

Business Report

PM Project - Coded

By Gagandeep Singh

Problem - 1.....	2
Data Description:.....	2
Structure of the Data.....	3
Data Type:.....	4
Statistical Summary.....	4
Univariate Analysis.....	4
Observations -	10
Bivariate and Multivariate Analysis.....	11
Observations -	12
vflt and fork.....	13
pgfree and ppgout.....	13
usr and swrite.....	14
usr and pflt.....	14
Problem 1 - Data Pre-processing.....	15
Missing Value Treatment (if needed).....	15
Outlier Treatment.....	16
Feature Engineering.....	17
Encode the data.....	17
Checking for duplicates.....	17
Train-test split.....	18
Problem 1- Model Building - Linear regression.....	18
Linear Regression Model (using OLS).....	18
Observations:.....	20
Testing the Assumptions of Linear Regression.....	22
Linearity and Independence of predictors.....	22
Test for Normality.....	22
Test for Homoscedasticity.....	22
Observation.....	23
Predictions.....	24
Observations –	24
Problem 1- Model Building - Linear regression.....	24
Impact of Relevant Variables:.....	25
Problem 1 - Business Insights & Recommendations.....	25
Key Takeaways -	25
Problem 2 - Define the problem and perform exploratory Data Analysis.....	26
Data Description.....	26
Data Structure.....	26
Observations.....	28
Problem 2 - Data Pre-processing.....	28
Outlier Treatment.....	28
Univariate Analysis.....	29
Bivariate Analysis.....	30
Multivariate Analysis.....	31
Outlier Treatment.....	33
Problem 2 - Model Building and Compare the Performance of the Models.....	35

CART.....	35
Feature Importance.....	35
Regularising the Decision Tree.....	36
AUC and ROC for the training data.....	36
AUC and ROC for the test data.....	36
Classification report of train and test.....	37
Confusion Matrix for the training data.....	37
Confusion Matrix for testing data.....	38
Summary -.....	39
Logistics Regression.....	40
Logistic Regression Model.....	41
Model Evaluation.....	41
Summary.....	45
Linear Discriminant Analysis.....	45
Null Value Treatment.....	45
LDA Model.....	46
Linear Discriminant Function.....	46
Summary: The confusion matrix and classification report for test data.....	48
Summary : The confusion matrix and classification report for training data.....	49
Probability prediction for the training and test data -.....	50
AUC and ROC for the training data.....	51
Performance Metrics Overview:.....	51
CART (Decision Tree) Test Data Report:.....	51
Logistic Regression Test Data Report:.....	52
LDA (Linear Discriminant Analysis) Test Data Report:.....	52
Comparison:.....	52
Conclusion:.....	52
Problem 2 - Business Insights & Recommendations.....	52

Problem - 1

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 Description:

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

Structure of the Data

	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 x 22 columns

```
<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
```

- Number of Rows: 8192
- Number of Columns: 22

- Memory Usage: 1.4+ MB
- Range Index: 0 to 8191
- Data Types: Float, Int, and Object

Data Type:

The different datatypes in the dataset are as follows

1. There are 8 columns in the with int64 data type
2. There are 1 column in the with object data type
3. There are 13 columns in the with float64 data type

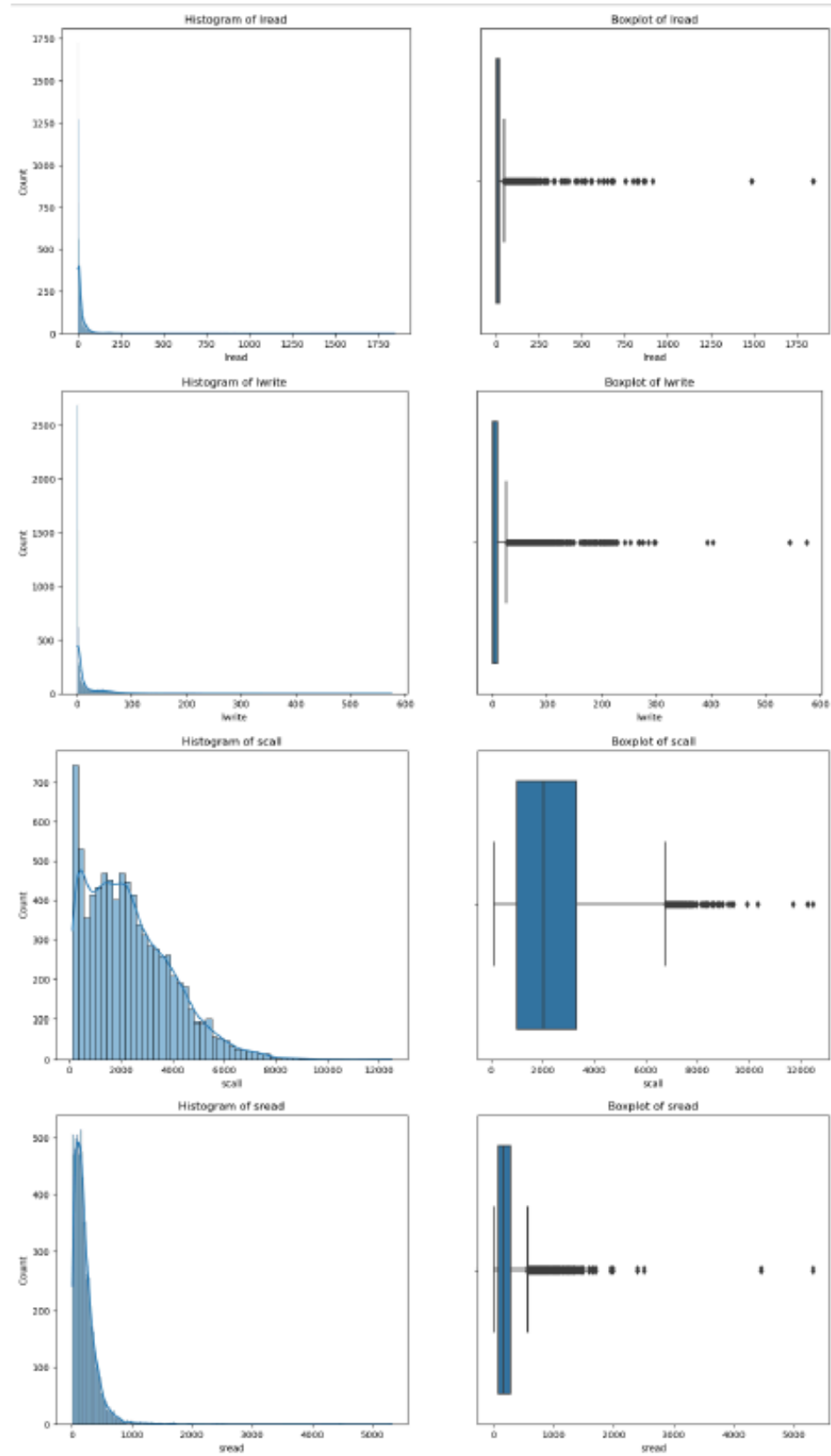
	count	mean	std	min	25%	50%	75%	max
lread	8192.0	1.955969e+01	53.353799	0.0	2.0	7.0	20.000	1845.00
lwrite	8192.0	1.310620e+01	29.891726	0.0	0.0	1.0	10.000	575.00
scall	8192.0	2.306318e+03	1633.617322	109.0	1012.0	2051.5	3317.250	12493.00
sread	8192.0	2.104800e+02	198.980146	6.0	86.0	166.0	279.000	5318.00
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

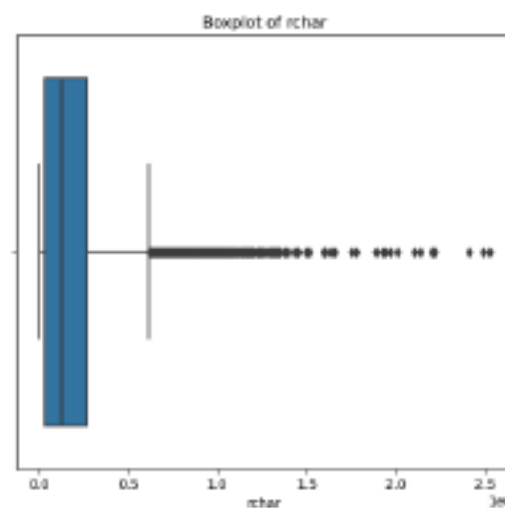
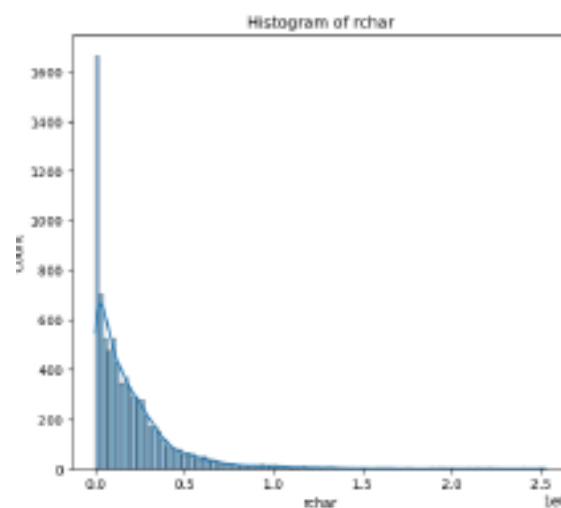
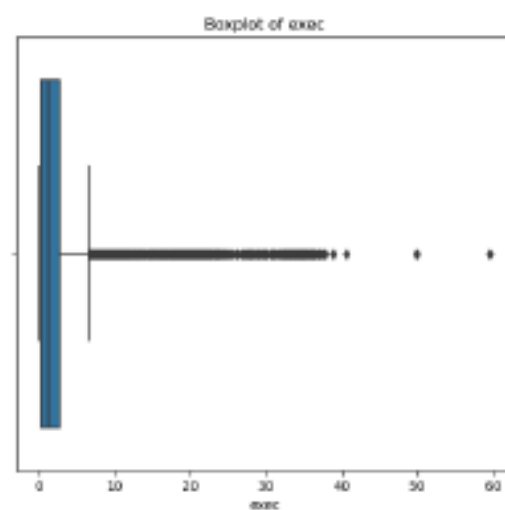
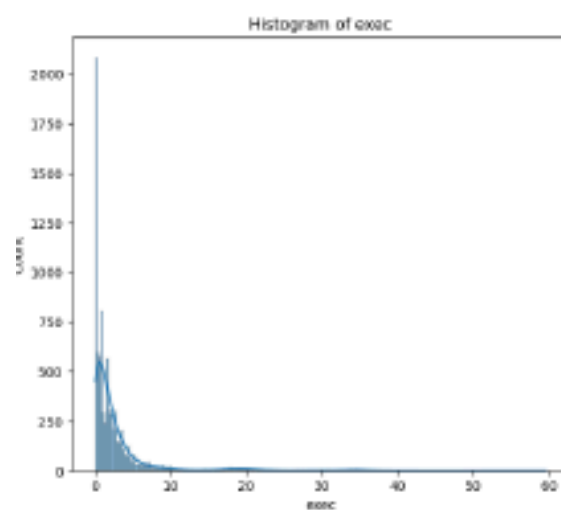
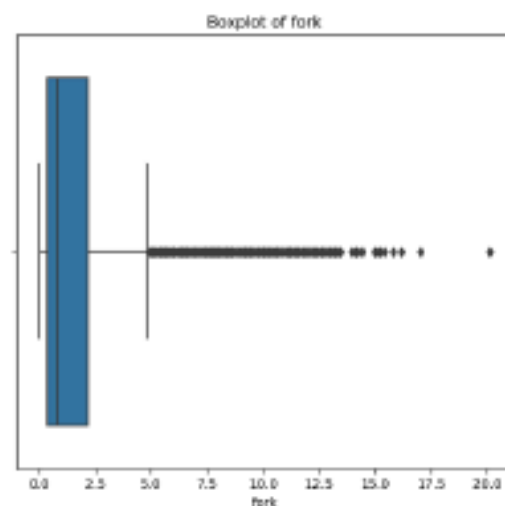
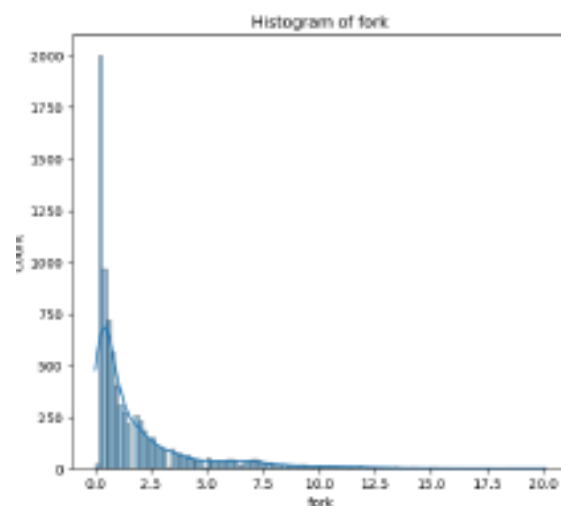
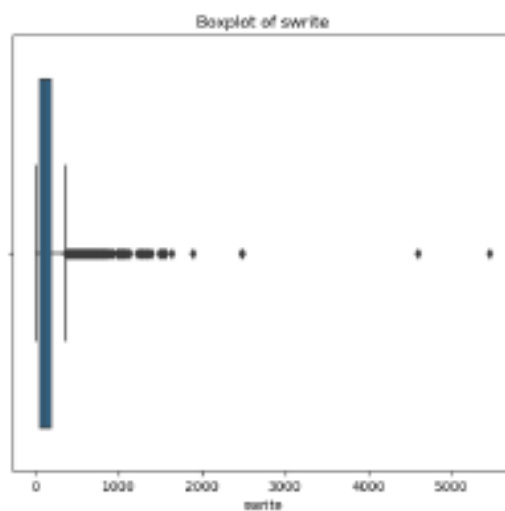
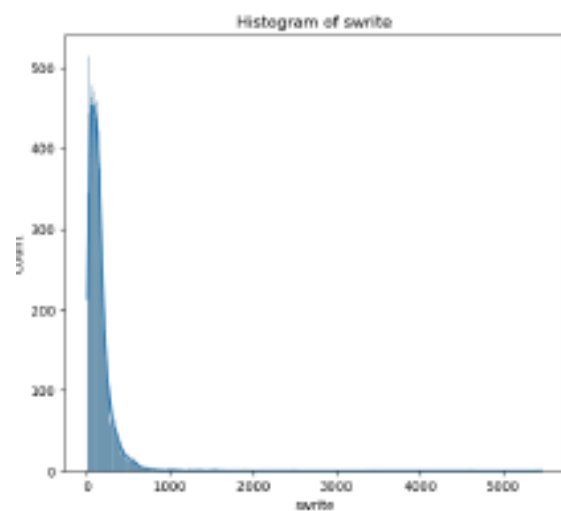
Statistical Summary

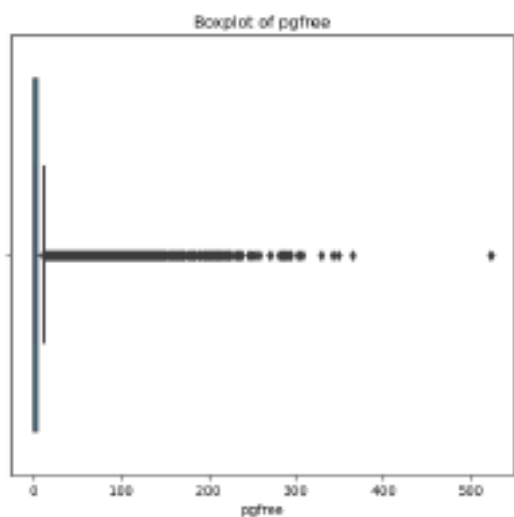
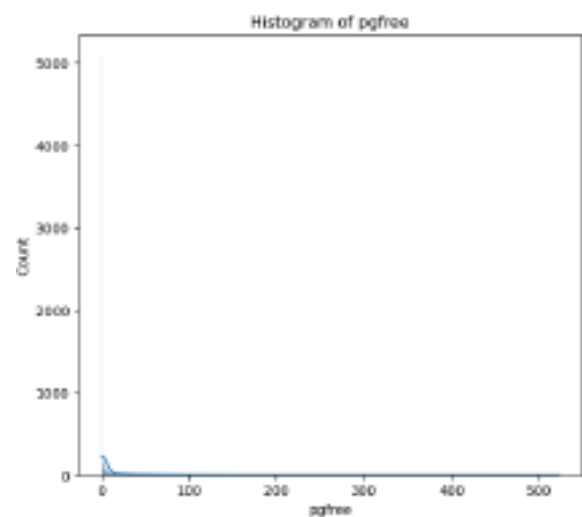
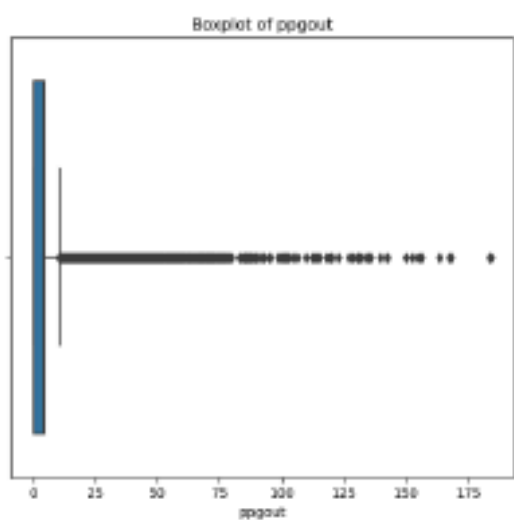
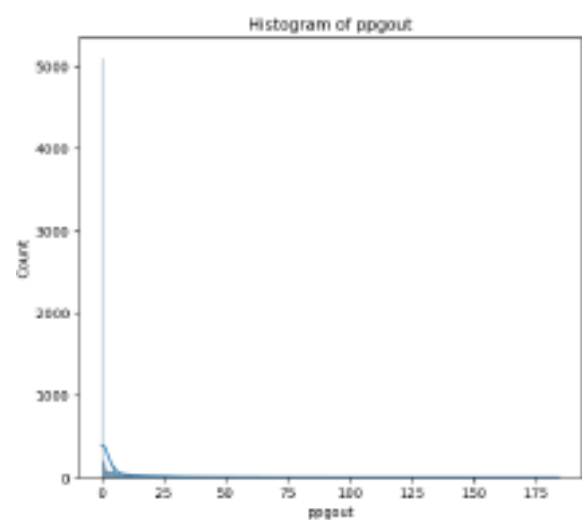
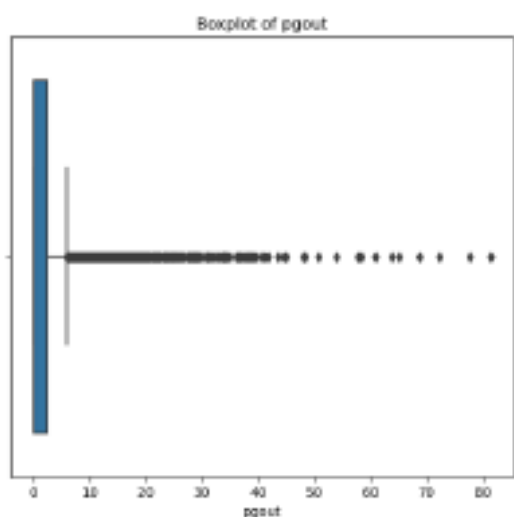
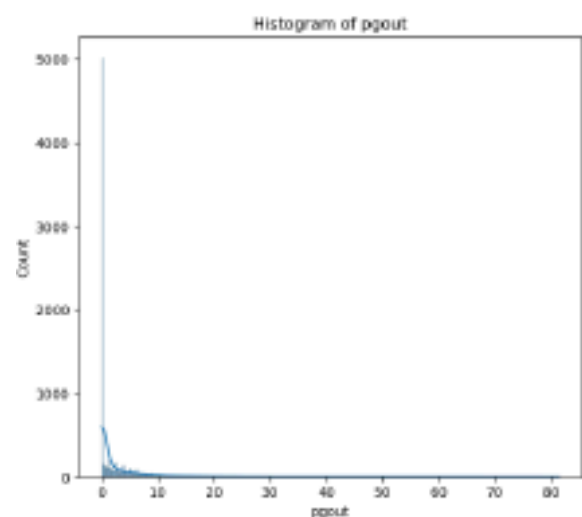
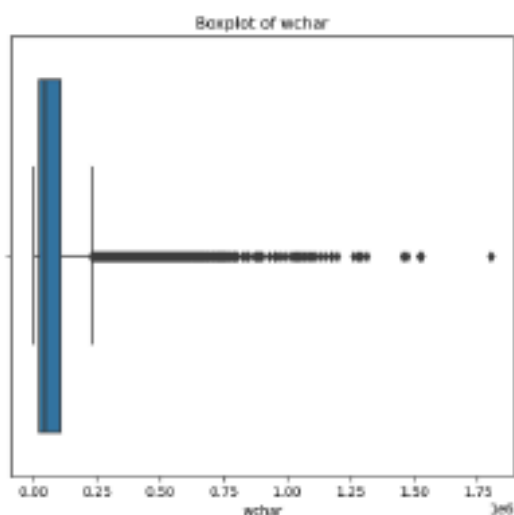
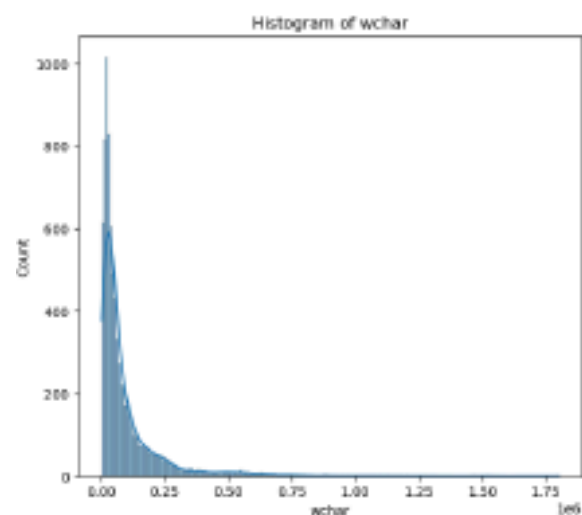
Most columns in the dataset have minimum values of zero, except for "scall," "sread," "rchar," "wchar," "vflt," "freemem," and "freeswap."

All columns exhibit a positively skewed distribution, meaning the tail of the distribution extends towards the higher values, with the mean exceeding the median in each case.

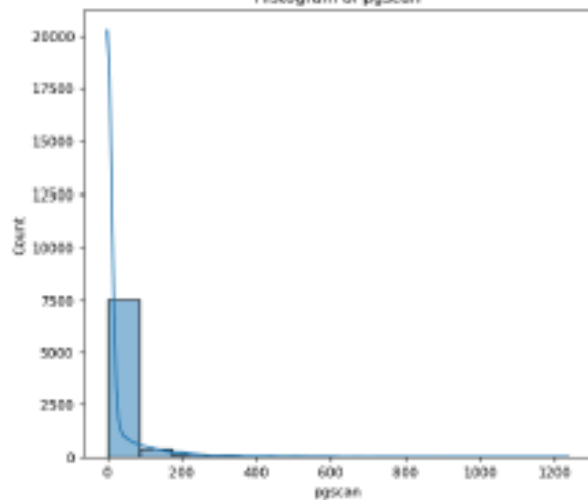
Univariate Analysis



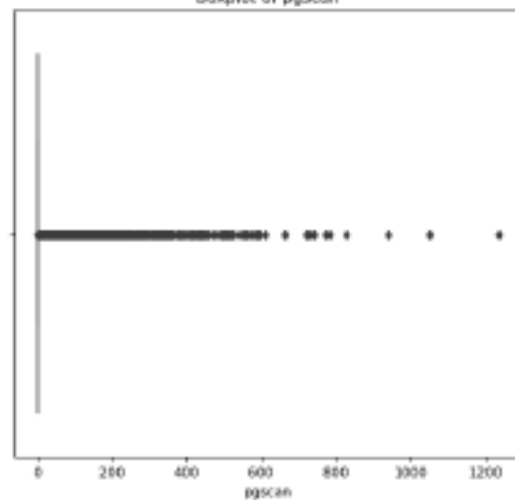




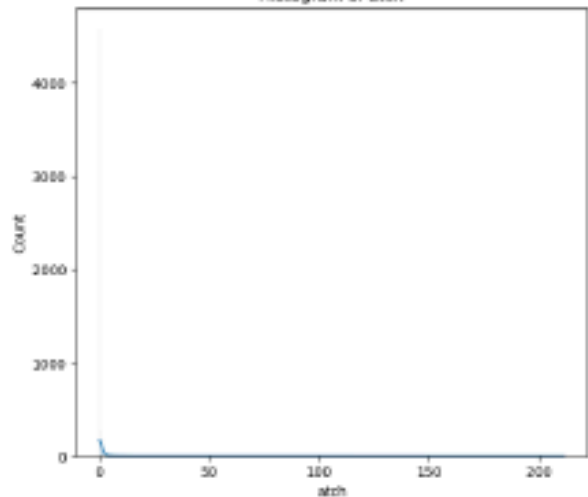
Histogram of pgscan



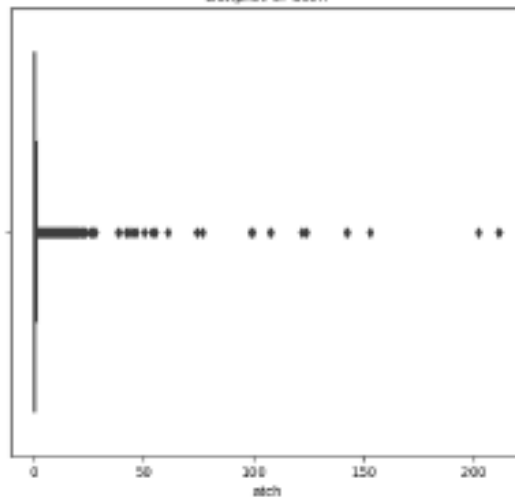
Boxplot of pgscan



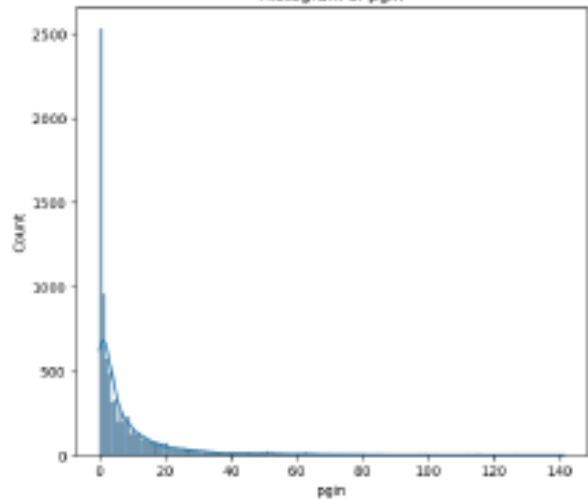
Histogram of atch



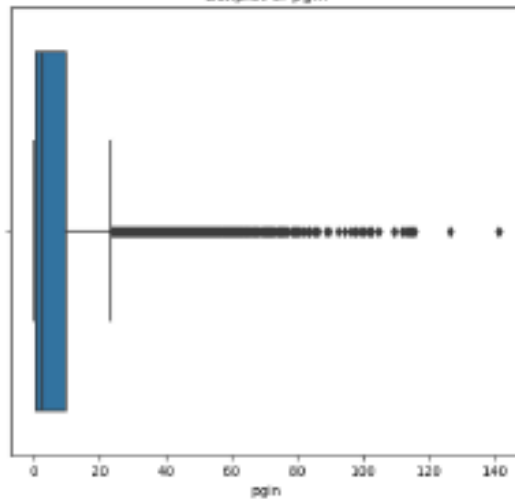
Boxplot of atch



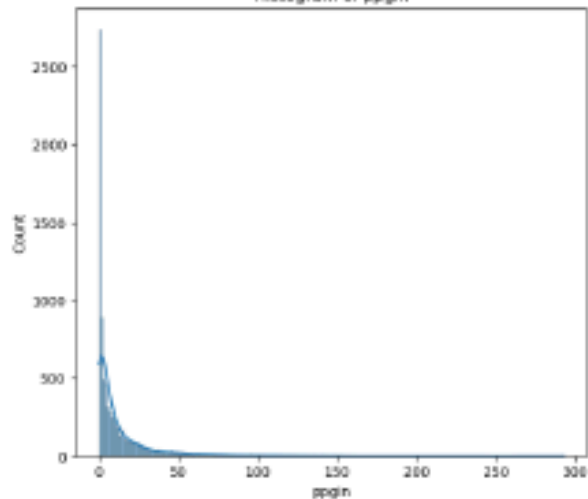
Histogram of pgin



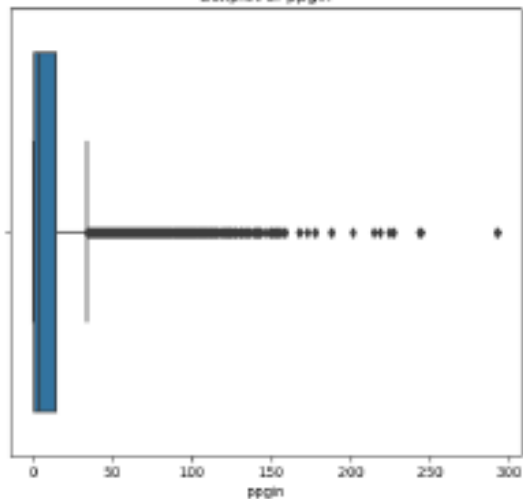
Boxplot of pgin



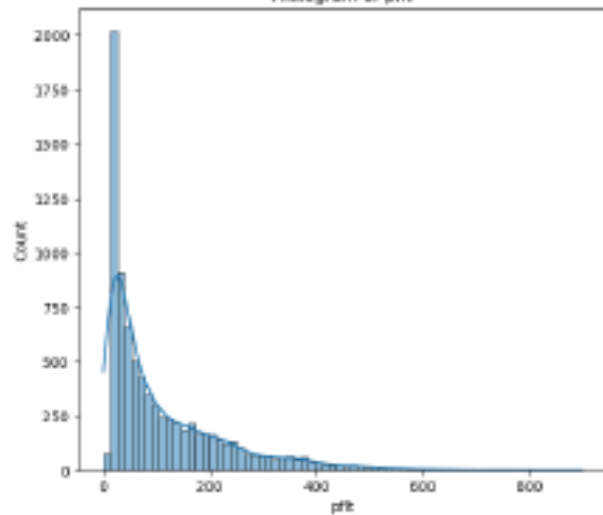
Histogram of ppgin



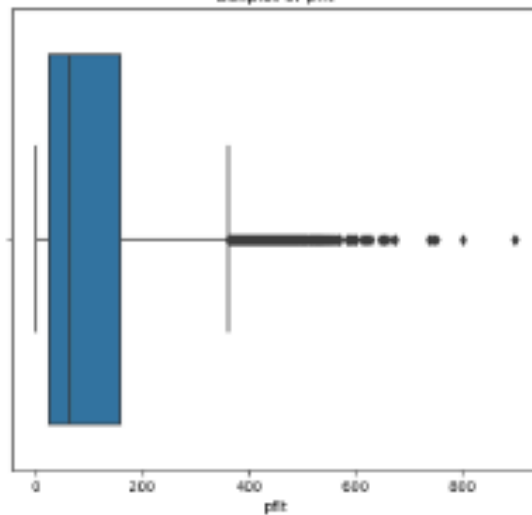
Boxplot of ppgin



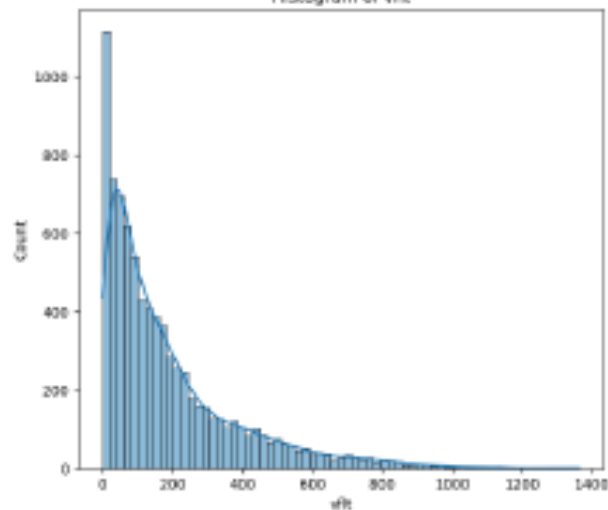
Histogram of pfit



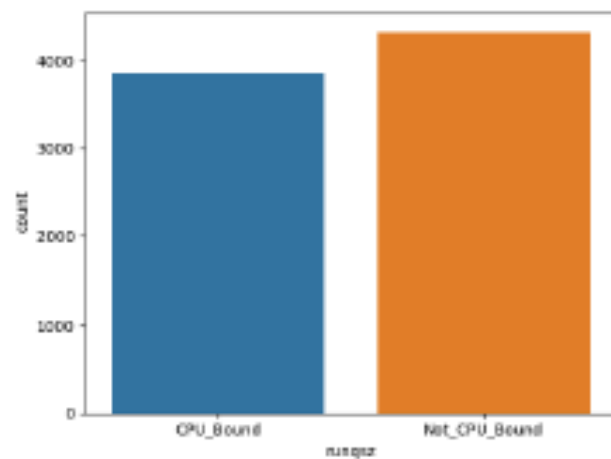
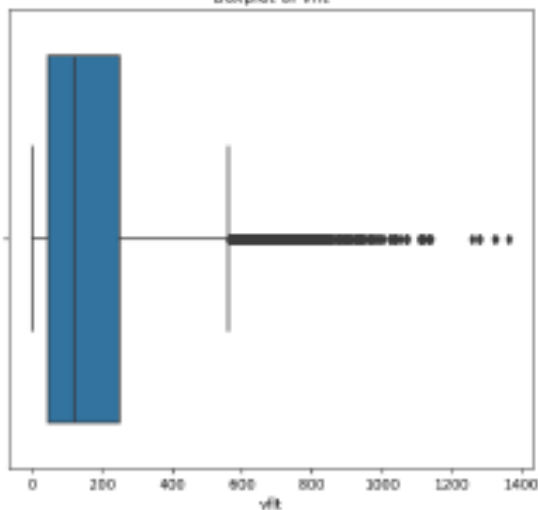
Boxplot of pfit



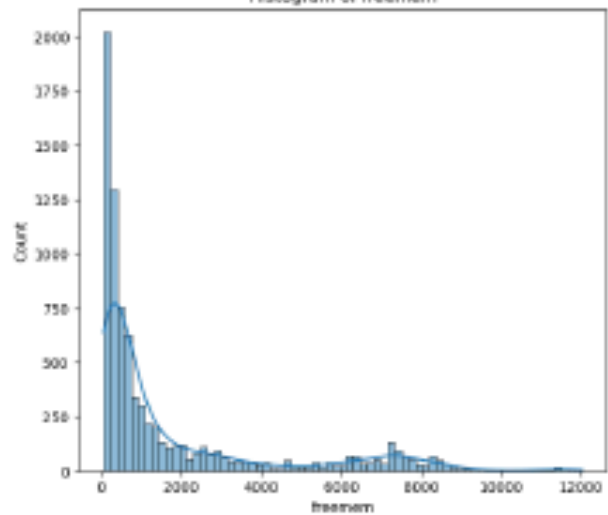
Histogram of vfit



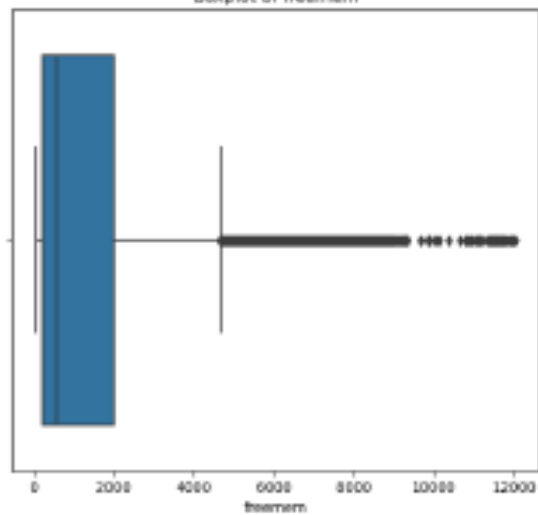
Boxplot of vfit

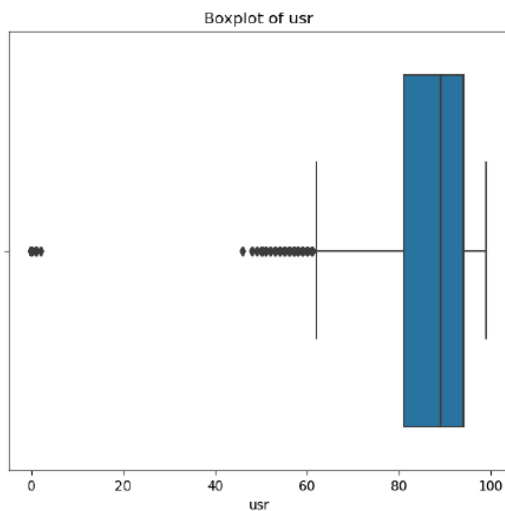
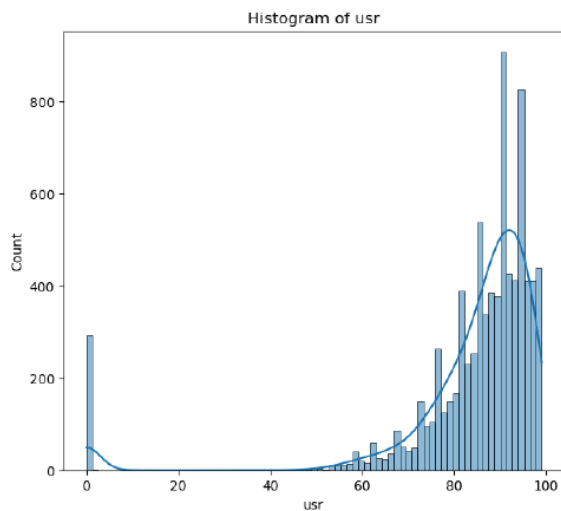
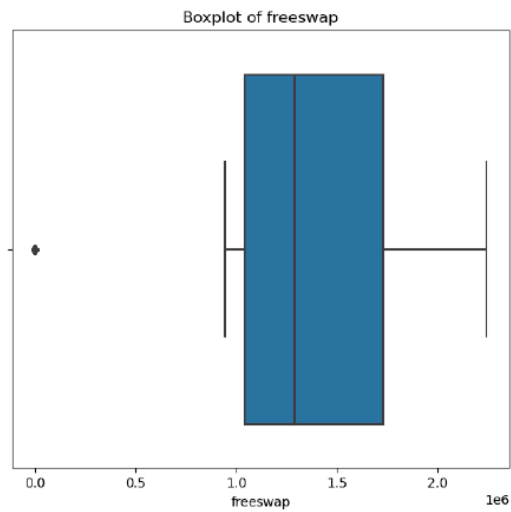
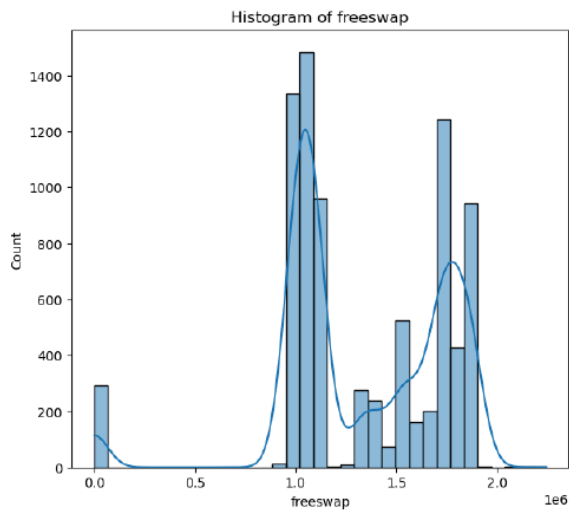


Histogram of freemem



Boxplot of freemem





Observations -

Outliers and Data Skewness:

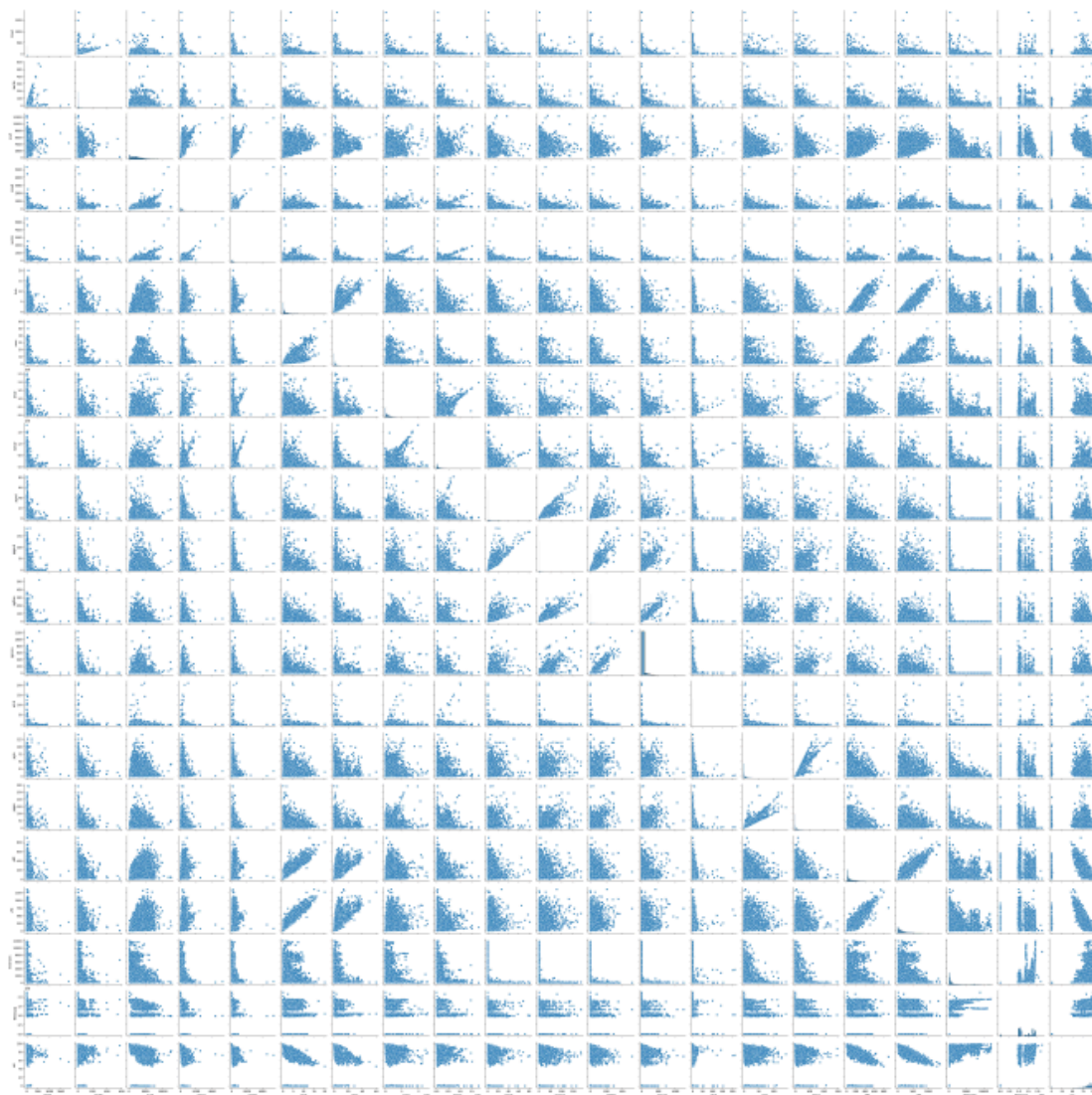
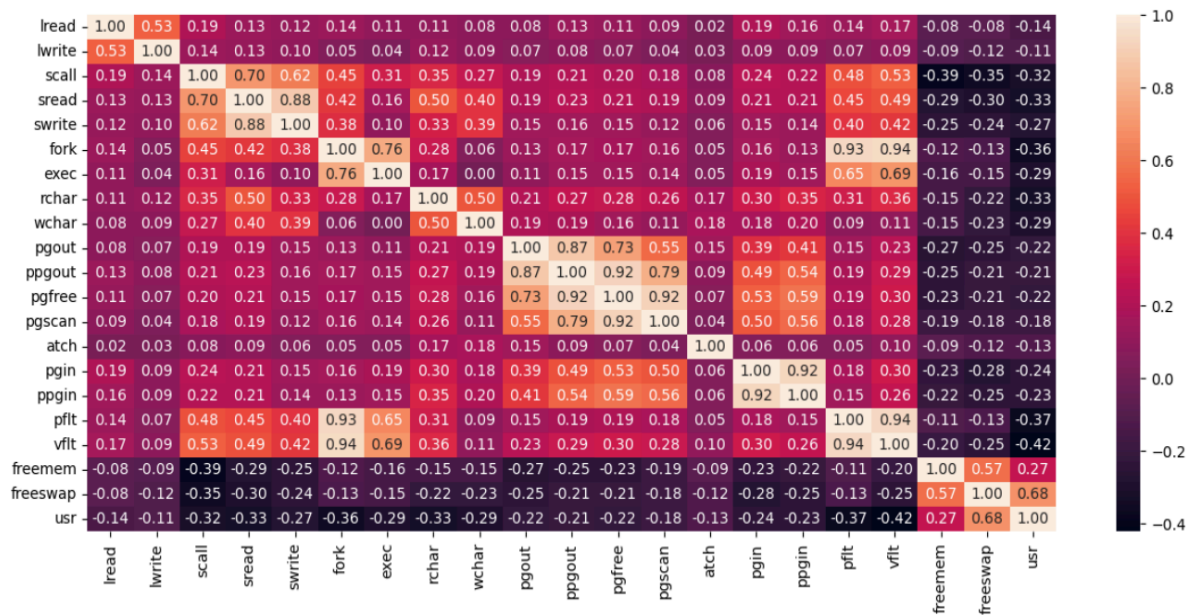
Many columns exhibit outliers, leading to data skewness. Outliers are data points that significantly differ from other observations in the dataset. They can skew statistical measures like the mean and impact the distribution of data. Skewed data distributions can be either positively skewed (right-skewed), where the tail of the distribution extends to the right, or negatively skewed (left-skewed), where the tail extends to the left. Run Queue Size (runqsz):

The "runqsz" column represents the process run queue size. It indicates the number of processes waiting to run on the CPU. Observations suggest that the dataset contains more non-CPU-bound processes, as indicated by a higher count of such observations. This imbalance implies that there are fewer high values (possibly indicating CPU-bound processes) compared to non-CPU-bound processes in the dataset. Highest Count for usr at 90 Percent:

Among the observations for the "usr" column, the highest count occurs at the 90th percentile, with a count of 459. This suggests that a significant number of observations have "usr" values at or below the 90th percentile, indicating a concentration of data in the lower percentiles. Skewness in Variables like vflt, pflt, ppgin, pgin, pgscan:

Variables such as "vflt," "pflt," "ppgin," "pgin," and "pgscan" exhibit right skewness. Right skewness implies that the tail of the distribution extends to the right, indicating that there are fewer high values compared to lower values in the dataset. This skewness can affect statistical analyses and modeling, as it may lead to biased estimates and influence the interpretation of results. It's important to consider transformations or other techniques to address skewness when analyzing or modeling such data.

Bivariate and Multivariate Analysis



Observations -

Here's a summary of the key observations you've made:

High Correlation (≥ 0.9):

Variables pflt and vflt exhibit the highest correlation with fork, with correlation coefficients of 0.93 and 0.94, respectively.

Strong Correlation (≥ 0.7):

Variables scall and sread show a strong correlation of 0.70.

Moderate Correlation (0.4 - 0.59):

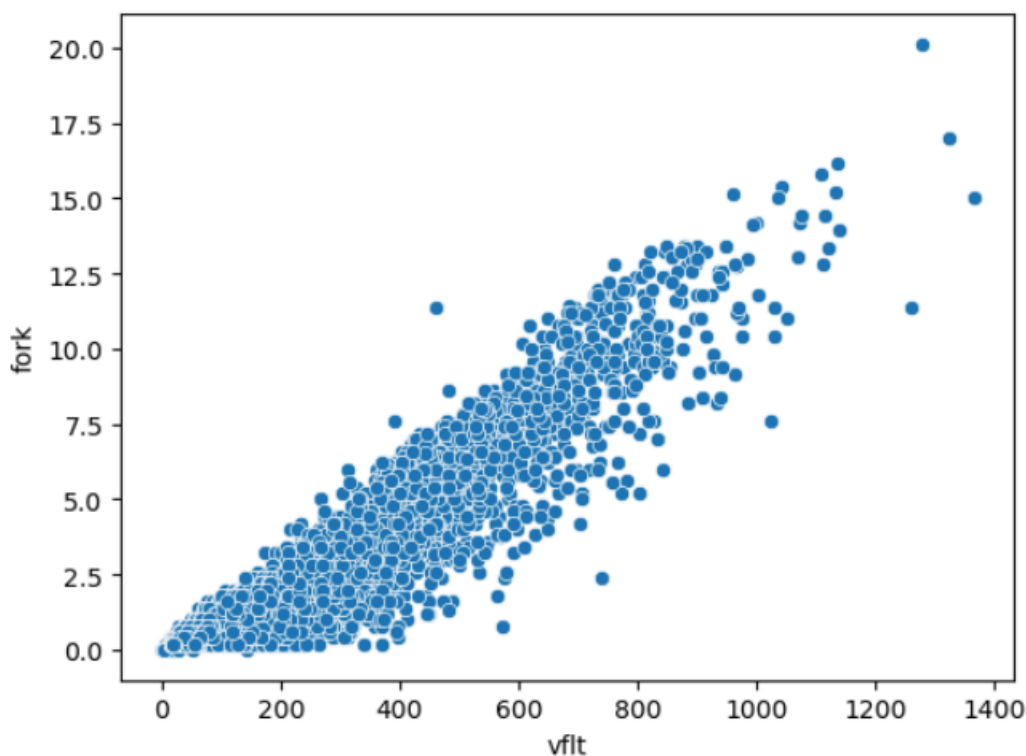
Several pairs of variables exhibit moderate correlation, including pgout and pgfree, pgout and pgscan, exec and fork, exec and pflt, exec and vflt, lwrite and lread, fork and scall, pflt and scall, rchar and sread, pgin and pgout, pgfree and pgin, swrite and pflt, pflt and sread, pflt and scall, vflt and swrite, vflt and sread, vflt and scall, freemem and freeswap.

Weak or No Correlation (< 0.4):

Other variables not mentioned above exhibit weak or no correlation with each other.

vflt and fork

<Axes: xlabel='vflt', ylabel='fork'>



Observations:

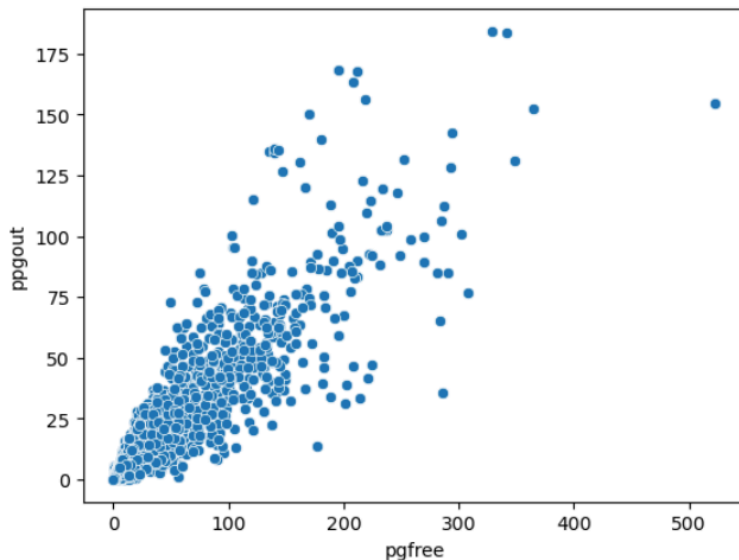
The variables 'vflt' and 'fork' exhibit a positive correlation.

As the count of page faults (vflt) rises, there's a corresponding increase in the occurrence of system forks.

The majority of observations fall within the range of 0-to-1000-page faults (vflt), as depicted in the plot.

pgfree and ppgout

<Axes: xlabel='pgfree', ylabel='pggout'>



Observations:

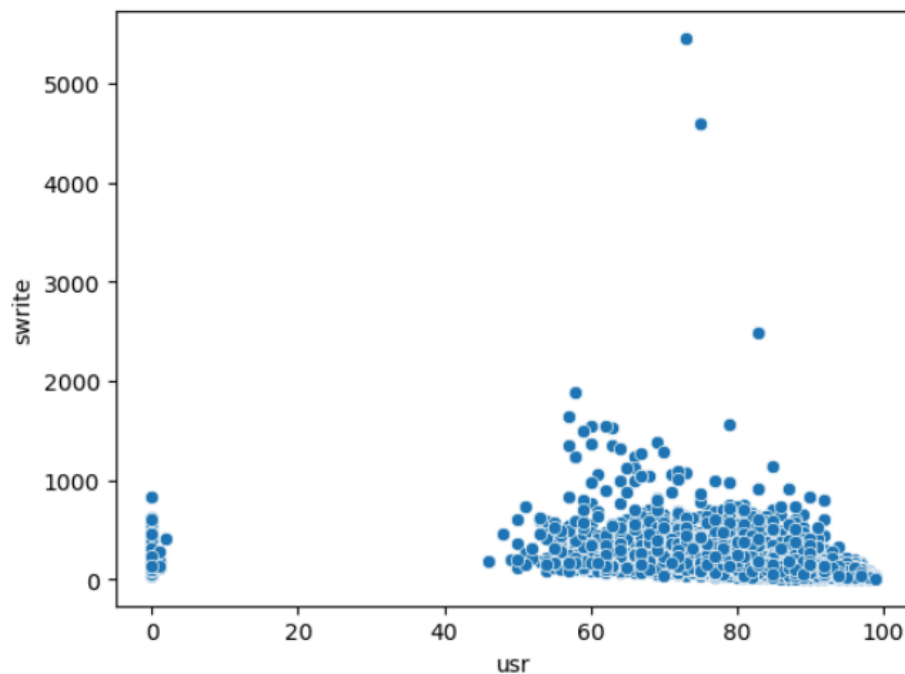
A highly notable positive correlation of 0.92 exists between the variables 'pgfree' and 'pgout'.

As the rate of pages per second being placed on the free list (pgfree) increases, there is a corresponding rise in the rate of pages being paged out per second (pgout).

The bulk of observations fall within the range of 0 to 250 for 'pgfree', as evident from the plot.

usr and swrite

<Axes: xlabel='usr', ylabel='swrite'>



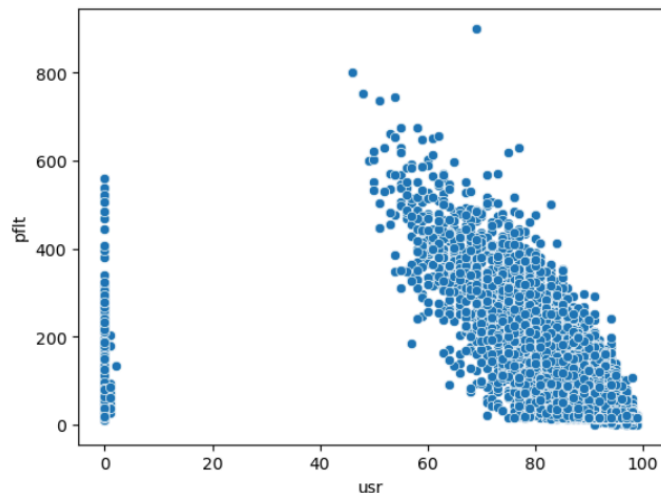
Observations -

There is no significant pattern or correlation between usr and swrite.

The data reveals that the majority of observations either cluster around 0 or within the range of 40% to 100% for 'usr' (the percentage of time CPUs spend running in user mode). Interestingly, there are no observations falling within the range of 2% to 40% for 'usr'.

usr and pflt

<Axes: xlabel='usr', ylabel='pflt'>



Observations -

The patterns are not clearly visible in the plot but we can see that there is a slight negative correlation between user and pflt.

The data reveals that the majority of observations either cluster around 0 or within the range of 40% to 100% for 'usr' (the percentage of time CPUs spend running in user mode). Interestingly, there are no observations falling within the range of 2% to 40% for 'usr'.

Problem 1 - Data Pre-processing Missing Value Treatment (if needed)

```
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 are missing values present in rchar and wchar. To address this issue, we will impute the missing values with the median of the respective column, considering the data type is float.


```

<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

```

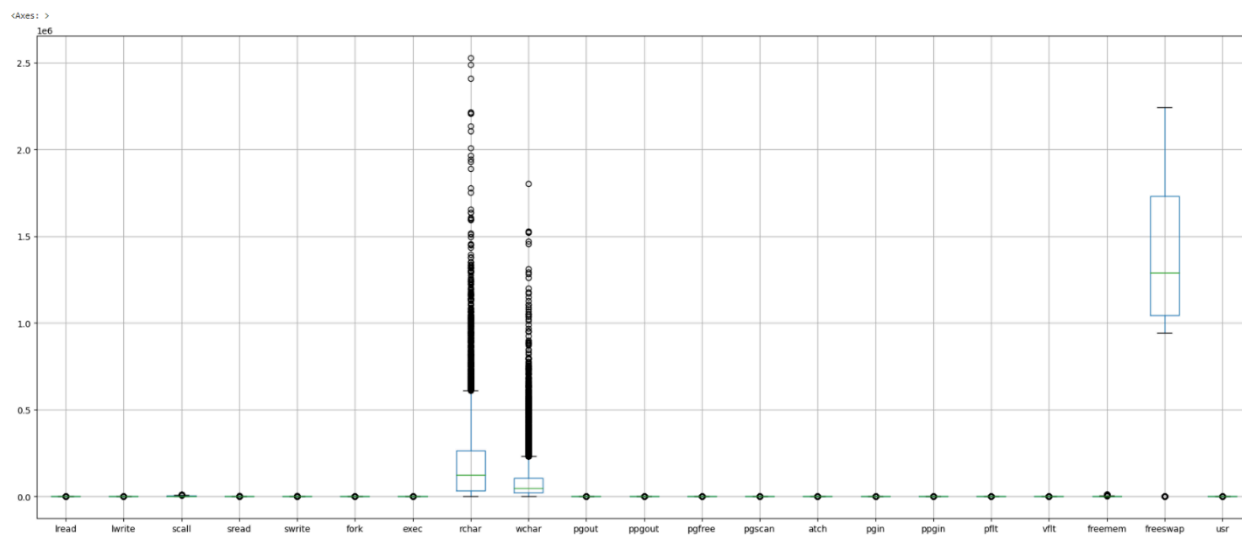
```

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

```

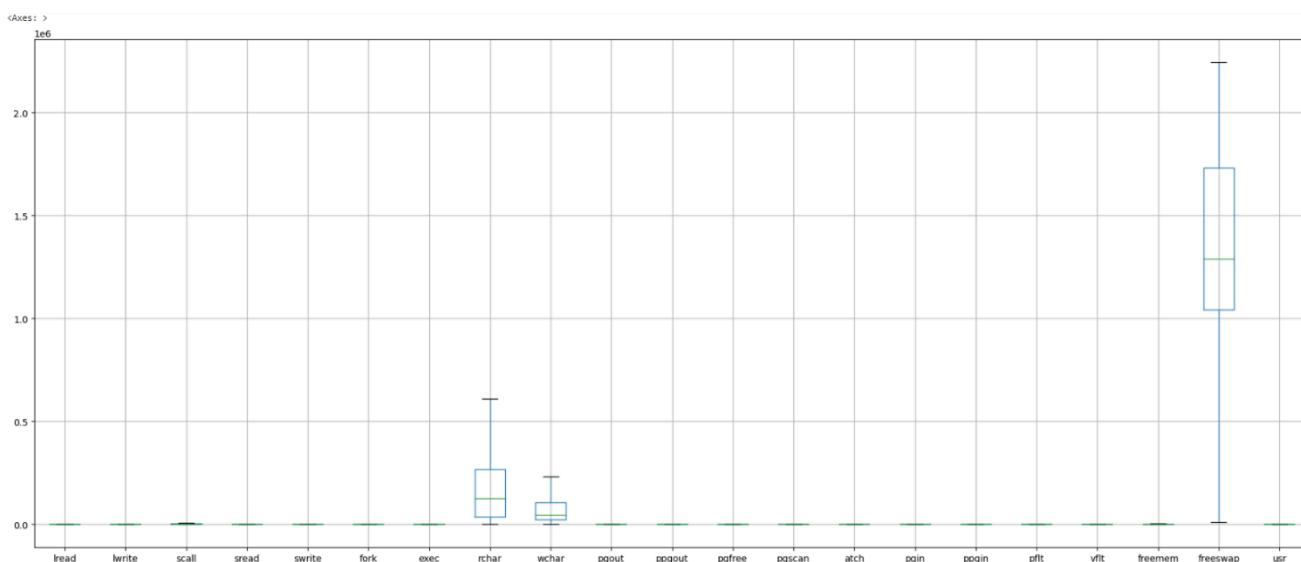
The missing values are imputed with median as shown in the output image above.

Outlier Treatment



As can be seen, outliers are present in every variable.

```
The number of outliers in lread is 753
The number of outliers in lwrite is 1305
The number of outliers in scall is 108
The number of outliers in sread is 340
The number of outliers in swrite is 495
The number of outliers in fork is 943
The number of outliers in exec is 710
The number of outliers in rchar is 465
The number of outliers in wchar is 817
The number of outliers in pgout is 988
The number of outliers in ppgout is 1315
The number of outliers in pgfree is 1555
The number of outliers in pgscan is 1744
The number of outliers in atch is 1209
The number of outliers in pgin is 789
The number of outliers in ppgin is 821
The number of outliers in pflt is 395
The number of outliers in vflt is 484
The number of outliers in freemem is 1185
The number of outliers in freeswap is 294
The number of outliers in usr is 430
```



The outliers have been addressed for all the variables.

Feature Engineering

	lread	lwrite	scall	sread	swrite	fork	exec	rchar	wchar	pgout	...	pgscan	atch	pgin	ppgin	pflt	vflt		runqsz	freemem	freeswap	usr
0	1.0	0.0	2147.0	79.0	68.0	0.2	0.2	40671.0	53995.0	0.0	...	0.0	0.0	1.6	2.6	16.00	26.40		CPU_Bound	4659.125	1730946.0	95.0
1	0.0	0.0	170.0	18.0	21.0	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	4659.125	1869002.0	97.0
2	15.0	3.0	2162.0	159.0	119.0	2.0	2.4	125473.5	31950.0	0.0	...	0.0	1.2	6.0	9.4	150.20	220.20		Not_CPU_Bound	702.000	1021237.0	87.0
3	0.0	0.0	160.0	12.0	16.0	0.2	0.2	125473.5	8670.0	0.0	...	0.0	0.0	0.2	0.2	15.60	16.80		Not_CPU_Bound	4659.125	1863704.0	98.0
4	5.0	1.0	330.0	39.0	38.0	0.4	0.4	125473.5	12185.0	0.0	...	0.0	0.0	1.0	1.2	37.80	47.60		Not_CPU_Bound	633.000	1760253.0	90.0

5 rows × 22 columns

Converting Instant_bookable from 'Not_CPU_Bound' and 'CPU_Bound' to Boolean

	lread	lwrite	scall	sread	swrite	fork	exec	rchar	wchar	pgout	...	pgscan	atch	pgin	ppgin	pflt	vflt	runqsz	freemem	freeswap	usr
0	1.0	0.0	2147.0	79.0	68.0	0.2	0.2	40671.0	53995.0	0.0	...	0.0	0.0	1.6	2.6	16.00	26.40	True	4659.125	1730946.0	95.0
1	0.0	0.0	170.0	18.0	21.0	0.2	0.2	448.0	8385.0	0.0	...	0.0	0.0	0.0	0.0	15.63	16.83	False	4659.125	1869002.0	97.0
2	15.0	3.0	2162.0	159.0	119.0	2.0	2.4	125473.5	31950.0	0.0	...	0.0	1.2	6.0	9.4	150.20	220.20	False	702.000	1021237.0	87.0
3	0.0	0.0	160.0	12.0	16.0	0.2	0.2	125473.5	8670.0	0.0	...	0.0	0.0	0.2	0.2	15.60	16.80	False	4659.125	1863704.0	98.0
4	5.0	1.0	330.0	39.0	38.0	0.4	0.4	125473.5	12185.0	0.0	...	0.0	0.0	1.0	1.2	37.80	47.60	False	633.000	1760253.0	90.0

5 rows × 22 columns

In the "runqsz" column, instances labeled as "Not_CPU_Bound" have been substituted with the value False, while instances labeled as "CPU_Bound" have been substituted with the value True.

Encode the data

	lread	lwrite	scall	sread	swrite	fork	exec	rchar	wchar	pgout	...	pgscan	atch	pgin	ppgin	pflt	vflt	freemem	freeswap	usr	runqsz_True
0	1.0	0.0	2147.0	79.0	68.0	0.2	0.2	40671.0	53995.0	0.0	...	0.0	0.0	1.6	2.6	16.00	26.40	4659.125	1730946.0	95.0	1
1	0.0	0.0	170.0	18.0	21.0	0.2	0.2	448.0	8385.0	0.0	...	0.0	0.0	0.0	0.0	15.63	16.83	4659.125	1869002.0	97.0	0
2	15.0	3.0	2162.0	159.0	119.0	2.0	2.4	125473.5	31950.0	0.0	...	0.0	1.2	6.0	9.4	150.20	220.20	702.000	1021237.0	87.0	0
3	0.0	0.0	160.0	12.0	16.0	0.2	0.2	125473.5	8670.0	0.0	...	0.0	0.0	0.2	0.2	15.60	16.80	4659.125	1863704.0	98.0	0
4	5.0	1.0	330.0	39.0	38.0	0.4	0.4	125473.5	12185.0	0.0	...	0.0	0.0	1.0	1.2	37.80	47.60	633.000	1760253.0	90.0	0

5 rows × 22 columns

The encoding process has been completed successfully. Now, we have a new column called runqsz_True, where the values 1 and 0 correspond to true or false, indicating whether the process is CPU-bound or not CPU-bound, respectively.

Checking for duplicates

```
Number of duplicate rows = 0
```

Since there are no duplicate entries in the dataset, there's no need for us to remove any records.

Train-test split

	lread	lwrite	scall	sread	swrite	fork	exec	rchar	wchar	pgout	...	pgfree	pgscan	atch	pgin	ppgin	pflt	vflt	freemem	freeswap	runqsz_True
0	1.0	0.0	2147.0	79.0	68.0	0.2	0.2	40671.0	53995.0	0.0	...	0.0	0.0	0.0	1.6	2.6	16.00	26.40	4659.125	1730946.0	1
1	0.0	0.0	170.0	18.0	21.0	0.2	0.2	448.0	8385.0	0.0	...	0.0	0.0	0.0	0.0	0.0	15.63	16.83	4659.125	1869002.0	0
2	15.0	3.0	2162.0	159.0	119.0	2.0	2.4	125473.5	31950.0	0.0	...	0.0	0.0	1.2	6.0	9.4	150.20	220.20	702.000	1021237.0	0
3	0.0	0.0	160.0	12.0	16.0	0.2	0.2	125473.5	8670.0	0.0	...	0.0	0.0	0.0	0.2	0.2	15.60	16.80	4659.125	1863704.0	0
4	5.0	1.0	330.0	39.0	38.0	0.4	0.4	125473.5	12185.0	0.0	...	0.0	0.0	0.0	1.0	1.2	37.80	47.60	633.000	1760253.0	0

5 rows × 21 columns

We proceed with splitting the dataset into training and testing sets. X contains all the predictor variables, while y comprises the 'usr' variable, which serves as our target variable. As indicated in the preceding output, the data has been split successfully.

Problem 1- Model Building - Linear regression

Linear Regression Model (using OLS)

```

=====
                        OLS Regression Results
=====
Dep. Variable:          usr      R-squared:          0.794
Model:                  OLS      Adj. R-squared:       0.794
Method:                  Least Squares      F-statistic:       1183.
Date:                    Fri, 19 Apr 2024      Prob (F-statistic): 0.00
Time:                    21:13:06      Log-Likelihood:    -17870.
No. Observations:        6144      AIC:               3.578e+04
Df Residuals:            6123      BIC:               3.592e+04
Df Model:                 20
Covariance Type:         nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
const	85.6549	0.286	299.223	0.000	85.094	86.216
lread	-0.0641	0.009	-7.391	0.000	-0.081	-0.047
lwrite	0.0483	0.013	3.801	0.000	0.023	0.073
scall	-0.0007	6.09e-05	-11.114	0.000	-0.001	-0.001
sread	0.0005	0.001	0.549	0.583	-0.001	0.002
swrite	-0.0056	0.001	-4.020	0.000	-0.008	-0.003
fork	0.0353	0.128	0.277	0.782	-0.215	0.285
exec	-0.3048	0.050	-6.116	0.000	-0.402	-0.207
rchar	-5.176e-06	4.72e-07	-10.964	0.000	-6.1e-06	-4.25e-06
wchar	-5.3e-06	1.01e-06	-5.272	0.000	-7.27e-06	-3.33e-06
pgout	-0.4206	0.087	-4.846	0.000	-0.591	-0.250
ppgout	-0.0240	0.076	-0.315	0.753	-0.174	0.126
pgfree	0.0623	0.046	1.343	0.179	-0.029	0.153
pgscan	-1.06e-14	6.12e-17	-173.090	0.000	-1.07e-14	-1.05e-14
atch	0.6539	0.138	4.722	0.000	0.382	0.925
pgin	0.0230	0.027	0.838	0.402	-0.031	0.077
ppgin	-0.0716	0.019	-3.759	0.000	-0.109	-0.034
pflt	-0.0337	0.002	-17.552	0.000	-0.037	-0.030
vflt	-0.0054	0.001	-3.978	0.000	-0.008	-0.003
freemem	-0.0005	4.93e-05	-9.577	0.000	-0.001	-0.000
freeswap	8.896e-06	1.84e-07	48.436	0.000	8.54e-06	9.26e-06
runqsz_True	-1.6293	0.122	-13.347	0.000	-1.869	-1.390

```

=====
Omnibus:                  1154.251      Durbin-Watson:       2.015
Prob(Omnibus):            0.000      Jarque-Bera (JB):    2416.770
Skew:                     -1.106      Prob(JB):            0.00
Kurtosis:                  5.133      Cond. No.            2.01e+22
=====

```

Notes:

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

[2] The smallest eigenvalue is 3.03e-29. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.

Given our current R-squared value of 79.4, our next step is to investigate multicollinearity and eliminate any variables contributing to it in an effort to boost the R-squared value.

Let's take a look at the VIF values of the predictors –

```

VIF values:
const      25.505365
lread      5.329904
lwrite     4.321098
scall      2.960670
sread      6.411855
swrite     5.597305
fork       13.007090
exec       3.200317
rchar      2.122906
wchar      1.595891
pgout      11.240801
ppgout     29.300444
pgfree     16.538594
pgscan     NaN
atch       1.873416
pgin       13.824074
ppgin      13.987579
pflt       11.958409
vflt       15.573372
freemem    1.958029
freeswap   1.837506
runqsz_True 1.156618
dtype: float64

C:\Users\Gagan\anaconda3\Lib\site-packages\statsmodels\regression\linear_model.py:1781: RuntimeWarning: invalid value encountered in scalar divide
  return 1 - self.ssr/self.centered_tss

```

AS few predictors have VIF values > 2 therefore there is some multicollinearity in the data

We remove those predictors with multicollinearity due to which there is least impact on the adjusted R²

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

- 1) Initially, we'll exclude the variable "fork" due to its VIF value exceeding 10.

```
R-squared(fork): 0.794  
Adjusted R-squared(fork): 0.794
```

- 2) Removing predictor 'pgout' as VIF>10

```
R-squared(pgout): 0.794  
Adjusted R-squared(pgout): 0.793
```

- 3) Removing predictor 'pgfree' as VIF>10

```
R-squared(pgfree): 0.794  
Adjusted R-squared(pgfree): 0.794
```

- 4) Removing predictor 'ppgout' as VIF>10

```
R-squared(ppgout): 0.794  
Adjusted R-squared(ppgout): 0.794
```

- 5) Removing predictor 'pgin' as VIF>10

```
R-squared(pgin): 0.794  
Adjusted R-squared(pgin): 0.794
```

- 6) Removing predictor 'ppgin' as VIF>10

```
R-squared(ppgin): 0.794  
Adjusted R-squared(ppgin): 0.793
```

- 7) Removing predictor 'pflt' as VIF>10

```
R-squared(pflt): 0.784  
Adjusted R-squared(pflt): 0.783
```

- 8) Removing predictor 'vflt' as VIF>10

```
R-squared(vflt): 0.794  
Adjusted R-squared(vflt): 0.793
```

Observations:

It appears that for most predictor variables with a VIF greater than 10, their individual R-squared values are comparable to the overall model's R-squared. This suggests that removing these variables may have minimal to no impact on the model's explanatory power. In other words, dropping these variables is unlikely to substantially affect the model's performance.

Dropping all of these variables now

```
R-squared: 0.739  
Adjusted R-squared: 0.738
```

let's check the VIF of the predictors after removing these variables

VIF values:

```
const      25.505365
lread      5.329904
lwrite     4.321098
scall      2.960670
sread      6.411855
swrite     5.597305
fork       13.007090
exec       3.200317
rchar      2.122906
wchar      1.595891
pgout      11.240801
ppgout     29.300444
pgfree     16.538594
pgscan     NaN
atch       1.873416
pgin       13.824074
ppgin      13.987579
pflt       11.958409
vflt       15.573372
freemem    1.958029
freeswap   1.837506
runqsz_True 1.156618
dtype: float64
```

```
C:\Users\Gagan\anaconda3\Lib\site-packages\statsmodels\regression\linear_model.py:1781: RuntimeWarning: invalid value encountered in scalar divide
return 1 - self.ssr/self.centered_tss
```

Now that 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.

```
OLS Regression Results
=====
Dep. Variable:          usr      R-squared:                0.739
Model:                  OLS      Adj. R-squared:           0.738
Method:                 Least Squares      F-statistic:        1334.
Date:                   Fri, 19 Apr 2024    Prob (F-statistic):    0.00
Time:                   21:13:07           Log-Likelihood:       -18605.
No. Observations:       6144              AIC:                3.724e+04
Df Residuals:           6130              BIC:                3.733e+04
Df Model:               13
Covariance Type:        nonrobust
=====
               coef      std err          t      P>|t|      [0.025      0.975]
-----
const          86.5003      0.316     273.844      0.000      85.881      87.120
lread         -0.1481      0.009    -15.691      0.000     -0.167     -0.130
lwrite         0.1573      0.014     11.291      0.000      0.130      0.185
scall         -0.0005      6.77e-05    -7.068      0.000     -0.001     -0.000
sread         -0.0029      0.001     -2.670      0.008     -0.005     -0.001
swrite         -0.0181      0.001    -12.214      0.000     -0.021     -0.015
exec          -1.5560      0.039    -39.815      0.000     -1.633     -1.479
rchar         -7.443e-06    5.23e-07   -14.245      0.000    -8.47e-06    -6.42e-06
wchar         1.51e-06     1.1e-06     1.375      0.169    -6.43e-07     3.66e-06
pgscan        1.578e-13    6.1e-16    258.685      0.000     1.57e-13     1.59e-13
atch          -0.0540      0.134     -0.403      0.687     -0.317      0.209
pgin          -0.0824      0.010     -8.321      0.000     -0.102     -0.063
freemem       -0.0005      5.39e-05   -10.099      0.000     -0.001     -0.000
freeswap       8.26e-06     2.02e-07    40.791      0.000     7.86e-06     8.66e-06
runqsz_True   -1.3247      0.137     -9.677      0.000     -1.593     -1.056
=====
Omnibus:            817.459    Durbin-Watson:           2.010
Prob(Omnibus):      0.000    Jarque-Bera (JB):        1386.133
Skew:               -0.894    Prob(JB):                1.01e-301
Kurtosis:           4.488    Cond. No.                4.07e+22
=====
```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 7.36e-30. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.

Now as observed from above the predictors, 'wchar' and 'atch' has p-value>0.05 we remove that and build the model

```

=====
                        OLS Regression Results
=====
Dep. Variable:          usr      R-squared:          0.794
Model:                  OLS      Adj. R-squared:       0.794
Method:                 Least Squares      F-statistic:       1245.
Date:                   Fri, 19 Apr 2024    Prob (F-statistic):    0.00
Time:                   21:13:07          Log-Likelihood:      -17870.
No. Observations:       6144             AIC:               3.578e+04
Df Residuals:           6124             BIC:               3.592e+04
Df Model:               19
Covariance Type:        nonrobust
=====
                        coef      std err          t      P>|t|      [0.025      0.975]
-----
const                85.6739         0.285      300.247      0.000      85.115      86.233
lread               -0.0643         0.009      -7.416      0.000     -0.081     -0.047
lwrite              0.0484         0.013       3.807      0.000      0.023      0.073
scall               -0.0007        6.09e-05     -11.102      0.000     -0.001     -0.001
sread              0.0005         0.001       0.542      0.588     -0.001      0.002
swrite             -0.0056         0.001      -4.011      0.000     -0.008     -0.003
fork               0.0321         0.128       0.252      0.801     -0.218      0.282
exec              -0.3032         0.050      -6.089      0.000     -0.401     -0.206
rchar             -5.204e-06      4.71e-07     -11.050      0.000     -6.13e-06     -4.28e-06
wchar             -5.282e-06      1.01e-06     -5.255      0.000     -7.25e-06     -3.31e-06
pgout             -0.4167         0.087      -4.808      0.000     -0.587     -0.247
ppgout            -0.0268         0.076      -0.351      0.725     -0.176      0.123
pgfree            0.0626         0.046       1.348      0.178     -0.028      0.154
pgscan            -2.082e-14      8.95e-17     -232.659      0.000     -2.1e-14     -2.06e-14
atch              0.6545         0.138       4.727      0.000      0.383      0.926
ppgin             -0.0567         0.007      -8.416      0.000     -0.070     -0.043
pflt              -0.0338         0.002     -17.643      0.000     -0.038     -0.030
vflt              -0.0053         0.001      -3.910      0.000     -0.008     -0.003
freemem           -0.0005         4.93e-05     -9.574      0.000     -0.001     -0.000
freeswap          8.886e-06      1.83e-07     48.492      0.000      8.53e-06      9.24e-06
runqsz_True       -1.6286         0.122     -13.343      0.000     -1.868     -1.389
=====
Omnibus:              1155.714      Durbin-Watson:       2.015
Prob(Omnibus):         0.000      Jarque-Bera (JB):     2423.249
Skew:                  -1.107      Prob(JB):             0.00
Kurtosis:              5.137      Cond. No.             1.00e+22
=====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.22e-28. 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.

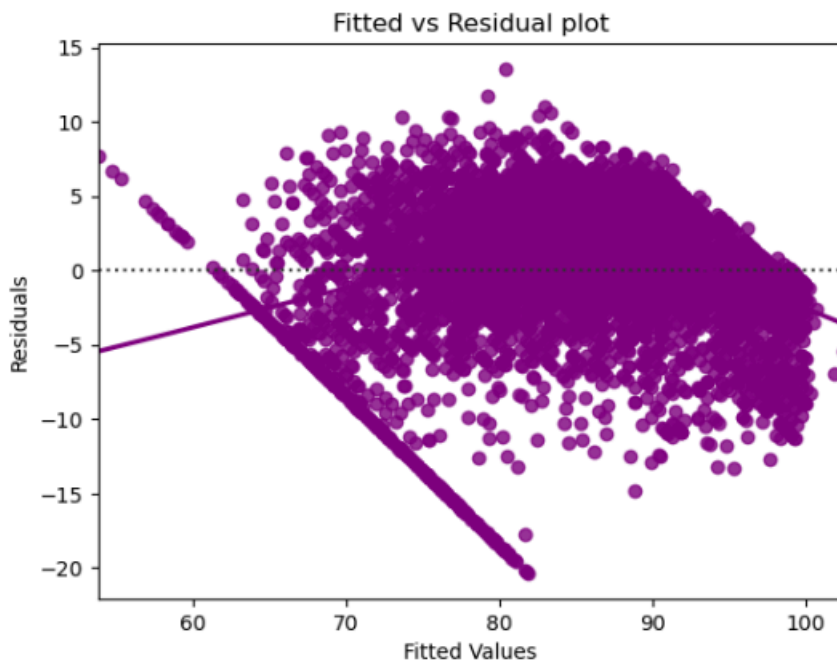
Testing the Assumptions of Linear Regression

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

1. Linearity
2. Independence
3. Homoscedasticity
4. Normality of error terms
5. No strong Multicollinearity

Linearity and Independence of predictors

	Actual Values	Fitted Values	Residuals
0	94.0	89.752157	4.247843
1	96.0	88.171298	7.828702
2	77.0	77.616803	-0.616803
3	93.0	97.890175	-4.890175
4	86.0	86.561922	-0.561922



There is no significant pattern

Test for Normality

```
C:\Users\Gagan\anaconda3\Lib\site-packages\scipy\stats\_morestats.py:1882: UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
ShapiroResult(statistic=0.9399019479751587, pvalue=2.1019476964872256e-44)
```

Test for Homoscedasticity

0.012154361139571362

The model built `olsmod_6` satisfies all assumptions of Linear Regression

```
OLS Regression Results
Dep. Variable:  usr      R-squared:  0.794
Model:  OLS      Adj. R-squared:  0.793
Method:  Least Squares      F-statistic:  1242.
Date:  Fri, 19 Apr 2024 Prob (F-statistic): 0.00
Time:  21:13:12      Log-Likelihood: -17877.
No. Observations: 6144      AIC:  3.579e+04
Df Residuals:  6124      BIC:  3.593e+04
Df Model:  19
Covariance Type: nonrobust
```

	coef	std err	t	P> t	[0.025	0.975]
const	85.6898	0.286	299.182	0.000	85.128	86.251
lread	-0.0668	0.009	-7.713	0.000	-0.084	-0.050
lwrite	0.0508	0.013	4.004	0.000	0.026	0.076
scall	-0.0007	6.1e-05	-11.047	0.000	-0.001	-0.001
sread	0.0006	0.001	0.583	0.560	-0.001	0.002
swrite	-0.0056	0.001	-4.028	0.000	-0.008	-0.003
fork	0.0535	0.128	0.419	0.675	-0.197	0.304
exec	-0.3028	0.050	-6.069	0.000	-0.401	-0.205
rchar	-5.393e-06	4.69e-07	-11.496	0.000	-6.31e-06	-4.47e-06
wchar	-5.306e-06	1.01e-06	-5.272	0.000	-7.28e-06	-3.33e-06
pgout	-0.4010	0.087	-4.623	0.000	-0.571	-0.231
ppgout	-0.0414	0.076	-0.543	0.587	-0.191	0.108
pgfree	0.0609	0.046	1.311	0.190	-0.030	0.152
pgscan	-1.561e-14	7.39e-17	-211.122	0.000	-1.58e-14	-1.55e-14
atch	0.6629	0.139	4.783	0.000	0.391	0.935
pgin	-0.0736	0.010	-7.567	0.000	-0.093	-0.055
pflt	-0.0337	0.002	-17.546	0.000	-0.037	-0.030
vflt	-0.0055	0.001	-4.005	0.000	-0.008	-0.003
freemem	-0.0005	4.94e-05	-9.589	0.000	-0.001	-0.000
freeswap	8.871e-06	1.84e-07	48.280	0.000	8.51e-06	9.23e-06
runqsz_True	-1.6220	0.122	-13.275	0.000	-1.862	-1.382

```
Omnibus: 1149.712 Durbin-Watson: 2.015
Prob(Omnibus): 0.000 Jarque-Bera (JB): 2402.792
Skew: -1.103 Prob(JB): 0.00
Kurtosis: 5.127 Cond. No. 5.16e+22
```

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 4.58e-30. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.

The model equation will be as follows:

```
usr = 86.50033823675241 + -0.14806765525164184 * ( lread ) + 0.15727702982106334 * ( lwrite )  
+ -0.000478830255032549 * ( scall ) + -0.0029127025131697806 * ( sread ) +  
-0.018149980926498173 * ( swrite ) + -1.5559973118441364 * ( exec ) +  
-7.443259987671406e-06 * ( rchar ) + 1.5095931737171688e-06 * ( wchar ) +  
1.578219677698472e-13 * ( pgscan ) + -0.05403060711564556 * ( atch ) +  
-0.08238648737961496 * ( pgin ) + -0.0005440167919623178 * ( freemem ) +  
8.259861605713757e-06 * ( freeswap ) + -1.324707901823066 * ( runqsz_True )
```

Observation

- For every 1 unit increase in lread (reads per second between system memory and user memory), the usr decreases by a factor of -0.149.
- For every 1 unit increase in lwrite (writes per second between system memory and user memory), the usr increases by a factor of 0.159.
- For every 1 unit increase in scall (system calls of all types per second), the usr decreases by a factor of 0.000478.
- For every 1 unit increase in sread (system read calls per second), the usr decreases by a factor of 0.00307.
- For every 1 unit increase in swrite (system write calls per second), the usr decreases by a factor of 0.0177.
- For every 1 unit increase in exec (system exec calls per second), the usr decreases by a factor of 1.561.
- For every 1 unit increase in rchar (characters transferred per second by system read calls), the usr decreases by a factor of approximately 7.21e-06.
- For every 1 unit increase in pgin (page-in requests per second), the usr decreases by a factor of 0.082.
- For every 1 unit increase in freemem (memory pages available to user processes), the usr decreases by a factor of 0.000537.
- For every 1 unit increase in freeswap (disk blocks available for page swapping), the usr increases by a factor of approximately 8.26e-06.
- If the property has a run queue size (runqsz), the usr decreases by a factor of 1.29.

	lread	lwrite	scall	sread	swrite	fork	exec	rchar	wchar	pgout	...	pgscan	atch	pgin	ppgin	pflt	vflt	freemem	freeswap	usr	runqsz_True
0	1.0	0.0	2147.0	79.0	68.0	0.2	0.20	40671.0	53995.0	0.0	...	0.0	0.0	1.6000	2.60	16.00	26.40	4659.125	1730946.0	95.0	1
1	0.0	0.0	170.0	18.0	21.0	0.2	0.20	448.0	8385.0	0.0	...	0.0	0.0	0.0000	0.00	15.63	16.83	4659.125	1869002.0	97.0	0
2	15.0	3.0	2162.0	159.0	119.0	2.0	2.40	125473.5	31950.0	0.0	...	0.0	1.2	6.0000	9.40	150.20	220.20	702.000	1021237.0	87.0	0
3	0.0	0.0	160.0	12.0	16.0	0.2	0.20	125473.5	8670.0	0.0	...	0.0	0.0	0.2000	0.20	15.60	16.80	4659.125	1863704.0	98.0	0
4	5.0	1.0	330.0	39.0	38.0	0.4	0.40	125473.5	12185.0	0.0	...	0.0	0.0	1.0000	1.20	37.80	47.60	633.000	1760253.0	90.0	0
...
8187	16.0	12.0	3009.0	360.0	244.0	1.6	5.81	405250.0	85282.0	6.0	...	0.0	0.6	23.5125	33.60	139.28	270.74	387.000	986647.0	80.0	1
8188	4.0	0.0	1596.0	170.0	146.0	2.4	1.80	89489.0	41764.0	3.8	...	0.0	0.8	3.8000	4.40	122.40	212.60	263.000	1055742.0	90.0	0
8189	16.0	5.0	3116.0	289.0	190.0	0.6	0.60	325948.0	52640.0	0.4	...	0.0	0.4	23.5125	33.60	60.20	219.80	400.000	969106.0	87.0	0
8190	32.0	25.0	5180.0	254.0	179.0	1.2	1.20	62571.0	29505.0	1.4	...	0.0	0.4	23.0500	24.25	93.19	202.81	141.000	1022458.0	83.0	1
8191	2.0	0.0	985.0	55.0	46.0	1.6	4.80	111111.0	22256.0	0.0	...	0.0	0.2	3.4000	6.20	91.80	110.00	659.000	1756514.0	94.0	1

8192 rows x 22 columns

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

Predictions

```
rmse1 = np.sqrt(mean_squared_error(y_train, y_pred_train))
rmse1
```

4.998761093503711

```
rmse2 = np.sqrt(mean_squared_error(y_test, y_pred_test))
rmse2
```

5.23218971637551

Observations –

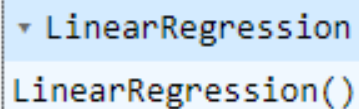
Columns not present in the training data are dropped from the test data to maintain feature consistency.

Model performance is assessed using root mean squared error (RMSE) on both the training and test datasets.

The RMSE value for the training data is approximately 4.99.

The RMSE value for the test data is approximately 5.23.

Problem 1- Model Building - Linear regression



```
LinearRegression()
LinearRegression()
```

Linear Regression model has been created

Let us explore the coefficients for each of the independent attributes

```
The coefficient for const is 0.0
The coefficient for lread is -0.14806765525157983
The coefficient for lwrite is 0.15727702982092817
The coefficient for scall is -0.00047883025503225635
The coefficient for sread is -0.0029127025131867874
The coefficient for swrite is -0.018149980926494926
The coefficient for exec is -1.5559973118434975
The coefficient for rchar is -7.443259987617579e-06
The coefficient for wchar is 1.5095931736649332e-06
The coefficient for pgscan is 6.661338147750939e-16
The coefficient for atch is -0.054030607118203455
The coefficient for pgin is -0.08238648737968349
The coefficient for freemem is -0.0005440167919655747
The coefficient for freeswap is 8.259861605658392e-06
The coefficient for runqsz_True is -1.3247079018233419
```

Let us check the intercept for the model

```
The intercept for our model is 86.50033823687217
```

R square on training data

0.7388207780955344

RMSE on Training data
4.9987610935037115

RMSE on Testing data
5.232189716374651

When comparing the RMSE values computed using Statsmodels and Sklearn on both the training and testing data, it can be observed that they are identical. Specifically, the RMSE value for the training data is 4.99, while for the testing data, it is 5.23.

Impact of Relevant Variables:

lread (Reads) and lwrite (Writes): These variables have positive coefficients, indicating that an increase in transfers per second between system memory and user memory is associated with a higher outcome value. This suggests that optimizing memory read and write operations can lead to improved system performance.

exec (Executions): With a negative coefficient, an increase in the number of system exec calls per second is associated with a decrease in the outcome. This implies that excessive execution of system processes may have a detrimental effect on overall system performance.

Problem 1 - Business Insights & Recommendations

Key Takeaways -

- **Optimize Memory Operations:** Given the positive impact of memory read and write operations on the outcome, focusing on optimizing these operations can lead to better system performance. This might involve improving memory management algorithms or enhancing hardware capabilities related to memory transfers.
 - **Monitor and Control System Executions:** Since a higher frequency of system exec calls has a negative impact on the outcome, it's essential to monitor and control the number of such executions. This could involve optimizing system processes, reducing unnecessary executions, or implementing efficient scheduling algorithms.
 - **Resource Allocation:** Considering the impact of variables related to memory utilization and system executions, it's crucial to allocate system resources effectively. Balancing resource allocation between memory-intensive tasks and system processes can help optimize overall system performance.
 - **Proactive System Monitoring:** Continuous monitoring of system performance metrics such as memory utilization, system calls, and run queue size is vital. This allows for early detection of performance bottlenecks or resource constraints, enabling proactive measures to maintain system efficiency.
 - **Further Investigation:** While the provided equation sheds light on the impact of various variables, further investigation may be necessary to understand the complex interactions and dependencies within the system. This could involve conducting additional experiments, analysing historical data trends, or exploring advanced modelling techniques.
 - By incorporating these insights into system management practices, businesses can enhance the efficiency and reliability of their systems, ultimately improving user experience and operational effectiveness.
-

Problem 2 - Define the problem and perform exploratory Data Analysis

Objective 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=verlow, 2, 3, 4=high
- Media exposure (binary) Good, not good
- Contraceptive method used (class attribute) No,Yes

Data Structure

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

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

no.of rows: 1473 no.of columns: 10

```

<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 Check for null values in the dataset:

```

Wife_age           71
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

```

To Check the duplicate values in the dataset

```
Number of duplicate rows = 80
```

	Wife_age	Wife_education	Husband_education	No_of_children_born	Wife_religion	Wife_Working	Husband_Occupation	Standard_of_living_index	Media_exposure	Contraceptive_method_used
79	38.0	Tertiary	Tertiary	1.0	Scientology	Yes	1	Very High	Exposed	No
167	26.0	Tertiary	Tertiary	1.0	Scientology	No	1	Very High	Exposed	No
224	47.0	Tertiary	Tertiary	4.0	Scientology	No	1	Very High	Exposed	No
270	30.0	Tertiary	Tertiary	2.0	Scientology	No	1	Very High	Exposed	No
299	26.0	Tertiary	Tertiary	1.0	Scientology	No	1	Very High	Exposed	No
...
1367	44.0	Tertiary	Tertiary	5.0	Scientology	Yes	1	Very High	Exposed	Yes
1387	NaN	Secondary	Tertiary	2.0	Scientology	Yes	2	Very High	Exposed	Yes
1423	NaN	Tertiary	Tertiary	2.0	Non-Scientology	No	1	Very High	Exposed	Yes
1440	NaN	Tertiary	Tertiary	1.0	Non-Scientology	Yes	2	Very High	Exposed	Yes
1447	NaN	Tertiary	Tertiary	2.0	Non-Scientology	Yes	2	Very High	Exposed	Yes

80 rows x 10 columns

Observations

Removing the duplicate value

```
Number of duplicate rows = 0
```

Duplicate rows have been taken care of

```
no.of rows: 1393 no.of columns: 10
```

Checking the null values in the dataset

```

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

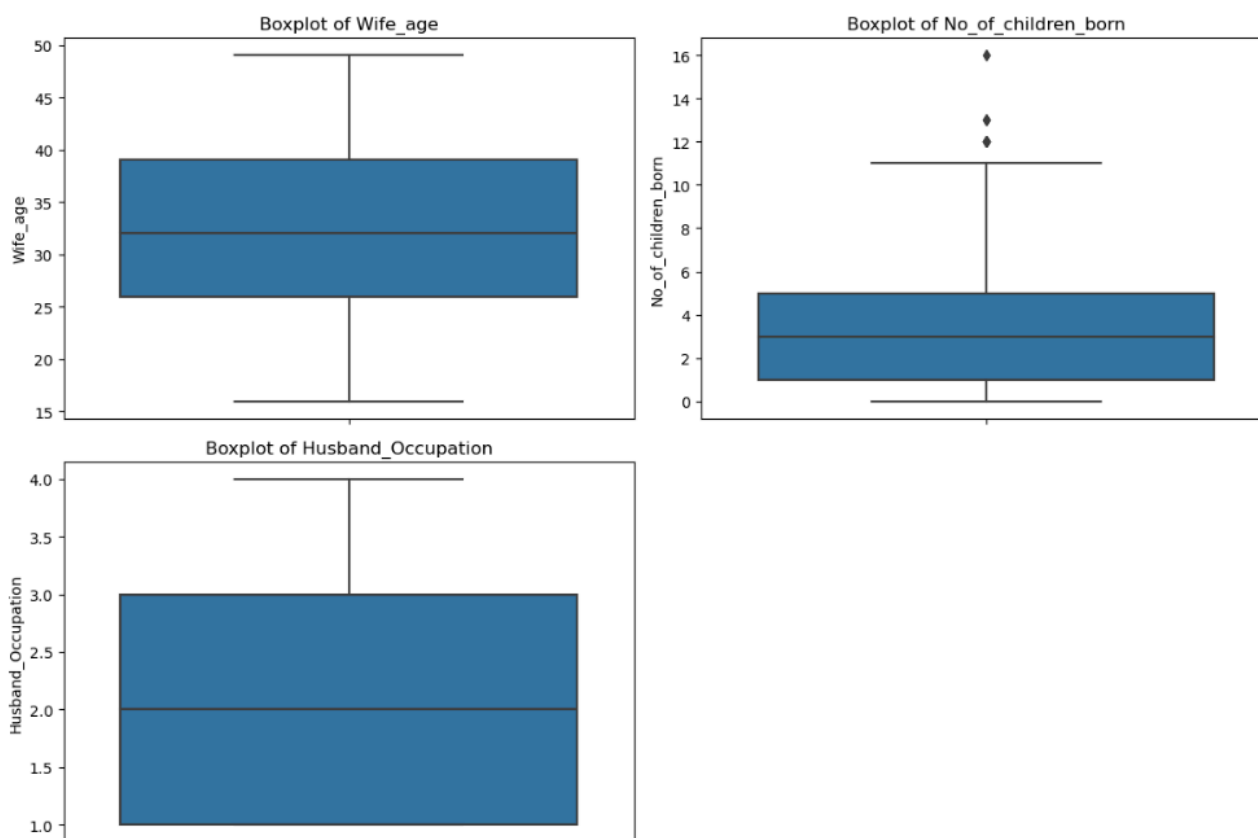
```

	count	mean	std	min	25%	50%	75%	max
Wife_age	1326.0	32.557315	8.289259	16.0	26.0	32.0	39.0	49.0
No_of_children_born	1372.0	3.290816	2.399697	0.0	1.0	3.0	5.0	16.0
Husband_Occupation	1393.0	2.174444	0.854590	1.0	1.0	2.0	3.0	4.0

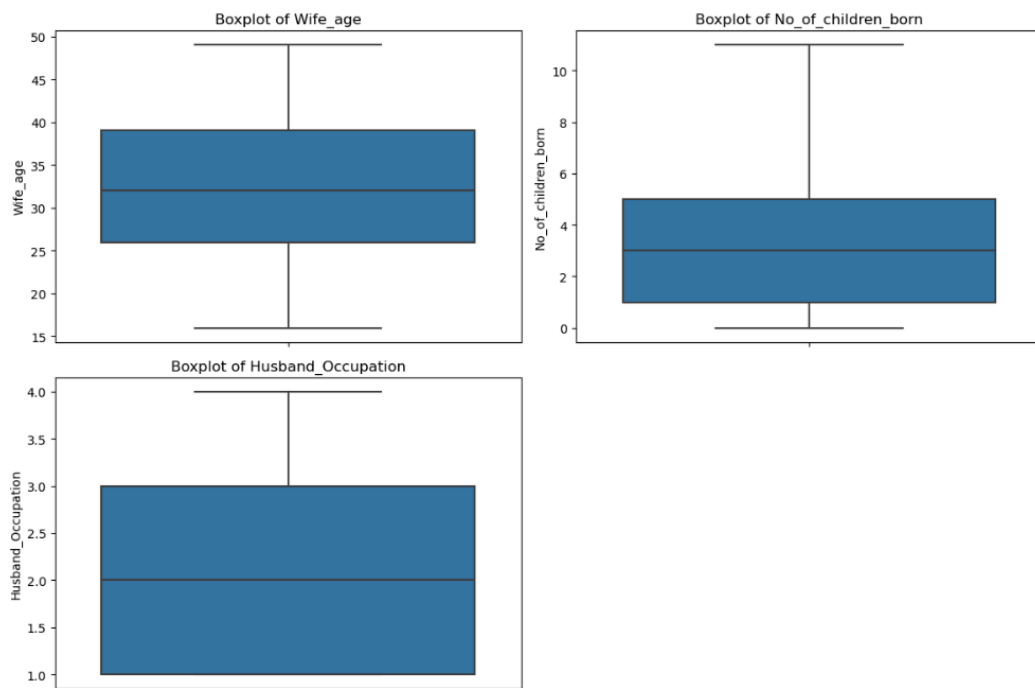
Problem 2 - Data Pre-processing

Outlier Treatment

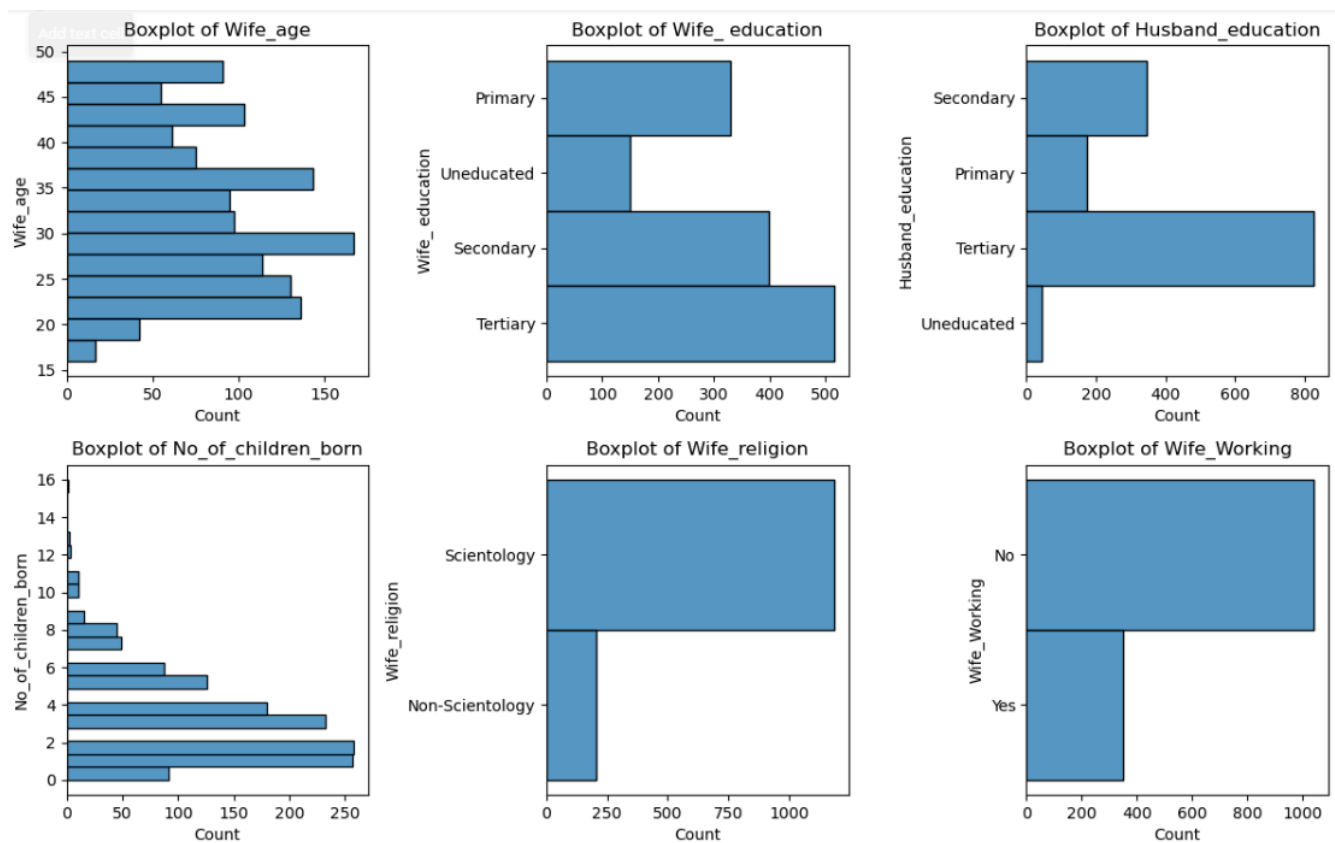
To check for outliers, we will be plotting the box plots.

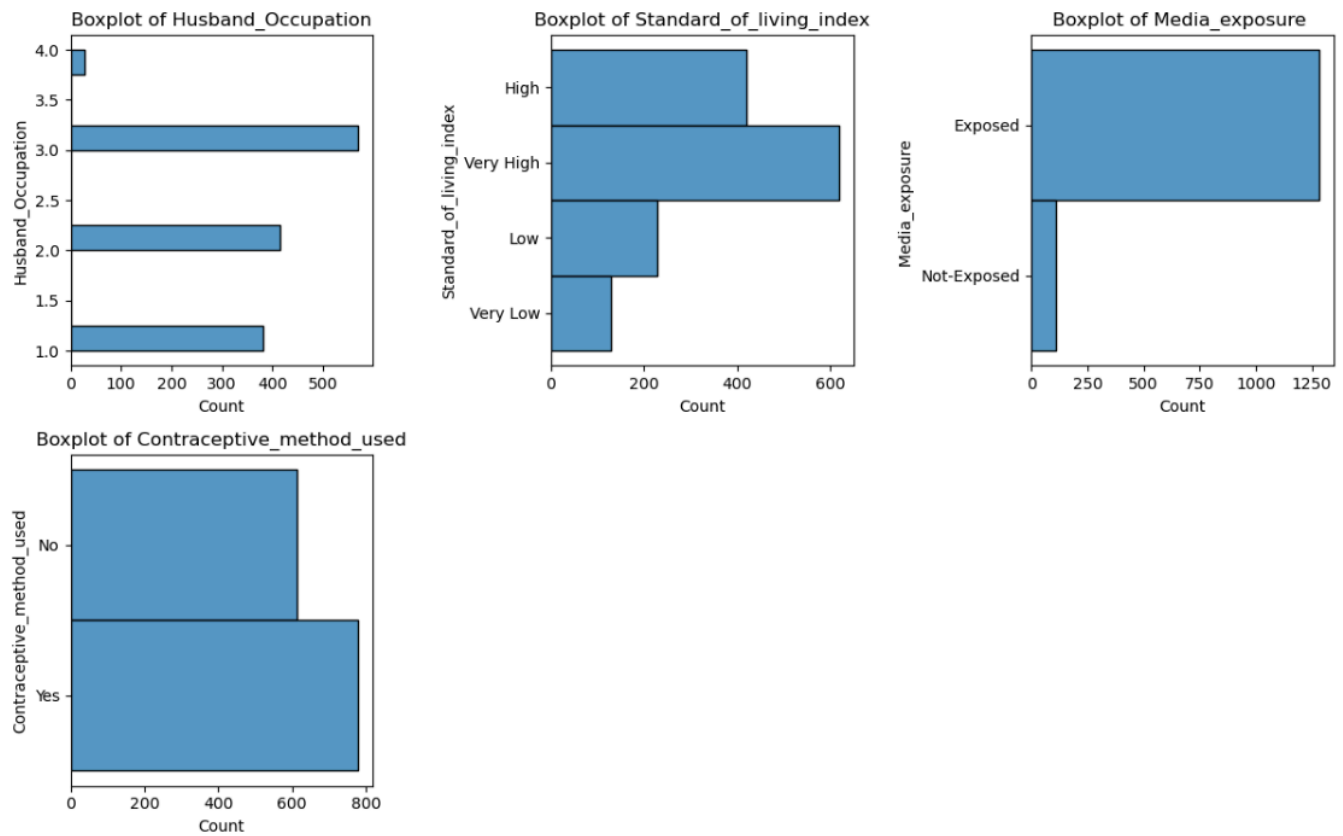


After treating the outliers

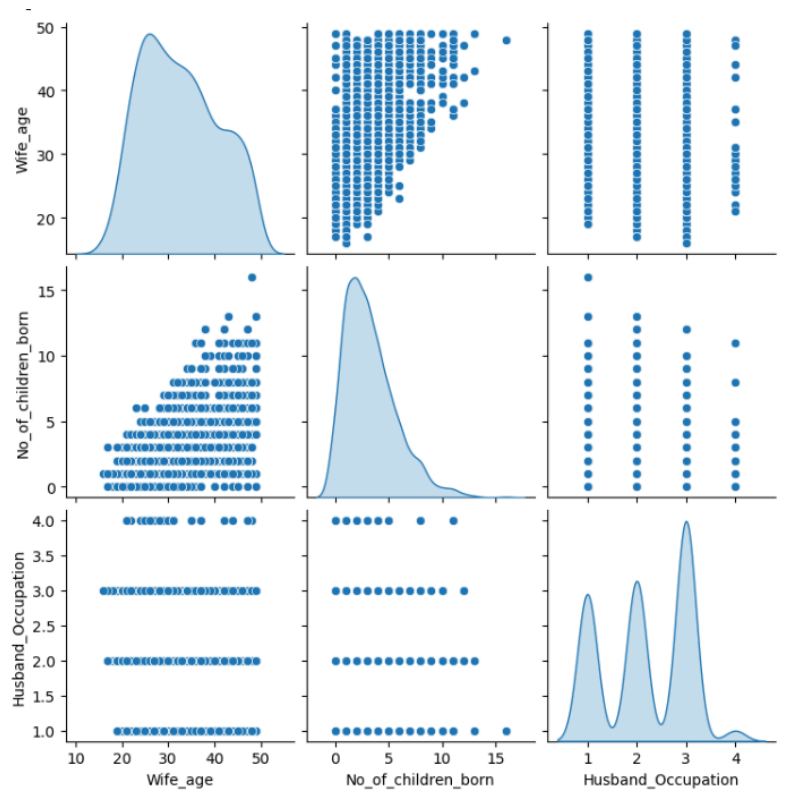


Univariate Analysis





Bivariate Analysis

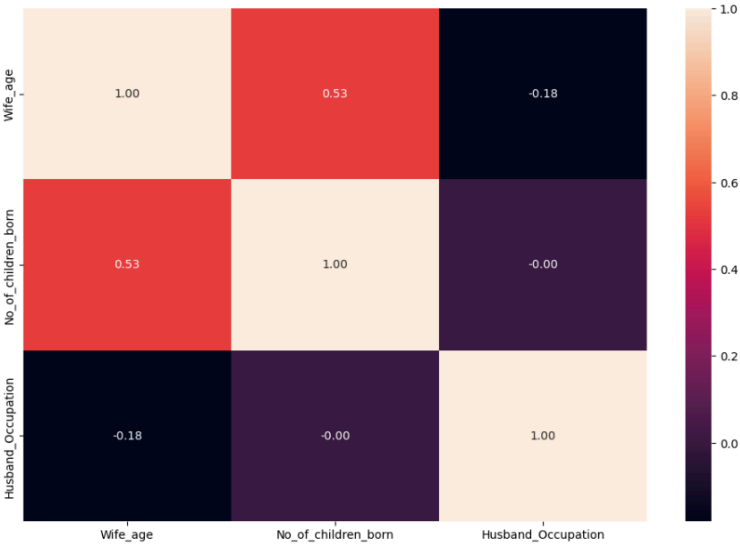


- There is no variance in the depth of variables, scattered data will help models to perform well
- Each variable has equivalent contribution of Contraceptive_method_used dependent variable

Checking for Correlations:

	Wife_age	No_of_children_born	Husband_Occupation
Wife_age	1.000000	0.539520	-0.189798
No_of_children_born	0.539520	1.000000	-0.025032
Husband_Occupation	-0.189798	-0.025032	1.000000

Multivariate Analysis



Converting all objects to categorical codes

	Wife_age	Wife_education	Husband_education	No_of_children_born	Wife_religion	Wife_Working	Husband_Occupation	Standard_of_living_index	Media_exposure	Contraceptive_method_used
0	24.0	2	3	3.0	Scientology	No	2	High	Exposed	No
1	45.0	1	3	10.0	Scientology	No	3	Very High	Exposed	No
2	43.0	2	3	7.0	Scientology	No	3	Very High	Exposed	No
3	42.0	3	2	9.0	Scientology	No	3	High	Exposed	No
4	36.0	3	3	8.0	Scientology	No	3	Low	Exposed	No

Top 5 record of the dataset

	Wife_age	Wife_education	Husband_education	No_of_children_born	Wife_religion	Wife_Working	Husband_Occupation	Standard_of_living_index	Media_exposure	Contraceptive_method_used
0	24.0	2	3	3.0	1	0	2	3	1	0
1	45.0	1	3	10.0	1	0	3	1	1	0
2	43.0	2	3	7.0	1	0	3	1	1	0
3	42.0	3	2	9.0	1	0	3	3	1	0
4	36.0	3	3	8.0	1	0	3	2	1	0

To view describe the dataset

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Wife_age	1326.0	NaN	NaN	NaN	32.557315	8.289259	16.0	26.0	32.0	39.0	49.0
Wife_education	1393.0	4.0	4.0	515.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Husband_education	1393.0	4.0	4.0	827.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
No_of_children_born	1372.0	NaN	NaN	NaN	3.290816	2.399697	0.0	1.0	3.0	5.0	16.0
Wife_religion	1393.0	2.0	1.0	1186.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Wife_Working	1393.0	2.0	0.0	1043.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Husband_Occupation	1393.0	NaN	NaN	NaN	2.174444	0.854590	1.0	1.0	2.0	3.0	4.0
Standard_of_living_index	1393.0	4.0	1.0	618.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Media_exposure	1393.0	2.0	1.0	1284.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Contraceptive_method_used	1393.0	2.0	1.0	779.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Info of the Dataset

```
<class 'pandas.core.frame.DataFrame'>
Index: 1393 entries, 0 to 1472
Data columns (total 10 columns):
#   Column                                Non-Null Count  Dtype
---  ---
0   Wife_age                             1326 non-null   float64
1   Wife_education                       1393 non-null   object
2   Husband_education                    1393 non-null   object
3   No_of_children_born                  1372 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: 119.7+ KB
```

convert the object into int8 datatype

```
<class 'pandas.core.frame.DataFrame'>
Index: 1393 entries, 0 to 1472
Data columns (total 10 columns):
#   Column                                Non-Null Count  Dtype
---  ---
0   Wife_age                             1326 non-null   float64
1   Wife_education                       1393 non-null   int8
2   Husband_education                    1393 non-null   int8
3   No_of_children_born                  1372 non-null   float64
4   Wife_religion                        1393 non-null   int8
5   Wife_Working                         1393 non-null   int8
6   Husband_Occupation                  1393 non-null   int64
7   Standard_of_living_index              1393 non-null   int8
8   Media_exposure                       1393 non-null   int8
9   Contraceptive_method_used            1393 non-null   int8
dtypes: float64(2), int64(1), int8(7)
memory usage: 53.1 KB
```

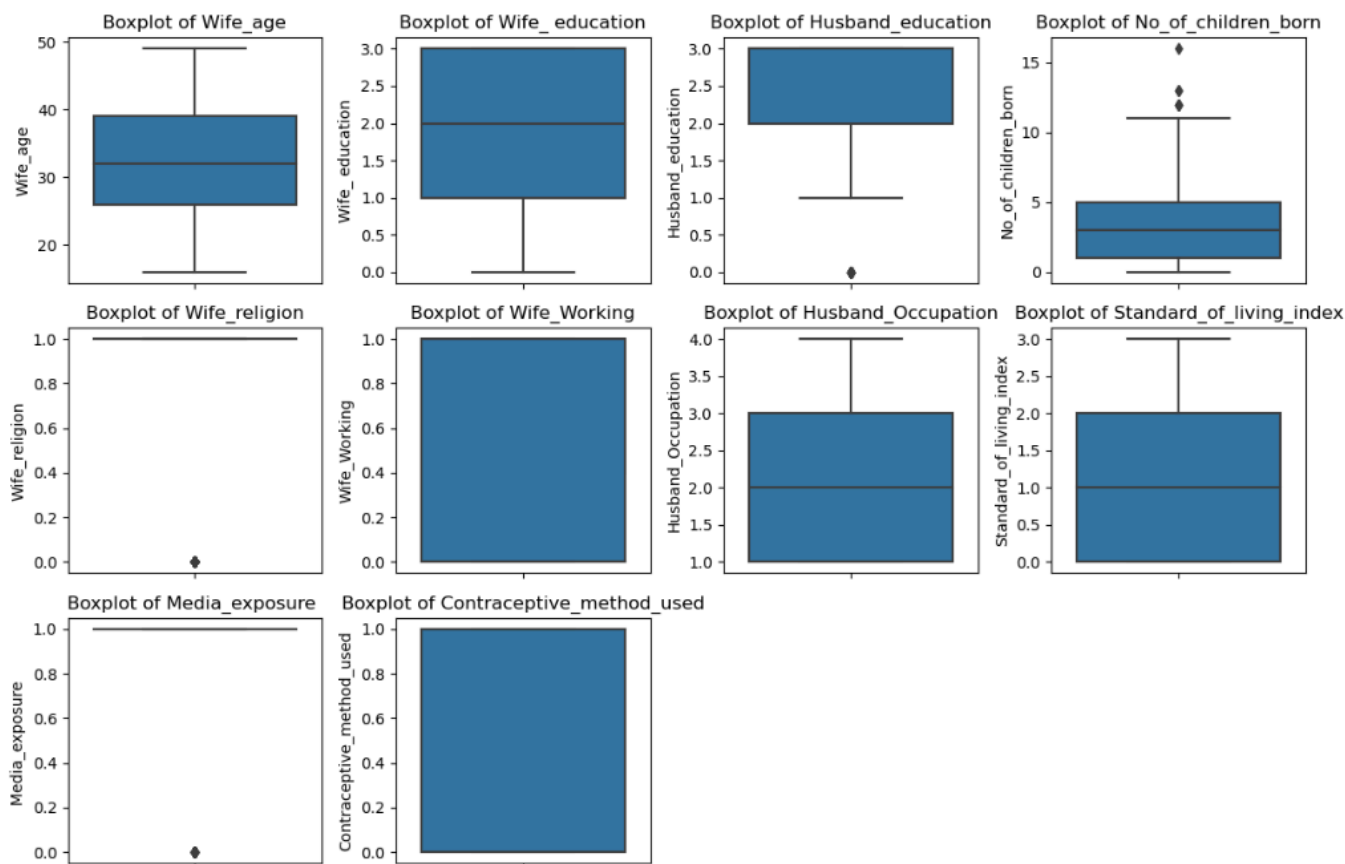
To view the dtype info

```

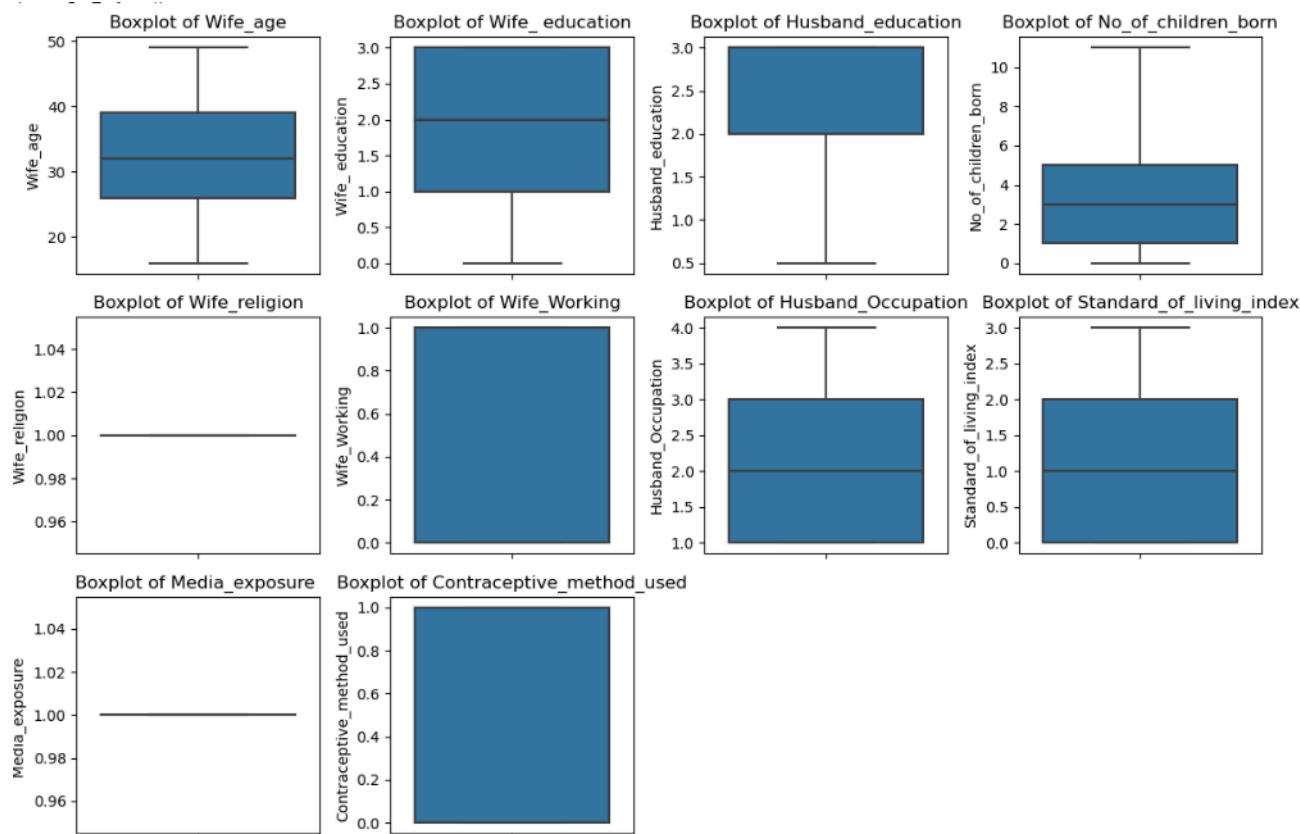
<class 'pandas.core.frame.DataFrame'>
Index: 1393 entries, 0 to 1472
Data columns (total 10 columns):
#   Column                                Non-Null Count  Dtype  
---  -
0   Wife_age                               1326 non-null   float64
1   Wife_education                         1393 non-null   int8    
2   Husband_education                     1393 non-null   int8    
3   No_of_children_born                   1372 non-null   float64
4   Wife_religion                         1393 non-null   int8    
5   Wife_Working                          1393 non-null   int8    
6   Husband_Occupation                    1393 non-null   int64   
7   Standard_of_living_index              1393 non-null   int8    
8   Media_exposure                        1393 non-null   int8    
9   Contraceptive_method_used             1393 non-null   int8    
dtypes: float64(2), int64(1), int8(7)
memory usage: 53.1 KB

```

Outlier Treatment



Post outlier Treatment -



Checking for Correlations

	Wife_age	Wife_education	Husband_education	No_of_children_born	Wife_religion	Wife_Working	Husband_Occupation	Standard_of_living_index	Media_exposure	Contraceptive_method_used
Wife_age	1.000000	-0.058209	-0.054183	0.541569	NaN	0.032528	-0.189798	-0.156799	NaN	-0.097251
Wife_education	-0.058209	1.000000	0.612967	-0.197466	NaN	0.058523	-0.370799	-0.269542	NaN	0.228341
Husband_education	-0.054183	0.612967	1.000000	-0.191812	NaN	-0.002612	-0.324801	-0.246578	NaN	0.141775
No_of_children_born	0.541569	-0.197466	-0.191812	1.000000	NaN	-0.103768	-0.022776	0.003592	NaN	0.121452
Wife_religion	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Wife_Working	0.032528	0.058523	-0.002612	-0.103768	NaN	1.000000	-0.013669	-0.087576	NaN	-0.042433
Husband_Occupation	-0.189798	-0.370799	-0.324801	-0.022776	NaN	-0.013669	1.000000	0.228363	NaN	-0.040438
Standard_of_living_index	-0.156799	-0.269542	-0.246578	0.003592	NaN	-0.087576	0.228363	1.000000	NaN	-0.108264
Media_exposure	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Contraceptive_method_used	-0.097251	0.228341	0.141775	0.121452	NaN	-0.042433	-0.040438	-0.108264	NaN	1.000000

Check the Heatmap



Problem 2 - Model Building and Compare the Performance of the Models

CART

1. Capture the target column ("Contraceptive_method_used") into separate vectors
2. Splitting data into training and test set for independent attributes
3. Import the decisiontreeclassifier library

```
DecisionTreeClassifier ⓘ ?  
DecisionTreeClassifier()
```

Feature Importance

Importance of features in the tree building (The importance of a feature is computed as the (normalized) total reduction of the criterion brought by that feature. It is also known as the Gini importance)

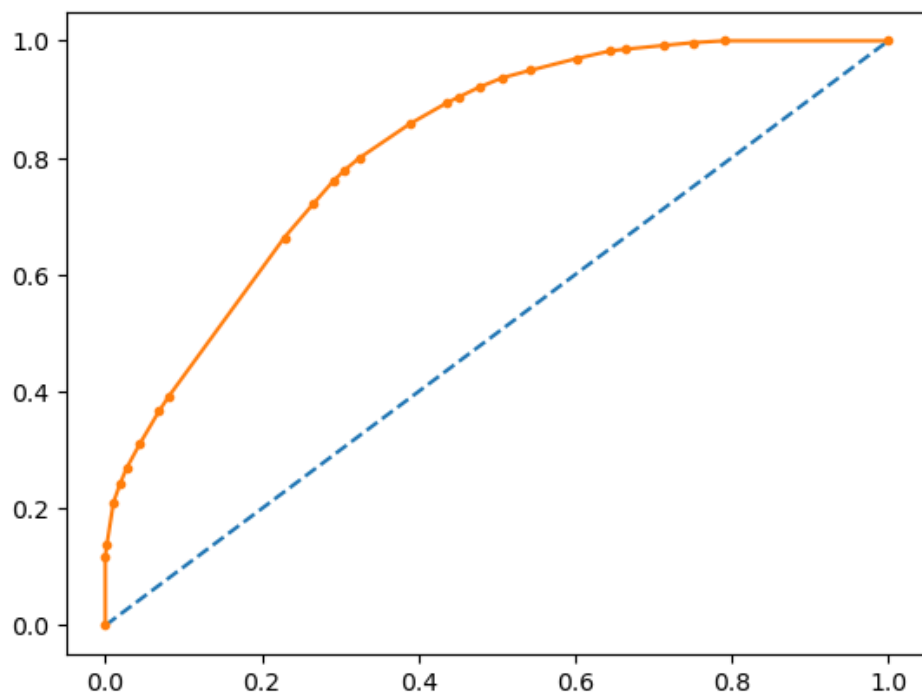
	Imp
Wife_age	0.358909
Wife_education	0.095365
Husband_education	0.076999
No_of_children_born	0.240930
Wife_religion	0.000000
Wife_Working	0.029485
Husband_Occupation	0.087755
Standard_of_living_index	0.110556
Media_exposure	0.000000

Regularising the Decision Tree

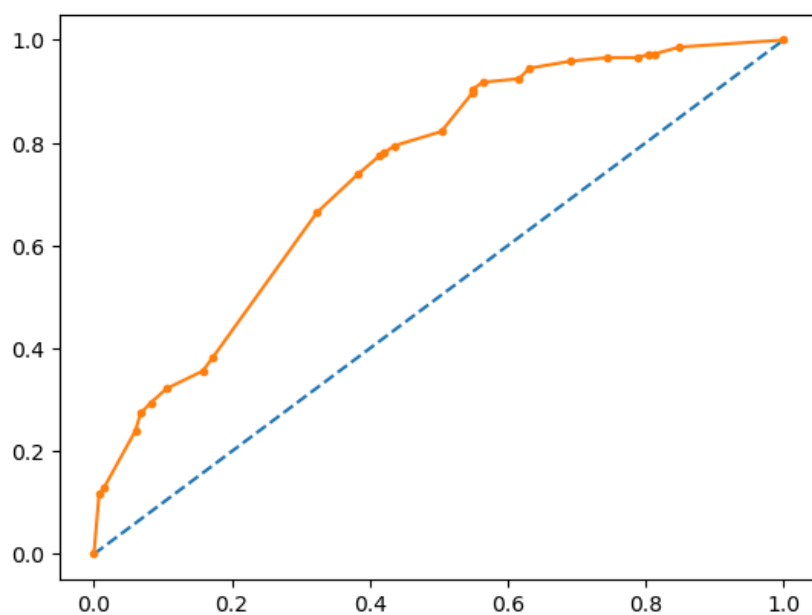
```
DecisionTreeClassifier ⓘ ?  
DecisionTreeClassifier(max_depth=7, min_samples_leaf=10, min_samples_split=30)
```

	Imp
Wife_age	0.358909
Wife_education	0.095365
Husband_education	0.076999
No_of_children_born	0.240930
Wife_religion	0.000000
Wife_Working	0.029485
Husband_Occupation	0.087755
Standard_of_living_index	0.110556
Media_exposure	0.000000

AUC and ROC for the training data



AUC and ROC for the test data

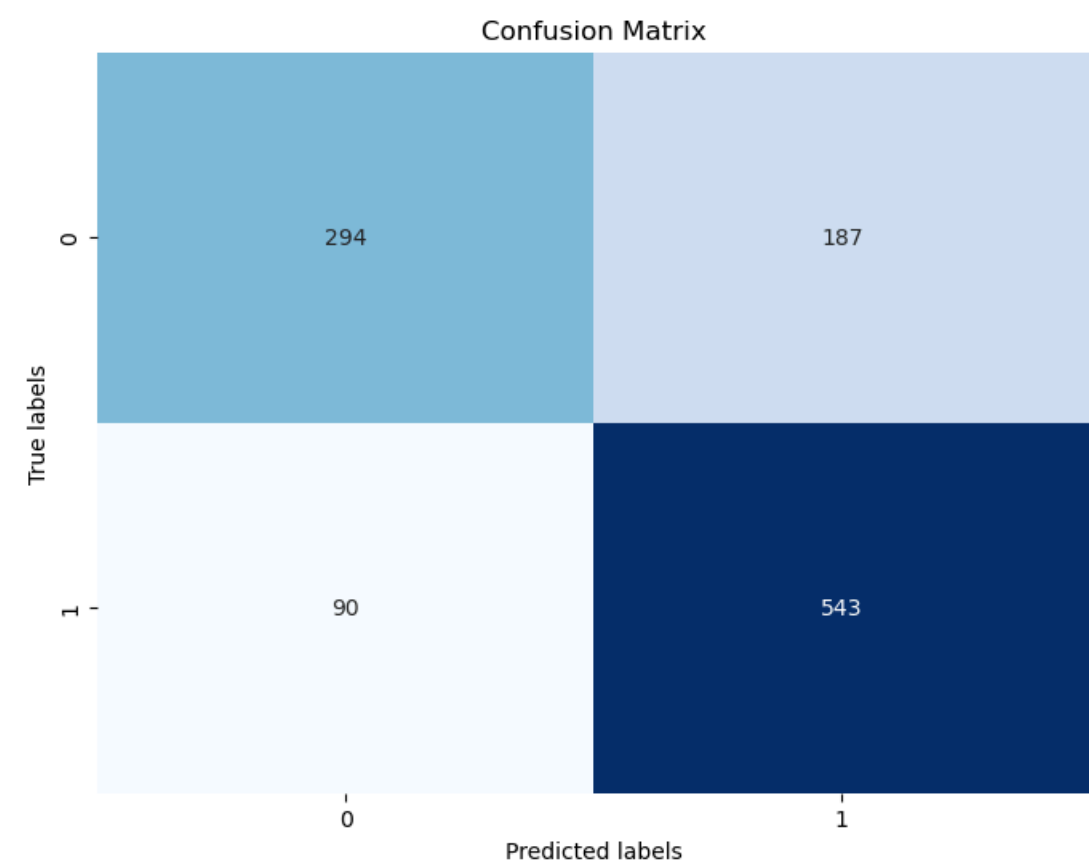


Classification report of train and test

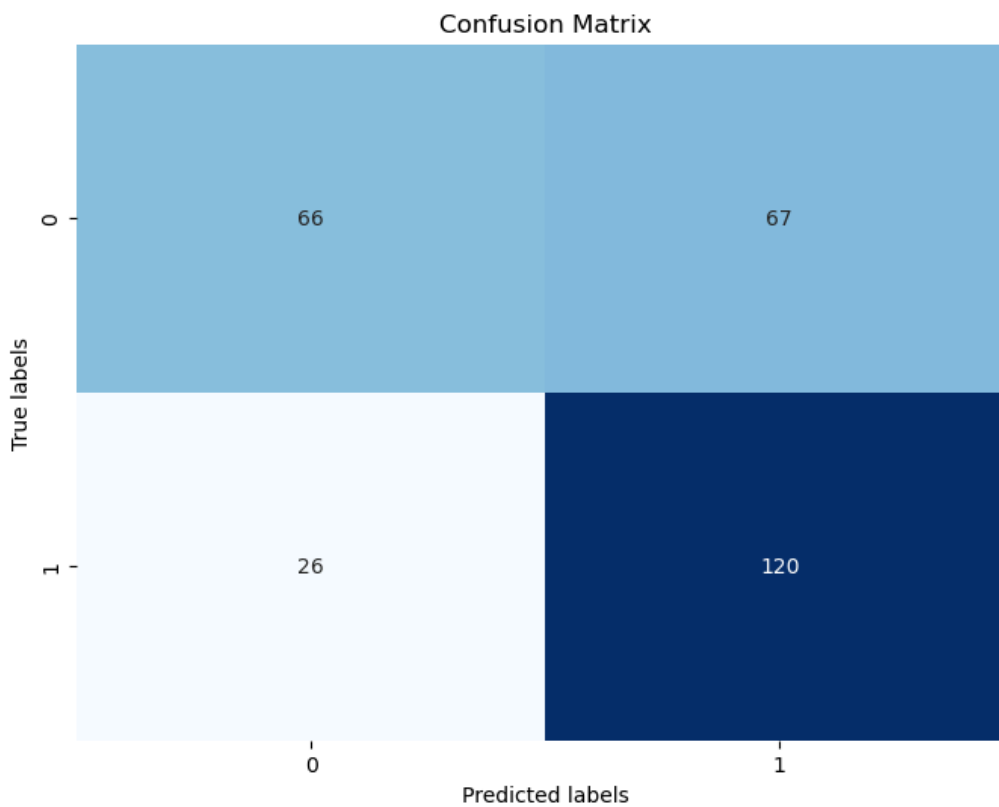
	precision	recall	f1-score	support
0.0	0.77	0.61	0.68	481
1.0	0.74	0.86	0.80	633
accuracy			0.75	1114
macro avg	0.75	0.73	0.74	1114
weighted avg	0.75	0.75	0.75	1114

	precision	recall	f1-score	support
0.0	0.72	0.50	0.59	133
1.0	0.64	0.82	0.72	146
accuracy			0.67	279
macro avg	0.68	0.66	0.65	279
weighted avg	0.68	0.67	0.66	279

Confusion Matrix for the training data



Confusion Matrix for testing data



Accuracy of the decision tree model on train data is 0.7271

0.751346499102334

Accuracy of the decision tree model on test data is 0.6379

0.6666666666666666

Summary -

Precision:

Class 0 (No contraceptive): The model achieved a precision of 0.72, indicating that when it predicts a woman will not use contraceptives, it is correct approximately 72% of the time.

Class 1 (Contraceptive used): The precision for class 1 is 0.64, meaning that when the model predicts a woman will use contraceptives, it is accurate around 64% of the time.

Recall:

Class 0 (No contraceptive): The recall for class 0 is 0.50, suggesting that the model correctly identifies about 50% of the instances where women do not use contraceptives.

Class 1 (Contraceptive used): The recall is 0.82, indicating that the model captures approximately 82% of the instances where women use contraceptives.

F1-score:

Class 0 (No contraceptive): The F1-score is 0.59, which represents the harmonic mean of precision and recall for class 0.

Class 1 (Contraceptive used): The F1-score is 0.72, indicating the balance between precision and recall for class 1.

Accuracy:

The overall accuracy of the Decision Tree (CART) model on the test data is 0.67, suggesting that it correctly predicts the contraceptive usage status for approximately 67% of the instances.

In summary, the Decision Tree (CART) model demonstrates promising performance in predicting contraceptive method usage, particularly in identifying instances where contraceptives are used (class 1). However, there is room for improvement in accurately predicting instances where contraceptives are not used (class 0).

Logistics Regression

Logistic Regression Model Using logistic regression we are trying to predict the dependent variable; logistic regression is used in predicting the categorical dependent variable. To perform the regression, model the data set has to be all numeric, to achieve this we have encoded all the object data in the dataset to numeric.

1. Creating a copy of the original data frame
2. Import the LabelEncoder library
3. Defining a Label Encoder object instance

```
LabelEncoder
LabelEncoder()
```

Applying the created Label Encoder object for the target class

	Wife_age	Wife_education	Husband_education	No_of_children_born	Wife_religion	Wife_Working	Husband_Occupation	Standard_of_living_index	Media_exposure	Contraceptive_method_used
0	24.0	1.0	2.0	3.0	1.0	0.0	2.0	2.0	1.0	0
1	45.0	0.0	2.0	10.0	1.0	0.0	3.0	0.0	1.0	0
2	43.0	1.0	2.0	7.0	1.0	0.0	3.0	0.0	1.0	0
3	42.0	2.0	1.0	9.0	1.0	0.0	3.0	2.0	1.0	0
4	36.0	2.0	2.0	8.0	1.0	0.0	3.0	1.0	1.0	0

```
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
```

Null value Treatment

	Wife_age	Wife_education	Husband_education	No_of_children_born	Wife_religion	Wife_Working	Husband_Occupation	Standard_of_living_index	Media_exposure	Contraceptive_method_used
0	24.0	1.0	2.0	3.0	1.0	0.0	2.0	2.0	1.0	0
1	45.0	0.0	2.0	10.0	1.0	0.0	3.0	0.0	1.0	0
2	43.0	1.0	2.0	7.0	1.0	0.0	3.0	0.0	1.0	0
3	42.0	2.0	1.0	9.0	1.0	0.0	3.0	2.0	1.0	0
4	36.0	2.0	2.0	8.0	1.0	0.0	3.0	1.0	1.0	0

Converting the other 'object' type variables as dummy variables

	Wife_age	Wife_education	Husband_education	No_of_children_born	Wife_religion	Wife_Working	Husband_Occupation	Standard_of_living_index	Media_exposure	Contraceptive_method_used
0	24.0	1.0	2.0	3.0	1.0	0.0	2.0	2.0	1.0	0
1	45.0	0.0	2.0	10.0	1.0	0.0	3.0	0.0	1.0	0
2	43.0	1.0	2.0	7.0	1.0	0.0	3.0	0.0	1.0	0
3	42.0	2.0	1.0	9.0	1.0	0.0	3.0	2.0	1.0	0
4	36.0	2.0	2.0	8.0	1.0	0.0	3.0	1.0	1.0	0

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

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

```
Contraceptive_method_used
1    0.559809
0    0.440191
Name: proportion, dtype: float64
```

Logistic Regression Model

Fit the model to the training data

```
LogisticRegression
LogisticRegression(C=10000000000.0, max_iter=10000, n_jobs=2,
                    solver='newton-cg', verbose=True)
```

Getting the probabilities on the test set

	0	1
0	0.304510	0.695490
1	0.563891	0.436109
2	0.392231	0.607769
3	0.367412	0.632588
4	0.305900	0.694100

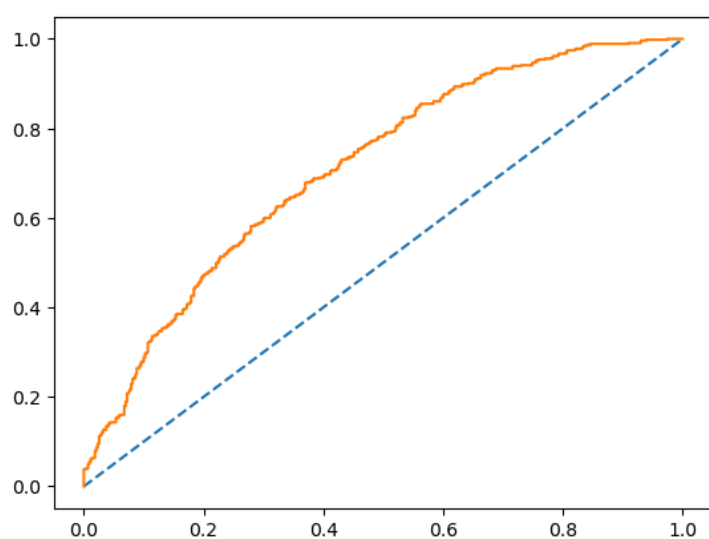
Model Evaluation

Model Score: 0.6594871794871795

AUC Value closer to 1 tells that there is good separability between the predicted classes and thus the model is good for prediction

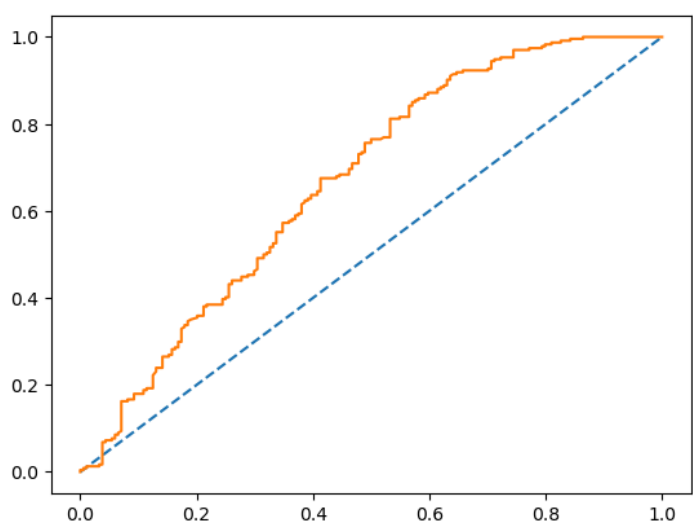
ROC Curve visually represents the above concept where the plot should be as far as possible from the diagonal.

AUC and ROC for the training data

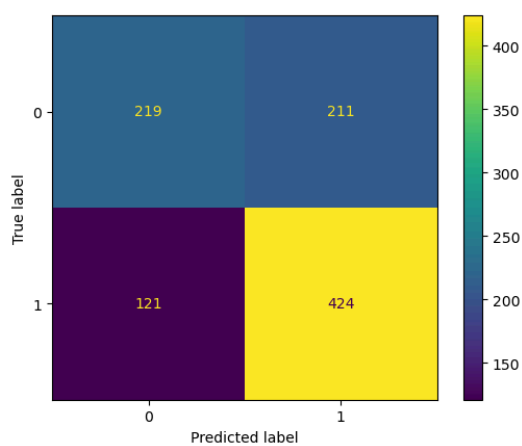


Accuracy of Test Data = 0.638755980861244

AUC and ROC for the test data



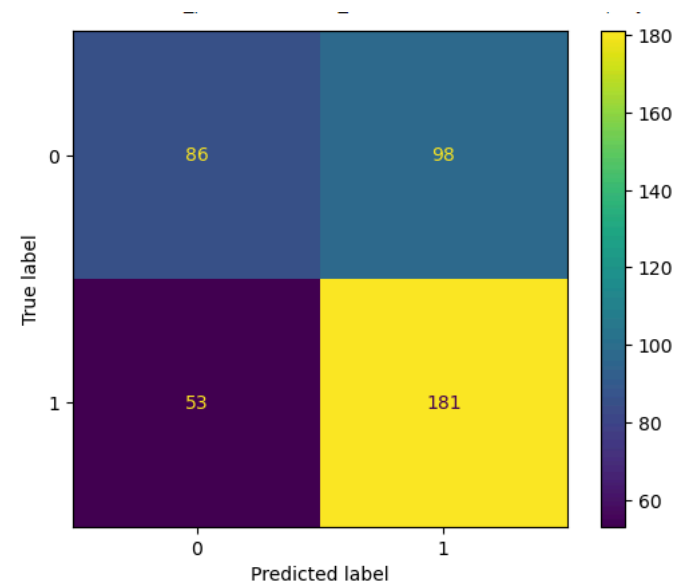
Confusion Matrix for the training data



Classification report

	precision	recall	f1-score	support
0	0.64	0.51	0.57	430
1	0.67	0.78	0.72	545
accuracy			0.66	975
macro avg	0.66	0.64	0.64	975
weighted avg	0.66	0.66	0.65	975

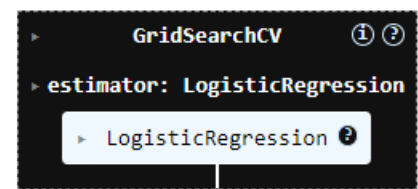
Confusion Matrix for test data



Classification report

	precision	recall	f1-score	support
0	0.62	0.47	0.53	184
1	0.65	0.77	0.71	234
accuracy			0.64	418
macro avg	0.63	0.62	0.62	418
weighted avg	0.64	0.64	0.63	418

Applying GridSearchCV for Logistic Regression

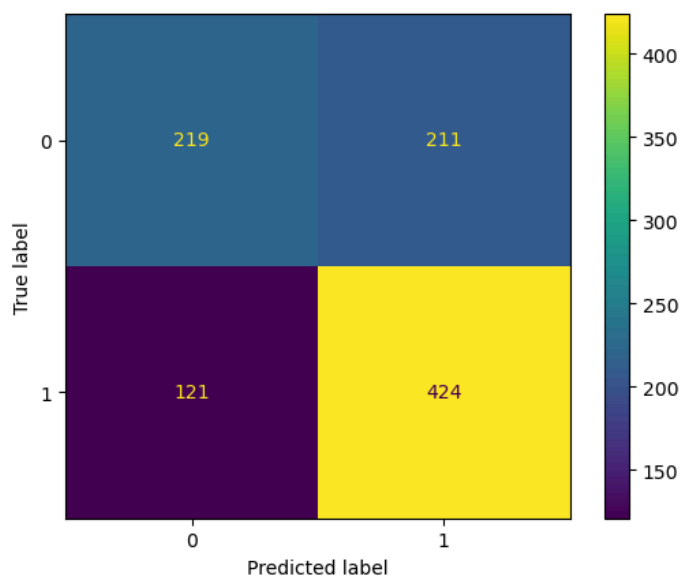


Getting the probabilities on the test set

	0	1
0	0.304829	0.695171
1	0.564242	0.435758
2	0.393028	0.606972
3	0.366988	0.633012
4	0.306586	0.693414

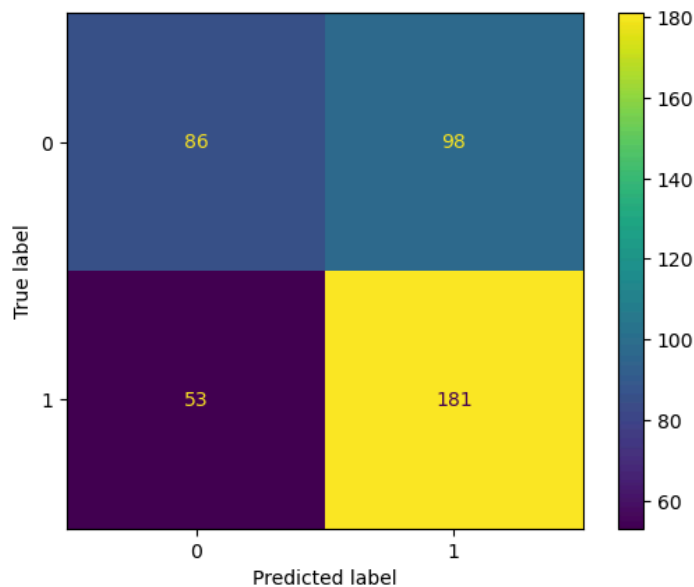
Confusion matrix on the training data

	precision	recall	f1-score	support
0	0.64	0.51	0.57	430
1	0.67	0.78	0.72	545
accuracy			0.66	975
macro avg	0.66	0.64	0.64	975
weighted avg	0.66	0.66	0.65	975



Confusion matrix on the Test data

	precision	recall	f1-score	support
0	0.62	0.47	0.53	184
1	0.65	0.77	0.71	234
accuracy			0.64	418
macro avg	0.63	0.62	0.62	418
weighted avg	0.64	0.64	0.63	418



Summary

Evaluation of Logistic Regression Model

Analysis:

The logistic regression model demonstrates a relatively adept capability in forecasting contraceptive utilization (class 1), as evidenced by precision (0.65) and recall (0.77) metrics.

However, there exists an opportunity for enhancement in predicting instances of non-usage (class 0), as precision (0.62) and recall (0.47) scores for this category imply.

Comprehensive Report:

For the "No contraceptive" category (class 0):

Precision: 0.62

Recall: 0.47

F1-score: 0.53

For the "Contraceptive used" category (class 1):

Precision: 0.65

Recall: 0.77

F1-score: 0.71

The weighted average F1-score, amalgamating both classes, stands at 0.66, denoting the holistic effectiveness of the model.

With an overall accuracy of 0.66, the model demonstrates competence in forecasting across both categories.

Precision and recall metrics furnish valuable insights into the model's efficacy in precisely identifying instances of contraceptive usage. These metrics serve as pivotal indicators for a thorough assessment of the model's performance.

Linear Discriminant Analysis

Creating a copy of the original data frame

	Wife_age	Wife_education	Husband_education	No_of_children_born	Wife_religion	Wife_Working	Husband_Occupation	Standard_of_living_index	Media_exposure	Contraceptive_method_used
0	24.0	1.0	2.0	3.0	1.0	0.0	2.0	2.0	1.0	0
1	45.0	0.0	2.0	10.0	1.0	0.0	3.0	0.0	1.0	0
2	43.0	1.0	2.0	7.0	1.0	0.0	3.0	0.0	1.0	0
3	42.0	2.0	1.0	9.0	1.0	0.0	3.0	2.0	1.0	0
4	36.0	2.0	2.0	8.0	1.0	0.0	3.0	1.0	1.0	0

Null Value Treatment

Display the first few rows of the DataFrame after imputation

	Wife_age	Wife_education	Husband_education	No_of_children_born	Wife_religion	Wife_Working	Husband_Occupation	Standard_of_living_index	Media_exposure	Contraceptive_method_used
0	24.0	1.0	2.0	3.0	1.0	0.0	2.0	2.0	1.0	0
1	45.0	0.0	2.0	10.0	1.0	0.0	3.0	0.0	1.0	0
2	43.0	1.0	2.0	7.0	1.0	0.0	3.0	0.0	1.0	0
3	42.0	2.0	1.0	9.0	1.0	0.0	3.0	2.0	1.0	0
4	36.0	2.0	2.0	8.0	1.0	0.0	3.0	1.0	1.0	0

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,)

LDA Model

```
LinearDiscriminantAnalysis
LinearDiscriminantAnalysis()
```

Generate Coefficients and intercept for the Linear Discriminant Function

Intercept value is 0.2601

Coefficients for the Linear Discriminant Function

```
array([[ -7.26686167e-02,  5.18797397e-01,  1.98861282e-01,
         2.56295241e-01,  1.97044936e-15, -1.79831249e-01,
         1.52789903e-01, -1.89292026e-01,  0.00000000e+00]])
```

To view the columns of X-train

```
Index(['Wife_age', 'Wife_education', 'Husband_education',
       'No_of_children_born', 'Wife_religion', 'Wife_Working',
       'Husband_Occupation', 'Standard_of_living_index', 'Media_exposure'],
      dtype='object')
```

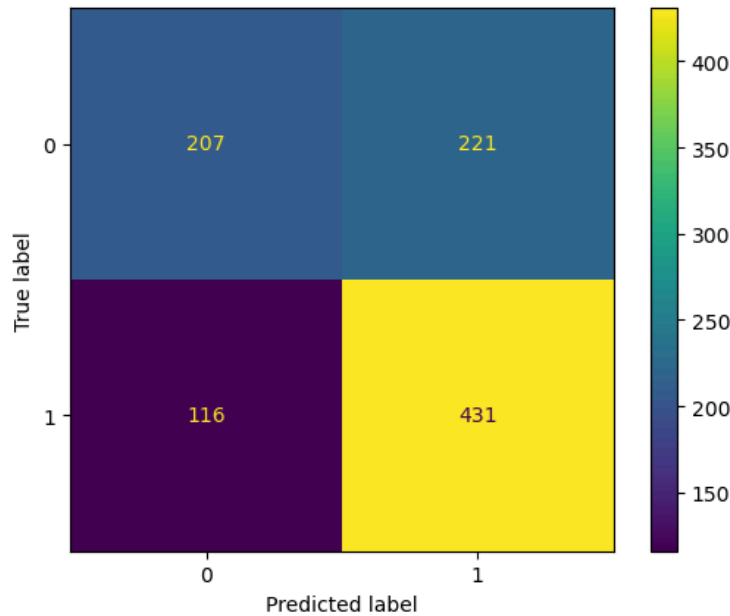
Linear Discriminant Function

$$= -0.26013643 + (-7.26686167 \times 10^{-2} * \text{'Wife_age'}) + (5.18797397 \times 10^{-1} * \text{'Wife_education'}) + (1.98861282 \times 10^{-1} * \text{'Husband_education'}) + (2.56295241 \times 10^{-1} * \text{'No_of_children_born'}) + (1.97044936 \times 10^{-1} * \text{'Wife_religion'}) + (-1.79831249 \times 10^{-1} * \text{'Wife_Working'}) + (1.52789903 \times 10^{-1} * \text{'Husband_Occupation'}) + (-1.89292026 \times 10^{-1} * \text{'Standard_of_living_index'}) + (\text{'Media_exposure'} * 0.00000000 \times 10^0)$$

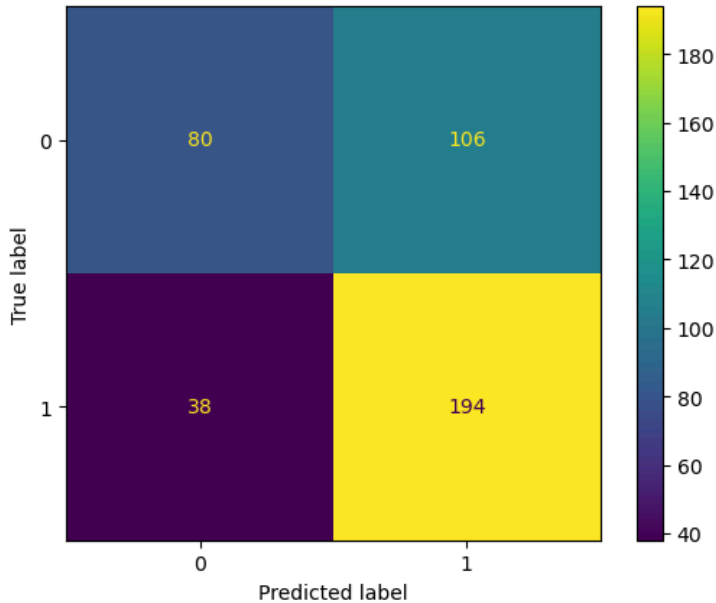
Rounded up coefficients

```
array([[ -0.07,  0.52,  0.2 ,  0.26,  0.  , -0.18,  0.15, -0.19,  0.  ]])
```

Training Data and Test Data Confusion Matrix Comparison



Confusion matrix on the training data



Training Data and Test Data Classification Report Comparison

Classification Report of the training data:

	precision	recall	f1-score	support
0	0.64	0.48	0.55	428
1	0.66	0.79	0.72	547
accuracy			0.65	975
macro avg	0.65	0.64	0.64	975
weighted avg	0.65	0.65	0.65	975

Classification Report of the test data:

	precision	recall	f1-score	support
0	0.68	0.43	0.53	186
1	0.65	0.84	0.73	232
accuracy			0.66	418
macro avg	0.66	0.63	0.63	418
weighted avg	0.66	0.66	0.64	418

Summary: The confusion matrix and classification report for test data

Confusion Matrix:

The confusion matrix is a table with rows representing the actual classes and columns representing the predicted classes.

Each cell shows the number of instances that were predicted to belong to a particular class (column) but actually belonged to a different class (row).

In the image, the confusion matrix shows two classes (likely 0 and 1).

Out of 186 actual class 0 instances, 127 were correctly classified, and 59 were incorrectly classified as class 1 (shown in the top row).

Out of 232 actual class 1 instances, 151 were correctly classified, and 81 were incorrectly classified as class 0 (shown in the bottom row).

Classification Report:

The report provides a more detailed breakdown of the model's performance on each class.

Precision: This metric represents the proportion of predicted positive cases that were actually positive.

Class 0: Precision = 0.68 (meaning 68% of predicted class 0 were truly class 0).

Class 1: Precision = 0.65 (meaning 65% of predicted class 1 were truly class 1).

Recall: This metric represents the proportion of actual positive cases that were predicted positive.

Class 0: Recall = 0.43 (meaning the model only identified 43% of actual class 0 instances correctly).

Class 1: Recall = 0.84 (meaning the model correctly classified 84% of actual class 1 instances).

F1-Score: This metric is the harmonic mean of precision and recall, and it takes both precision and recall into account.

Class 0: F1-Score = 0.53

Class 1: F1-Score = 0.73

Accuracy: This metric represents the overall proportion of predictions that were correct.

Accuracy = 0.66 (meaning the model correctly classified 66% of the instances).

Overall Performance:

The model seems to have a slightly better performance in predicting class 1 (recall of 0.84) compared to class 0 (recall of 0.43). However, the precision for both classes is around 0.65, which means that there is a significant number of misclassified instances for both classes.

Summary : The confusion matrix and classification report for training data

Overall Performance:

Accuracy: 65% - This means the model correctly classified 65% of the instances in the training data.

Class-wise Performance:

The report shows the performance for two classes (possibly positive and negative). Here's a breakdown for each class:

Class 0:

Precision: 0.64 - Out of all instances predicted as class 0, 64% were actually class 0. (Low false positives)

Recall: 0.48 - Out of all actual class 0 instances, the model only identified 48% correctly. (High false negatives)

F1-Score: 0.55 - This is a harmonic mean between precision and recall, indicating a moderate balance between the two metrics for class 0.

Class 1:

Precision: 0.66 - Similar to class 0, 66% of predicted class 1 instances were truly class 1.

Recall: 0.79 - The model performed better in identifying actual class 1 instances, correctly classifying 79% of them. (Low false negatives)

F1-Score: 0.72 - This indicates a better balance between precision and recall for class 1 compared to class 0.

Key Takeaways:

Similar to the test data, the model struggles with identifying class 0 instances, having a low recall (0.48). This means there might be a significant number of actual class 0 instances that were incorrectly classified as class 1 (false negatives) during training.

Class 1 has a higher recall (0.79), suggesting the model is better at identifying actual positives in this class during training.

Overall accuracy is moderate (0.65), which is typical for a training dataset.

Training vs Test Data Performance:

It's important to compare the model's performance on training and test data. In this case, both the training and test data have similar accuracy (around 65-66%). However, the model's recall for class 0 seems to be lower on the test data (0.43) compared to training data (0.48). This suggests that the model might be overfitting the training data to some extent.

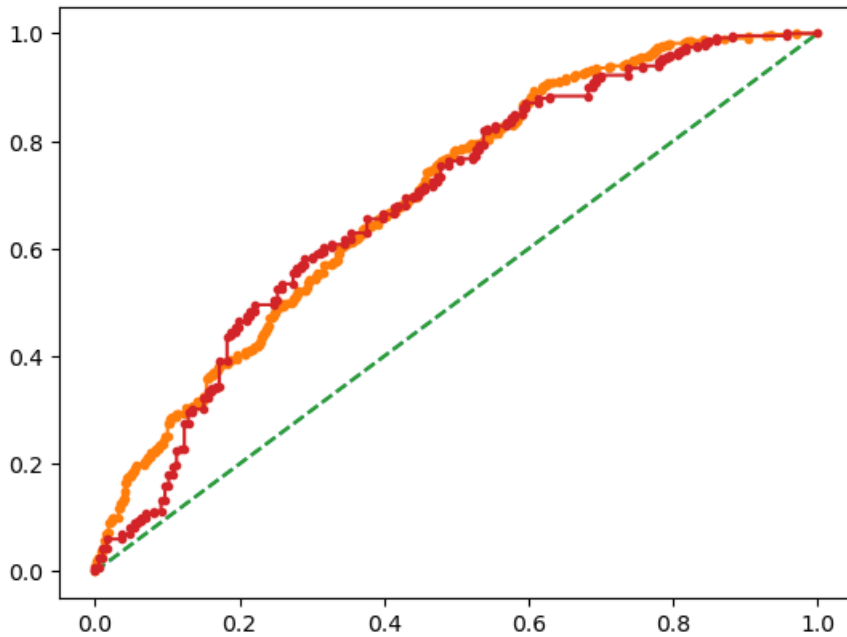
Probability prediction for the training and test data -

Here is the probability prediction for both training and test data.

```
array([0.5315517 , 0.38952506, 0.69286713, 0.58327292, 0.14563503,
       0.81566304, 0.44352753, 0.62262665, 0.71436959, 0.14094974,
       0.3700017 , 0.20528217, 0.75151882, 0.58749504, 0.33057091,
       0.77420103, 0.72757446, 0.707044 , 0.75283237, 0.57891589,
       0.46139431, 0.62443545, 0.64771244, 0.58668999, 0.67307932,
       0.80509963, 0.85889242, 0.50633801, 0.66642152, 0.54679076,
       0.61356152, 0.77221521, 0.88155922, 0.55389003, 0.43244332,
       0.65807807, 0.60024373, 0.79093456, 0.49353353, 0.35589469,
       0.43349865, 0.35671666, 0.4626679 , 0.71062766, 0.28203307,
       0.67757385, 0.36469292, 0.52569608, 0.74416011, 0.70132229,
       0.59314271, 0.5530575 , 0.55099915, 0.72675488, 0.8045251 ,
       0.20223583, 0.46650841, 0.75541202, 0.67547835, 0.79719029,
       0.48938602, 0.68847511, 0.41085524, 0.78061641, 0.24018917,
       0.45806664, 0.67126909, 0.63892843, 0.57489335, 0.6111075 ,
       0.51028252, 0.0809925 , 0.53645043, 0.6666422 , 0.58319447,
       0.63839609, 0.57585022, 0.53265206, 0.45575359, 0.75468884,
       0.56900634, 0.5523172 , 0.53428834, 0.65069991, 0.6670307 ,
       0.35341599, 0.76952322, 0.5257247 , 0.33478096, 0.7056522 ,
       0.80448533, 0.58789254, 0.37775147, 0.64832167, 0.71411534,
       0.69655649, 0.41943722, 0.70350719, 0.19238397, 0.75253799,
       0.68679536, 0.67114959, 0.37870438, 0.64742963, 0.68960881,
       0.66042473, 0.22231735, 0.55415924, 0.3667856 , 0.64483891,
       0.78665095, 0.68549614, 0.79839256, 0.50296852, 0.54538013,
       0.73161223, 0.60082224, 0.73290122, 0.71913025, 0.78599397,
       0.45791192, 0.52916664, 0.78650354, 0.49895678, 0.20641304,
       0.66233381, 0.75959927, 0.72410101, 0.85983293, 0.72073661,
       0.7033242 , 0.52819232, 0.21760458, 0.74715209, 0.68812896,
       0.73213508, 0.8644047 , 0.4288408 , 0.11365865, 0.57953889,
       0.78730646, 0.76722064, 0.84960227, 0.47717126, 0.52675321,
       0.53318682, 0.53425978, 0.25708681, 0.55787555, 0.71606493,
       0.57550368, 0.5552974 , 0.28531542, 0.55874913, 0.35257666,
       0.63720733, 0.69495462, 0.68316821, 0.56922288, 0.55778486,
       0.53046551, 0.76658164, 0.79900278, 0.65154722, 0.59755095,
       0.39557019, 0.64540932, 0.71632594, 0.74614388, 0.36554071,
       0.33546978, 0.72165673, 0.53825038, 0.8092383 , 0.59418888,
       0.44251978, 0.68972612, 0.85606112, 0.25170784, 0.39482547,
       0.57104939, 0.27567927, 0.57326709, 0.35615399, 0.70748353,
       0.68209198, 0.34232177, 0.72605084, 0.71691187, 0.6889429 ,
       0.46972967, 0.78413868, 0.69165472, 0.78348803, 0.1918443 ,
       0.73366596, 0.41111155, 0.55281391, 0.3788318 , 0.41629942,
       0.66420745, 0.44269789, 0.63849691, 0.71326391, 0.78861308,
```

AUC and ROC for the training data

AUC for the Training Data: 0.695
AUC for the Test Data: 0.686



Performance Metrics Overview:

CART (Decision Tree) Test Data Report:

- Precision (Class 0): 0.72
- Recall (Class 0): 0.50
- F1-score (Class 0): 0.59
- Precision (Class 1): 0.64
- Recall (Class 1): 0.82
- F1-score (Class 1): 0.72
- Accuracy: 0.67

Logistic Regression Test Data Report:

- Precision (Class 0): 0.62
- Recall (Class 0): 0.47
- F1-score (Class 0): 0.53
- Precision (Class 1): 0.65
- Recall (Class 1): 0.77
- F1-score (Class 1): 0.71
- Accuracy: 0.64

LDA (Linear Discriminant Analysis) Test Data Report:

- Precision (Class 0): 0.68
- Recall (Class 0): 0.43
- F1-score (Class 0): 0.53
- Precision (Class 1): 0.65
- Recall (Class 1): 0.84
- F1-score (Class 1): 0.73
- Accuracy: 0.66

Comparison:

- Accuracy: CART (Decision Tree) model performs the best with an accuracy of 0.67, followed by LDA with 0.66 and Logistic Regression with 0.64.
- Precision: CART model has the highest precision for class 0 (0.72), while Logistic Regression has the highest precision for class 1 (0.65).
- Recall: LDA model achieves the highest recall for class 1 (0.84), indicating its ability to capture true positive instances.
- F1-score: CART model shows the highest F1-score for class 1 (0.72), balancing precision and recall effectively.

Conclusion:

Considering the comprehensive evaluation of performance metrics, the CART (Decision Tree) model emerges as the optimal choice among the tested models for this classification task. With its superior accuracy, balanced precision and recall, and impressive F1-score, the CART model demonstrates robustness and effectiveness in predicting contraceptive method usage based on the provided dataset. Thus, it is recommended for deployment in real-world scenarios where accurate classification is paramount.

Problem 2 - Business Insights & Recommendations

Insights and Recommendations -

Targeting Education: Invest in educational initiatives for both wives and husbands, as higher education levels are strongly correlated with increased contraceptive usage.

Family Planning Programs: Implement family planning programs targeting couples with a higher number of children, as they are more likely to use contraceptives.

Promoting Female Employment: Encourage female workforce participation, as it is associated with lower contraceptive usage. Addressing barriers to employment for women can help improve contraceptive access and usage.

Standard of Living Improvement: Focus on improving the standard of living to decrease contraceptive usage, as higher standard of living index is associated with reduced usage.

Occupational Factors: Explore the specific occupations associated with higher contraceptive usage and tailor interventions or services accordingly.

Age Considerations: Consider age-specific interventions, as contraceptive usage tends to decrease with increasing wife age
