Predictive Modelling

ANIL KUMAR PGBPDSBA AUG’2023

Contents

Linear Regression

1.0 Problem Statement – Comp-active Dataset

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

SUMMARY

The Given dataset contains data collected from sun Sparcstation 20/712 with 128Mbytes of memory running in a multi-user university department.

A Model need to find out for predicting 'usr' and check how each attribute affects the system to be in 'usr' mode using a list of system attributes. The following process carried out for building a model as follows,

Importing the required libraries regarding linear regression.

Reading the excel files.

1.1 EDA- Exploratory Data Analysis

- Checking the shape of the dataset

  Dataset contains 8192 rows and 22 columns

```
(8192, 22)
```

- Displaying few rows of the dataset

```
data.head(5) ##  Complete the code to view top 5 rows of the data
```

	lread	lwrite	scall	sread	swrite	fork	exec	rchar	wchar	pgout	ppgout	pgfree	pgscan	atch	pgin	ppgin	pflt	vflt	runqsz	freemen
0	1	0	2147	79	68	0.20000	0.20000	40671.00000	53995.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.60000	2.60000	16.00000	26.40000	CPU_Bound	4670
1	0	0	170	18	21	0.20000	0.20000	448.00000	8385.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	15.63000	16.83000	Not_CPU_Bound	7278
2	15	3	2162	159	119	2.00000	2.40000	NaN	31950.00000	0.00000	0.00000	0.00000	0.00000	1.20000	6.00000	9.40000	150.20000	220.20000	Not_CPU_Bound	702
3	0	0	160	12	16	0.20000	0.20000	NaN	8670.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.20000	0.20000	15.60000	16.80000	Not_CPU_Bound	7248
4	5	1	330	39	38	0.40000	0.40000	NaN	12185.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	1.20000	37.80000	47.60000	Not_CPU_Bound	633

- Checking the data types of the columns for the dataset

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

- Statistical summary of the dataset

Data set contains 13 Float values, 8 integer values and 1 object

	count	mean	std	min	25%	50%	75%	max
lread	8192.00000	19.55969	53.35380	0.00000	2.00000	7.00000	20.00000	1845.00000
lwrite	8192.00000	13.10620	29.89173	0.00000	0.00000	1.00000	10.00000	575.00000
scall	8192.00000	2306.31824	1633.61732	109.00000	1012.00000	2051.50000	3317.25000	12493.00000
sread	8192.00000	210.47998	198.98015	6.00000	86.00000	166.00000	279.00000	5318.00000
swrite	8192.00000	150.05823	160.47898	7.00000	63.00000	117.00000	185.00000	5456.00000
fork	8192.00000	1.88455	2.47949	0.00000	0.40000	0.80000	2.20000	20.12000
exec	8192.00000	2.79200	5.21246	0.00000	0.20000	1.20000	2.80000	59.56000
rchar	8088.00000	197385.72836	239837.49353	278.00000	34091.50000	125473.50000	267828.75000	2526649.00000
wchar	8177.00000	95902.99278	140841.70791	1498.00000	22916.00000	46619.00000	106101.00000	1801623.00000
pgout	8192.00000	2.28532	5.30704	0.00000	0.00000	0.00000	2.40000	81.44000
ppgout	8192.00000	5.97723	15.21459	0.00000	0.00000	0.00000	4.20000	184.20000
pgfree	8192.00000	11.91971	32.36352	0.00000	0.00000	0.00000	5.00000	523.00000
pgscan	8192.00000	21.52685	71.14134	0.00000	0.00000	0.00000	0.00000	1237.00000
atch	8192.00000	1.12750	5.70835	0.00000	0.00000	0.00000	0.60000	211.58000
pgin	8192.00000	8.27796	13.87498	0.00000	0.60000	2.80000	9.76500	141.20000
ppgin	8192.00000	12.38859	22.28132	0.00000	0.60000	3.80000	13.80000	292.61000
pflt	8192.00000	109.79380	114.41922	0.00000	25.00000	63.80000	159.60000	899.80000
vflt	8192.00000	185.31580	191.00060	0.20000	45.40000	120.40000	251.80000	1365.00000
freemem	8192.00000	1763.45630	2482.10451	55.00000	231.00000	579.00000	2002.25000	12027.00000
freeswap	8192.00000	1328125.95984	422019.42696	2.00000	1042623.50000	1289289.50000	1730379.50000	2243187.00000
usr	8192.00000	83.96887	18.40190	0.00000	81.00000	89.00000	94.00000	99.00000

- Checking the Duplicate Values

No Duplicate Values found

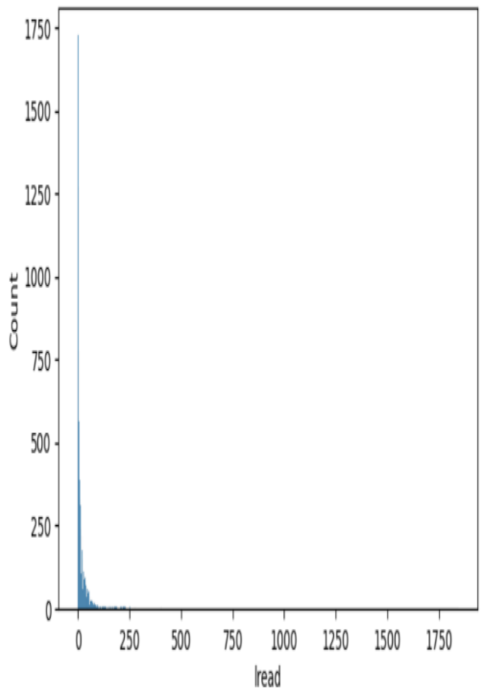
```
[11] df.duplicated().sum()

0
```

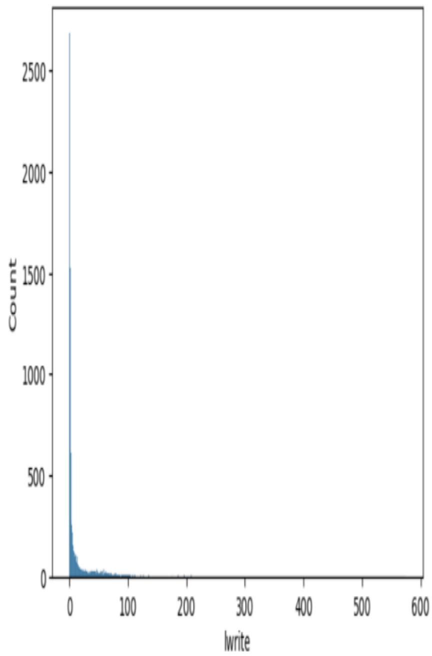
- Checking the number of unique values in each column

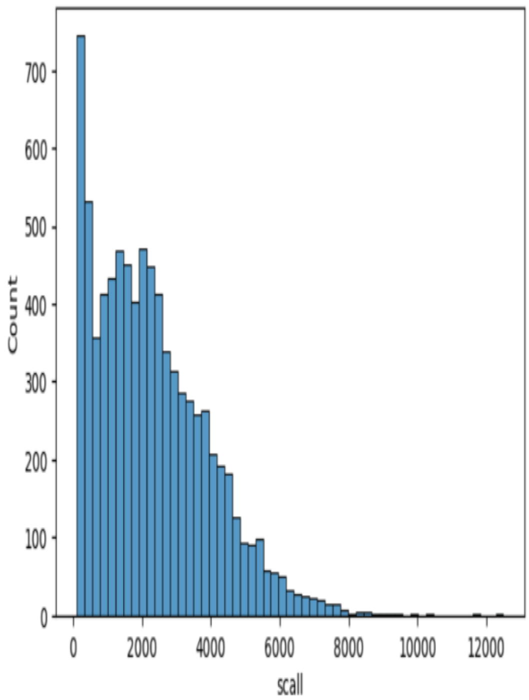
```python
# checking for non-unique values
data.nunique()
```

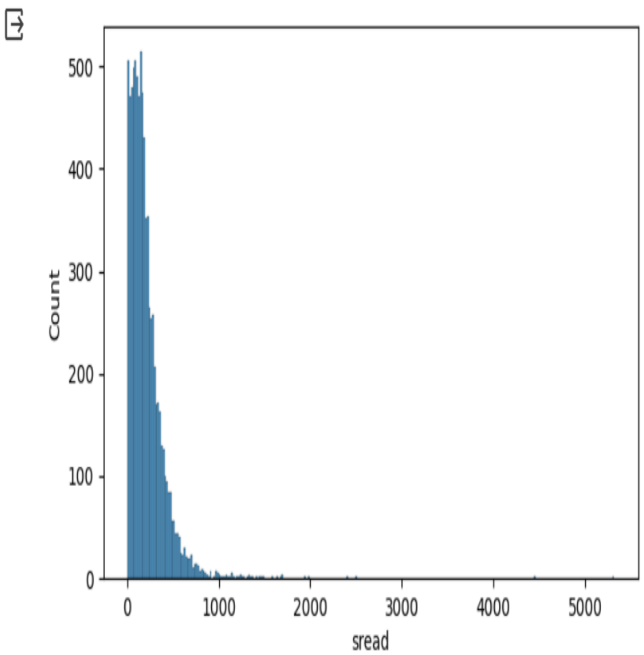
```
lread        235
lwrite       189
scall       4115
sread        794
swrite       640
fork         228
exec         386
rchar       7898
wchar       7925
pgout        404
ppgout       774
pgfree      1070
pgscan      1202
atch         253
pgin         832
ppgin       1072
pflt        2987
vflt        3799
runqsz         2
freemem     3165
freeswap    7658
usr           56
dtype: int64
```

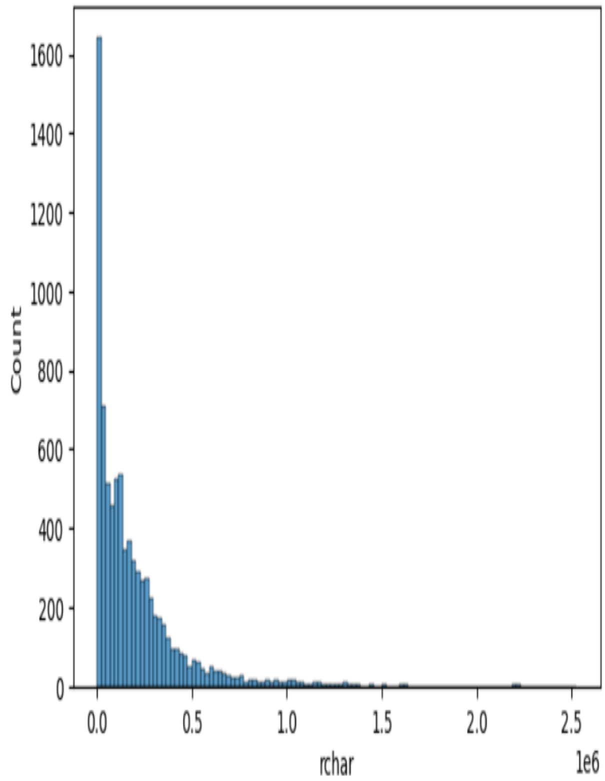
1.1.1 Univariate Analysis



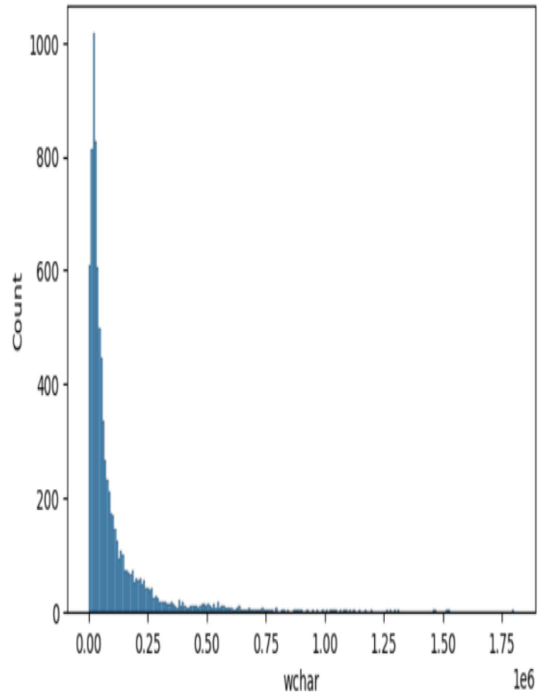






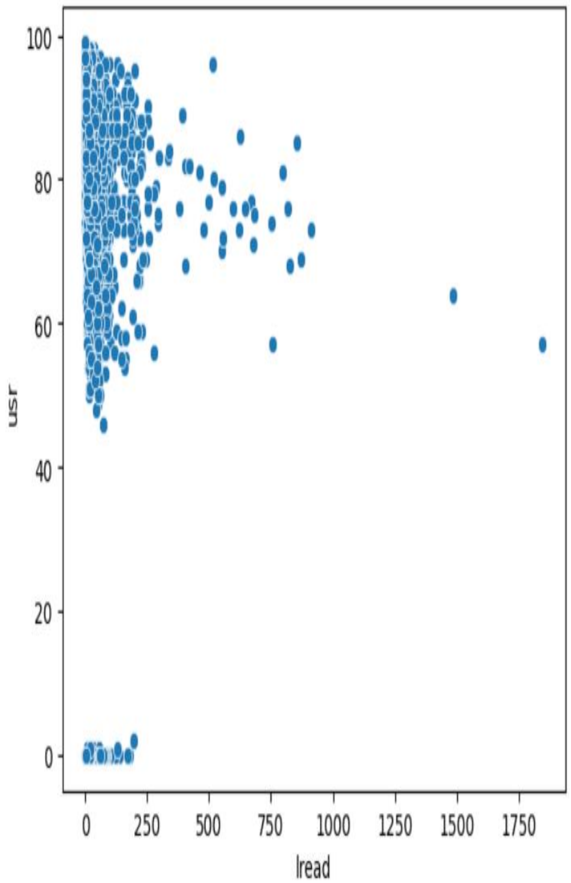


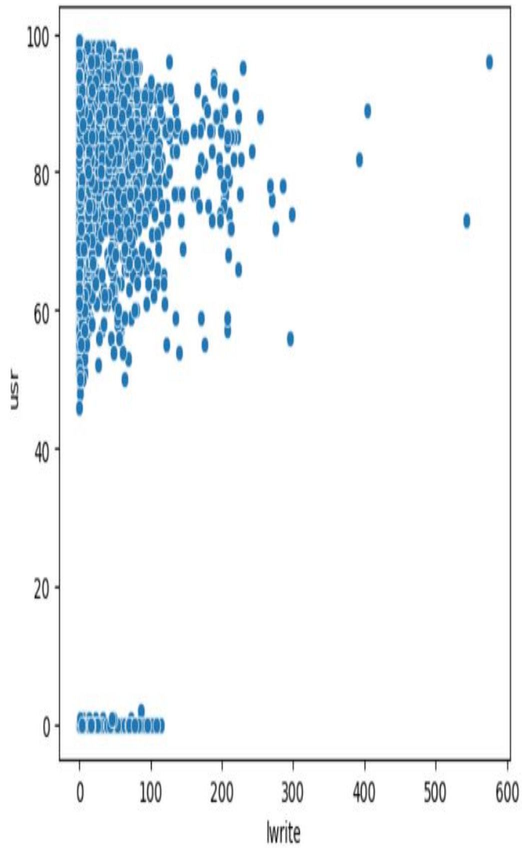
Count
rchar
1e6

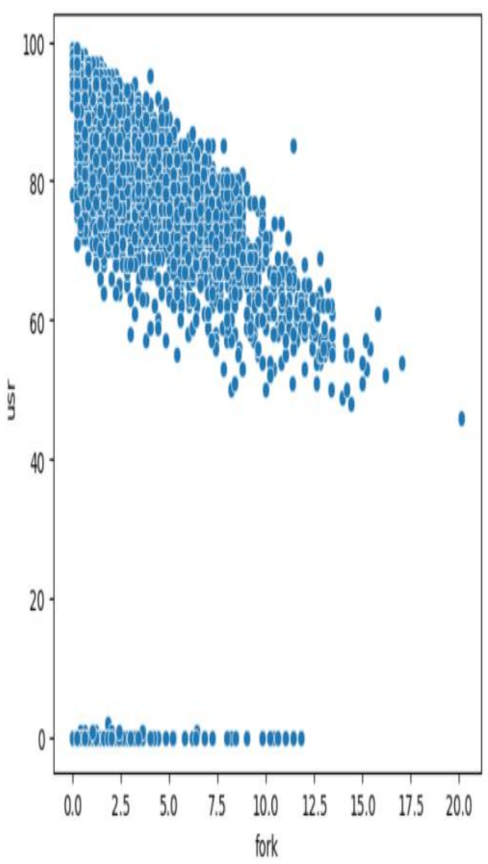


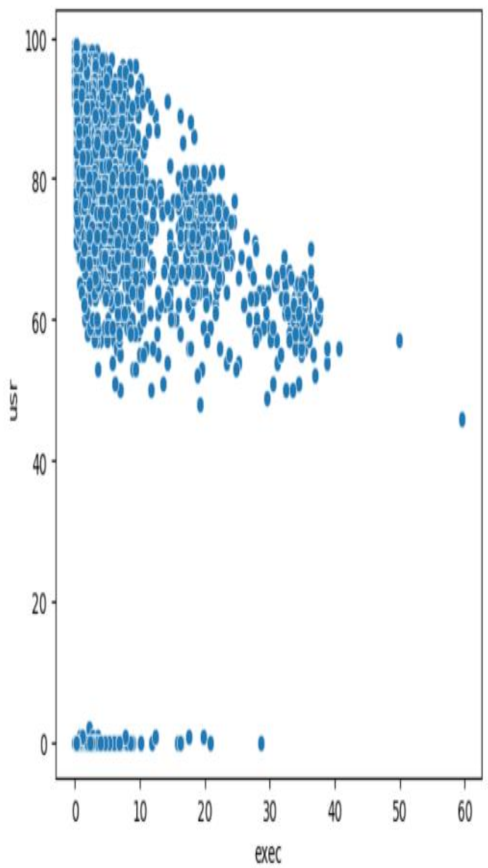
Count
wchar
1e6

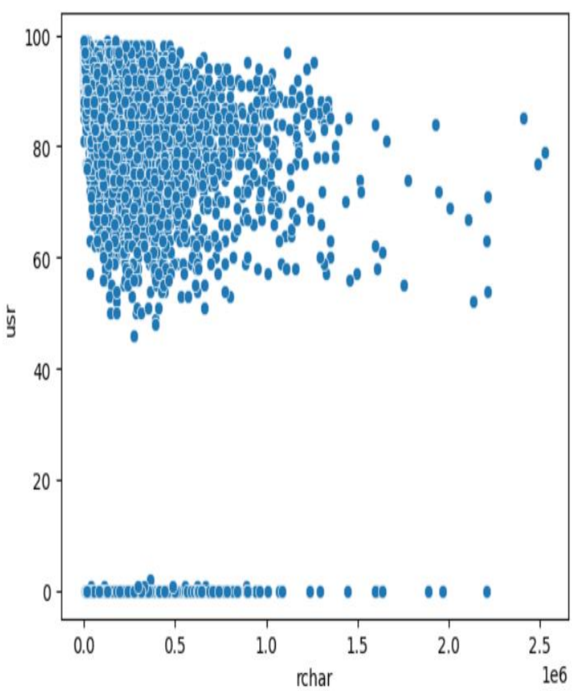
1.1.2 Perform Bivariate Analysis

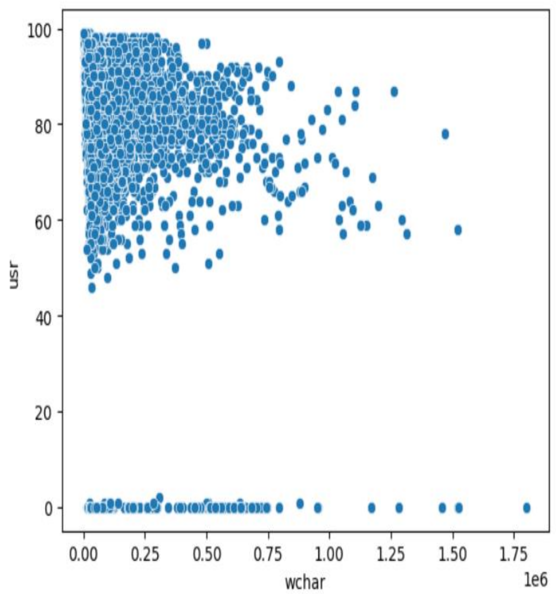




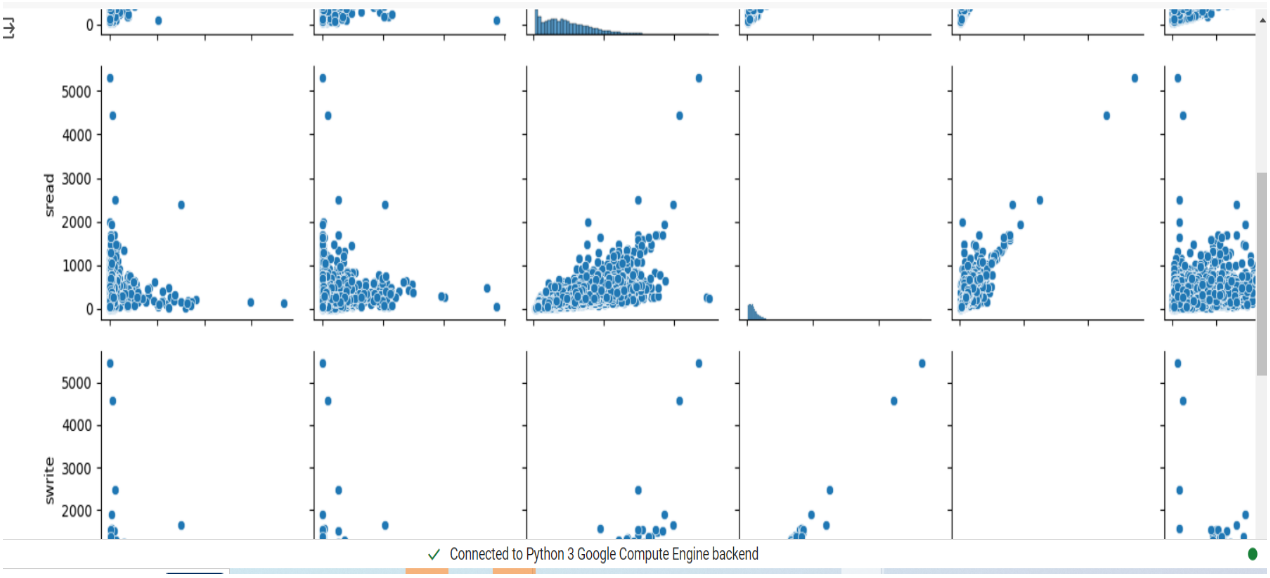

100
80
60
40
20
0
usr
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
fork


100
80
60
40
20
0
usr
0
10
20
30
40
50
60
exec


100
80
60
40
20
0
usr
0.0
0.5
1.0
1.5
2.0
2.5
1e6
rchar


100
80
60
40
20
0
usr
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1e6
wchar

1.1.3 Perform Multivariate Analysis.



1.2 Data Preprocessing

1.2.1 Check of Null Values

```
Missing Values:
 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
```

1.2.2 Impute Null Values

```
[ ] # Handle missing values (e.g., impute or drop)
    # For example, if 'lread' has missing values and you want to impute them with the mean:
    mean_lread = data['lread'].mean()
    data['lread'].fillna(mean_lread, inplace=True)
```

```
[ ] # Check for variables with zeros and handle accordingly
    # For example, if 'pgout' has zeros and they are not meaningful, you can replace them with NaN or drop the rows:
    data['pgout'].replace(0, np.nan, inplace=True)
    data.dropna(subset=['pgout'], inplace=True)
```

1.2.2(B) Check once again after Impute Null Values

```
# check once again missing values
data.isnull().sum()
```

```
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
```

1.2.3 Check for the values which are equal to zero. Do they have any meaning or do we need to change them or drop them

some variable has zero and they are ot meaningful, you can replace with NaN or drop the rows

```
# Check for variables with zeros and handle accordingly
# For example, if 'pgout' has zeros and they are not meaningful, you can replace them with NaN or drop the rows:
data['pgout'].replace(0, np.nan, inplace=True)
data.dropna(subset=['pgout'], inplace=True)
```

1.2.4 Check for Outliers

```
# Check for outliers using box plots or other techniques
# For example, to detect outliers in 'usr' column using the IQR method:
Q1 = data['usr'].quantile(0.25)
Q3 = data['usr'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = data[(data['usr'] < lower_bound) | (data['usr'] > upper_bound)]
print(outliers)
```

```
     lread  lwrite  scall  sread  swrite    fork    exec        rchar  \
7       21      18   2799    291     211  0.60000  0.40000  167537.00000
32       6       1   2414    249     163  1.20000  3.60000  167537.00000
51      50      65   2292    357     201  0.40000  1.80000  167537.00000
63       7       2   2404    211     153  1.40000  2.61000  167537.00000
144     13       0   8309   1407     519 10.80000 14.20000 2214883.00000
...    ...     ...    ...    ...     ...     ...      ...          ...
8002     8       6   1152    138      96  0.40000  0.40000   16017.00000
8017     3       1   1833    659     317  0.20000  0.20000  735962.00000
8084    73      83   4405    540     461  8.42000  2.40000  272948.00000
8086    60      78    862    387     131  0.20000  0.20000  626392.00000
8166    18       1   1135    188     135  2.40000 12.40000  485245.00000

          wchar   pgout  ppgout   pgfree  pgscan    atch     pgin  \
7   259868.00000 2.60000  4.80000  4.80000  0.00000 0.00000  1.00000
32  472149.00000 2.00000  2.60000  2.60000  0.80000 0.80000 11.00000
51   75166.00000 5.80000  9.20000 16.20000 15.40000 1.40000  5.40000
63   43705.00000 0.40000  0.40000  0.40000  0.00000 0.00000  5.01000
144  95033.00000 1.00000  1.40000  1.40000  0.00000 1.40000 14.20000
...          ...     ...      ...      ...      ...     ...      ...
8002  23243.00000 2.40000  2.79000  2.79000  0.00000 1.20000  3.59000
8017 516358.00000 7.60000 14.20000 20.80000 30.40000 3.00000 11.00000
```

```
8084  43129.00000 6.41000  8.82000  8.62000  0.00000 1.40000  6.21000
8086 639838.00000 1.20000  1.40000  1.40000  0.00000 0.00000  0.00000
8166 109897.00000 3.00000  3.80000  3.80000  0.00000 1.80000 14.80000

      ppgin     pflt     vflt    runqsz  freemem  freeswap  usr
7    1.00000  35.40000  71.00000  CPU_Bound      87       13    0
32   15.80000  61.00000 133.40000  CPU_Bound      89       10    0
51    8.00000  33.80000  87.00000  CPU_Bound      88       12    0
63    7.41000  76.95000 129.86000  CPU_Bound      89       11    0
144  22.00000 477.80000 831.20000  CPU_Bound     279   1093539   54
...       ...       ...       ...        ...     ...       ... ...
8002  3.79000  35.33000 141.72000  CPU_Bound      93        7    0
8017 16.00000  27.60000  62.20000  CPU_Bound      89       11    0
8084 11.22000 405.81000 612.22000  CPU_Bound      68       32    0
8086  0.00000  15.60000  34.40000  CPU_Bound      94        6    0
8166 19.20000  60.60000 139.00000  CPU_Bound      88       10    1

[204 rows x 22 columns]
```

1.2.5 Check Duplicates

```
[ ] # Check for duplicates and remove if necessary
    data.drop_duplicates(inplace=True)
```

1.3 Feature Engineering and Encoding

1.3.1 Encode the data (having string values) for Modelling

```python
# Perform feature engineering if required

# Encode categorical variables using one-hot encoding or label encoding
# For example, if 'attribute' is a categorical variable:
encoded_data = pd.get_dummies(data, columns=['runqsz'], drop_first=True)
```

1.3.2 Split the data into train and test (70:30).

Step 4: Train-Test Split

```python
from sklearn.model_selection import train_test_split

# Split the data into train and test sets
X = encoded_data.drop('usr', axis=1)
y = encoded_data['usr']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

1.3.3 Linear Regression Modelling

```
                    OLS Regression Results
==================================================================================
=
Dep. Variable:                usr   R-squared (uncentered):              0.972
Model:                        OLS   Adj. R-squared (uncentered):         0.972
Method:             Least Squares   F-statistic:                         3820.
Date:            Sat, 20 Jan 2024   Prob (F-statistic):                   0.00
Time:                    13:54:30   Log-Likelihood:                    -9343.6
No. Observations:            2319   AIC:                             1.873e+04
Df Residuals:                2298   BIC:                             1.885e+04
Df Model:                      21
Covariance Type:        nonrobust
==================================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
lread         -0.0149      0.005     -2.788      0.005      -0.025      -0.004
lwrite         0.0128      0.010      1.230      0.219      -0.008       0.033
scall          0.0024      0.000      9.481      0.000       0.002       0.003
sread          0.0009      0.003      0.307      0.759      -0.005       0.007
swrite         0.0013      0.003      0.365      0.715      -0.006       0.008
fork          -4.1244      0.425     -9.695      0.000      -4.959      -3.290
exec           0.1743      0.079      2.210      0.027       0.020       0.329
rchar      -4.733e-07   1.44e-06     -0.329      0.742   -3.29e-06    2.35e-06
```

wchar	-2.573e-06	2.28e-06	-1.130	0.259	-7.04e-06	1.89e-06
pgout	0.2290	0.086	2.659	0.008	0.060	0.398
ppgout	-0.0633	0.046	-1.370	0.171	-0.154	0.027
pgfree	-0.0823	0.023	-3.541	0.000	-0.128	-0.037
pgscan	0.0142	0.007	2.098	0.036	0.001	0.028
atch	0.1950	0.079	2.466	0.014	0.040	0.350
pgin	0.0457	0.045	1.025	0.305	-0.042	0.133
ppgin	0.0201	0.028	0.713	0.476	-0.035	0.075
pflt	-0.0372	0.008	-4.871	0.000	-0.052	-0.022
vflt	0.0500	0.006	9.071	0.000	0.039	0.061
freemem	0.0008	0.001	0.908	0.364	-0.001	0.003
freeswap	5.746e-05	5.14e-07	111.807	0.000	5.65e-05	5.85e-05
runqsz_Not_CPU_Bound	14.5413	0.554	26.271	0.000	13.456	15.627

==

Omnibus:	24.020	Durbin-Watson:	1.895
Prob(Omnibus):	0.000	Jarque-Bera (JB):	23.576
Skew:	-0.223	Prob(JB):	7.59e-06
Kurtosis:	2.789	Cond. No.	2.36e+06

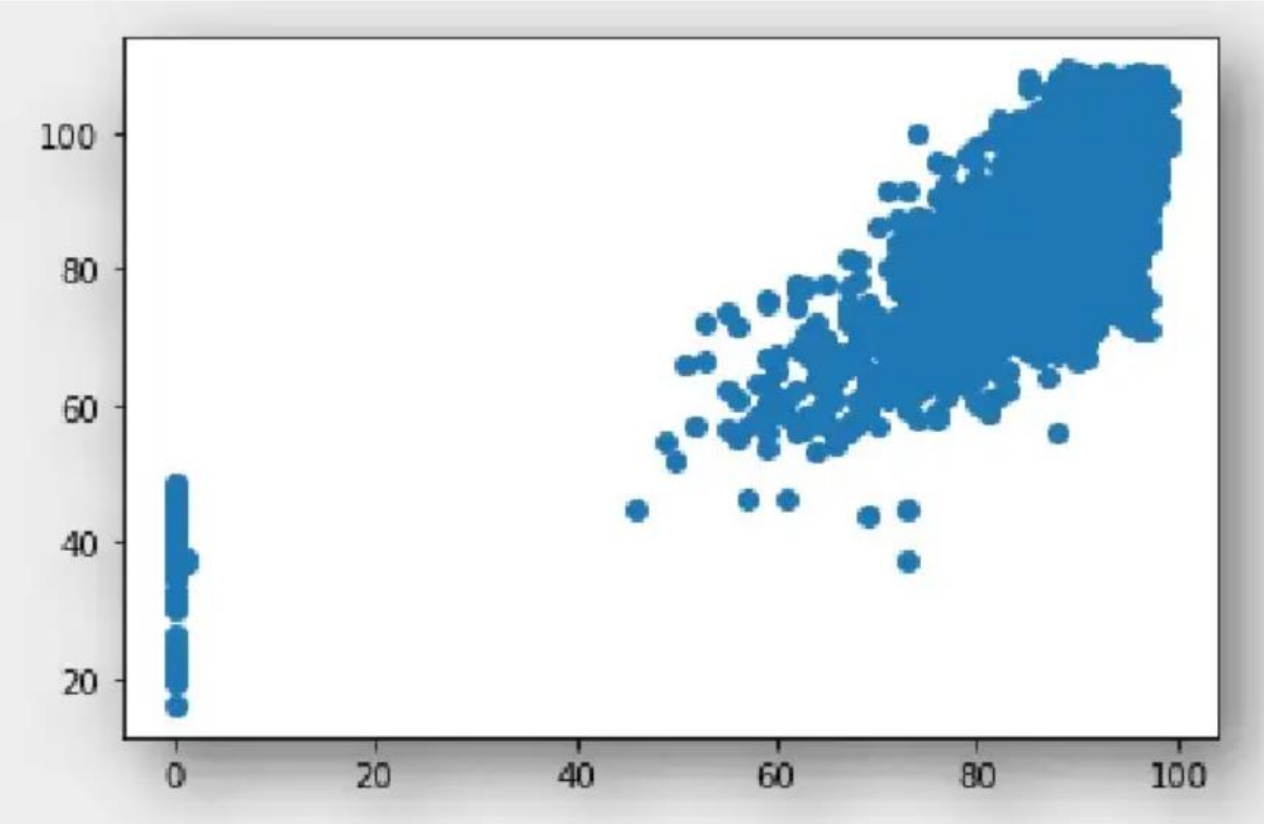
==

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 2.36e+06. This might indicate that there are strong multicollinearity or other numerical problems.

1.3.4 Create multiple models and check the performance of Predictions on Train and Test sets using Rsquare, RMSE & Adj Rsquare

```
Training Set:
R-squared: 0.7022833821208176
RMSE: 11.644767419087993

Test Set:
R-squared: 0.6686290812875155
RMSE: 11.726287748925351
```

With this data we can see there is strong correlation between the data of Y_test and y_prediction



1.4 Inference: Basis on these predictions, what are the business insights and recommendations

The Final linear equation of the given data,(44.64) * const + (-0.02) * lread + (0.0) * lwrite + (0.0) * scall + (-0.0) * sread + (-0.0)* swrite + (-1.72) * fork + (-0.09) * exec + (-0.0) * rchar + (-0.0) * wchar + (-0.17) * pgout +(0.1) * ppgout + (-0.07) * pgfree + (0.01) * pgscan + (-0.08) * atch + (0.09) * pgin + (-0.06) *ppgin + (-0.04) * pflt + (0.02) * vflt + (-0.0) * freemem + (0.0) * freeswap + (7.79) *runqsz_Not_CPU_Bound

When number of faults increases, 'usr' also getting increase by 0.02% and rest of all are in negative value. There are so many negative co-efficient are present in linear equation.

Except 'vflt' and 'rungsz' all co-efficient are decrease when implies.

Totally model was not good enough to predict the future data set as the Outliers dependent is more.

Even including the '0' as the data the linear regression model sensitive for the outliers, if we try to remove these 0, then the information from the data will change.

PART -2

2.0 Problem Statement: Contraceptive Method Dataset

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

2.1 EDA – Exploratory Data Analysis

- Data Ingestion: Read the dataset. Do the descriptive statistics and do null value condition check, check for duplicates and outliers and write an inference on it. Perform Univariate and Bivariate Analysis and Multivariate Analysis.

- Checking the shape of the dataset

```
[148] df.shape

(1473, 10)
```

- Displaying few rows 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	24.00000	Primary	Secondary	3.00000	Scientology	No	2	High	Exposed	No
1	45.00000	Uneducated	Secondary	10.00000	Scientology	No	3	Very High	Exposed	No
2	43.00000	Primary	Secondary	7.00000	Scientology	No	3	Very High	Exposed	No
3	42.00000	Secondary	Primary	9.00000	Scientology	No	3	High	Exposed	No
4	36.00000	Secondary	Secondary	8.00000	Scientology	No	3	Low	Exposed	No

- Checking the data types of the columns for the dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1473 entries, 0 to 1472
Data columns (total 11 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Wife_age                   1473 non-null   float64
 1   Wife_ education            1473 non-null   object
 2   Husband_education          1473 non-null   object
 3   No_of_children_born        1473 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
 10  Education_level            1473 non-null   object
dtypes: float64(2), int64(1), object(8)
memory usage: 126.7+ KB
```

- Statistical summary of the dataset

  There are 8 object variables, 1 integer type and 2 float types variables

- Checking for Missing Values

```
# Check for null values
print(df.isnull().sum())
```

```
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
```

- Impute Missing Values

```
# Impute missing values

df["Wife_age"].fillna(1, inplace=True)
```

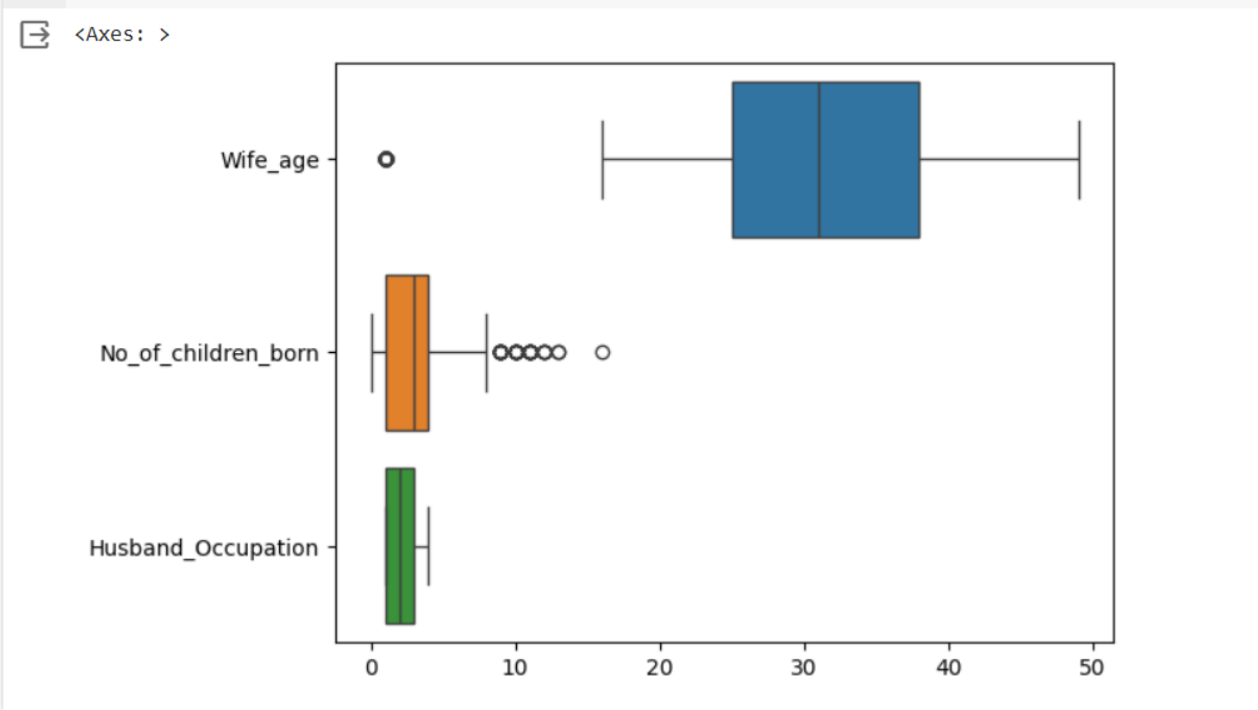
```
[154] # Impute missing values

df["No_of_children_born"].fillna(1, inplace=True)
```

- Checking for Missing Values Once Again

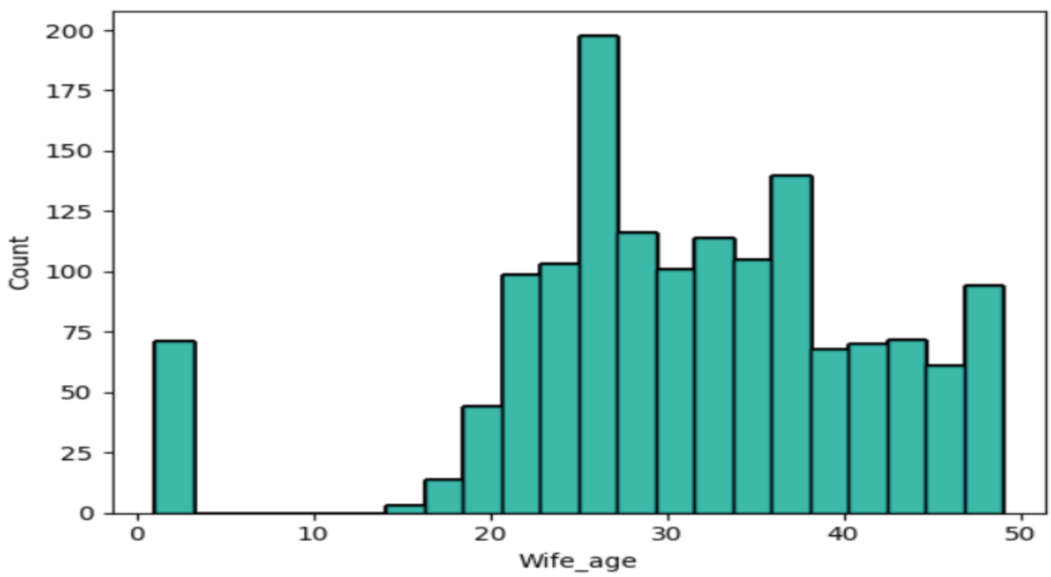
```python
print(df.isnull().sum())
```

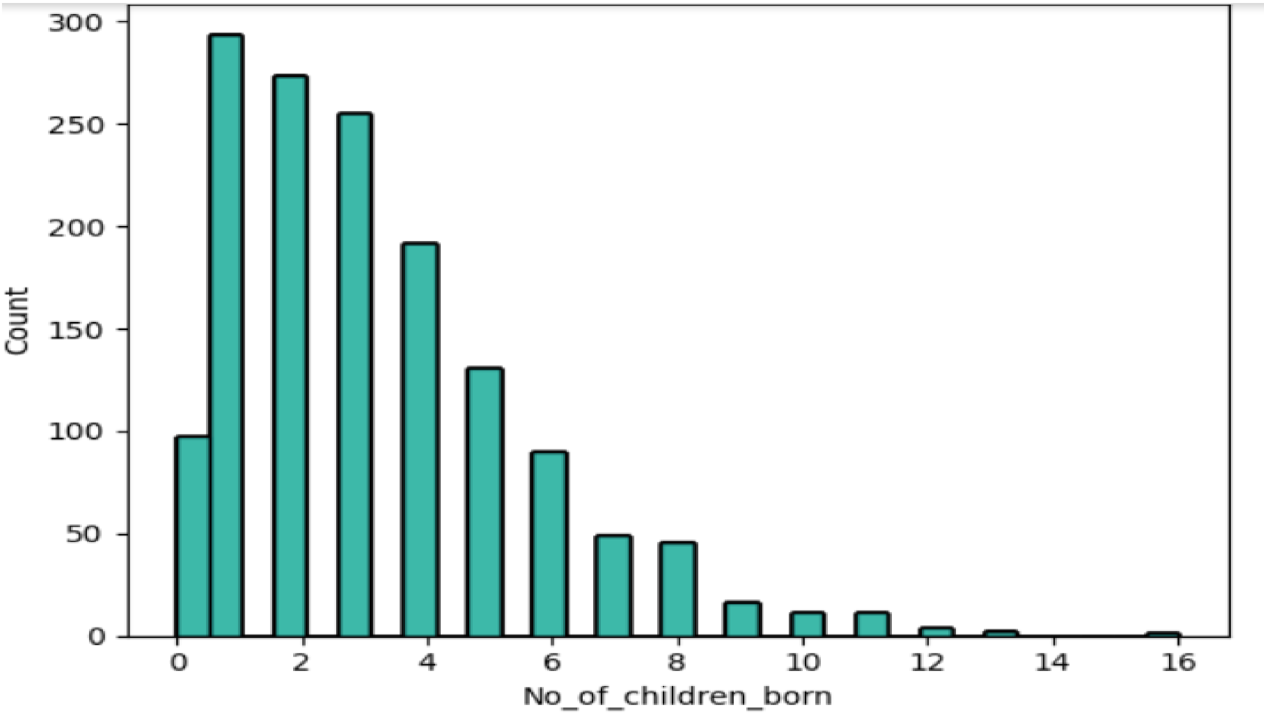
```
Wife_age                     0
Wife_ education              0
Husband_education            0
No_of_children_born          0
Wife_religion                0
Wife_Working                 0
Husband_Occupation           0
Standard_of_living_index     0
Media_exposure               0
Contraceptive_method_used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

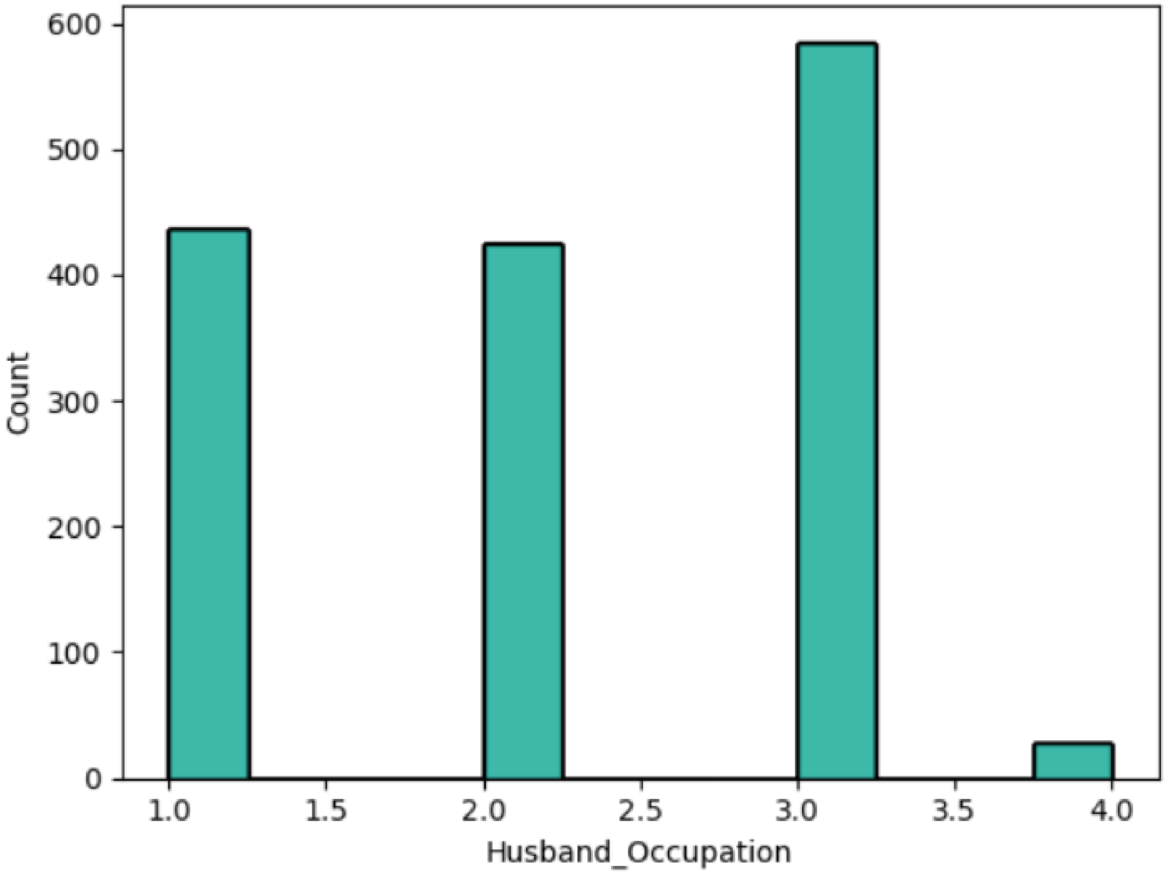
➢ 2.1.1 Check for outliers



➢ 2.1.2 Perform univariate analysis




Count
No_of_children_born


Count
Husband_Occupation

Before we move onto the plots with the object variables, we can change these object as numeric labelling by using the unique codes .Because our model created byLogistics Regression , LDA and the CART will not work on object variables. As the plot informed, some variables having outliers.

But we not going to treat those outliers as it may help us to predict the model by this in formation. For example, Wife_ education

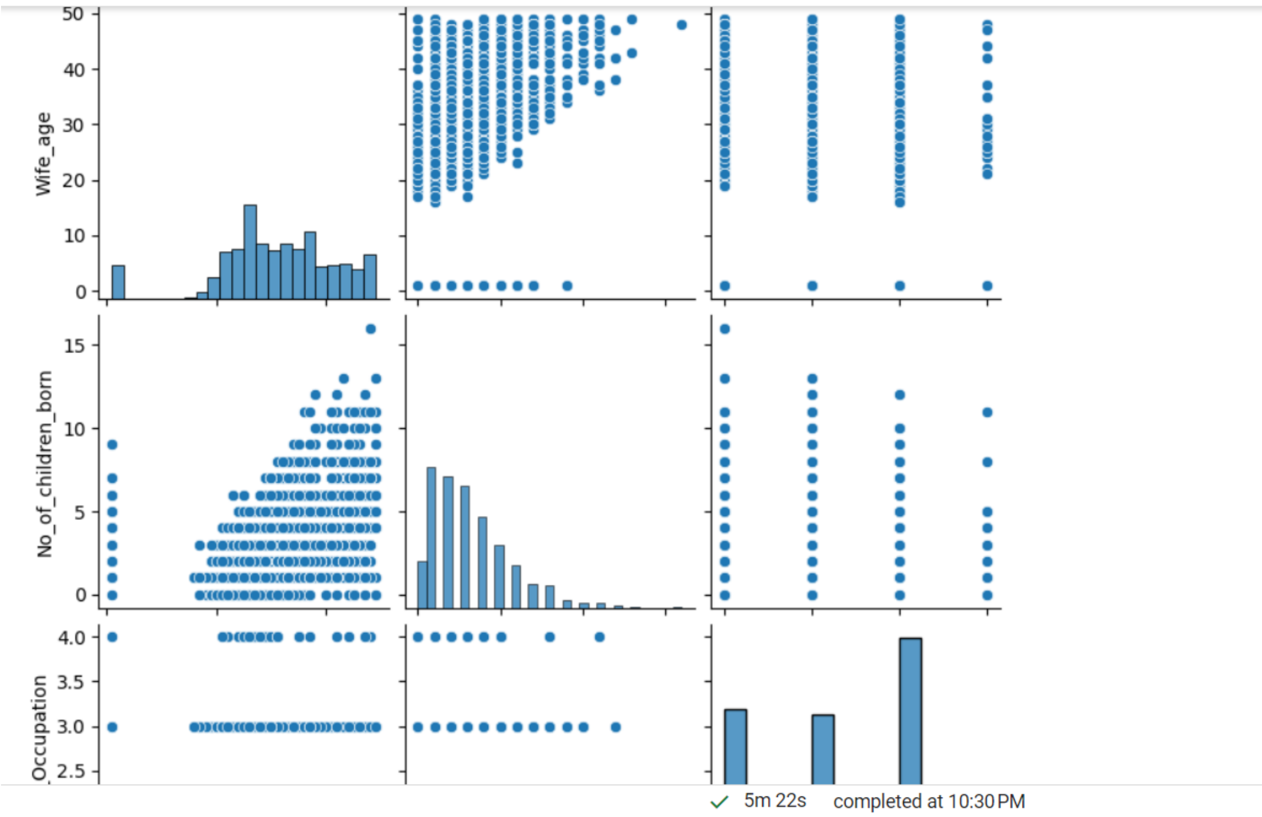
Tertiary 570

Secondary 405

Primary 327

Uneducated 150

- 2.1.3 Perform multivariate analysis



The Pair plot shows how the relationship between the every variables. Among these variables we need to predict those models that they use the contraceptive method or not.

2.2 **Data Preprocessing**

```
2.2.1 # Encode the data that has string values
data = pd.get_dummies(data)


2.2.2 # Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(data, data["Contraceptive method used"], test_size=0.3)
```

Let us create the x and y variable data with respect to ''Contraceptive_method_used''column as the target variable. Now x having every data except the target variable and y having only the target variable . Before we proceed the process , we need to import the required libraries or Checking it. In this encoding for Contraceptive_method_used 1 as yes and 0 as No.

As we already label encoding the object variables , there is no necessary to use Label Encoder form sklearn library.The encoding is for creating the dummy variables

.

```
2.2.3 # Fit a logistic regression model to the train set

logistic_regression = LogisticRegression()
logistic_regression.fit(X_train, y_train)


# Fit a LDA model to the train set
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)


# Fit a CART model to the train set
cart = DecisionTreeClassifier()
cart.fit(X_train, y_train)
```

```
# Fit the model

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

```
▾ LinearRegression
LinearRegression()
```

```
[34] lda = LinearDiscriminantAnalysis()

     lda.fit(X_train, y_train)
```

```
▾ LinearDiscriminantAnalysis
LinearDiscriminantAnalysis()
```

```
cart = DecisionTreeClassifier()

cart.fit(X_train, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

2.3.1 Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix,

```
# Compare the models based on the performance metrics
```

```
Logistic regression accuracy: 1.0
LDA accuracy: 0.6719457013574661
CART accuracy: 1.0
```

```
LDA confusion matrix:
[[ 87 102]
 [ 43 210]]
```

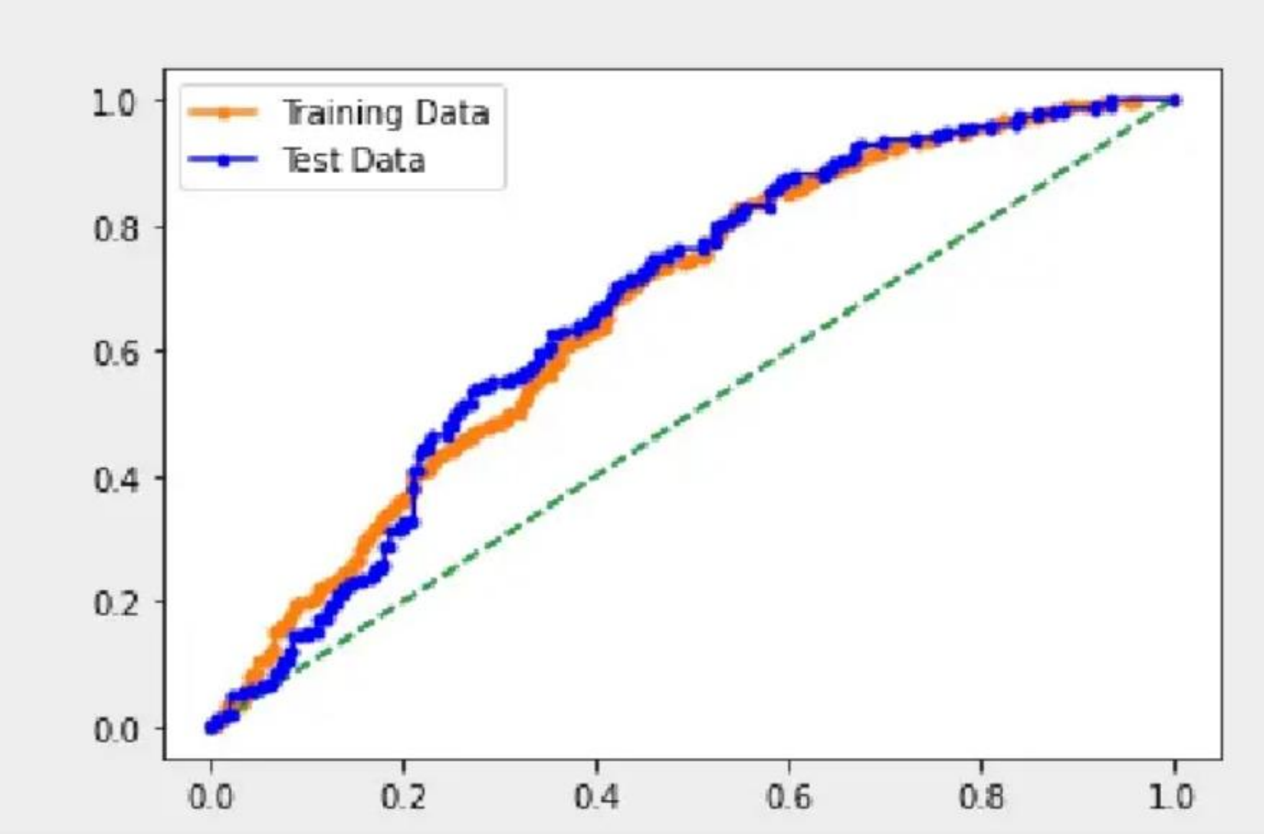
```
CART confusion matrix:
[[189   0]
 [  0 253]]
```

2.3.2 Plot ROC curve and get ROC_AUC score for each model Final Model: Compare Both the models and write inference which model is best/optimized.

```
Logistic regression ROC_AUC score: (array([0., 0., 1.]), array([0., 1., 1.]), array([2, 1, 0]))
LDA ROC_AUC score: (array([0.        , 0.53968254, 1.        ]), array([0.        , 0.83003953, 1.        ]), array([2, 1, 0]))
CART ROC_AUC score: (array([0., 0., 1.]), array([0., 1., 1.]), array([2, 1, 0]))
```

The results of the models will depend on the specific data set. However, in general, logistic regression is a good choice for binary classification problems. LDA is a good choice for problems where the data is normally distributed. CART is a good choice for problems where the data is not normally distributed

AUC curve for test and train data:



This plot shows the AUC curve of both the training data and Test data. And in the curve we can say the train data forming smooth curve ,where test data auc curve forming slightly different . The AUC curve for train data is 67.0%, and AUC for test data is 67.4%

The model accuracy on the training as well as the test set is about 67%, which is roughly the same proportion as the class 0 observations in the dataset. This model isa affected by a class imbalance problem. Since we only have 1473 observations, if re-build the same LDA model with more number of data points, an even better model could be built.

2.4 Inference: Basis on these predictions, what are the insights and recommendations.

From these above models , in Every models the Encoded label ‘1’(conceptive method used) predicted as high and the Accuracy and the F1 score of the models also favour for the label ’1’.
But we can’t conclude that the conceptive method used or not , but we can predict that the married women used the Conceptive method as prediction and their final prediction also showing the same things only.