.260
pgscan              -9.595e-15   2.44e-16    -39.299      0.000   -1.01e-14   -9.12
e-15
atch                    1.1999      0.360      3.334      0.001       0.494
1.905
pgin                    0.3343      0.071      4.686      0.000       0.194
0.474
ppgin                  -0.1707      0.050     -3.432      0.001      -0.268      -
0.073
pflt                   -0.0443      0.004    -10.142      0.000      -0.053      -
0.036
freemem                -0.0023      0.000    -18.325      0.000      -0.003      -
0.002
freeswap             3.231e-05   4.73e-07     68.268      0.000    3.14e-05    3.32
e-05
runqsz_Not_CPU_Bound    7.0208      0.318     22.047      0.000       6.397
```

```
7.645
==============================================================================
Omnibus:                     1460.026   Durbin-Watson:                   2.062
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4349.996
Skew:                          -1.314   Prob(JB):                         0.00
Kurtosis:                       6.362   Cond. No.                     2.68e+22
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
[2] The smallest eigenvalue is 1.59e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

In [196…

```
vif_series3 = pd.Series(
    [variance_inflation_factor(X_train.values, i) for i in range(X_train.shape[1])]
    index=X_train.columns,
)
print("VIF values: \n\n{}\n".format(vif_series3))
```

```
VIF values: 

const                  28.641818
lread                   5.335455
lwrite                  4.327130
scall                   2.952947
sread                   6.374687
swrite                  5.595777
fork                   10.089700
exec                    3.235396
rchar                   2.123783
wchar                   1.558923
pgout                   6.450724
pgfree                  6.149223
pgscan                       NaN
atch                    1.864254
pgin                   13.602134
ppgin                  13.898845
pflt                    9.131802
freemem                 1.957966
freeswap                1.787695
runqsz_Not_CPU_Bound    1.156363
dtype: float64
```

Since there is no effect on adj. R-squared after dropping the 'ppgin' column, we can remove it from the training set.

In [197…

```
X_train = X_train.drop(["ppgin"], axis=1)
```

In [198…

```
olsmod_18 = sm.OLS(y_train, X_train)
olsres_18 = olsmod_18.fit()
print(olsres_18.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    usr   R-squared:                       0.618
Model:                            OLS   Adj. R-squared:                  0.617
Method:                 Least Squares   F-statistic:                     543.6
Date:                Sat, 16 Dec 2023   Prob (F-statistic):               0.00
Time:                        22:44:25   Log-Likelihood:                -21981.
No. Observations:                5734   AIC:                         4.400e+04
Df Residuals:                    5716   BIC:                         4.412e+04
Df Model:                          17
Covariance Type:            nonrobust
==================================================================================
====
                          coef    std err          t      P>|t|      [0.025      0.
975]
----------------------------------------------------------------------------------
----
const                  46.2665      0.791     58.483      0.000      44.716      4
7.817
lread                  -0.0759      0.023     -3.355      0.001      -0.120      -
0.032
lwrite                  0.0430      0.033      1.296      0.195      -0.022
0.108
scall                   0.0011      0.000      7.202      0.000       0.001
0.001
sread                   0.0019      0.003      0.763      0.445      -0.003
0.007
swrite                 -0.0056      0.004     -1.556      0.120      -0.013
0.001
fork                   -0.1854      0.293     -0.632      0.527      -0.760
0.389
exec                   -0.0340      0.131     -0.260      0.795      -0.290
0.222
rchar               -1.055e-05   1.22e-06     -8.645      0.000   -1.29e-05   -8.16
e-06
wchar               -8.054e-06   2.59e-06     -3.107      0.002   -1.31e-05   -2.97
e-06
pgout                  -0.7288      0.172     -4.249      0.000      -1.065      -
0.393
pgfree                  0.0918      0.073      1.251      0.211      -0.052
0.236
pgscan               6.226e-15   1.45e-15      4.306      0.000    3.39e-15    9.06
e-15
atch                    1.2241      0.360      3.398      0.001       0.518
1.930
pgin                    0.1036      0.024      4.327      0.000       0.057
0.151
pflt                   -0.0444      0.004    -10.150      0.000      -0.053      -
0.036
freemem                -0.0024      0.000    -18.348      0.000      -0.003      -
0.002
freeswap             3.226e-05   4.73e-07     68.124      0.000    3.13e-05    3.32
e-05
runqsz_Not_CPU_Bound    7.0012      0.319     21.968      0.000       6.376
7.626
==============================================================================
```

```
Omnibus:                     1461.106   Durbin-Watson:                   2.059
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4348.873
Skew:                          -1.315   Prob(JB):                         0.00
Kurtosis:                       6.359   Cond. No.                     3.11e+22
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
[2] The smallest eigenvalue is 1.18e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

In [199…

```
vif_series4 = pd.Series(
    [variance_inflation_factor(X_train.values, i) for i in range(X_train.shape[1])]
    index=X_train.columns,
)
print("VIF values: \n\n{}\n".format(vif_series4))
```

```
VIF values: 

const                  28.594882
lread                   5.304009
lwrite                  4.316362
scall                   2.951826
sread                   6.374556
swrite                  5.595670
fork                   10.074886
exec                    3.235387
rchar                   2.090401
wchar                   1.558921
pgout                   6.445478
pgfree                  6.093623
pgscan                       NaN
atch                    1.863536
pgin                    1.529142
pflt                    9.131545
freemem                 1.957713
freeswap                1.785393
runqsz_Not_CPU_Bound    1.155990
dtype: float64
```

Since there is no effect on adj. R-squared after dropping the 'fork' column, we can remove it from the training set.

In [200…

```
X_train = X_train.drop(["fork"], axis=1)
```

In [201…

```
olsmod_19 = sm.OLS(y_train, X_train)
olsres_19 = olsmod_19.fit()
print(olsres_19.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                    usr   R-squared:                       0.618
Model:                            OLS   Adj. R-squared:                  0.617
Method:                 Least Squares   F-statistic:                     577.6
Date:                Sat, 16 Dec 2023   Prob (F-statistic):               0.00
Time:                        22:44:28   Log-Likelihood:                -21982.
No. Observations:                5734   AIC:                         4.400e+04
Df Residuals:                    5717   BIC:                         4.411e+04
Df Model:                          16                                         
Covariance Type:            nonrobust                                         
====================================================================================
====
                           coef    std err          t      P>|t|      [0.025      0.
975]
------------------------------------------------------------------------------------
----
const                   46.3033      0.789     58.692      0.000      44.757      4
7.850
lread                   -0.0767      0.023     -3.399      0.001      -0.121      -
0.032
lwrite                   0.0443      0.033      1.340      0.180      -0.021       
0.109
scall                    0.0012      0.000      7.319      0.000       0.001       
0.001
sread                    0.0020      0.003      0.772      0.440      -0.003       
0.007
swrite                  -0.0061      0.004     -1.708      0.088      -0.013       
0.001
exec                    -0.0622      0.123     -0.507      0.612      -0.303       
0.178
rchar                -1.057e-05   1.22e-06     -8.662      0.000    -1.3e-05   -8.18
e-06
wchar                -7.934e-06   2.58e-06     -3.070      0.002    -1.3e-05   -2.87
e-06
pgout                   -0.7284      0.172     -4.247      0.000      -1.065      -
0.392
pgfree                   0.0914      0.073      1.245      0.213      -0.053       
0.235
pgscan               -1.532e-14   2.27e-16    -67.378      0.000   -1.58e-14   -1.49
e-14
atch                     1.2293      0.360      3.414      0.001       0.523       
1.935
pgin                     0.1042      0.024      4.356      0.000       0.057       
0.151
pflt                    -0.0466      0.003    -17.305      0.000      -0.052      -
0.041
freemem                 -0.0024      0.000    -18.361      0.000      -0.003      -
0.002
freeswap              3.224e-05   4.73e-07     68.153      0.000    3.13e-05    3.32
e-05
runqsz_Not_CPU_Bound     6.9968      0.319     21.961      0.000       6.372       
7.621
==============================================================================
Omnibus:                     1460.051   Durbin-Watson:                   2.059
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4348.704
```

```
Skew:                          -1.314   Prob(JB):                        0.00
Kurtosis:                       6.361   Cond. No.                    4.18e+21
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
[2] The smallest eigenvalue is 6.54e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

In [202…

```
vif_series5 = pd.Series(
    [variance_inflation_factor(X_train.values, i) for i in range(X_train.shape[1])]
    index=X_train.columns,
)
print("VIF values: \n\n{}\n".format(vif_series5))
```

```
VIF values: 

const                  28.440419
lread                   5.285069
lwrite                  4.298019
scall                   2.914853
sread                   6.373458
swrite                  5.390263
exec                    2.856973
rchar                   2.089364
wchar                   1.550686
pgout                   6.445377
pgfree                  6.093041
pgscan                       NaN
atch                    1.862553
pgin                    1.526800
pflt                    3.458168
freemem                 1.957226
freeswap                1.782829
runqsz_Not_CPU_Bound    1.155448
dtype: float64
```

Since there is no effect on adj. R-squared after dropping the 'sread' column, we can remove it from the training set.

In [203…

```
X_train = X_train.drop(["sread"], axis=1)
```

In [204…

```
olsmod_20 = sm.OLS(y_train, X_train)
olsres_20 = olsmod_20.fit()
print(olsres_20.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    usr   R-squared:                       0.618
Model:                            OLS   Adj. R-squared:                  0.617
Method:                 Least Squares   F-statistic:                     616.2
Date:                Sat, 16 Dec 2023   Prob (F-statistic):               0.00
Time:                        22:44:31   Log-Likelihood:                -21982.
No. Observations:                5734   AIC:                         4.400e+04
Df Residuals:                    5718   BIC:                         4.410e+04
Df Model:                          15
Covariance Type:            nonrobust
====================================================================================
====
                           coef    std err          t      P>|t|      [0.025      0.
975]
------------------------------------------------------------------------------------
----
const                   46.3342      0.788     58.809      0.000      44.790      4
7.879
lread                   -0.0774      0.023     -3.431      0.001      -0.122      -
0.033
lwrite                   0.0455      0.033      1.377      0.168      -0.019
0.110
scall                    0.0012      0.000      7.906      0.000       0.001
0.001
swrite                  -0.0043      0.003     -1.599      0.110      -0.009
0.001
exec                    -0.0668      0.122     -0.545      0.586      -0.307
0.173
rchar                -1.015e-05   1.09e-06     -9.295      0.000   -1.23e-05   -8.01
e-06
wchar                -8.119e-06   2.57e-06     -3.155      0.002   -1.32e-05   -3.07
e-06
pgout                   -0.7270      0.171     -4.239      0.000      -1.063      -
0.391
pgfree                   0.0920      0.073      1.253      0.210      -0.052
0.236
pgscan                1.416e-14   2.74e-16     51.613      0.000    1.36e-14    1.47
e-14
atch                     1.2221      0.360      3.395      0.001       0.516
1.928
pgin                     0.1037      0.024      4.337      0.000       0.057
0.151
pflt                    -0.0464      0.003    -17.299      0.000      -0.052      -
0.041
freemem                 -0.0024      0.000    -18.351      0.000      -0.003      -
0.002
freeswap              3.221e-05   4.71e-07     68.351      0.000    3.13e-05    3.31
e-05
runqsz_Not_CPU_Bound     6.9962      0.319     21.960      0.000       6.372
7.621
==============================================================================
Omnibus:                     1461.798   Durbin-Watson:                   2.058
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4358.573
Skew:                          -1.315   Prob(JB):                         0.00
Kurtosis:                       6.366   Cond. No.                     3.42e+22
```

```
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
[2] The smallest eigenvalue is 9.74e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

In [205…

```
vif_series6 = pd.Series(
    [variance_inflation_factor(X_train.values, i) for i in range(X_train.shape[1])]
    index=X_train.columns,
)
print("VIF values: \n\n{}\n".format(vif_series6))
```

```
VIF values: 

const                   28.366808
lread                    5.277543
lwrite                   4.288733
scall                    2.657189
swrite                   3.013887
exec                     2.850220
rchar                    1.673113
wchar                    1.537416
pgout                    6.444663
pgfree                   6.092363
pgscan                        NaN
atch                     1.861273
pgin                     1.525797
pflt                     3.436271
freemem                  1.956658
freeswap                 1.769115
runqsz_Not_CPU_Bound     1.155441
dtype: float64
```

Since there is no effect on adj. R-squared after dropping the 'lread' column, we can remove it from the training set.

In [206…

```
X_train = X_train.drop(["lread"], axis=1)
```

In [207…

```
olsmod_21 = sm.OLS(y_train, X_train)
olsres_21 = olsmod_21.fit()
print(olsres_21.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    usr   R-squared:                       0.617
Model:                            OLS   Adj. R-squared:                  0.616
Method:                 Least Squares   F-statistic:                     658.1
Date:                Sat, 16 Dec 2023   Prob (F-statistic):               0.00
Time:                        22:44:34   Log-Likelihood:                -21988.
No. Observations:                5734   AIC:                         4.401e+04
Df Residuals:                    5719   BIC:                         4.411e+04
Df Model:                          14
Covariance Type:            nonrobust
==================================================================================
====
                         coef    std err          t      P>|t|      [0.025      0.
975]
----------------------------------------------------------------------------------
----
const                 46.2239      0.788     58.663      0.000      44.679      4
7.769
lwrite                -0.0530      0.016     -3.234      0.001      -0.085      -
0.021
scall                  0.0012      0.000      7.740      0.000       0.001
0.001
swrite                -0.0044      0.003     -1.647      0.100      -0.010
0.001
exec                  -0.1082      0.122     -0.887      0.375      -0.347
0.131
rchar              -1.018e-05   1.09e-06     -9.313      0.000   -1.23e-05   -8.04
e-06
wchar              -7.725e-06   2.57e-06     -3.002      0.003   -1.28e-05   -2.68
e-06
pgout                 -0.7135      0.172     -4.158      0.000      -1.050      -
0.377
pgfree                 0.0842      0.073      1.146      0.252      -0.060
0.228
pgscan               -3.2e-14   5.66e-16    -56.575      0.000   -3.31e-14   -3.09
e-14
atch                   1.2018      0.360      3.336      0.001       0.496
1.908
pgin                   0.0902      0.024      3.821      0.000       0.044
0.137
pflt                  -0.0485      0.003    -18.569      0.000      -0.054      -
0.043
freemem               -0.0023      0.000    -18.303      0.000      -0.003      -
0.002
freeswap            3.231e-05   4.71e-07     68.609      0.000    3.14e-05    3.32
e-05
runqsz_Not_CPU_Bound   7.1012      0.317     22.372      0.000       6.479
7.723
==============================================================================
Omnibus:                     1449.267   Durbin-Watson:                   2.057
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4280.375
Skew:                          -1.307   Prob(JB):                         0.00
Kurtosis:                       6.329   Cond. No.                     2.03e+22
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
[2] The smallest eigenvalue is 2.76e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

In [208…

```
vif_series7 = pd.Series(
    [variance_inflation_factor(X_train.values, i) for i in range(X_train.shape[1])]
    index=X_train.columns,
)
print("VIF values: \n\n{}\n".format(vif_series7))
```

```
VIF values:

const                  28.319577
lwrite                  1.052264
scall                   2.650823
swrite                  3.013285
exec                    2.822530
rchar                   1.673014
wchar                   1.534358
pgout                   6.441278
pgfree                  6.086528
pgscan                       NaN
atch                    1.860771
pgin                    1.484476
pflt                    3.254523
freemem                 1.956495
freeswap                1.762948
runqsz_Not_CPU_Bound    1.144769
dtype: float64
```

Since there is no effect on adj. R-squared after dropping the 'pgout' column, we can remove it from the training set.

In [209…

```
X_train = X_train.drop(["pgout"], axis=1)
```

In [210…

```
olsmod_22 = sm.OLS(y_train, X_train)
olsres_22 = olsmod_22.fit()
print(olsres_22.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    usr   R-squared:                       0.616
Model:                            OLS   Adj. R-squared:                  0.615
Method:                 Least Squares   F-statistic:                     705.4
Date:                Sat, 16 Dec 2023   Prob (F-statistic):               0.00
Time:                        22:44:37   Log-Likelihood:                -21996.
No. Observations:                5734   AIC:                         4.402e+04
Df Residuals:                    5720   BIC:                         4.411e+04
Df Model:                          13
Covariance Type:            nonrobust
==================================================================================
====
                           coef    std err          t      P>|t|      [0.025      0.
975]
----------------------------------------------------------------------------------
----
const                   46.1862      0.789     58.536      0.000      44.639      4
7.733
lwrite                  -0.0528      0.016     -3.220      0.001      -0.085      -
0.021
scall                    0.0011      0.000      7.617      0.000       0.001
0.001
swrite                  -0.0042      0.003     -1.574      0.116      -0.009
0.001
exec                    -0.0905      0.122     -0.741      0.458      -0.330
0.149
rchar                -1.005e-05   1.09e-06     -9.186      0.000   -1.22e-05   -7.91
e-06
wchar                -8.355e-06   2.57e-06     -3.248      0.001   -1.34e-05   -3.31
e-06
pgfree                  -0.1686      0.041     -4.084      0.000      -0.249      -
0.088
pgscan               -8.945e-14   1.64e-15    -54.526      0.000   -9.27e-14   -8.62
e-14
atch                     0.8081      0.348      2.322      0.020       0.126
1.490
pgin                     0.0883      0.024      3.734      0.000       0.042
0.135
pflt                    -0.0488      0.003    -18.634      0.000      -0.054      -
0.044
freemem                 -0.0023      0.000    -18.075      0.000      -0.003      -
0.002
freeswap              3.232e-05   4.72e-07     68.541      0.000    3.14e-05    3.32
e-05
runqsz_Not_CPU_Bound     7.0201      0.317     22.127      0.000       6.398
7.642
==============================================================================
Omnibus:                     1449.457   Durbin-Watson:                   2.055
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4268.814
Skew:                          -1.309   Prob(JB):                         0.00
Kurtosis:                       6.319   Cond. No.                     7.82e+21
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
```

```
cified.
[2] The smallest eigenvalue is 1.87e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

In [211…

```python
vif_series8 = pd.Series(
    [variance_inflation_factor(X_train.values, i) for i in range(X_train.shape[1])]
    index=X_train.columns,
)
print("VIF values: \n\n{}\n".format(vif_series8))
```

```
VIF values: 

const                  28.315818
lwrite                  1.052259
scall                   2.648774
swrite                  3.012409
exec                    2.819098
rchar                   1.671676
wchar                   1.529035
pgfree                  1.917372
pgscan                       NaN
atch                    1.732230
pgin                    1.483892
pflt                    3.253088
freemem                 1.950475
freeswap                1.762866
runqsz_Not_CPU_Bound    1.140440
dtype: float64
```

Now we do not have multicollinearity in our data, the p-values of the coefficients have become reliable and we can remove the non-significant predictor variables.

In [212…

```python
print(olsres_22.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                    usr   R-squared:                       0.616
Model:                            OLS   Adj. R-squared:                  0.615
Method:                 Least Squares   F-statistic:                     705.4
Date:                Sat, 16 Dec 2023   Prob (F-statistic):               0.00
Time:                        22:44:53   Log-Likelihood:                -21996.
No. Observations:                5734   AIC:                         4.402e+04
Df Residuals:                    5720   BIC:                         4.411e+04
Df Model:                          13                                         
Covariance Type:            nonrobust                                         
====================================================================================
====
                          coef    std err          t      P>|t|      [0.025      0.
975]
------------------------------------------------------------------------------------
----
const                  46.1862      0.789     58.536      0.000      44.639      4
7.733
lwrite                 -0.0528      0.016     -3.220      0.001      -0.085      -
0.021
scall                   0.0011      0.000      7.617      0.000       0.001       
0.001
swrite                 -0.0042      0.003     -1.574      0.116      -0.009       
0.001
exec                   -0.0905      0.122     -0.741      0.458      -0.330       
0.149
rchar               -1.005e-05   1.09e-06     -9.186      0.000   -1.22e-05   -7.91
e-06
wchar               -8.355e-06   2.57e-06     -3.248      0.001   -1.34e-05   -3.31
e-06
pgfree                 -0.1686      0.041     -4.084      0.000      -0.249      -
0.088
pgscan              -8.945e-14   1.64e-15    -54.526      0.000   -9.27e-14   -8.62
e-14
atch                    0.8081      0.348      2.322      0.020       0.126       
1.490
pgin                    0.0883      0.024      3.734      0.000       0.042       
0.135
pflt                   -0.0488      0.003    -18.634      0.000      -0.054      -
0.044
freemem                -0.0023      0.000    -18.075      0.000      -0.003      -
0.002
freeswap             3.232e-05   4.72e-07     68.541      0.000    3.14e-05    3.32
e-05
runqsz_Not_CPU_Bound    7.0201      0.317     22.127      0.000       6.398       
7.642
==============================================================================
Omnibus:                     1449.457   Durbin-Watson:                   2.055
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4268.814
Skew:                          -1.309   Prob(JB):                         0.00
Kurtosis:                       6.319   Cond. No.                     7.82e+21
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
```

```
cified.
[2] The smallest eigenvalue is 1.87e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

As observed from above the predictor 'exec' and 'swrite' are having p-value>0.05 so we need to remove them and build the model again.

In [213…

```
X_train = X_train.drop(["exec"], axis=1)
```

In [214…

```
olsmod_23 = sm.OLS(y_train, X_train)
olsres_23 = olsmod_23.fit()
print(olsres_23.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    usr   R-squared:                       0.616
Model:                            OLS   Adj. R-squared:                  0.615
Method:                 Least Squares   F-statistic:                     764.2
Date:                Sat, 16 Dec 2023   Prob (F-statistic):               0.00
Time:                        22:44:57   Log-Likelihood:                -21997.
No. Observations:                5734   AIC:                         4.402e+04
Df Residuals:                    5721   BIC:                         4.411e+04
Df Model:                          12
Covariance Type:            nonrobust
=====================================================================================
====
                           coef    std err          t      P>|t|      [0.025      0.
975]
-------------------------------------------------------------------------------------
----
const                   46.1384      0.786     58.673      0.000      44.597      4
7.680
lwrite                  -0.0533      0.016     -3.254      0.001      -0.085      -
0.021
scall                    0.0011      0.000      7.649      0.000       0.001
0.001
swrite                  -0.0036      0.003     -1.414      0.157      -0.009
0.001
rchar                -1.006e-05   1.09e-06     -9.192      0.000   -1.22e-05   -7.91
e-06
wchar                -8.486e-06   2.57e-06     -3.306      0.001   -1.35e-05   -3.45
e-06
pgfree                  -0.1689      0.041     -4.092      0.000      -0.250      -
0.088
pgscan               -7.607e-14   1.38e-15    -55.076      0.000   -7.88e-14   -7.34
e-14
atch                     0.7953      0.348      2.288      0.022       0.114
1.477
pgin                     0.0865      0.024      3.678      0.000       0.040
0.133
pflt                    -0.0501      0.002    -27.430      0.000      -0.054      -
0.047
freemem                 -0.0023      0.000    -18.061      0.000      -0.003      -
0.002
freeswap              3.233e-05   4.71e-07     68.612      0.000    3.14e-05    3.33
e-05
runqsz_Not_CPU_Bound     7.0250      0.317     22.148      0.000       6.403
7.647
==============================================================================
Omnibus:                     1444.061   Durbin-Watson:                   2.055
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4242.905
Skew:                          -1.305   Prob(JB):                         0.00
Kurtosis:                       6.309   Cond. No.                     8.85e+20
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
```

[2] The smallest eigenvalue is 1.46e-26. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.

In [215…

```
X_train = X_train.drop(["swrite"], axis=1)
```

In [216…

```
olsmod_24 = sm.OLS(y_train, X_train)
olsres_24 = olsmod_24.fit()
print(olsres_24.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    usr   R-squared:                       0.616
Model:                            OLS   Adj. R-squared:                  0.615
Method:                 Least Squares   F-statistic:                     833.3
Date:                Sat, 16 Dec 2023   Prob (F-statistic):               0.00
Time:                        22:44:59   Log-Likelihood:                -21998.
No. Observations:                5734   AIC:                         4.402e+04
Df Residuals:                    5722   BIC:                         4.410e+04
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
====
                           coef    std err          t      P>|t|      [0.025      0.
975]
------------------------------------------------------------------------------
----
const                   45.9859      0.779     59.032      0.000      44.459      4
7.513
lwrite                  -0.0532      0.016     -3.246      0.001      -0.085      -
0.021
scall                    0.0010      0.000      8.269      0.000       0.001
0.001
rchar                 -1.01e-05   1.09e-06     -9.231      0.000   -1.22e-05   -7.95
e-06
wchar                -9.472e-06   2.47e-06     -3.835      0.000   -1.43e-05   -4.63
e-06
pgfree                  -0.1696      0.041     -4.109      0.000      -0.250      -
0.089
pgscan                1.218e-13   2.17e-15     56.222      0.000    1.18e-13    1.26
e-13
atch                     0.8150      0.347      2.346      0.019       0.134
1.496
pgin                     0.0870      0.024      3.701      0.000       0.041
0.133
pflt                    -0.0508      0.002    -28.931      0.000      -0.054      -
0.047
freemem                 -0.0023      0.000    -18.005      0.000      -0.003      -
0.002
freeswap              3.237e-05   4.71e-07     68.799      0.000    3.14e-05    3.33
e-05
runqsz_Not_CPU_Bound     7.0222      0.317     22.138      0.000       6.400
7.644
==============================================================================
Omnibus:                     1442.466   Durbin-Watson:                   2.054
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4235.457
Skew:                          -1.303   Prob(JB):                         0.00
Kurtosis:                       6.306   Cond. No.                     4.61e+20
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
[2] The smallest eigenvalue is 5.38e-26. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

After dropping the features causing strong multicollinearity and the statistically insignificant ones, our model performance hasn't dropped sharply. This shows that these variables did not have much predictive power.

Testing the Assumptions of Linear Regression

These assumptions are essential conditions that should be met before we draw inferences regarding the model estimates or use the model to make a prediction.

For Linear Regression, we need to check if the following assumptions hold:

Linearity
Independence
Homoscedasticity
Normality of error terms
No strong Multicollinearity

In [217…

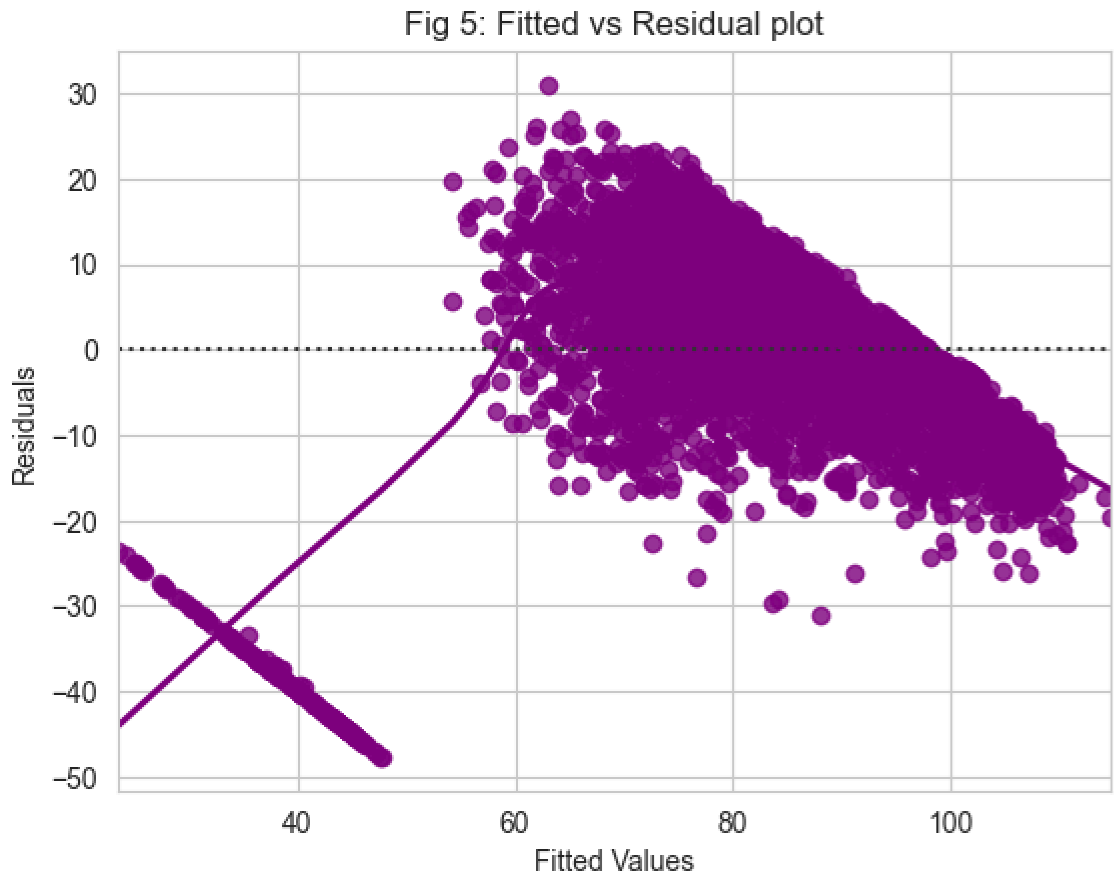
```
df_pred = pd.DataFrame()

df_pred["Actual Values"] = y_train.values.flatten()  # actual values
df_pred["Fitted Values"] = olsres_24.fittedvalues.values  # predicted values
df_pred["Residuals"] = olsres_24.resid.values  # residuals

df_pred.head()
```

Out[217…

	Actual Values	Fitted Values	Residuals
0	91	85.342968	5.657032
1	94	83.100859	10.899141
2	0	44.073705	-44.073705
3	83	72.385813	10.614187
4	94	100.082202	-6.082202

In [218…

```
# let us plot the fitted values vs residuals
sns.set_style("whitegrid")
sns.residplot(
    data=df_pred, x="Fitted Values", y="Residuals", color="purple", lowess=True
)
plt.xlabel("Fitted Values")
plt.ylabel("Residuals")
plt.title("Fig 5: Fitted vs Residual plot")
plt.show()
```

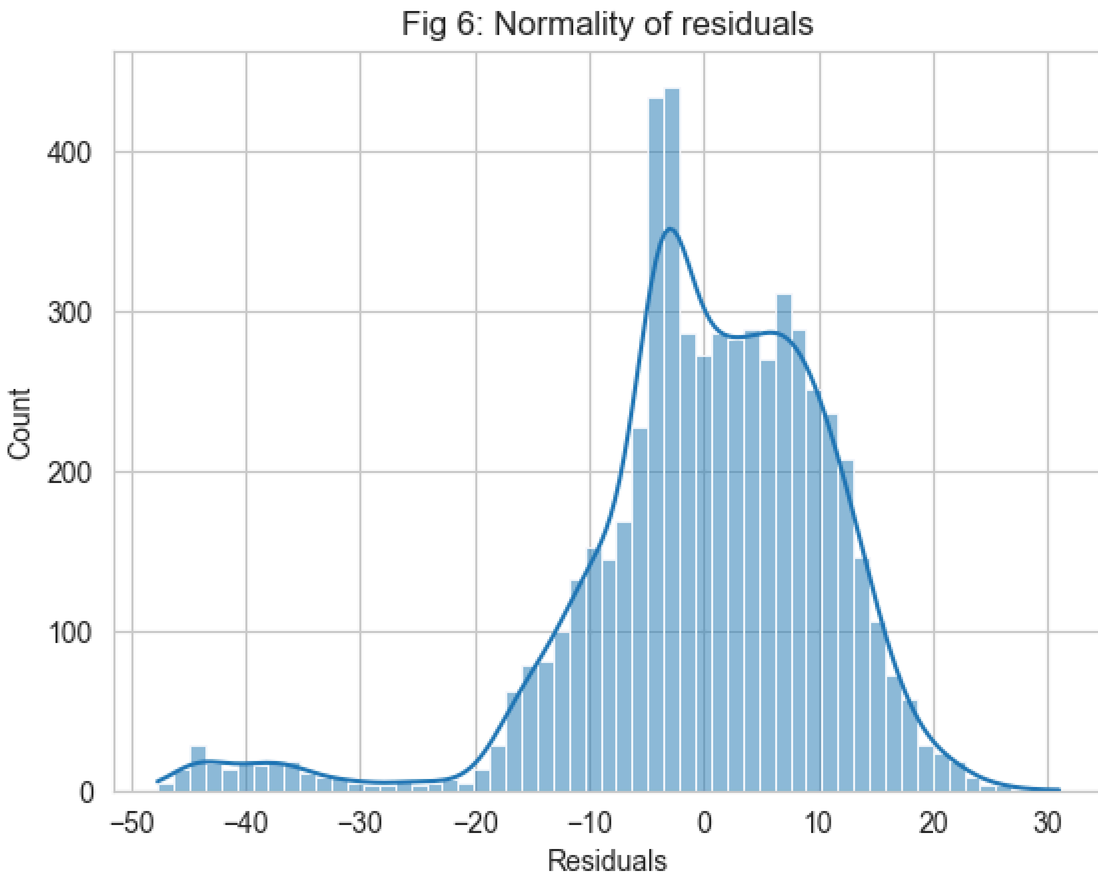


No pattern in the data thus the assumption of linearity and independence of predictors satisfied

Test for Normality

In [219…

```
sns.histplot(df_pred["Residuals"], kde=True)
plt.title("Fig 6: Normality of residuals")
plt.show()
```

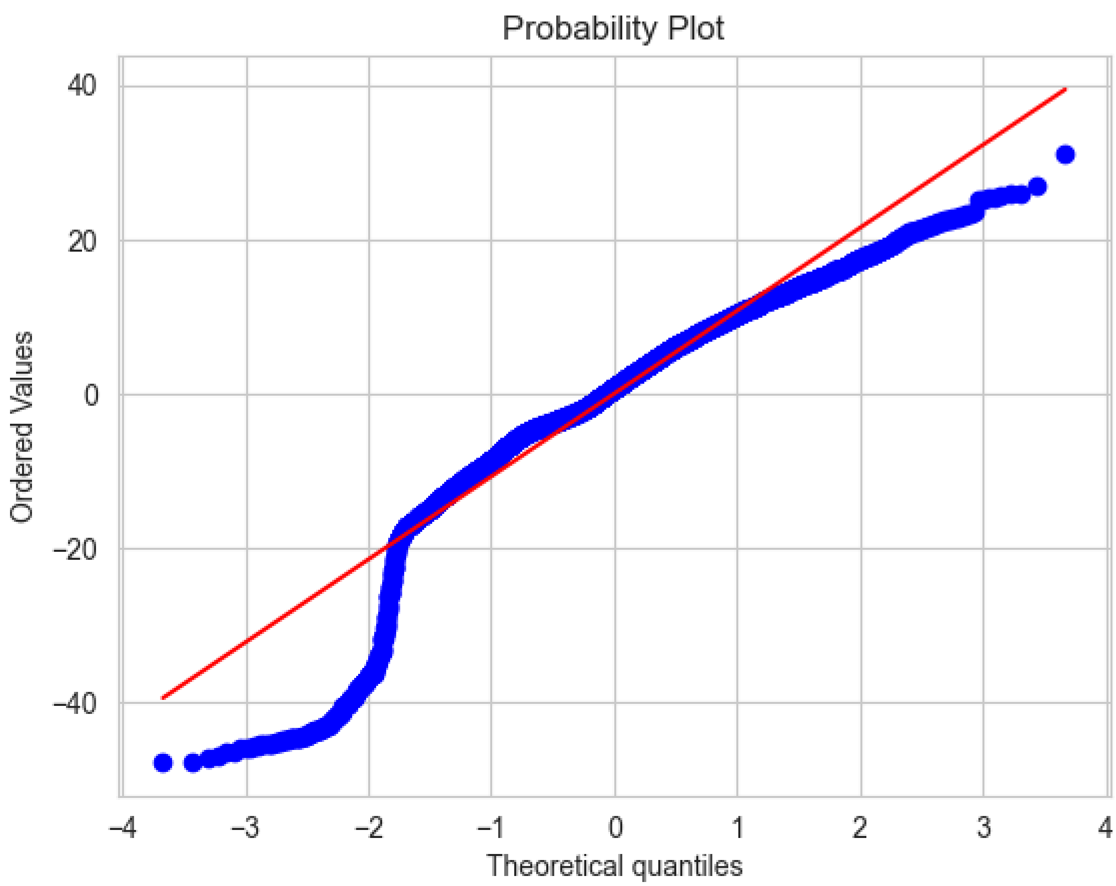


- The residual terms are normally distributed (as an approximation).

The QQ plot of residuals is used to visually check the normality assumption. The normal probability plot of residuals should approximately follow a straight line.

In [124…

```
stats.probplot(df_pred["Residuals"], dist="norm", plot=pylab)
plt.show()
```



In [125…
```
stats.shapiro(df_pred["Residuals"])
```

Out[125…
```
ShapiroResult(statistic=0.9149823188781738, pvalue=0.0)
```

- Since p-value < 0.05, the residuals are not normal as per shapiro test.
- The residuals are not normal. However, as an approximation, we can accept this distribution as close to being normal.

Test for Homoscedasticity

In [126…
```
name = ["F statistic", "p-value"]
test = sms.het_goldfeldquandt(df_pred["Residuals"], X_train)
lzip(name, test)
```

Out[126…
```
[('F statistic', 1.1241815614919726), ('p-value', 0.0008857227357494505)]
```

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

Summary of final model (olsres_24)

In [127…
```
print(olsres_24.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    usr   R-squared:                       0.616
Model:                            OLS   Adj. R-squared:                  0.615
Method:                 Least Squares   F-statistic:                     833.3
Date:                Sat, 16 Dec 2023   Prob (F-statistic):               0.00
Time:                        15:54:51   Log-Likelihood:                -21998.
No. Observations:                5734   AIC:                         4.402e+04
Df Residuals:                    5722   BIC:                         4.410e+04
Df Model:                          11
Covariance Type:            nonrobust
==================================================================================
====
                           coef    std err          t      P>|t|      [0.025      0.
975]
----------------------------------------------------------------------------------
----
const                   45.9859      0.779     59.032      0.000      44.459      4
7.513
lwrite                  -0.0532      0.016     -3.246      0.001      -0.085      -
0.021
scall                    0.0010      0.000      8.269      0.000       0.001
0.001
rchar                 -1.01e-05   1.09e-06     -9.231      0.000   -1.22e-05   -7.95
e-06
wchar                -9.472e-06   2.47e-06     -3.835      0.000   -1.43e-05   -4.63
e-06
pgfree                  -0.1696      0.041     -4.109      0.000      -0.250      -
0.089
pgscan                1.218e-13   2.17e-15     56.222      0.000    1.18e-13    1.26
e-13
atch                     0.8150      0.347      2.346      0.019       0.134
1.496
pgin                     0.0870      0.024      3.701      0.000       0.041
0.133
pflt                    -0.0508      0.002    -28.931      0.000      -0.054      -
0.047
freemem                 -0.0023      0.000    -18.005      0.000      -0.003      -
0.002
freeswap              3.237e-05   4.71e-07     68.799      0.000    3.14e-05    3.33
e-05
runqsz_Not_CPU_Bound     7.0222      0.317     22.138      0.000       6.400
7.644
==============================================================================
Omnibus:                     1442.466   Durbin-Watson:                   2.054
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4235.457
Skew:                          -1.303   Prob(JB):                         0.00
Kurtosis:                       6.306   Cond. No.                     4.61e+20
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
[2] The smallest eigenvalue is 5.38e-26. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

The model equation will be as follows

In [241…
```
# Let us write the equation of linear regression
Equation = "usr ="
print(Equation, end=" ")
for i in range(len(X_train.columns)):
    if i == 0:
        print(olsres_24.params[i], "+", end=" ")
    elif i != len(X_train.columns) - 1:
        print(
            olsres_24.params[i],
            "* (",
            X_train.columns[i],
            ")",
            "+",
            end="  ",
        )
    else:
        print(olsres_24.params[i], "* (", X_train.columns[i], ")")
```

```
usr = 45.985898637332326 + -0.05320083210526674 * ( lwrite ) +  0.001006460871296824
9 * ( scall ) +  -1.00977450267907e-05 * ( rchar ) +  -9.472019738498376e-06 * ( wch
ar ) +  -0.16958079612018762 * ( pgfree ) +  1.218464047535082e-13 * ( pgscan ) +
0.8149653158320619 * ( atch ) +  0.08703312601751777 * ( pgin ) +  -0.05084827884843
015 * ( pflt ) +  -0.0022976787041281256 * ( freemem ) +  3.23720833103961e-05 * ( f
reeswap ) +  7.022150216030361 * ( runqsz_Not_CPU_Bound )
```

Observations

- R-squared of the model is 0.616 and adjusted R-squared is 0.615, which shows that the model is able to explain ~61% variance in the data.
- 1 unit increase in the lwrite lead to a 0.053 times decrease in the usr
- 1 unit increase in the scall lead to a 0.001 times increase in the usr
- 1 unit increase in the pgfree lead to a 0.169 times decrease in the usr
- 1 unit increase in the atch lead to a 0.815 times increase in the usr
- 1 unit increase in the pgin lead to a 0.087 times increase in the usr
- 1 unit increase in the pflt lead to a 0.050 times decrease in the usr
- 1 unit increase in the freemem lead to a 0.002 times decrease in the usr
- 1 unit increase in the runqsz (Not_CPU_Bound) lead to a 7.022 times increase in the usr

Predictions

In [221…
```
X_train.columns
```

Out[221…
```
Index(['const', 'lwrite', 'scall', 'rchar', 'wchar', 'pgfree', 'pgscan',
       'atch', 'pgin', 'pflt', 'freemem', 'freeswap', 'runqsz_Not_CPU_Bound'],
      dtype='object')
```

In [222…
```
X_test.columns
```

Out[222…
```
Index(['const', 'lread', 'lwrite', 'scall', 'sread', 'swrite', 'fork', 'exec',
       'rchar', 'wchar', 'pgout', 'ppgout', 'pgfree', 'pgscan', 'atch', 'pgin',
       'ppgin', 'pflt', 'vflt', 'freemem', 'freeswap', 'runqsz_Not_CPU_Bound'],
      dtype='object')
```

In [223…
```
# dropping columns from the test data that are not there in the training data
X_test1 = X_test.drop(
    ['lread','sread','swrite','fork','exec','pgout','ppgout','ppgin','vflt'], axis=
)
```

In [224…
```
# let's make predictions on the test set
y_pred_test = olsres_24.predict(X_test1)
y_pred_train = olsres_24.predict(X_train)
```

In [225…
```
# RMSE on the Training data
rmse1 = np.sqrt(mean_squared_error(y_train, y_pred_train))
rmse1
```

Out[225…
```
11.216874606014489
```

In [226…
```
# RMSE on the Testing data
rmse2 = np.sqrt(mean_squared_error(y_test, y_pred_test))
rmse2
```

Out[226…
```
11.950559873138983
```

Linear Regression using (sklearn)

In [227…
```
# invoke the LinearRegression function and find the bestfit model on training data

regression_model = LinearRegression()
regression_model.fit(X_train, y_train)
```

Out[227…
```
▾ LinearRegression
LinearRegression()
```

In [228…
```
# Let us explore the coefficients for each of the independent attributes

for idx, col_name in enumerate(X_train.columns):
    print("The coefficient for {} is {}".format(col_name, regression_model.coef_[0]
```

```
The coefficient for const is 0.0
The coefficient for lwrite is -0.053200832105185024
The coefficient for scall is 0.0010064608712971775
The coefficient for rchar is -1.0097745026791706e-05
The coefficient for wchar is -9.472019738505921e-06
The coefficient for pgfree is -0.16958079612022087
The coefficient for pgscan is 9.020562075079397e-16
The coefficient for atch is 0.8149653158308979
The coefficient for pgin is 0.08703312601757876
The coefficient for pflt is -0.05084827884842716
The coefficient for freemem is -0.002297678704127836
The coefficient for freeswap is 3.237208331035267e-05
The coefficient for runqsz_Not_CPU_Bound is 7.022150216030472
```

In [229…

```
# Let us check the intercept for the model

intercept = regression_model.intercept_[0]

print("The intercept for model is {}".format(intercept))
```

```
The intercept for model is 45.98589863735059
```

In [235…

```
# R square on training data
regression_model.score(X_train, y_train)
```

Out[235… 0.6156760900762905

62% of the variation in the usr is explained by the predictors in the model for train set

In [238…

```
# R square on test data
regression_model.score(X_test1, y_test)
```

Out[238… 0.6081862810909436

61% of the variation in the usr is explained by the predictors in the model for test set

In [141…

```
# RMSE on Training data
predicted_train=regression_model.fit(X_train, y_train).predict(X_train)
np.sqrt(metrics.mean_squared_error(y_train,predicted_train))
```

Out[141… 11.216874606014489

In [142…

```
# RMSE on Testing data
predicted_test=regression_model.fit(X_train, y_train).predict(X_test1)
np.sqrt(metrics.mean_squared_error(y_test,predicted_test))
```

Out[142… 11.950559873140755

Actionable Insights and Recommendations:

- scall, atc, pgin and runqsz can be increased to improve usr %
- lwrite, pflt, freemem can be decreased to improve usr %

In [ ]:

```

```