PREDICTIVE MODELLING BUSSINESS REPORT

PGPDSBA 2022 BATCH
BY AKASH JHA

CONTENTS:

PROJECT OBJECTIVE:

Problem 2: Logistic Regression, LDA, CART.

You are a statistician at the Republic of Indonesia Ministry of Health, and you are provided with a data of 1473 females collected from a Contraceptive Prevalence Survey. The samples are married women who were either not pregnant or do not know if they were at the time of the survey.?

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

2.1. 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.

2.2. Do not scale the data. Encode the data (having string values) for Modelling. Data Split: Split the data into train and test (70:30). Apply Logistic Regression and LDA (linear discriminant analysis) and CART.

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

2.4. Inference: Basis on these predictions, what are the insights and recommendations. Please explain and summarise the various steps performed in this project. There should be proper business interpretation and actionable insights present.

Problem 2 -Logistic Regression,LDA,CART

Data Dictionary and Data Overview:

Data Dictionary:

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=very low, 2, 3, 4=high
9. Media exposure (binary) Good, not good
10. Contraceptive method used (class attribute) No, Yes

Importing Libraries:

To import the dataset and perform Exploratory Data Analysis on the given dataset we imported the following packages:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import scipy.stats as stats
sns.set(style='white')
sns.set(style='whitegrid',color_codes=True)

import sklearn
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import confusion_matrix
```

Solution 2.1:

Read the data and do exploratory data analysis. Describe the data briefly. (Check the null values, Data types, shape, EDA, etc,).

Importing Data:
The dataset is imported in Jupyter Notebook by function **pd.read_csv() ,** Stored dataset in **"data" .** Top Five Rows viewed by function **data.head().**

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

Shape of Dataset:

```
data.shape
```

```
(1473, 10)
```

Structure of Dataset:

Structure of dataset can be computed by using .info () function.

```
data.info()
```

```
<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
```

There are 7 variables of Object type, 2 float type & 1 Integer type.

Check for duplicated & Null values:

There are 71 Nan values in Wife age and 21 Nan values in No_of _children, we need to impute.

```
data.isna().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
```

Clean Dataset:

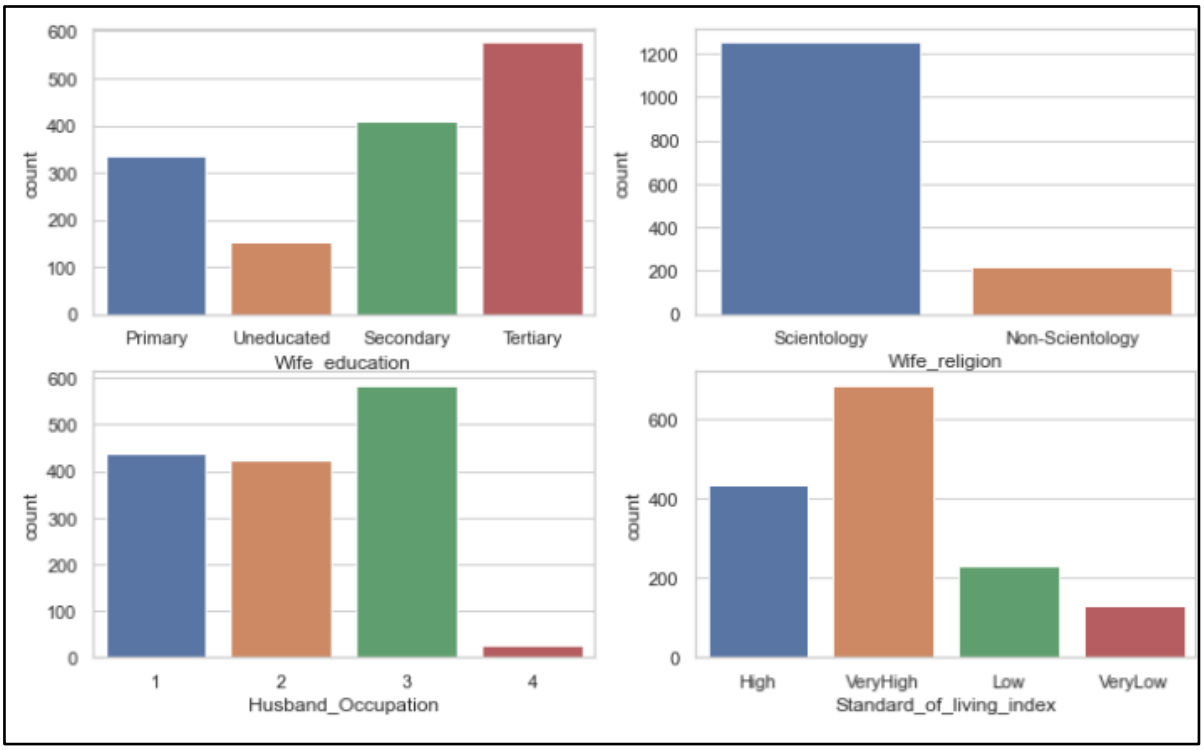
As there are nan values and Duplicate values in Data, we need to impute it with median.
Change datatype of columns, Removes spaces from columns names.

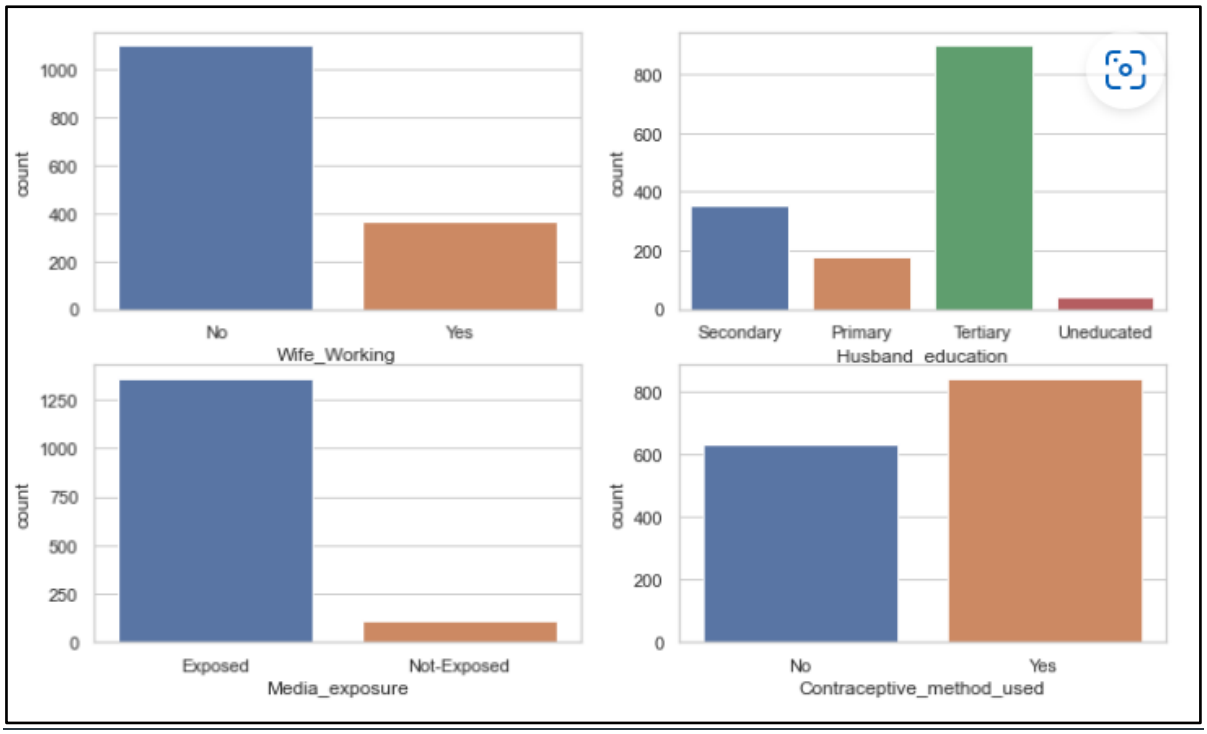
```
#Clean Data
data['Wife_age'] = data['Wife_age'].fillna(data['Wife_age'].median())
data['No_of_children_born']=data['No_of_children_born'].fillna(data['No_of_children_born'].median())

#Change type of husband occupation as its object encoded
data['Husband_Occupation']=data['Husband_Occupation'].astype('object')
#Remove space from Column Names
data.columns = data.columns.str.replace(' ', '')
```

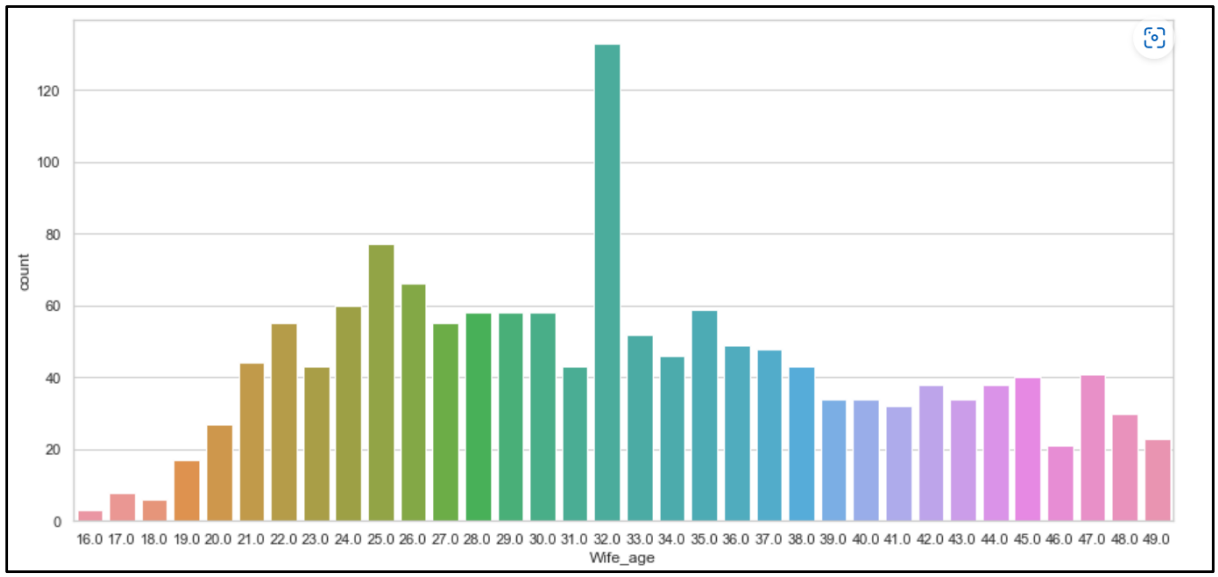
```
data['Standard_of_living_index']= np.where(data['Standard_of_living_index']== 'Very High','VeryHigh' ,data['Standard_of_livin

data['Standard_of_living_index']= np.where(data['Standard_of_living_index']== 'Very Low','VeryLow' ,data['Standard_of_living_
```

Univariate Analysis:

We plot histogram of all numerical variables by function sns.countplot() from seaborn package.





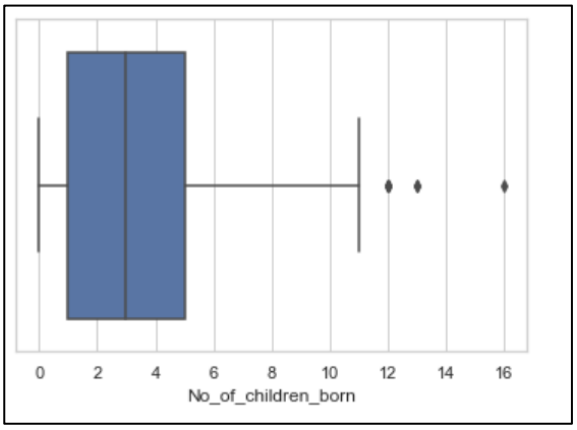


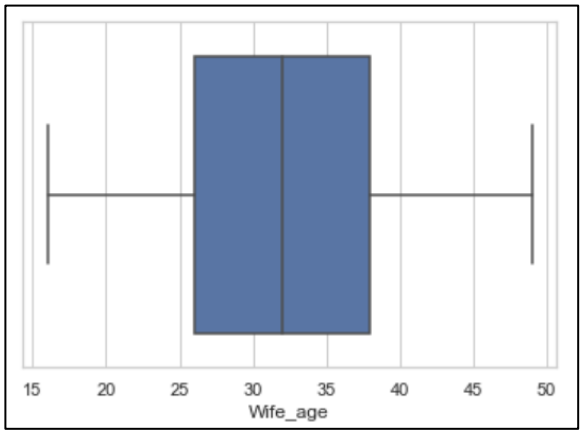
Inference:

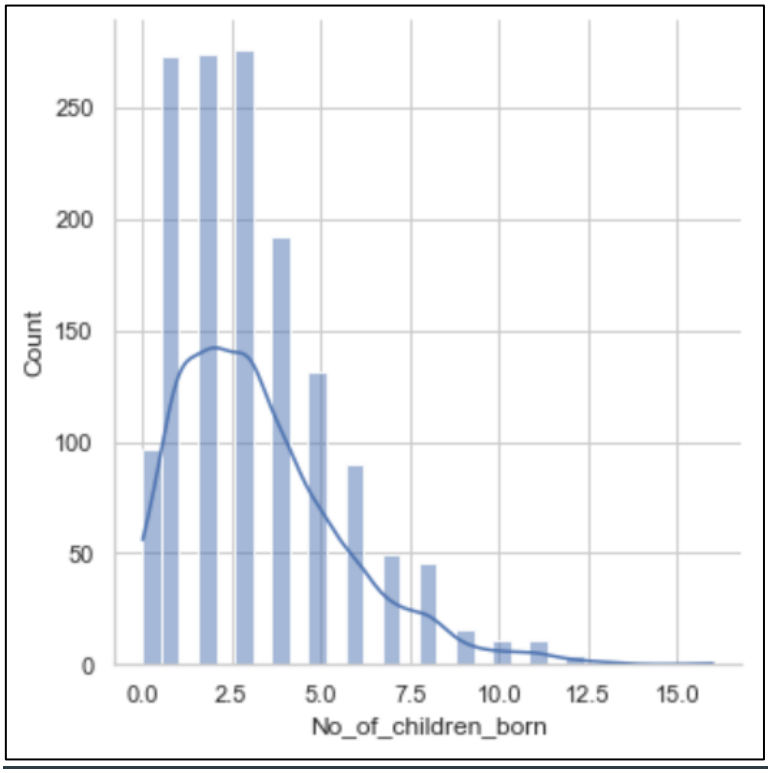
Following are Observations from Count Plots

1) From Count plots we can conclude high numbers of women are exposed to media-exposure.
2) Women are more in Tertiary category compared to others in education, more women are educated.
3) Working Wife's are less in numbers.
4) Standard of living is good for most of them.
5) Most of them are from Scientology religion.

Boxplot of Variables to Check for Outliers:



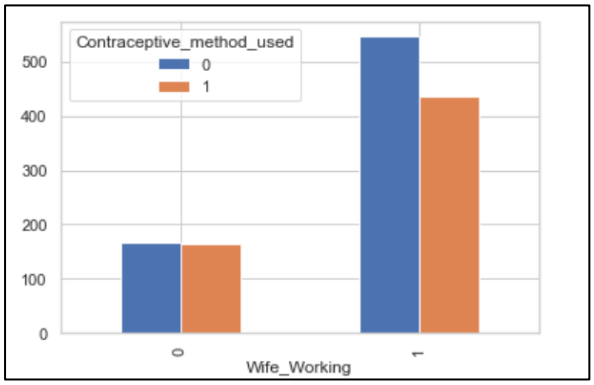


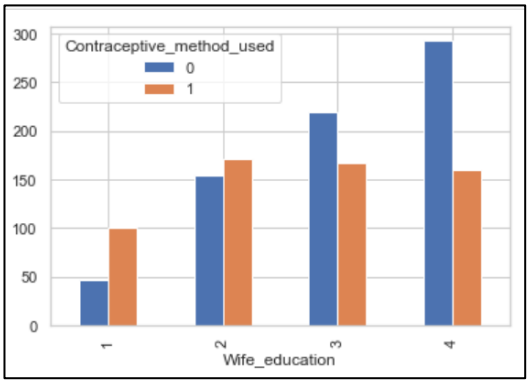


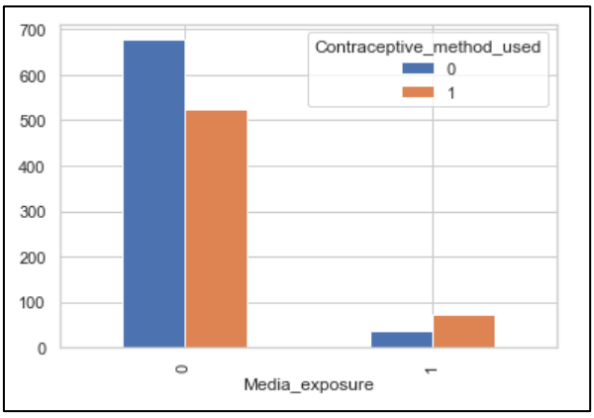
Inference:

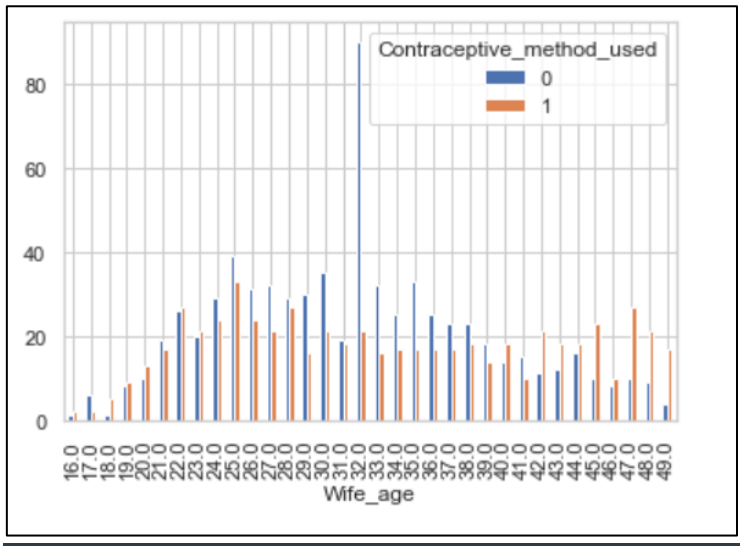
After Plotting Boxplot for variables, we can conclude there are few outliers in No_of_children_born Since there are very small numbers of outliers we can treat or not, it won't make any difference in our analysis. Still, we had treated outliers.

Bivariate Analysis:









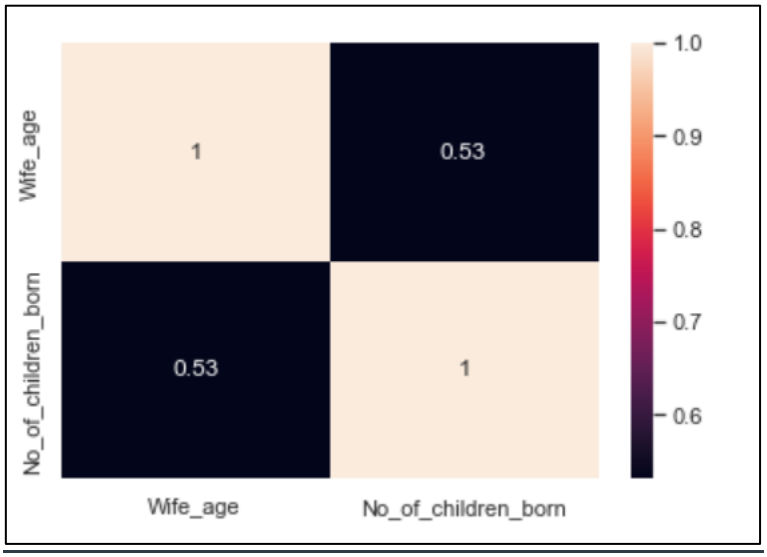
Inference:

- **Contraceptive method encoded as Yes:0 & No:1**
- By Wife age Vs Contraceptive Plot, we can say, Mid Age (25-40) age group are highest to take contraceptive method.
- High Educated Women are taking high amount of medicine to prevent pregnancy compared to low level educated women.
- There are large number of non-working wife who are using contraceptive method.
- Media Exposed wife highly tend to use contraceptive method compare to no-Exposed ones.

Multivariate Analysis:

Heat Map (Relationship Analysis)

We will now plot a Heat Map or Correlation Matrix to evaluate the relationship between different variables in our dataset. This graph can help us to check for any correlations between different variables.



Inference:

→ As per Heat Map, following variables are correlated.

→ Wife _age & No of Children are corelated.

Treating Outliers by Following Function:

```
# Treating One Column Outliers,No_of_children_born
col =['No_of_children_born']
Q1=df1[col].quantile(0.25)
Q3=df1[col].quantile(0.75)
IQR = Q3-Q1

condition=~((df1[col]< (Q1 - 1.5*IQR)) | (df1[col] > (Q3 + 1.5 * IQR))).any(axis=1)

fill_df1=df1[condition]
```

2.2 Do not scale the data. Encode the data (having string values) for Modelling. Data Split: Split the data into train and test (70:30). Apply Logistic Regression and LDA (linear discriminant analysis) and CART.

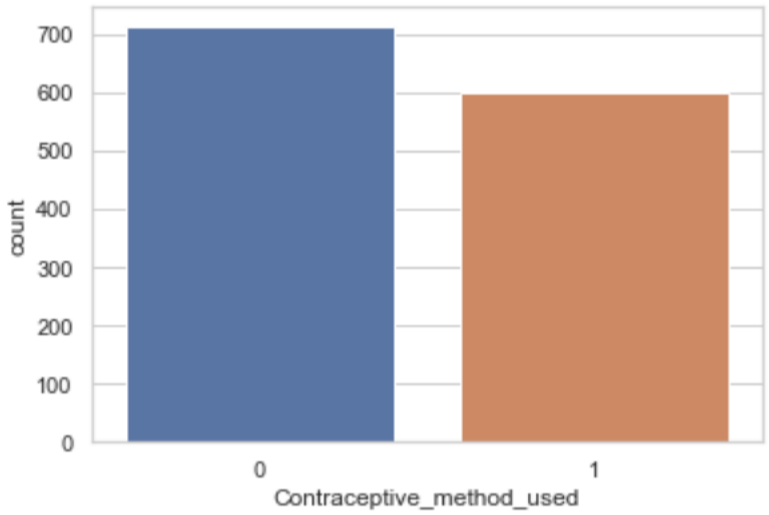
While Linear Regression helps us predicting continuous target variable, Logistic Regression helps us for predicting a discrete a target variable. Logistic Regression is one of the "Whitebox" algorithms which helps us in determining the probability values and the corresponding cut-offs. Logistic regression is used to solve such problem which gives us the corresponding probability outputs and then we can decide the appropriate cut-off points to get the target class outputs.

Evaluation of Logistic regression model- Performance measurement of classification algorithms is judge by confusion matrix which comprise the classification count values of actual and predicted labels.

In the given dataset, the target variable – contraceptive_method_used.

Variables can be encoded into numerical values for model creation analytical purposes.

```
sns.countplot(df1.Contraceptive_method_used);
```




```
df1['Contraceptive_method_used'].value_counts(1)
```

```
0    0.543031
1    0.456969
Name: Contraceptive_method_used, dtype: float64
```

YES:0 – 54% in Dataset, NO:1 – 46% in Dataset

The distribution seems to be fine, with 46% for no and 54% for yes

Let's now initiate the model:

We need to copy all the predictor variables into X Data frame & copy the target Variable in Y Data frame.

Spilt Data with Test & Train Set, with size of (70:30)

```
# Train Test Spilt
X2= fill_df1.drop('Contraceptive_method_used',axis=1)
Y2= fill_df1.pop('Contraceptive_method_used')
```

```
X1_train,X1_test,y1_train,y1_test=train_test_split(X2,Y2,test_size=0.30,random_state=1)
```

Checking the data split for the dependent variable – Y in both train and test data. The percentage split between no and yes seems to be almost same at 45% and 55%, respectively for both train and test data sets.

```
y1_train.value_counts(1)
```

```
0    0.565646
1    0.434354
Name: Contraceptive_method_used, dtype: float64
```

```
y1_test.value_counts(1)
```

```
1    0.507653
0    0.492347
Name: Contraceptive_method_used, dtype: float64
```

The data proportion seems to be reasonable, and we can continue with our model building as next steps.

As next steps, we will initiate the LogisticRegression function and will then fit the Logistic Regression model. There after we will predict on the training and test data set.

Logistic Regression Model

```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.metrics import confusion_matrix,classification_report,plot_confusion_matrix
```

```
model1=LogisticRegression()
model1.fit(X1_train,y1_train)
```

```
4]: LogisticRegression()
```

```
model1.fit(X1_train,y1_train)
```

```
5]: LogisticRegression()
```

```
model1.intercept_
```

```
6]: array([1.10212039])
```

```
model1.coef_
```

```
7]: array([[ 0.07944226, -0.44334872, -0.1316103 , -0.35154588, -0.66263508,
         0.02166494, -0.25928592, -0.17063442,  0.55687751]])
```

After fitting model, Intercept – 1.10 and model coefficient are above mentioned.
Media Exposure seems most impactful variable as per Logistic regression but.
Cons of Logistic- Classifier constructs linear boundaries and the interpretation of coefficients value is difficult.

Predictions:

```
y1_predict=model1.predict(X1_test)

y1_trainpredict=model1.predict(X1_train)

model_score=model1.score(X1_test,y1_test)

model_score_train=model1.score(X1_train,y1_train)

print(model_score_train)
```

```
0.6673960612691466
```

```
print(model_score)
```

```
0.6096938775510204
```

Model Accuracy scores:
Training data: model. Score (X1_train, y1_train) at 66.1%
Test data: model. Score (X1_test, y1_test) at 60.6%

The accuracy scores aren't too different and can be considered as right fit models avoiding the scenarios of underfit and overfit models.

Probabilities Prediction:

```
y1_predictprob= model1.predict_proba(X1_test)
pd.DataFrame(y1_predictprob).head()
```

	0	1
0	0.611803	0.388197
1	0.738735	0.261265
2	0.699772	0.300228
3	0.644874	0.355126
4	0.325091	0.674909

Confusion Matrix & Classification Report:

```
print(model_score)
print(confusion_matrix(y1_test,y1_predict))
print(classification_report(y1_test,y1_predict))
```

```
0.6096938775510204
[[158  35]
 [118  81]]
              precision    recall  f1-score   support

           0       0.57      0.82      0.67       193
           1       0.70      0.41      0.51       199

    accuracy                           0.61       392
   macro avg       0.64      0.61      0.59       392
weighted avg       0.64      0.61      0.59       392
```

As we can see, For 0 (yes) precision is 0.57 and 1(No) precision is 0.70, Recall is 0.70 and 0.41

Note:

Precision: tells us how many predictions are actually positive, out of predicted positive

Recall: tells us how many predictions are positive and predicted positive.

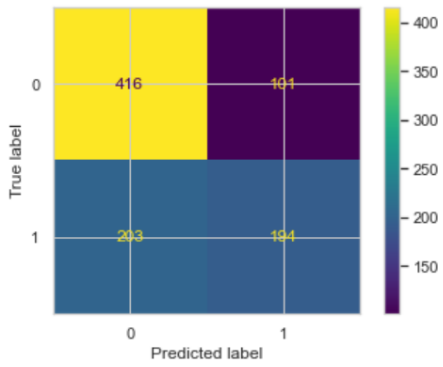
- Precision: 57% predicted positive are actually positive. (Uses Contraceptive method).
- Recall: Out of all using Contraceptive method women's , 82% are actually predicted correctly who are using Contraceptive method.

Confusion Matrix Training Set:

```
# Confusion matrix for training Data
confusion_matrix(y1_train,y1_trainpredict)
```

```
array([[416, 101],
       [203, 194]], dtype=int64)
```

```
plot_confusion_matrix(model1,X1_train,y1_train);
```



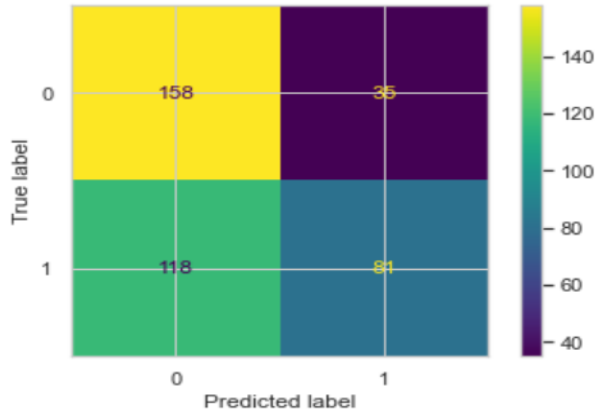
Confusion Matrix Testing Set:

```
# confusion_matrix for test data
confusion_matrix(y1_test,y1_predict)
```

```
array([[158,  35],
       [118,  81]], dtype=int64)
```

```
plot_confusion_matrix(model1,X1_test,y1_test)
```

```
<sklearn.metrics._plot.confusion_matrix.Confus
```



Linear Discriminant Analysis Model

Fitting model in LDA

```
# Build LDA
clif=LinearDiscriminantAnalysis()
model=clif.fit(X1_train,y1_train)
model
```

```
LinearDiscriminantAnalysis()
```

Generate Coefficients and intercept for the Linear Discriminant Function

```
model.intercept_
```

```
array([1.10819295])
```

```
model.coef_
```

```
array([[ 0.07849492, -0.43795411, -0.12939559, -0.34158978, -0.69467487,
         0.00845422, -0.25831741, -0.18569624,  0.5621857 ]])
```

Linear Discriminant Function

1.10 +(0.07*Wife_age)+(-0.43* Wife_education)+(-0.12 * Husband_education)+(-0.34 * No_of_children_born)+(-0.69 * Wife_religion)+(0.008 * Wife_Working)+(-0.25 * Husband_Occupation)+(-0.18 * Standard_of_living_index)+(0.56 * Media_exposure)

According to LDA Function Media Exposure is Most influential Coefficient.

Prediction:

```
# train Prediction
pred_class_train=model.predict(X1_train)

# test Prediction
pred_class_test=model.predict(X1_test)
```

Classification Report of Both Training & Testing Dataset:

```
Classification Report of training Data:

              precision    recall  f1-score   support

           0       0.67      0.81      0.73       517
           1       0.66      0.48      0.55       397

    accuracy                           0.67       914
   macro avg       0.66      0.64      0.64       914
weighted avg       0.67      0.67      0.66       914


Classification Report of test Data:

              precision    recall  f1-score   support

           0       0.57      0.82      0.68       193
           1       0.70      0.40      0.51       199

    accuracy                           0.61       392
   macro avg       0.64      0.61      0.59       392
weighted avg       0.64      0.61      0.59       392
```

Test Data:
Accuracy – 61%
Precision – 57%
Recall – 82%

Training Data:
Accuracy – 67%
Precision -67%
Recall – 81 %

CART MODEL

The cart model is built with various combinations of parameters.

Firstly, Convert all Object Data type into Categorical Data Type.

```
for feature in tree_df1.columns:
    if tree_df1[feature].dtype=='object':
        tree_df1[feature]=pd.Categorical(tree_df1[feature]).codes
```

Spilt Data into 70:30 ratio into Test and Train data.

```
x_train,x_test,train_labels,test_labels=train_test_split(X1,Y1,test_size=0.30,random_state=1)
```

Fit model in DecisionTreeClassifier

```
from sklearn.tree import DecisionTreeClassifier

dt_model=DecisionTreeClassifier(criterion='gini',random_state=1)
```

Used Criterion as Gini

Without Regularized data below are mentioned important variables.

```
print(pd.DataFrame(dt_model.feature_importances_,columns=['Imp'],index=x_train.columns))
```

```
                            Imp
Wife_age                0.294452
Wife_education          0.131818
Husband_education       0.082594
No_of_children_born     0.239607
Wife_religion           0.032646
Wife_Working            0.047535
Husband_Occupation      0.076268
Standard_of_living_index  0.084276
Media_exposure          0.010804
```

Regularize model by giving max depth=5 sub criteria:

```
from sklearn import tree

train_char_label = ['No', 'Yes']
ld_Tree_File = open('d:ld_Tree_File.dot','w')
dot_data = tree.export_graphviz(dt_model,
                                out_file=ld_Tree_File,
                                feature_names = list(x_train),
                                class_names = list(train_char_label))

ld_Tree_File.close()
```

Lets Regularized

```
reg_dt_model=DecisionTreeClassifier(criterion='gini',random_state=1,max_depth=5)
```

Important features of regularized model:

```
rint(pd.DataFrame(reg_dt_model.feature_importances_,columns=['Imp'],index=x_train.columns).sort_values('Imp',ascending=Fals
```

	Imp
No_of_children_born	0.466978
Wife_age	0.315046
Wife_education	0.143365
Husband_education	0.050356
Wife_religion	0.013640
Husband_Occupation	0.007890
Standard_of_living_index	0.002726
Wife_Working	0.000000
Media_exposure	0.000000

Getting Prediction Classes of this model:

```
ytest_predict
```

```
array([0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1,
       1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1], dtype=int8)
```

Getting the Predicted Probabilities

```
pd.DataFrame(ytest_predict_prob).head()
```

	0	1
0	0.832512	0.167488
1	0.832512	0.167488
2	0.668478	0.331522
3	0.668478	0.331522
4	0.153846	0.846154

Score for training and testing model :

```
reg_dt_model.score(x_train,train_labels)
```

```
0.7352297592997812
```

```
reg_dt_model.score(x_test,test_labels)
```

```
0.6530612244897959
```

Score Training – 73%

Score Testing- 65%

ROC CURVE (RECEIVER OPERATING CHARACTERISTIC CURVE)

An ROC curve (receiver operating characteristic curve) is **a graph showing the performance of a classification model at all classification thresholds**. This curve plots two parameters: True Positive Rate. False Positive Rate.

ROC Score for Training Model:

```
from sklearn.metrics import roc_auc_score,roc_curve
```

```
probs=reg_dt_model.predict_proba(x_train)
```

```
probs=probs[:,1]
```

```
auc=roc_auc_score(train_labels,probs)
auc
```

```
0.7781207216600324
```

ROC Score for Training Model = 77%

ROC Score for Testing Model:

```
probs1=reg_dt_model.predict_proba(x_test)
probs1=probs1[:,1]
```

```
auc=roc_auc_score(test_labels,probs1)
```

```
auc
```

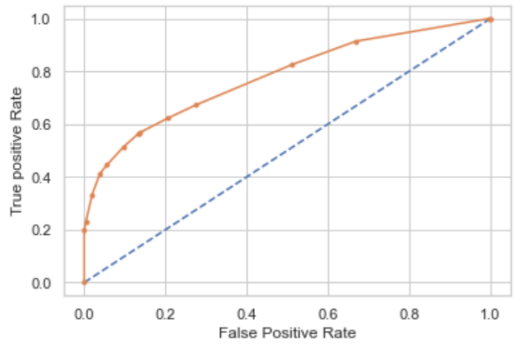
```
0.6777931106308746
```

ROC Curve for Testing Model = 67%

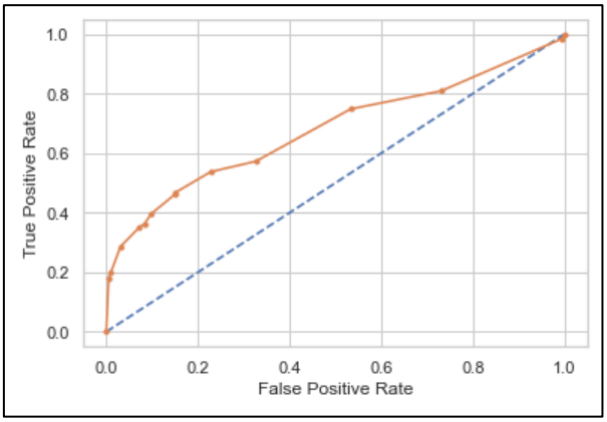
So, for testing and training Model ROC Curve Score is nearabout similar, we can say Model is not Overfitted or Under Fitted.

ROC Curve for training and testing Model:

```
plt.plot([0,1],[0,1],linestyle='--')
plt.plot(fpr,tpr,marker='.')
plt.xlabel('False Positive Rate')
plt.ylabel('True positive Rate')
plt.show()
```



Training Model



Testing Model

Classification Report for Training and testing Model of CART Model:

```
print(classification_report(train_labels,ytrain_predict))
```

	precision	recall	f1-score	support
0	0.72	0.87	0.79	517
1	0.76	0.56	0.65	397
accuracy			0.74	914
macro avg	0.74	0.72	0.72	914
weighted avg	0.74	0.74	0.73	914

```
print(classification_report(test_labels,ytest_predict))
```

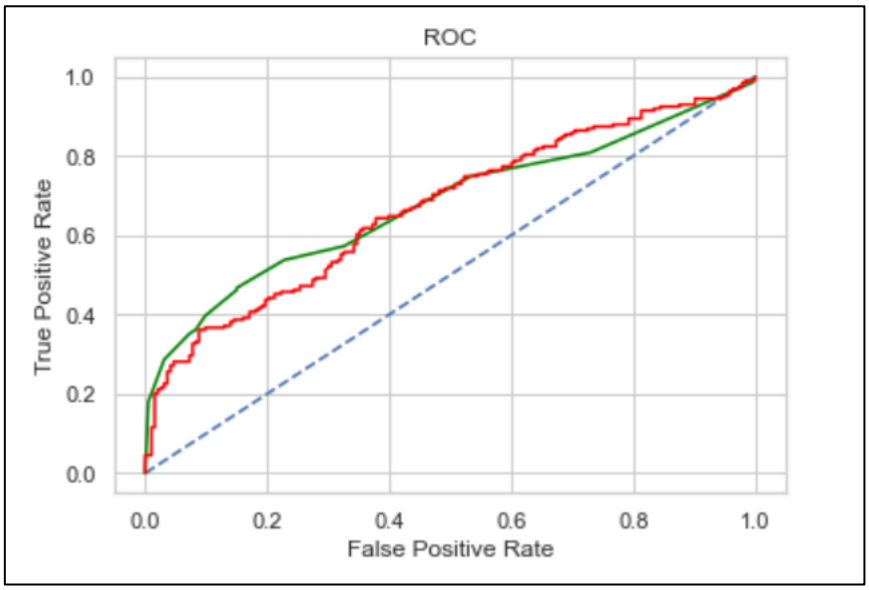
	precision	recall	f1-score	support
0	0.61	0.85	0.71	193
1	0.76	0.46	0.58	199
accuracy			0.65	392
macro avg	0.68	0.66	0.64	392
weighted avg	0.68	0.65	0.64	392

Test Data:
Accuracy – 65%
Precision – 85%
Recall – 61%
F1 Score- 71%

Training Data:
Accuracy – 74%
Precision -72%
Recall – 87 %
F1 Score – 79%

2.4) Inference: Based on the whole Analysis, what are the business insights and recommendations?

Comparing Different models



By Looking Comparison Cart Model seems to have accuracy ratio compared to other two model.

- → No_of_Childern, Wife_Age, Wife_Education found to be important factor for predicting if Individual will use Contraceptive method.
- → Government should focus more on Women Education.
- → More No of Children motivates people to use Contraceptive method.
- → Government should initiate plans for reducing cost of living.

PROJECT OBJECTIVE:

Problem 1: Linear Regression.

The comp-activ databases is a collection of a computer systems activity measures.
The data was collected from a Sun SPARCstation 20/712 with 128 Mbytes of memory running in a multi-user university department. Users would typically be doing a large variety of tasks ranging from accessing the internet, editing files, or running very cpu-bound programs.

As you are a budding data scientist you thought to find out a linear equation to build a model to predict 'usr'(Portion of time (%) that cpus run in user mode) and to find out how each attribute affects the system to be in 'usr' mode using a list of system attributes.

Data Dictionary:

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.

Q1.1. Read the data and do exploratory data analysis. Describe the data briefly. (Check the null values, Data types, shape, EDA). Perform Univariate and Bivariate Analysis.

Exploratory Data Analysis or (EDA) is understanding the data sets by summarizing their main characteristics often plotting them visually. This step is very important especially when we arrive at modelling the data. Plotting in EDA consists of Histograms, Box plot, pairplot and many more. It often takes much time to explore the data. Through the process of EDA, we can define the problem statement or definition on our data set which is very important.

Imported the required libraries.

To build the Linear Regression Model on our dataset we need to import the following packages:

```
#Import important libraries
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set(color_codes=True)

import os as os
import sklearn
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score,roc_curve
import statsmodels.api as sm
```

➢ Importing the dataset - “data.csv”

➢ Data Summary and Exploratory Data Analysis:

✓ Checking if the data is being imported properly

✓ Head: The top 5 rows of the dataset are viewed using head () function

	lread	lwrite	scall	sread	swrite	fork	exec	rchar	wchar	pgout	...	pgscan	atch	pgin	ppgin	pflt	vflt	runqsz	freemem	freesw
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	1730
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	1869
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	1021
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	1863
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	1760

Dimension of the Dataset:

The Dimension or shape of the dataset can be shown using shape function. It shows that the dataset given to us has 8192 rows and 22 columns or variables.

Structure of the Dataset:

Structure of the dataset can be computed using .info() function

```
 #   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
```

Descriptive Statistics for the dataset :

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
lread	8192.0	NaN	NaN	NaN	19.559692	53.353799	0.0	2.0	7.0	20.0	1845.0
lwrite	8192.0	NaN	NaN	NaN	13.106201	29.891726	0.0	0.0	1.0	10.0	575.0
scall	8192.0	NaN	NaN	NaN	2306.318237	1633.617322	109.0	1012.0	2051.5	3317.25	12493.0
sread	8192.0	NaN	NaN	NaN	210.47998	198.980146	6.0	86.0	166.0	279.0	5318.0
swrite	8192.0	NaN	NaN	NaN	150.058228	160.47898	7.0	63.0	117.0	185.0	5456.0
fork	8192.0	NaN	NaN	NaN	1.884554	2.479493	0.0	0.4	0.8	2.2	20.12
exec	8192.0	NaN	NaN	NaN	2.791998	5.212456	0.0	0.2	1.2	2.8	59.56
rchar	8088.0	NaN	NaN	NaN	197385.728363	239837.493526	278.0	34091.5	125473.5	267828.75	2526649.0
wchar	8177.0	NaN	NaN	NaN	95902.992785	140841.707911	1498.0	22916.0	46619.0	106101.0	1801623.0
pgout	8192.0	NaN	NaN	NaN	2.285317	5.307038	0.0	0.0	0.0	2.4	81.44
ppgout	8192.0	NaN	NaN	NaN	5.977229	15.21459	0.0	0.0	0.0	4.2	184.2
pgfree	8192.0	NaN	NaN	NaN	11.919712	32.36352	0.0	0.0	0.0	5.0	523.0
pgscan	8192.0	NaN	NaN	NaN	21.526849	71.14134	0.0	0.0	0.0	0.0	1237.0
atch	8192.0	NaN	NaN	NaN	1.127505	5.708347	0.0	0.0	0.0	0.6	211.58
pgin	8192.0	NaN	NaN	NaN	8.27796	13.874978	0.0	0.6	2.8	9.765	141.2
ppgin	8192.0	NaN	NaN	NaN	12.388586	22.281318	0.0	0.6	3.8	13.8	292.61
pflt	8192.0	NaN	NaN	NaN	109.793799	114.419221	0.0	25.0	63.8	159.6	899.8
vflt	8192.0	NaN	NaN	NaN	185.315796	191.000603	0.2	45.4	120.4	251.8	1365.0
runqsz	8192	2	Not_CPU_Bound	4331	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freemem	8192.0	NaN	NaN	NaN	1763.456299	2482.104511	55.0	231.0	579.0	2002.25	12027.0
freeswap	8192.0	NaN	NaN	NaN	1328125.959839	422019.426957	2.0	1042623.5	1289289.5	1730379.5	2243187.0
usr	8192.0	NaN	NaN	NaN	83.968872	18.401905	0.0	81.0	89.0	94.0	99.0

→ Freeswap: Minimum value is 2.0 compare to 25% value which is 1042623 , which means data have outliers, 50% value is 1289289 compare to max value of 2243187 , which concludes data is right skewed.

→ Freemem: 50% value is 579 and 75% value is 2002 with mean of 1783, which means majority of data falls under 50% to 75% marks. At lower level.

→ Ppgout: it only shows some value in between 75% to 100%, which means most of data value is considered as Zero, this variable will not be relevant for analysis.

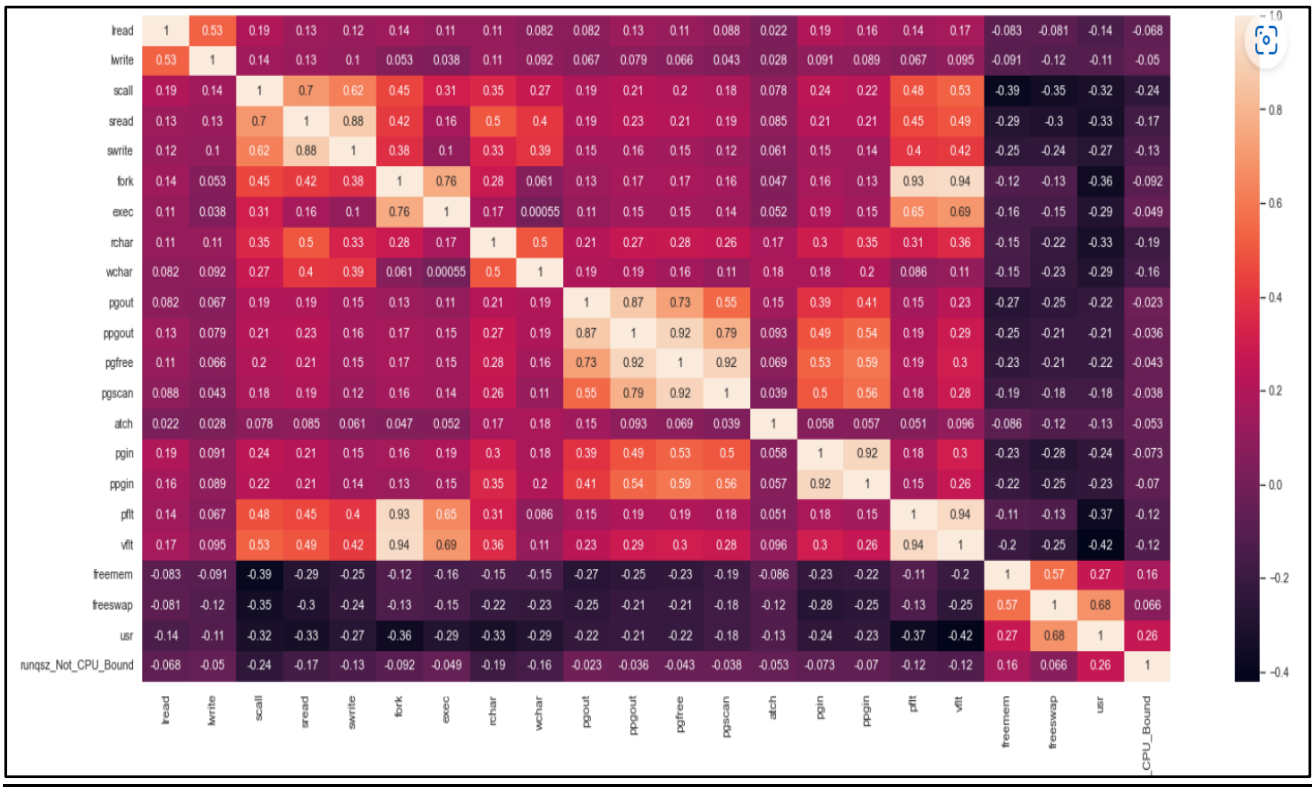
→ User: usr is predicted variable, mean is 83 and 89 value is of 50% , which means most of them fall under lower level.

Missing Value Imputation:

Imputing missing Values by mean of whole column

```
for column in data.columns:
    if data[column].dtype!='object':
        mean=data[column].mean()
        data[column]=data[column].fillna(mean)

data.isna().sum()
```

Checking Correlation in the data using Heatmap:



Observations:

→ High correlation between the different features like freemem, freeswap and user.

→ Less correlation between table with the other features.

→ Depth is negatively correlated with most the other features except for freeswap.

Data Cleaning:

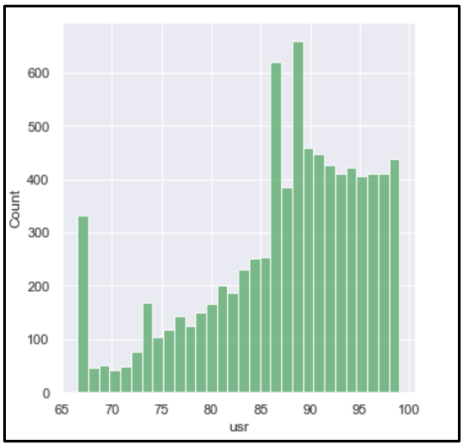
Impute 0 in Usr column

```
1 data['usr']=data['usr'].replace({0:np.nan})
2 data['usr'].fillna(data['usr'].median(),inplace=True)
```

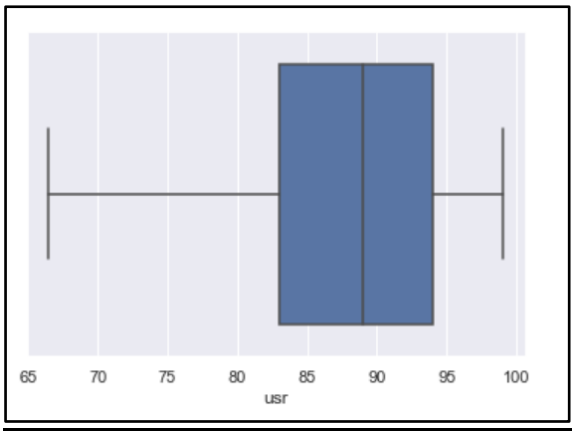
Univariate & Bivariate Analysis:

Define Function for univariate Analysis:

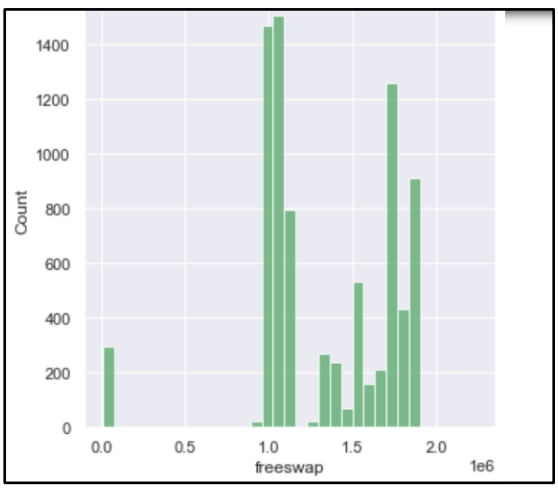
```
def univariateAnalysis(column,nbins):
    print ('Description of '+ column)
    print('--------------------------')
    print(data[column].describe(),end='')

    plt.figure()
    print('Distribution of '+ column)
    print('--------------------------')
    sns.displot(data[column],kde=False,color='g')

    plt.figure()
    print('Boxplot of '+ column)
    print('--------------------------')
    ax=sns.boxplot(x=data[column])
    plt.show()
```

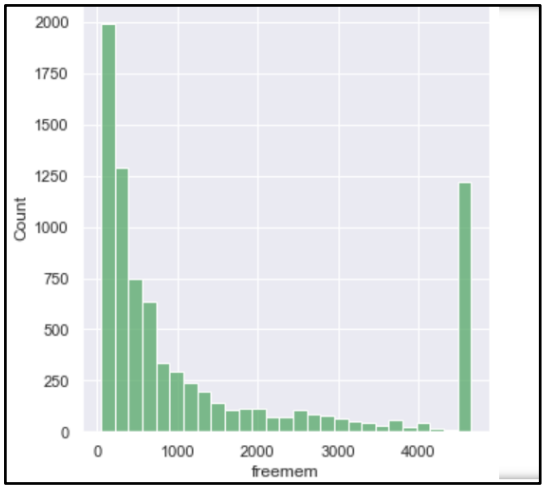


Count Plot of Usr



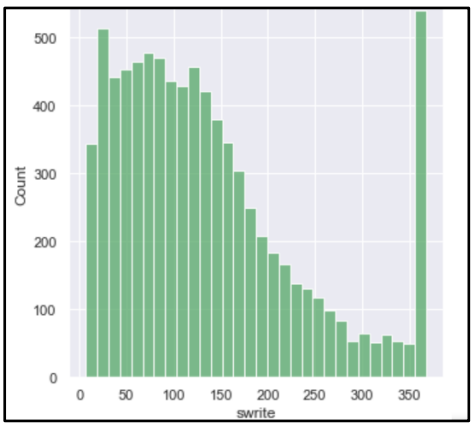
Usr: Majority of falls in between 85 to 98, and from Boxplot we can see usr is right tail skewed, variable does not follow normal distribution.

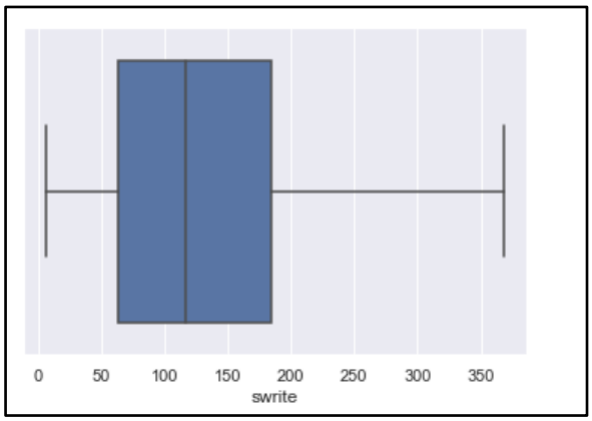




Freeswap: Most of them fall in range between 1 to 1.5, variable is right tail skewed, so number of disk blocks available are high for page swapping.

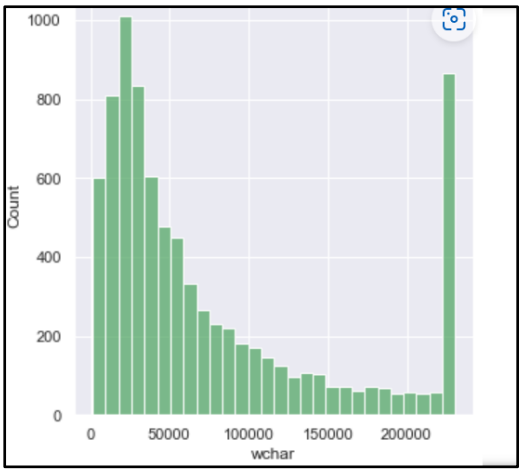
Freemem: Memory pages available for user processes are majority in lesser side.

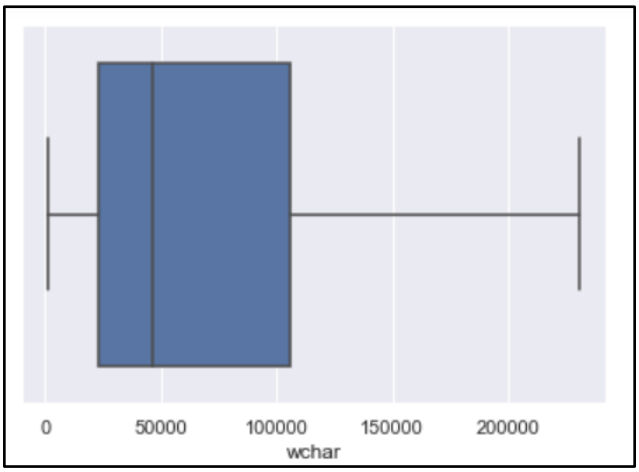




Swrite: Number of system write calls per second data is evenly distributed, but little left tail skewed

.





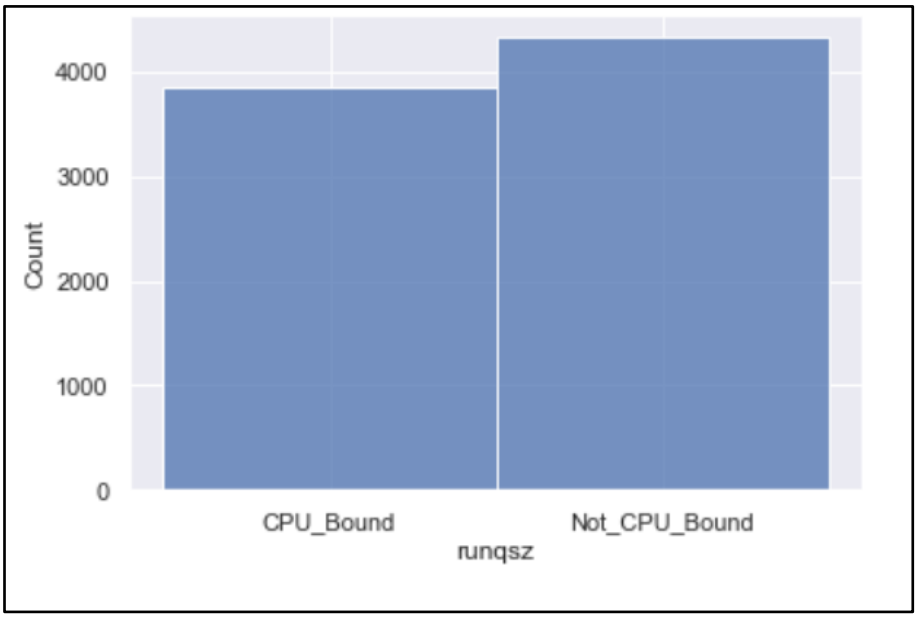
Wchar: Number of characters transferred per second by system write calls is majority on lower range.

Observations:

→ Outliers: Large number of outliers are present in few of the variables.

→ Target variable Usr , is right skewed data and with no outliers.

→ No variable can be Considered as Normal distribution.

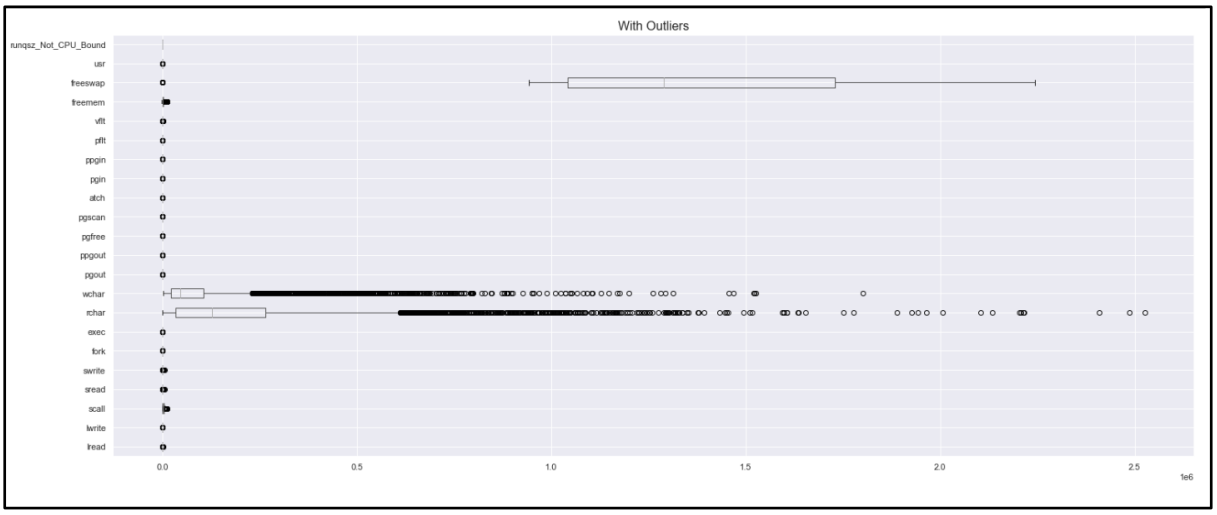


Runqsz is Object Data Type variable and independent variable

→ The number of kernel threads in memory that are waiting for a CPU to run are more in Not CPU_bound compared to CPU_Bound.

Treatment of outliers by IQR method.

Box Plots Before outliers' treatment.

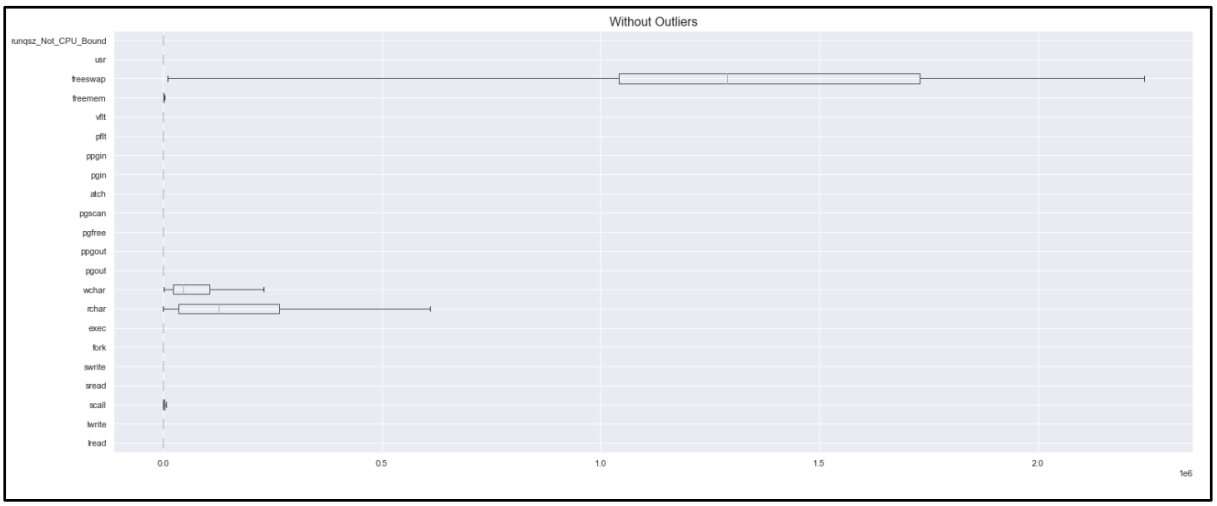


To treat outlier first define function and treat with IQR method:

```
def remove_outliers(col):
    sorted (col)
    Q1,Q3=np.percentile(col,[25,75])
    IQR=Q3-Q1
    lower_range=Q1-(1.5*IQR)
    upper_range=Q3+(1.5*IQR)
    return lower_range,upper_range
```

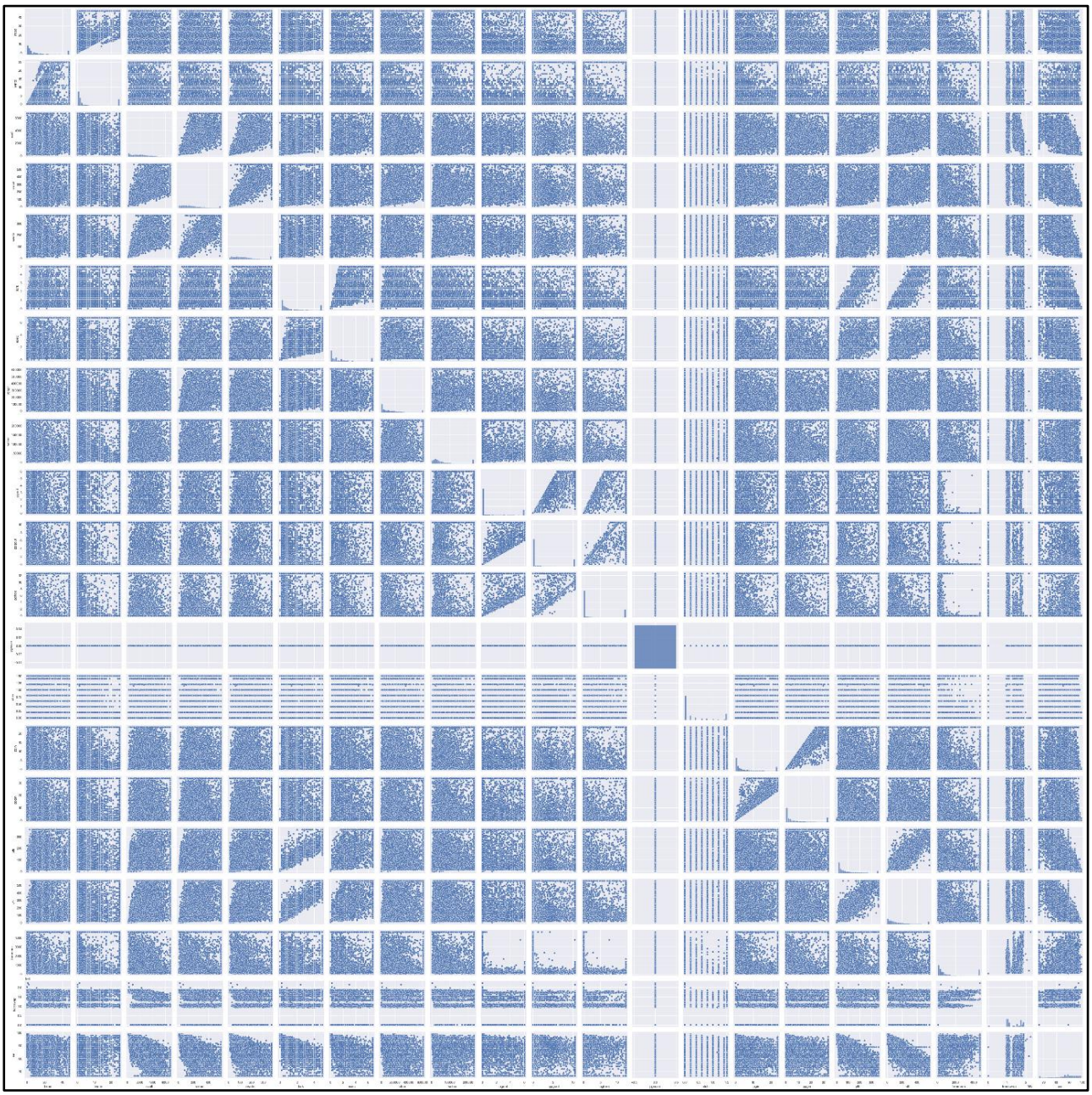
```
for column in data[cols].columns:
    lr,ur=remove_outliers(data[column])
    data[column]=np.where(data[column]>ur,ur,data[column])
    data[column]=np.where(data[column]<lr,lr,data[column])

```

Boxplot after treating Outliers:

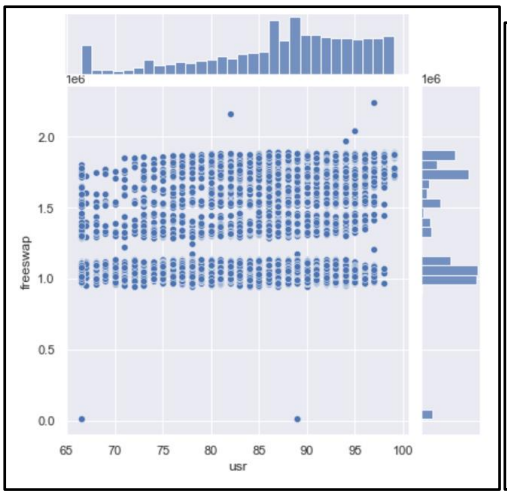


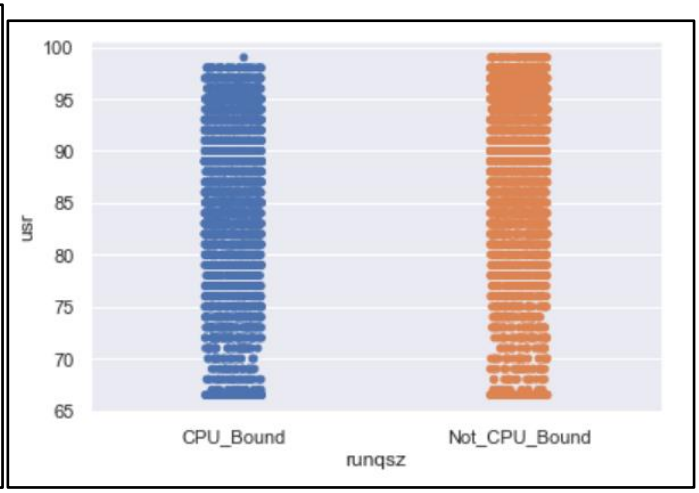
Bivariate Analysis:

Pair Plot :

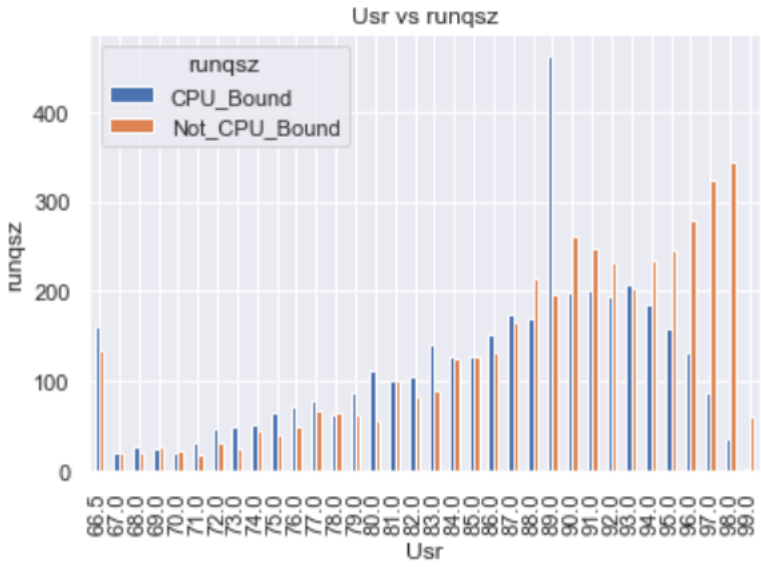


Pair plot allows us to see both distribution of single variable and relationships between two variables.






```
1 pd.crosstab(data.usr,data1.runqsz).plot(kind='bar')
2 plt.title('Usr vs runqsz')
3 plt.xlabel('Usr')
4 plt.ylabel('runqsz')
5 plt.show()
```



Usr Vs runqsz

→ Portion of time (%) that cpus run in user mode at higher range from 83 to 93 is more from Not_CPU_Bound.

→ 89.0 percent is highest for CPU_Bound.

Standardizing the Numerical Variables:

Standardize Numerical variables before doing Spilt with StandardScalar:

```
from sklearn.preprocessing import scale
from sklearn.preprocessing import StandardScaler
```

```
scaler=StandardScaler()
df1_scaled=scaler.fit_transform(data)
```

Q1.3. Encode the data (having string values) for Modelling. Data Split: Split the data into test and train (70:30). Apply Linear regression. Performance Metrics: Check the performance of Predictions on Train and Test sets using Rsquare, RMSE.

Answer:

Create Dummy Variable, as there is only one object data type variable (Runsqz), we can apply dummy function keeping drop first=True

For this case study, I have used one-hot encoding for independent variables using the function "drop_function = True" or can be referred as dummy encoding, which takes into consideration the (Kn-1) encoding. Dummy encoding converts it into n-1 variables.

```
data= pd.get_dummies(data,columns=['runqsz'],drop_first=True)
```

Without scaling the data:

✓ Copy all the predictor variables into X dataframe x = df2.drop('usr', axis=1)

✓ Copy target into the y dataframe. y = df2[['usr']]

✓ **Split x and y into training and test set in 70:30 ratio =**

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30 , random_state=1)

```
df2=pd.DataFrame(df1_scaled,columns=data.columns)
```

```
output=df2['usr']
```

pilt data in Training & Test set (70:30)

```
X_train,X_test,y_train,y_test=train_test_split(df2,output,test_size=0.30,random_state=1)#70:30
```

```
X=df2.drop(['usr'],axis=1)
y=df2['usr']
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.30,random_state=1)#70:30
```

Import Linear Regression from Sklearn.linear model libraries.

```
from sklearn.linear_model import LinearRegression
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

Fit Model in Linear Regression By scikit Learn method and impute coefficient, Intercept.

```
lm=LinearRegression()
lm.fit(X,y)
```

```
LinearRegression()
```

```
lm.coef_
```

```
array([-7.87353147e-02,  3.67339598e-02, -2.64127999e-01,  1.42375921e-02,
       -7.71163130e-02,  2.52170912e-02, -9.92497489e-02, -6.52860705e-02,
       -4.51431246e-02, -6.87745337e-02, -2.09913348e-02,  4.15537839e-02,
       -1.38777878e-17,  2.45298776e-02, -7.07793432e-02, -5.91514045e-02,
       -2.43651005e-01, -2.76773668e-01,  7.50825338e-02, -6.90250408e-02,
       -3.29997218e-02])
```

```
lm.intercept_
```

```
-3.8487321655071575e-17
```

→ After Scaling Data, we got Intercept of model as -3.84 and Coefficient of each variable as shown in above figure.

→ Freeswap and freemem variables have highest coefficient.

Fit Model in Linear Regression By stats model method and impute coefficient, Intercept.

Fit Model:

Add Constant in X Dataframe:

```
X=sm.add_constant(X)
```

```
X1=sm.add_constant(X)
olsmod = sm.OLS(y_train, X_train)
olsres = olsmod.fit()
```

Summary :

```
print(olsres.summary())
```

```
                            OLS Regression Results
==============================================================================================
Dep. Variable:                    usr   R-squared (uncentered):                   0.867
Model:                            OLS   Adj. R-squared (uncentered):              0.867
Method:                 Least Squares   F-statistic:                              1868.
Date:                Thu, 19 Jan 2023   Prob (F-statistic):                        0.00
Time:                        23:13:34   Log-Likelihood:                         -2414.1
No. Observations:                5734   AIC:                                      4868.
Df Residuals:                    5714   BIC:                                      5001.
Df Model:                          20
Covariance Type:            nonrobust
==============================================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------------
lread         -0.0819      0.011     -7.213      0.000      -0.104      -0.060
lwrite         0.0347      0.010      3.415      0.001       0.015       0.055
scall         -0.2645      0.008    -31.683      0.000      -0.281      -0.248
sread          0.0046      0.012      0.370      0.711      -0.020       0.029
swrite        -0.0777      0.012     -6.688      0.000      -0.101      -0.055
fork           0.0518      0.017      2.961      0.003       0.018       0.086
exec          -0.1093      0.009    -12.517      0.000      -0.126      -0.092
rchar         -0.0624      0.007     -8.802      0.000      -0.076      -0.049
wchar         -0.0442      0.006     -7.208      0.000      -0.056      -0.032
pgout         -0.0555      0.017     -3.362      0.001      -0.088      -0.023
ppgout        -0.0406      0.027     -1.531      0.126      -0.093       0.011
pgfree         0.0426      0.020      2.147      0.032       0.004       0.081
pgscan     -3.429e-17   1.18e-17     -2.898      0.004   -5.75e-17   -1.11e-17
atch           0.0300      0.007      4.473      0.000       0.017       0.043
pgin          -0.0743      0.018     -4.079      0.000      -0.110      -0.039
ppgin         -0.0574      0.018     -3.130      0.002      -0.093      -0.021
pflt          -0.2607      0.017    -15.534      0.000      -0.294      -0.228
vflt          -0.2780      0.019    -14.378      0.000      -0.316      -0.240
freemem        0.0733      0.007     10.783      0.000       0.060       0.087
freeswap      -0.0731      0.007    -10.961      0.000      -0.086      -0.060
runqsz_Not_CPU_Bound    -0.0389      0.005     -7.421      0.000      -0.049      -0.029
==============================================================================
Omnibus:                      883.213   Durbin-Watson:                   1.969
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             7924.713
Skew:                          -0.459   Prob(JB):                         0.00
Kurtosis:                       8.686   Cond. No.                     3.13e+17
==============================================================================

Notes:
[1] R² 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 smallest eigenvalue is 4.47e-31. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

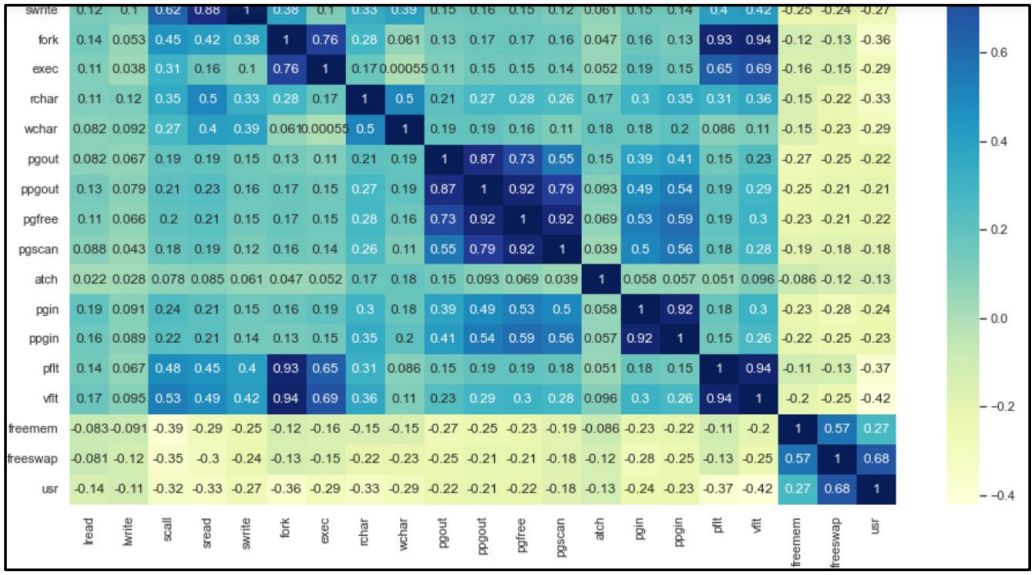
```
1 X.describe()
```

	const	lread	lwrite	scall	sread	swrite	fork	exec	rchar	wchar	...
count	8192.0	8.192000e+03	8.192000e+03	8.192000e+03	8.192000e+03	8.192000e+03	8.192000e+03	8.192000e+03	8.192000e+03	8.192000e+03	...
mean	1.0	-1.635790e-17	1.182594e-16	-4.025101e-18	5.826231e-17	-6.003770e-18	9.230965e-18	-9.240113e-17	-9.988213e-18	-7.922807e-17	...
std	0.0	1.000061e+00	1.000061e+00	1.000061e+00	1.000061e+00	1.000061e+00	1.000061e+00	1.000061e+00	1.000061e+00	1.000061e+00	...
min	1.0	-8.854815e-01	-7.165214e-01	-1.371933e+00	-1.320452e+00	-1.348317e+00	-9.790389e-01	-9.523530e-01	-1.028852e+00	-1.041890e+00	...
25%	1.0	-7.535451e-01	-7.165214e-01	-8.050764e-01	-7.753076e-01	-7.718049e-01	-7.276441e-01	-8.537400e-01	-8.306538e-01	-7.404325e-01	...
50%	1.0	-4.237040e-01	-6.088947e-01	-1.525324e-01	-2.301627e-01	-2.158828e-01	-4.762492e-01	-3.606748e-01	-2.978598e-01	-4.081622e-01	...
75%	1.0	4.338827e-01	3.597453e-01	6.420395e-01	5.398545e-01	4.841673e-01	4.036326e-01	4.282293e-01	4.905739e-01	4.252623e-01	...
max	1.0	2.215024e+00	1.974145e+00	2.812713e+00	2.512598e+00	2.368126e+00	2.100548e+00	2.351183e+00	2.472416e+00	2.173805e+00	...

8 rows × 22 columns

→ From above regression model summary, we can know pgscan is not important variable and we can drop this variable and try another model to get better coefficients and rsquared values.

Heat Map after dropping pgscan:



Seems usr is highly coorelated with freeswap:

Let's build model by taking freeswap.

```
1 y1_train=df2.pop('usr')
2 X1_train=X_train
```

uilding the linear model

```
1 X1_train_lm=sm.add_constant(X1_train[['freeswap']])
```

```
1 olsmod1 = sm.OLS(y_train, X1_train_lm)
2 olsres1 = olsmod1.fit()
```

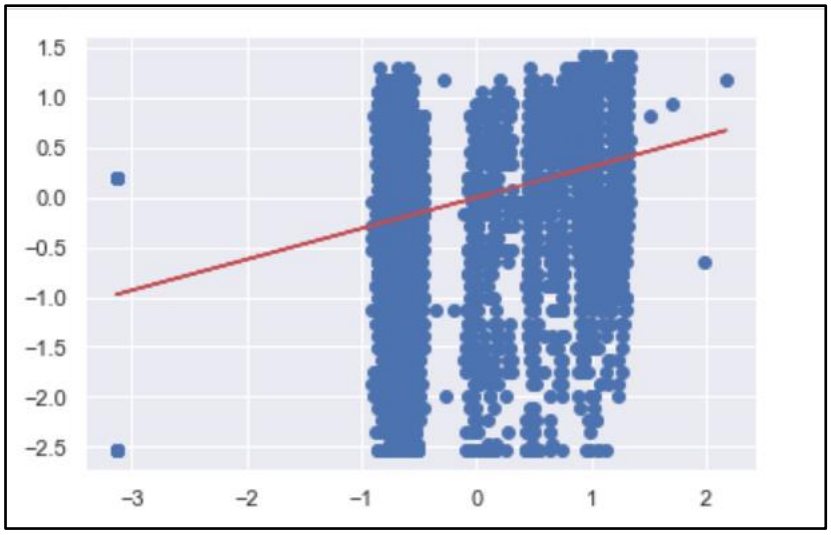
```
1 olsres1.params
```

```
const      -0.003690
freeswap    0.310191
dtype: float64
```

Intercept: -0.003 ,

Freeswap coefficient – 0.31.

<u>Scatter Plot to check Linear regression line for above model:</u>




```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    usr   R-squared:                       0.092
Model:                            OLS   Adj. R-squared:                  0.092
Method:                 Least Squares   F-statistic:                     584.1
Date:                Thu, 19 Jan 2023   Prob (F-statistic):          5.64e-123
Time:                        23:14:03   Log-Likelihood:                -7926.8
No. Observations:                5734   AIC:                         1.586e+04
Df Residuals:                    5732   BIC:                         1.587e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0037      0.013     -0.290      0.772      -0.029       0.021
freeswap       0.3102      0.013     24.169      0.000       0.285       0.335
==============================================================================
Omnibus:                      613.860   Durbin-Watson:                   1.997
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              833.030
Skew:                          -0.933   Prob(JB):                    1.29e-181
Kurtosis:                       3.085   Cond. No.                         1.02
==============================================================================

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

<u>Add second most highly correlated value freeman:</u>

```
1 X2_train_lm=sm.add_constant(X1_train[['freeswap','freemem']])
```

```
1 olsmod2= sm.OLS(y_train,X2_train_lm)
```

```
1 olsres2=olsmod2.fit()
```

```
1 olsres2.params
```

```
const      -0.002078
freeswap    0.145016
freemem     0.268509
dtype: float64
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    usr   R-squared:                       0.137
Model:                            OLS   Adj. R-squared:                  0.137
Method:                 Least Squares   F-statistic:                     456.4
Date:                Thu, 19 Jan 2023   Prob (F-statistic):          1.15e-184
Time:                        23:14:07   Log-Likelihood:                -7781.3
No. Observations:                5734   AIC:                         1.557e+04
Df Residuals:                    5731   BIC:                         1.559e+04
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0021      0.012     -0.167      0.867      -0.026       0.022
freeswap       0.1450      0.016      9.208      0.000       0.114       0.176
freemem        0.2685      0.016     17.275      0.000       0.238       0.299
==============================================================================
Omnibus:                      598.294   Durbin-Watson:                   1.997
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              805.116
Skew:                          -0.916   Prob(JB):                    1.48e-175
Kurtosis:                       3.108   Cond. No.                         2.02
==============================================================================

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

We have clearly improved model as the value of Rsquare went up to 0.13 from 0.09.

Adding all variables in model and Checking Coefficients of all Variables :

```
1 lr1.params
```

```
const                  -0.000702
lread                  -0.081883
lwrite                  0.034733
scall                  -0.264460
sread                   0.004570
swrite                 -0.077756
fork                    0.051802
exec                   -0.109325
rchar                  -0.062412
wchar                  -0.044220
pgout                  -0.055518
ppgout                 -0.040596
pgfree                  0.042601
atch                    0.029995
pgin                   -0.074348
ppgin                  -0.057441
pflt                   -0.260714
vflt                   -0.277928
freemem                 0.073250
freeswap               -0.073074
runqsz_Not_CPU_Bound   -0.038931
dtype: float64
```

Summary of Model:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    usr   R-squared:                       0.867
Model:                            OLS   Adj. R-squared:                  0.867
Method:                 Least Squares   F-statistic:                     1867.
Date:                Thu, 19 Jan 2023   Prob (F-statistic):               0.00
Time:                        23:14:13   Log-Likelihood:                -2414.1
No. Observations:                5734   AIC:                             4870.
Df Residuals:                    5713   BIC:                             5010.
Df Model:                          20
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0007      0.005     -0.144      0.886      -0.010       0.009
lread         -0.0819      0.011     -7.214      0.000      -0.104      -0.060
lwrite         0.0347      0.010      3.416      0.001       0.015       0.055
scall         -0.2645      0.008    -31.679      0.000      -0.281      -0.248
sread          0.0046      0.012      0.370      0.711      -0.020       0.029
swrite        -0.0778      0.012     -6.688      0.000      -0.101      -0.055
fork           0.0518      0.017      2.961      0.003       0.018       0.086
exec          -0.1093      0.009    -12.516      0.000      -0.126      -0.092
rchar         -0.0624      0.007     -8.800      0.000      -0.076      -0.049
wchar         -0.0442      0.006     -7.206      0.000      -0.056      -0.032
pgout         -0.0555      0.017     -3.362      0.001      -0.088      -0.023
ppgout        -0.0406      0.027     -1.532      0.126      -0.093       0.011
pgfree         0.0426      0.020      2.147      0.032       0.004       0.082
atch           0.0300      0.007      4.472      0.000       0.017       0.043
pgin          -0.0743      0.018     -4.079      0.000      -0.110      -0.039
ppgin         -0.0574      0.018     -3.130      0.002      -0.093      -0.021
pflt          -0.2607      0.017    -15.533      0.000      -0.294      -0.228
vflt          -0.2779      0.019    -14.374      0.000      -0.316      -0.240
freemem        0.0733      0.007     10.780      0.000       0.060       0.087
freeswap      -0.0731      0.007    -10.955      0.000      -0.086      -0.060
runqsz_Not_CPU_Bound    -0.0389      0.005     -7.419      0.000      -0.049      -0.029
==============================================================================
Omnibus:                      883.166   Durbin-Watson:                   1.969
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             7924.933
Skew:                          -0.459   Prob(JB):                         0.00
Kurtosis:                       8.686   Cond. No.                         18.7
==============================================================================

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

Intercept of the model is: -0.0007

Model score on the training data set is 86.7%

Model score on the test data is 86.7%

Create a dataframe that will contain the names of all the feature variables and their respective VIFs:

	Features	VIF
10	ppgout	29.40
11	pgfree	16.50
16	vflt	15.97
14	ppgin	13.95
13	pgin	13.81
5	fork	13.04
15	pflt	12.00
9	pgout	11.36
3	sread	6.42
4	swrite	5.59
0	lread	5.35
1	lwrite	4.33
6	exec	3.24
2	scall	2.96
7	rchar	2.13
17	freemem	1.96
12	atch	1.88
18	freeswap	1.84
8	wchar	1.58
19	runqsz_Not_CPU_Bound	1.16

According to this model freeswap, wchar, freemem, atch, scall, exec etc are variables of importance.

You can See sread and ppgout has very high p Value we can drop and check again:

Fit model in Linear regression:

Check Exponential Coefficient of variables:

```
0.9992977357174861
0.9212223070944083
1.0355247413262598
0.7683043984469929
0.9277923343206812
1.0528156014049193
0.8963107181385868
0.9405677365224313
0.9565666174593132
0.9460346484841066
0.9602486644763774
1.0435286449775096
1.0303644320075933
0.9282612641612425
0.9441487570204039
0.7705498628403511
0.7578103708264775
1.076039530505509
0.9293695375970822
0.961812186652716
```

Drop 'ppgout','sread','pgfree' Variable as it's P value and variance Inflation Factors are high, and fit Model for getting best model:

```
X3= X1_train.drop(['ppgout','sread','pgfree'],1,)
X3_train_lm=sm.add_constant(X3)
olsmod4=sm.OLS(y_train,X3_train_lm)
lr3=olsmod4.fit()
```

Summary :

OLS Regression Results

Dep. Variable:	usr	R-squared:	0.867
Model:	OLS	Adj. R-squared:	0.867
Method:	Least Squares	F-statistic:	2196.
Date:	Thu, 19 Jan 2023	Prob (F-statistic):	0.00
Time:	23:14:17	Log-Likelihood:	-2416.5
No. Observations:	5734	AIC:	4869.
Df Residuals:	5716	BIC:	4989.
Df Model:	17		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0007	0.005	-0.140	0.889	-0.010	0.009
lread	-0.0818	0.011	-7.213	0.000	-0.104	-0.060
lwrite	0.0345	0.010	3.398	0.001	0.015	0.054
scall	-0.2641	0.008	-33.060	0.000	-0.280	-0.248
swrite	-0.0748	0.009	-8.499	0.000	-0.092	-0.058
fork	0.0517	0.017	2.957	0.003	0.017	0.086
exec	-0.1095	0.009	-12.553	0.000	-0.127	-0.092
rchar	-0.0613	0.006	-9.590	0.000	-0.074	-0.049
wchar	-0.0449	0.006	-7.351	0.000	-0.057	-0.033
pgout	-0.0564	0.007	-8.028	0.000	-0.070	-0.043
atch	0.0303	0.007	4.515	0.000	0.017	0.043
pgin	-0.0747	0.018	-4.105	0.000	-0.110	-0.039
ppgin	-0.0565	0.018	-3.100	0.002	-0.092	-0.021
pflt	-0.2617	0.017	-15.602	0.000	-0.295	-0.229
freemem	0.0722	0.007	10.655	0.000	0.059	0.085
freeswap	-0.0733	0.007	-11.034	0.000	-0.086	-0.060
runqsz_Not_CPU_Bound	-0.0392	0.005	-7.474	0.000	-0.049	-0.029

Omnibus:	886.389	Durbin-Watson:	1.970
Prob(Omnibus):	0.000	Jarque-Bera (JB):	7901.460
Skew:	-0.464	Prob(JB):	0.00
Kurtosis:	8.675	Cond. No.	13.1

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

Our model is in the right fit zone. We have a good model with us. Our model is neither an underfit nor an overfit model and its working out fine for us.

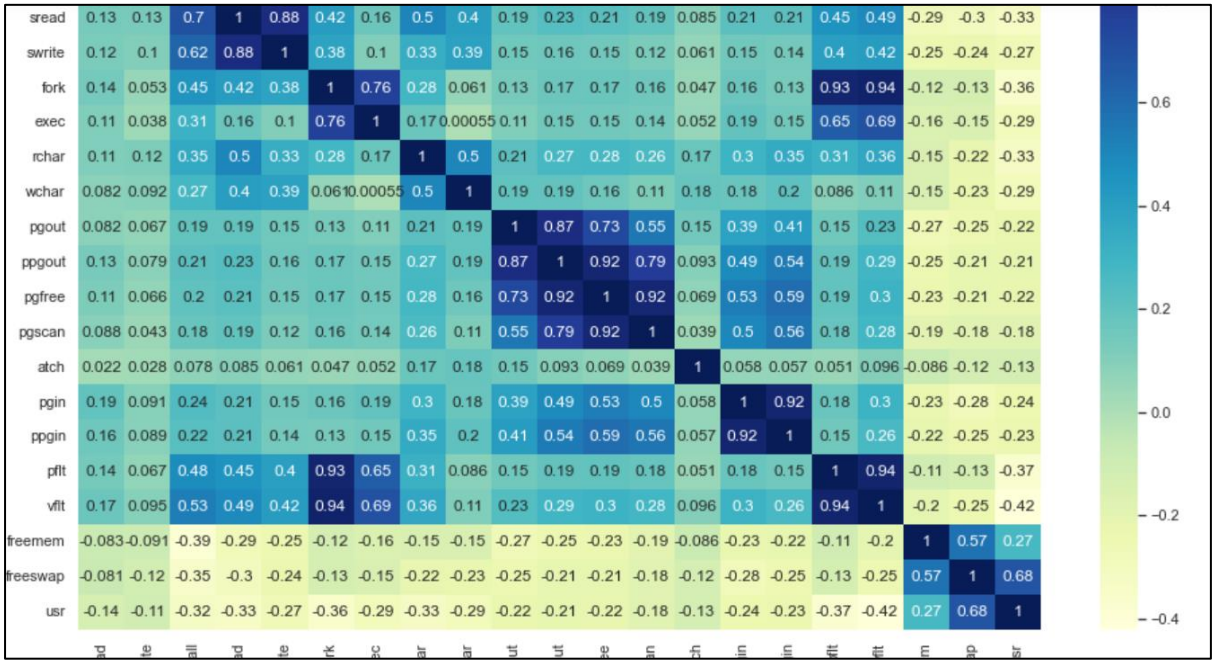
Q1.4 Inference: Basis on these predictions, what are the business insights and recommendations.

We have a database which have strong correlation between independent variables and hence we need to tackle with the issue of multicollinearity which can hinder the results of the model performance. Multicollinearity makes it difficult to understand how one variable influence the target variable. However, it does not affect the accuracy of the model. As a result, while creating the model, I had dropped a lot of independent variables displaying multicollinearity or the ones with no direct relation with the target variable.

While we looked at the data during univariate analysis, we were able to establish that Freeswap and freemem is strongly related with the usr variable, It can be established that freeswap will be a strong predictor in our model creation.

The same trend was displayed even in model creation and Variance Inflation Factor Check. The freeswap and freemem variable continues to display strong to low correlation with most of the variables, making its claim to be the most important predictor firm.

After the Linear Regression model was created, we can see the assumption coming true as the freeswap variable emerged as the single biggest factor impacting the target variable, followed with a few others within freemem variables. freeswap variable has the highest coefficient value as compared to the other studies variables for this test case.



We can then look at the Variance Inflation Factor (VIF) score to check the multicollinearity scores. As per the industry standards or at least for this case study, any variable with a VIF score of greater than 10 has been accepted to indicate severe collinearity. The table below indicates VIF for all the variables.

	features	VIF
13	vflt	15.79
11	ppgin	13.77
10	pgin	13.75
4	fork	12.99
12	pflt	11.98
0	lread	5.34
1	lwrite	4.31
5	exec	3.23
3	swrite	3.21
2	scall	2.71
8	pgout	2.05
14	freemem	1.95
9	atch	1.87
15	freeswap	1.83
6	rchar	1.73
7	wchar	1.57
16	runqsz_Not_CPU_Bound	1.16

VIF measures the intercorrelation among independent variables in a multiple regression model. In mathematical terms, the variance inflation factor for a regression model variable would be the ratio of the overall model variance to the variance of the model with a single independent variable. As an example, the VIF value for Wchar & Freeswap in the table above is the intercorrelation with other independent variables in the dataset and so on for other variables.

If we are looking to finetune the model, we can simply drop these columns from our Linear Regression model to see how the results pan out and can check our model performance.

The Freeswap , wchar, runqsz_Not_CPU_Bound , Freemem are Strong Predictor in change of User.