EAST WEST UNIVERSITY

CSE303: Statistics and Probability
[Spring 2023]

Project Report
Group No – 4

Submitted by:

Student ID	Student Name	Contribution Percentage	Signature
Zin Din Ziden	2020-2-60-132	30%	
Pulak Islam	2020-2-60-193	30%	
Safaet Abdullah	2019-3-60-104	20%	
Shishir Majumder	2020-2-60-131	20%	

1. Introduction

The main purpose of this project is to implement Python modules, data frames, and functions. We were given three datasets on great_customers, heart_disease and mobile_price_train, which we used in our project. There are basically three parts to this project. These are: data preprocessing, explorartory data analysis and hypothesis testing. In the great_customers dateset, there are 15 columns and 13600 rows. In the heart_disease dateset, there are 16 columns and 4239 rows. In the mobile_price_train dateset, there are 21 columns and 2001 rows.

In great_customers dateset it contains age, workclass, salary, education_rank, marital-status, occupation race, sex, mins_beerdrinking_year, mins_exercising_year, works_hours, tea_per_year, coffee_ per_ year. In the heart_disease dateset there are age, education, currentSmoker, cigsPerDay, BPMeds, preval-entStroke, prevalentHyp, diabetes, totChol, sysBP, diaBP, BMI, heartRate,glucos, TenYearCHD. Here TOTCHOL is a continuous variable for total cholesterol (mg/dL). Changes in total cholesterol (Tot-Chol), non-HDL cholesterol, low-density lipoprotein (LDL) and high-density lipoprotein (HDL) cholesterol, and triglyceride (TG) plasmatic levels during tocilizumab treatment. Changes in blood pressure during saline infusion. Systolic (SysBP) and diastolic (DiaBP) blood pressure over the 4 days of the trial. Mean values are shown, together with upper (up) and lower (lo) 95% confidence intervals. The rise in systolic BP was significant ($p = 0.02$) but there was no significant change in diastolic BP. Diastolic Blood Pressure(diaBP), The normal range of diastolic blood pressure should be 60 – 80 mm Hg.The pressure exerted on the walls of the arteries when the heart muscles relax in between two beats. In the mobile_price_train dateset there are battery_power, blue, clock_speed, dual_sim, fc, four_g, int_memory, m_dep, mobile_wt, n_cores, pc, px_height, px_width, ram, sc_h, sc_w, talk_time, three_g, touch_screen, WiFi, price_range.

2. Data Preprocessing

In this section we processed the data by cleaning the duplicate values and replacing the missing values with median or mode of that feature.

```
import pandas as pd
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.csv')

print(df.info())

print('Num of Columns: ', df.shape[1])
print('Num of Rows: ', df.shape[0])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   battery_power  2000 non-null   int64
 1   blue           2000 non-null   int64
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64
 4   fc             2000 non-null   int64
 5   four_g         2000 non-null   int64
 6   int_memory     2000 non-null   int64
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64
 9   n_cores        2000 non-null   int64
 10  pc             2000 non-null   int64
 11  px_height      2000 non-null   int64
 12  px_width       2000 non-null   int64
 13  ram            2000 non-null   int64
 14  sc_h           2000 non-null   int64
 15  sc_w           2000 non-null   int64
```

```
print("Showing amount of Null values")
print(df.isnull().sum())
print(df.isnull().any())

print("Showing amount of duplicate values")
print(df.duplicated().sum())
df.drop_duplicates(inplace=True)
```

```
Showing amount of Null values
battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
touch_screen     0
wifi             0
price_range      0
dtype: int64
battery_power    False
blue             False
clock_speed      False
dual_sim         False
fc               False
four_g           False
int_memory       False
m_dep            False
mobile_wt        False
n_cores          False
pc               False
px_height        False
px_width         False
ram              False
sc_h             False
sc_w             False
talk_time        False
three_g          False
touch_screen     False
wifi             False
price_range      False
dtype: bool
Showing amount of duplicate values
0
```

```
import pandas as pd
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\heart_disease.csv')

print(df.info())

print('Num of Columns: ', df.shape[1])
print('Num of Rows: ', df.shape[0])
```

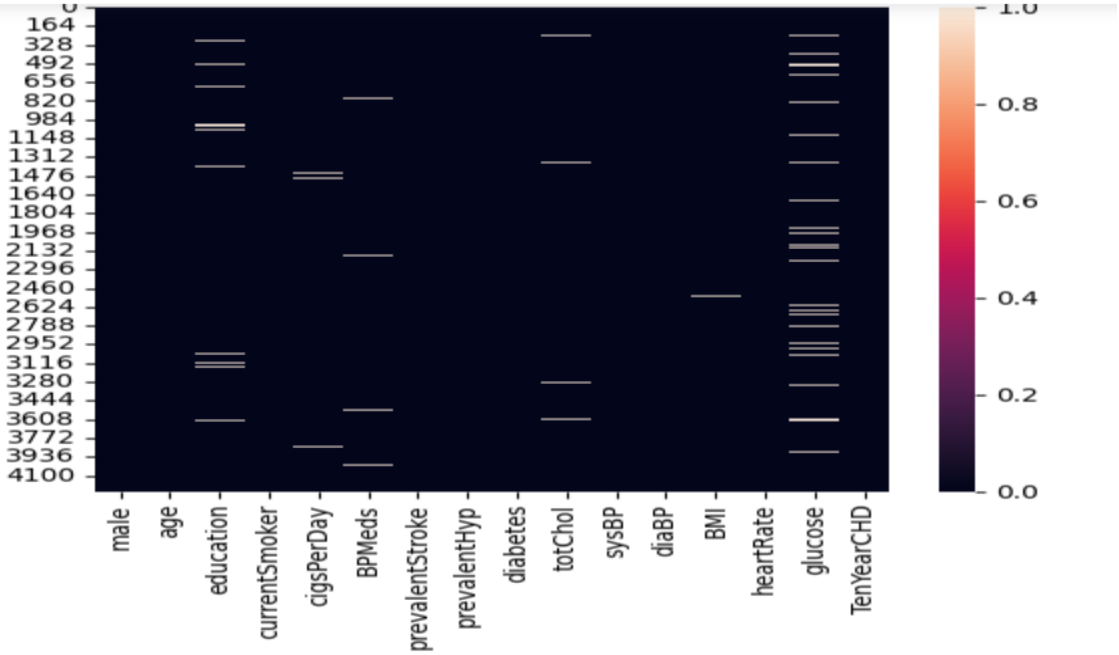
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4238 entries, 0 to 4237
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   male             4238 non-null   int64  
 1   age              4238 non-null   int64  
 2   education        4133 non-null   float64
 3   currentSmoker    4238 non-null   int64  
 4   cigsPerDay       4209 non-null   float64
 5   BPMeds           4185 non-null   float64
 6   prevalentStroke  4238 non-null   int64  
 7   prevalentHyp     4238 non-null   int64  
 8   diabetes         4238 non-null   int64  
 9   totChol          4188 non-null   float64
 10  sysBP            4238 non-null   float64
 11  diaBP            4238 non-null   float64
 12  BMI              4219 non-null   float64
 13  heartRate        4237 non-null   float64
 14  glucose          3850 non-null   float64
 15  TenYearCHD       4238 non-null   int64  
dtypes: float64(9), int64(7)
memory usage: 529.9 KB
None
Num of Columns:  16
Num of Rows:  4238
```

```
import seaborn as sns
import matplotlib.pyplot as plt

df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\heart_disease.csv')

print("Showing amount of Null values")
print(df.isnull().sum())
sns.heatmap(df.isna())
plt.show()

print("Showing amount of duplicate values")
print(df.duplicated().sum())
df.drop_duplicates(inplace=True)
```

```
Showing amount of Null values
male                0
age                 0
education         105
currentSmoker       0
cigsPerDay         29
BPMeds             53
prevalentStroke     0
prevalentHyp        0
diabetes            0
totChol            50
sysBP               0
diaBP               0
BMI                19
heartRate           1
glucose           388
TenYearCHD          0
```

View Insert Cell Kernel Widgets Help

Run Code Voilà




```
Showing amount of duplicate values
0
```

In [8]:
```
dff=pd.read_csv(r'C:\Users\mdzid\Documents\project303\heart_disease.csv')

dff['education'] = dff['education'].fillna(dff['education'].mode()[0])

c = dff["cigsPerDay"].median()
dff["cigsPerDay"].fillna(c, inplace = True)

b = dff["BPMeds"].fillna(method='ffill')
dff["BPMeds"].fillna(b, inplace = True)

t = dff["totChol"].median()
dff["totChol"].fillna(t, inplace = True)

bm = dff["BMI"].mean()
dff["BMI"].fillna(bm, inplace = True)

h = dff["heartRate"].median()
dff["heartRate"].fillna(h, inplace = True)

g = dff["glucose"].median()
dff["glucose"].fillna(g, inplace = True)

print(dff.isnull().sum())
```

```
dff["glucose"].fillna(g, inplace = True)

print(dff.isnull().sum())
```

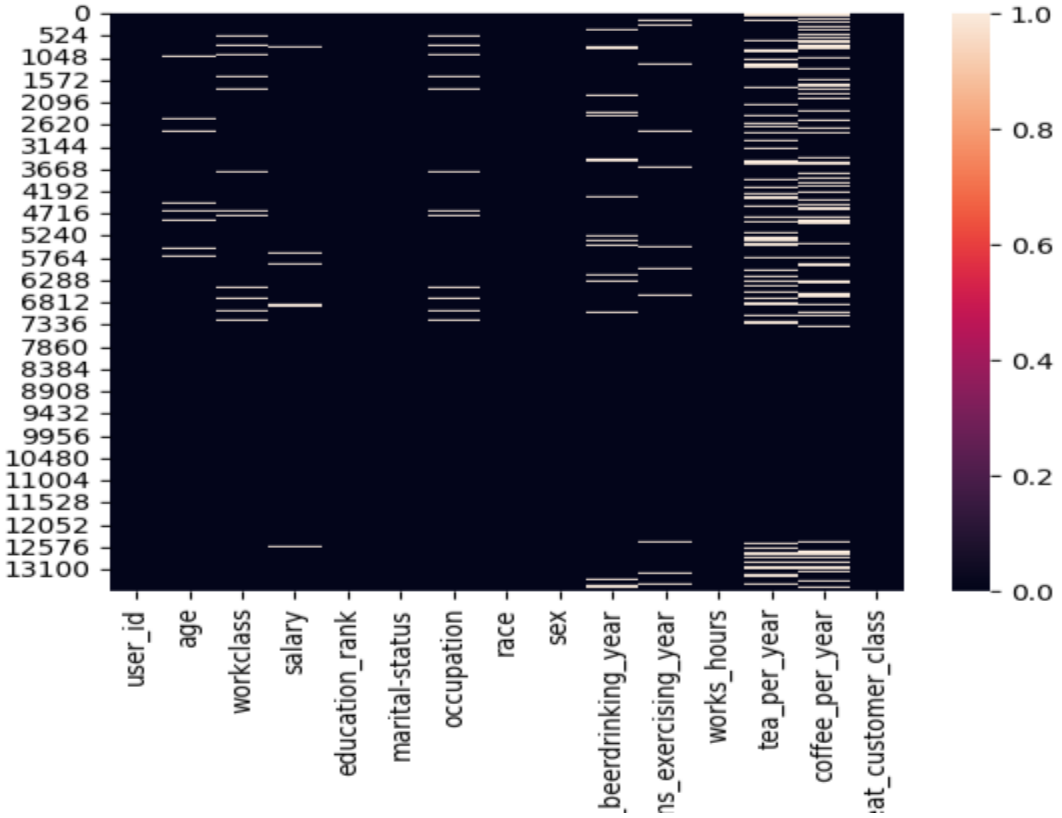
```
male               0
age                0
education          0
currentSmoker      0
cigsPerDay         0
BPMeds             0
prevalentStroke    0
prevalentHyp       0
diabetes           0
totChol            0
sysBP              0
diaBP              0
BMI                0
heartRate          0
glucose            0
TenYearCHD         0
dtype: int64
```

```
import pandas as pd
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\great_customers.csv')

print(df.info())

print('Num of Columns: ', df.shape[1])
print('Num of Rows: ', df.shape[0])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13599 entries, 0 to 13598
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   user_id                 13599 non-null  int64
 1   age                     13178 non-null  float64
 2   workclass               13056 non-null  object
 3   salary                  13177 non-null  float64
 4   education_rank          13599 non-null  int64
 5   marital-status          13599 non-null  object
 6   occupation              13056 non-null  object
 7   race                    13599 non-null  object
 8   sex                     13599 non-null  object
 9   mins_beerdrinking_year  13175 non-null  float64
 10  mins_exercising_year    13178 non-null  float64
 11  works_hours             13599 non-null  int64
 12  tea_per_year            11170 non-null  float64
 13  coffee_per_year         11188 non-null  float64
 14  great_customer_class    13599 non-null  int64
dtypes: float64(6), int64(4), object(5)
memory usage: 1.6+ MB
None
Num of Columns:  15
Num of Rows:  13599
```

```
]: import pandas as pd
   import seaborn as sns
   import matplotlib.pyplot as plt

   df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\great_customers.csv')

   print("Showing amount of Null values")
   print(df.isnull().sum())
   sns.heatmap(df.isna())
   plt.show()
```

```
Showing amount of Null values
user_id                     0
age                       421
workclass                 543
salary                    422
education_rank              0
marital-status              0
occupation                543
race                        0
sex                         0
mins_beerdrinking_year    424
mins_exercising_year      421
works_hours                 0
tea_per_year             2429
coffee_per_year          2411
great_customer_class        0
dtype: int64
```



mins_ mi gre

```
import pandas as pd
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\great_customers.csv')

print("Showing amount of duplicate values")
print(df.duplicated().sum())
df.drop_duplicates(inplace=True)
print("new row size")
print(df.shape[0])
```

```
Showing amount of duplicate values
5000
new row size
8599
```

```
import pandas as pd
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\great_customers.csv')

g = df["age"].median()
df["age"].fillna(g, inplace = True)

df['workclass'] = df['workclass'].fillna(df['workclass'].mode()[0])

s = df["salary"].mean()
df["salary"].fillna(s, inplace = True)

df['occupation'] = df['occupation'].fillna(df['occupation'].mode()[0])

df["mins_beerdrinking_year"].fillna(0, inplace = True)
df["mins_exercising_year"].fillna(0,inplace=True)

t = df["tea_per_year"].median()
df["tea_per_year"].fillna(t, inplace = True)

c = df["coffee_per_year"].median()
df["coffee_per_year"].fillna(c, inplace = True)
print("amount of null rows")
print(df.isnull().sum())
```

```
print("amount of null rows")
print(df.isnull().sum())
```

```
amount of null rows
user_id                 0
age                     0
workclass               0
salary                  0
education_rank          0
marital-status          0
occupation              0
race                    0
sex                     0
mins_beerdrinking_year  0
mins_exercising_year    0
works_hours             0
tea_per_year            0
coffee_per_year         0
great_customer_class    0
```

Now we divide the categorical value of this data set to count the frequency
And analyze the data more easily to draw charts or making hypothesis test

```
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\great_customers.csv')
catg = df.select_dtypes(object)
numc = df.select_dtypes(["int64","float64"])
print(catg.info())
print(numc.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13599 entries, 0 to 13598
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   workclass       13056 non-null  object
 1   marital-status  13599 non-null  object
 2   occupation      13056 non-null  object
 3   race            13599 non-null  object
 4   sex             13599 non-null  object
dtypes: object(5)
memory usage: 531.3+ KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13599 entries, 0 to 13598
Data columns (total 10 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   user_id                 13599 non-null  int64
 1   age                     13178 non-null  float64
 2   salary                  13177 non-null  float64
 3   education_rank          13599 non-null  int64
 4   mins_beerdrinking_year  13175 non-null  float64
 5   mins_exercising_year    13178 non-null  float64
 6   works_hours             13599 non-null  int64
 7   tea_per_year            11170 non-null  float64
 8   coffee_per_year         11188 non-null  float64
 9   great_customer_class    13599 non-null  int64
```

```
import pandas as pd
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\great_customers.csv')
catg = df.select_dtypes(object)
numc = df.select_dtypes(["int64","float64"])
print(catg.info())
print(numc.info())

catg["workclass"].replace("government",0,inplace=True)
catg["workclass"].replace("private",1,inplace=True)
catg["workclass"].replace("self_employed",2,inplace=True)

catg["marital-status"].replace("Divorced",0,inplace=True)
catg["marital-status"].replace("Married",1,inplace=True)
catg["marital-status"].replace("Never-married",2,inplace=True)
catg["marital-status"].replace("Widowed",3,inplace=True)

catg["race"].replace("caucasian",0,inplace=True)
catg["race"].replace("not_caucasian",1,inplace=True)

catg["sex"].replace("Male",0,inplace=True)
catg["sex"].replace("Female",1,inplace=True)

from sklearn.preprocessing import LabelEncoder
ob = LabelEncoder()
ob.fit_transform(catg["occupation"])

for col in catg:
    ob=LabelEncoder()
    catg[col]=ob.fit_transform(catg[col])
print(catg.info())
```

```
catg.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13599 entries, 0 to 13598
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   workclass       13599 non-null  int64
 1   marital-status  13599 non-null  int64
 2   occupation      13599 non-null  int32
 3   race            13599 non-null  int64
 4   sex             13599 non-null  int64
dtypes: int32(1), int64(4)
memory usage: 478.2 KB
```

```
print(catg.head())
```

```
   workclass  marital-status  occupation  race  sex
0          1               2           9     1    0
1          1               0           9     0    1
2          1               2           1     0    1
3          1               0           9     0    1
4          1               1           9     1    0
```

Exploratory Data Analysis:

In EDA at first we identified the outliers by finding the median, quartile and fence and we used python codes to draw the boxplots after cleaning the outliers to standardize the data of given dataset mobile-price-train.csv features for testing numerical datas

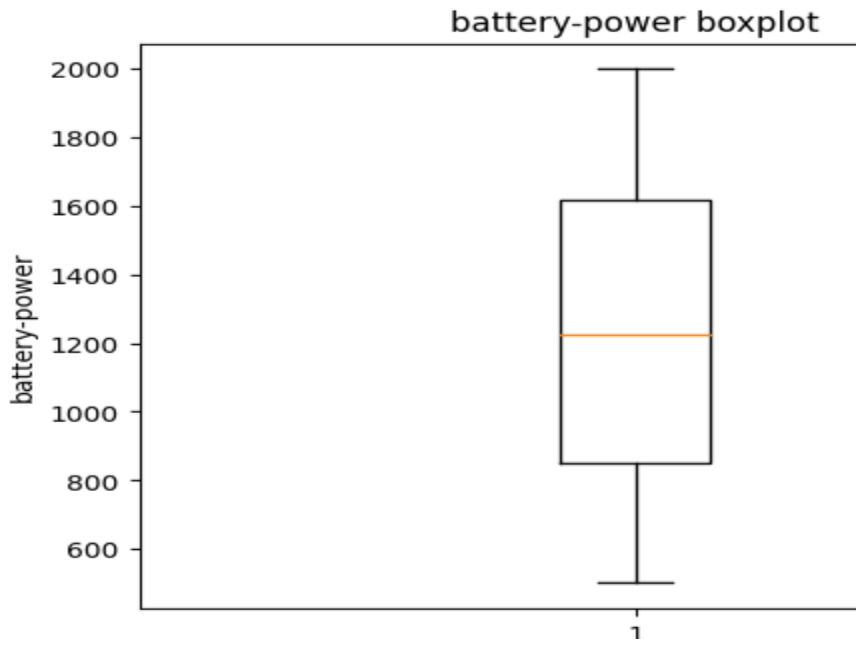
```
: import pandas as pd
  df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.csv')

  print(df.describe())
```

```
       battery_power         blue   clock_speed      dual_sim           fc  \
count    2000.000000  2000.000000  2000.000000  2000.000000  2000.000000
mean     1238.518500     0.495000     1.522250     0.509500     4.309500
std       439.418206     0.500116     0.816004     0.500035     4.341444
min       501.000000     0.000000     0.500000     0.000000     0.000000
25%       851.750000     0.000000     0.700000     0.000000     1.000000
50%      1226.000000     0.000000     1.500000     1.000000     3.000000
75%      1615.250000     1.000000     2.200000     1.000000     7.000000
max      1998.000000     1.000000     3.000000     1.000000    19.000000

            four_g   int_memory        m_dep     mobile_wt      n_cores  ...  \
count  2000.000000  2000.000000  2000.000000  2000.000000  2000.000000  ...
mean      0.521500    32.046500     0.501750   140.249000     4.520500  ...
std       0.499662    18.145715     0.288416    35.399655     2.287837  ...
min       0.000000     2.000000     0.100000    80.000000     1.000000  ...
25%       0.000000    16.000000     0.200000   109.000000     3.000000  ...
50%       1.000000    32.000000     0.500000   141.000000     4.000000  ...
75%       1.000000    48.000000     0.800000   170.000000     7.000000  ...
max       1.000000    64.000000     1.000000   200.000000     8.000000  ...


         px_height     px_width          ram         sc_h         sc_w
count  2000.000000  2000.000000  2000.000000  2000.000000  2000.000000
mean    645.108000  1251.515500  2124.213000    12.306500     5.767000
std     443.780811   432.199447  1084.732044     4.213245     4.356398
min       0.000000   500.000000   256.000000     5.000000     0.000000
25%     282.750000   874.750000  1207.500000     9.000000     2.000000
50%     564.000000  1247.000000  2146.500000    12.000000     5.000000
75%     947.250000  1633.000000  3064.500000    16.000000     9.000000
max    1960.000000  1998.000000  3998.000000    19.000000    18.000000

         talk_time      three_g  touch_screen         wifi  price_range
count  2000.000000  2000.000000   2000.000000  2000.000000  2000.000000
mean     11.011000     0.761500      0.503000     0.507000     1.500000
std       5.463955     0.426273      0.500116     0.500076     1.118314
min       2.000000     0.000000      0.000000     0.000000     0.000000
25%       6.000000     1.000000      0.000000     0.000000     0.750000
50%      11.000000     1.000000      1.000000     1.000000     1.500000
75%      16.000000     1.000000      1.000000     1.000000     2.250000
max      20.000000     1.000000      1.000000     1.000000     3.000000
```

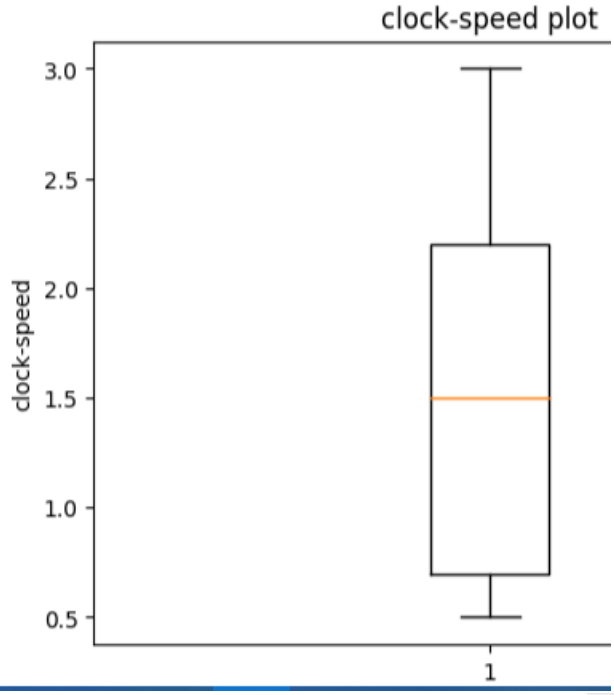
```
A = df['battery_power']
print('For bp: ')
print('Q1: ', A.quantile(0.25))
print('Q3: ', A.quantile(0.75))
print('Median: ', A.quantile(0.5))

IQR_A =  (A.quantile(0.75) - A.quantile(0.25))
print('IQR for Amount: ', IQR_A)
print('Lower fench: ', A.quantile(0.25)-(IQR_A * 1.5))
print('Upper fench: ', A.quantile(0.75)+(IQR_A * 1.5))
a_max = A.max()
a_min = A.min()
print('Minimum: ', a_min)
print('Maximum: ', a_max)
if (a_min < (A.quantile(0.25)-(IQR_A*1.5))):
    print("There are Outliers")
elif(a_max > (A.quantile(0.75)+(IQR_A * 1.5))):
    print('There are right outliers ')
else:
    print('There are no outliers in bp. \n')
```

```
For bp:
Q1:  851.75
Q3:  1615.25
Median:  1226.0
IQR for Amount:  763.5
Lower fench:  -293.5
Upper fench:  2760.5
Minimum:  501
Maximum:  1998
There are no outliers in bp.
```

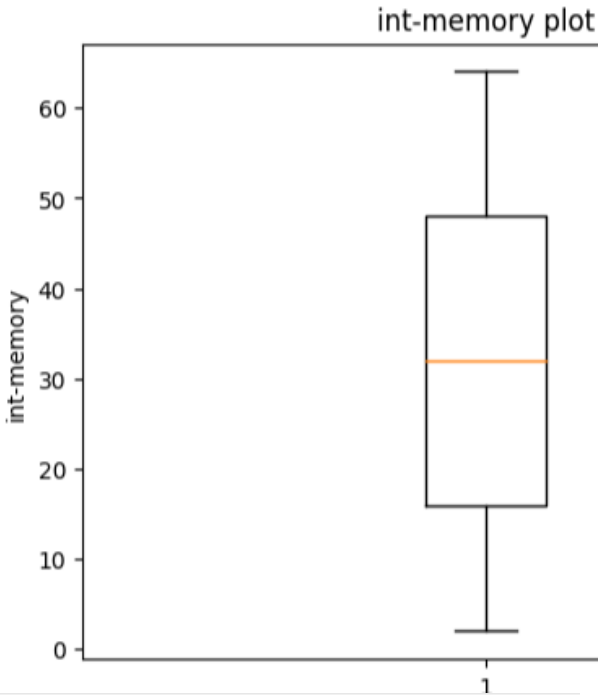
```
import matplotlib.pyplot as plt

plt.boxplot(x="battery_power", data=df)
plt.title("battery-power boxplot")
plt.ylabel("battery-power")
plt.show()
```



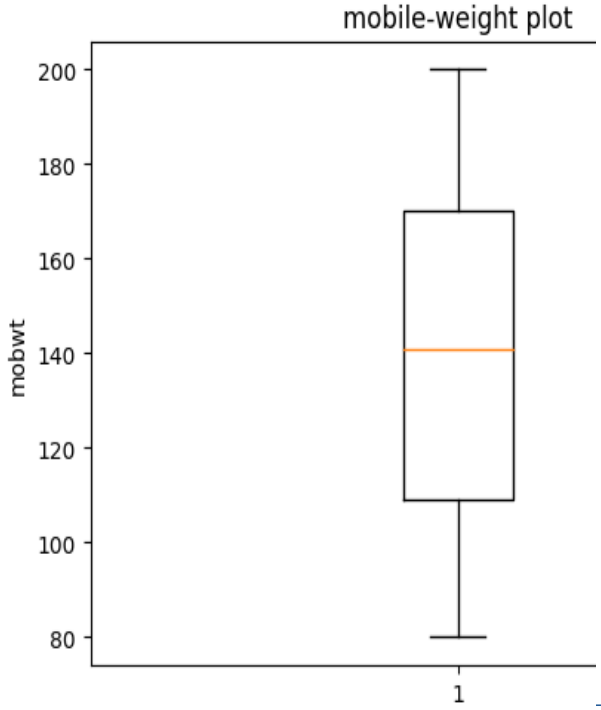
```
import matplotlib.pyplot as plt

plt.boxplot(x="clock_speed", data=df)
plt.title("clock-speed plot")
plt.ylabel("clock-speed")
plt.show()
```




```
import matplotlib.pyplot as plt

plt.boxplot(x="int_memory", data=df)
plt.title("int-memory plot")
plt.ylabel("int-memory")
plt.show()
```

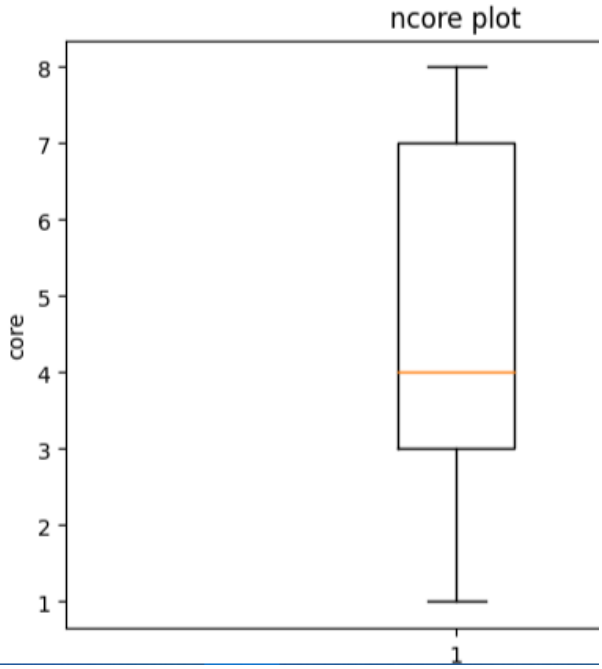



```
import matplotlib.pyplot as plt

plt.boxplot(x="mobile_wt", data=df)
plt.title("mobile-weight plot")
plt.ylabel("mobwt")
plt.show()
```



Run Code

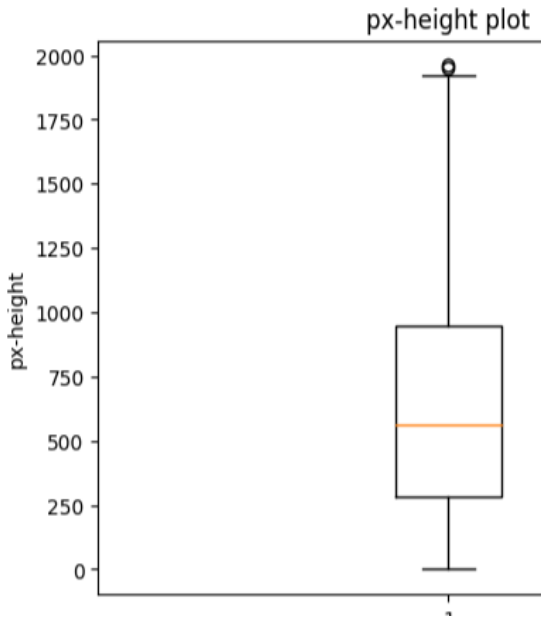
```
import matplotlib.pyplot as plt

plt.boxplot(x="n_cores", data=df)
plt.title("ncore plot")
plt.ylabel("core")
plt.show()
```



```
P = df['px_height']
print('For height: ')
print('Q1: ', P.quantile(0.25))
print('Q3: ', P.quantile(0.75))
print('Median: ', P.quantile(0.5))

IQR_P =  (P.quantile(0.75) - P.quantile(0.25))
print('IQR for mem: ', IQR_P)
print('Lower fench: ', P.quantile(0.25)-(IQR_P * 1.5))
print('Upper fench: ', P.quantile(0.75)+(IQR_P * 1.5))
p_max = P.max()
p_min = P.min()
print('Minimum: ', p_min)
print('Maximum: ', p_max)
if (p_min < (P.quantile(0.25)-(IQR_P*1.5))):
    print("There are Outliers")
elif(p_max > (P.quantile(0.75)+(IQR_P * 1.5))):
    print('There are right outliers ')
else:
    print('There are no outliers in core. \n')
```

```
For height:
Q1:  282.75
Q3:  947.25
Median:  564.0
IQR for mem:  664.5
Lower fench:  -714.0
Upper fench:  1944.0
Minimum:  0
Maximum:  1960
There are right outliers
```

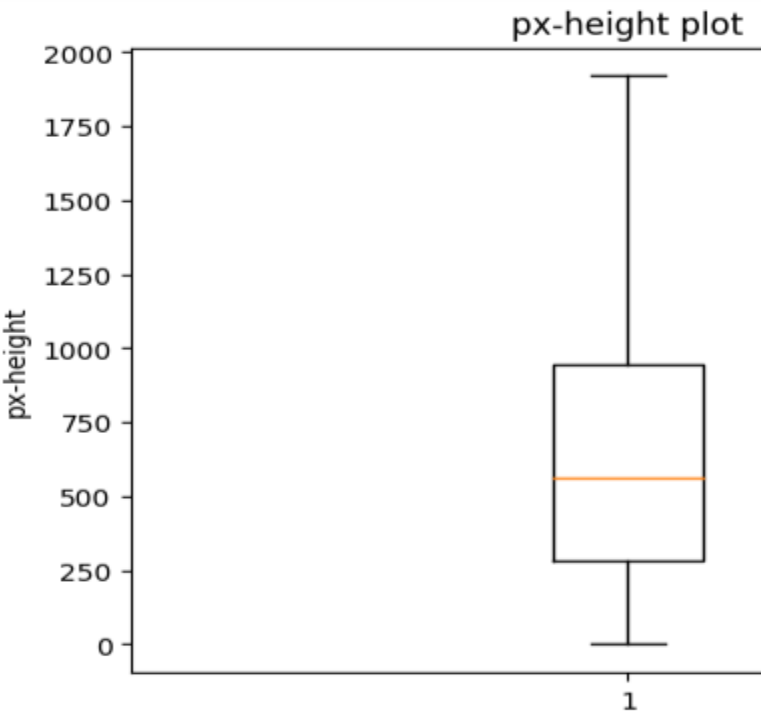
```
import matplotlib.pyplot as plt

plt.boxplot(x="px_height", data=df)
plt.title("px-height plot")
plt.ylabel("px-height")
plt.show()
```



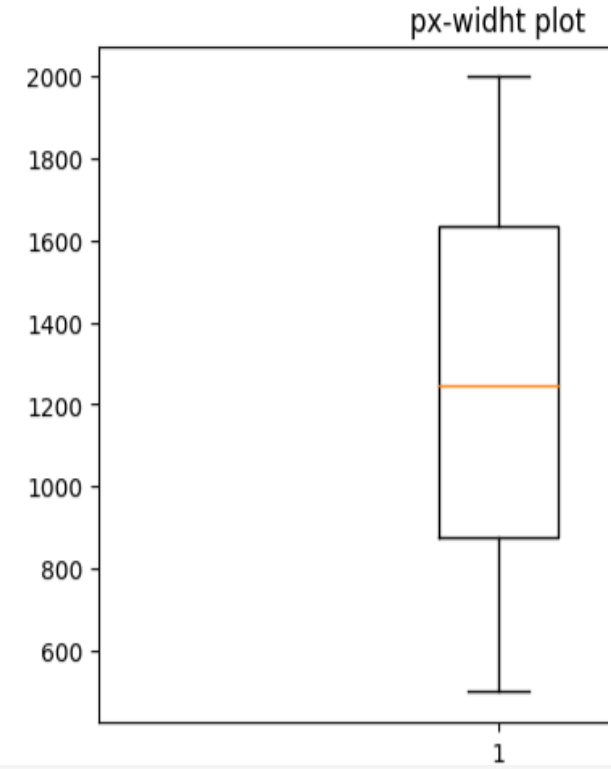

```
for x in df.index:
  if df.loc[x, "px_height"] > 1944:
    df.loc[x, "px_height"] = 564
```

```
for x in df.index:
  if df.loc[x, "px_height"]<-714:
    df.loc[x, "px_height"] = 564
```

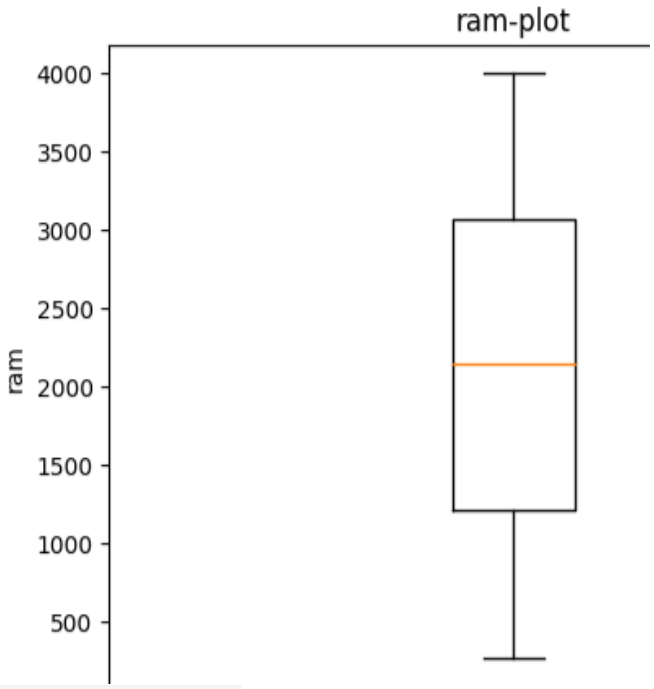
```
import matplotlib.pyplot as plt

plt.boxplot(x="px_height", data=df)
plt.title("px-height plot")
plt.ylabel("px-height")
plt.show()
```



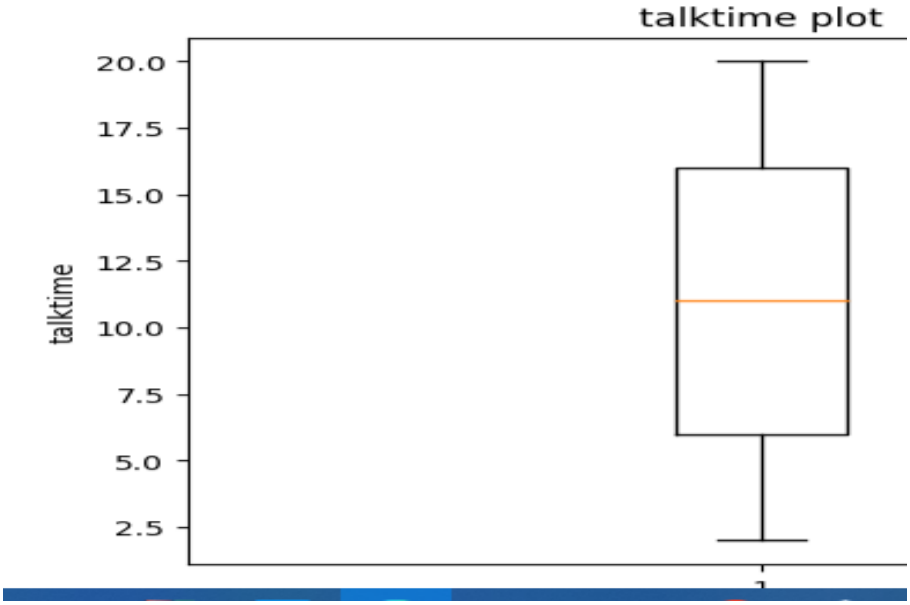
```
plt.boxplot(x='px_width',data=df)
plt.title('px-widht plot')
plt.show()
```




```
import matplotlib.pyplot as plt

plt.boxplot(x="ram", data=df)
plt.title("ram-plot")
plt.ylabel("ram")
plt.show()
```

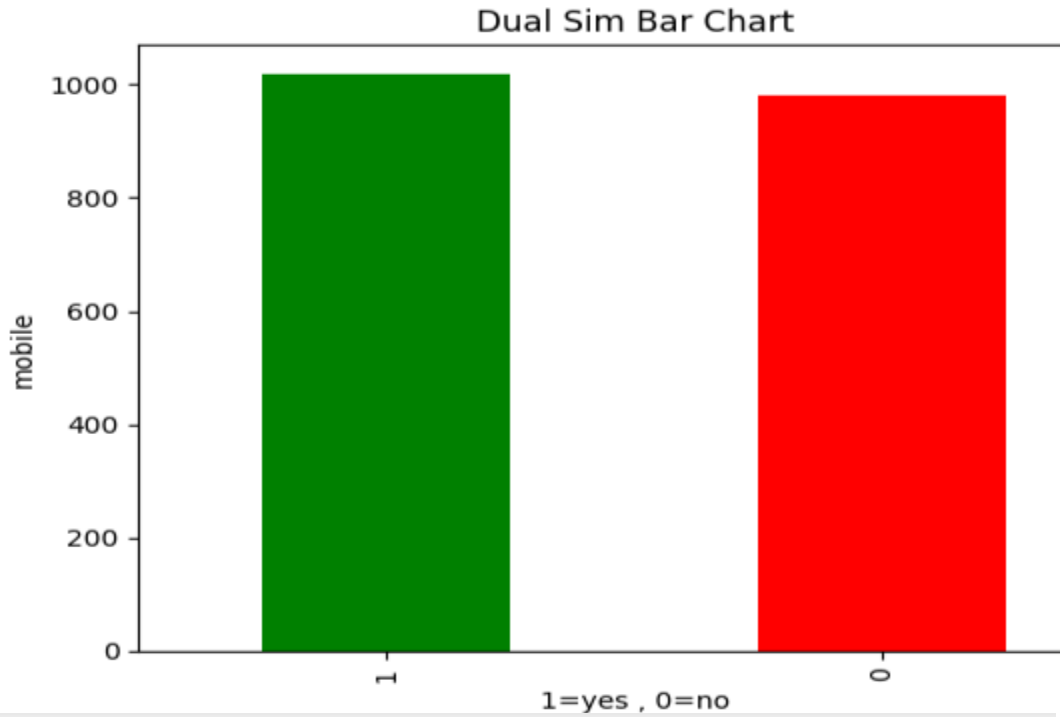



```
import matplotlib.pyplot as plt

plt.boxplot(x="talk_time", data=df)
plt.title("talktime plot")
plt.ylabel("talktime")
plt.show()
```

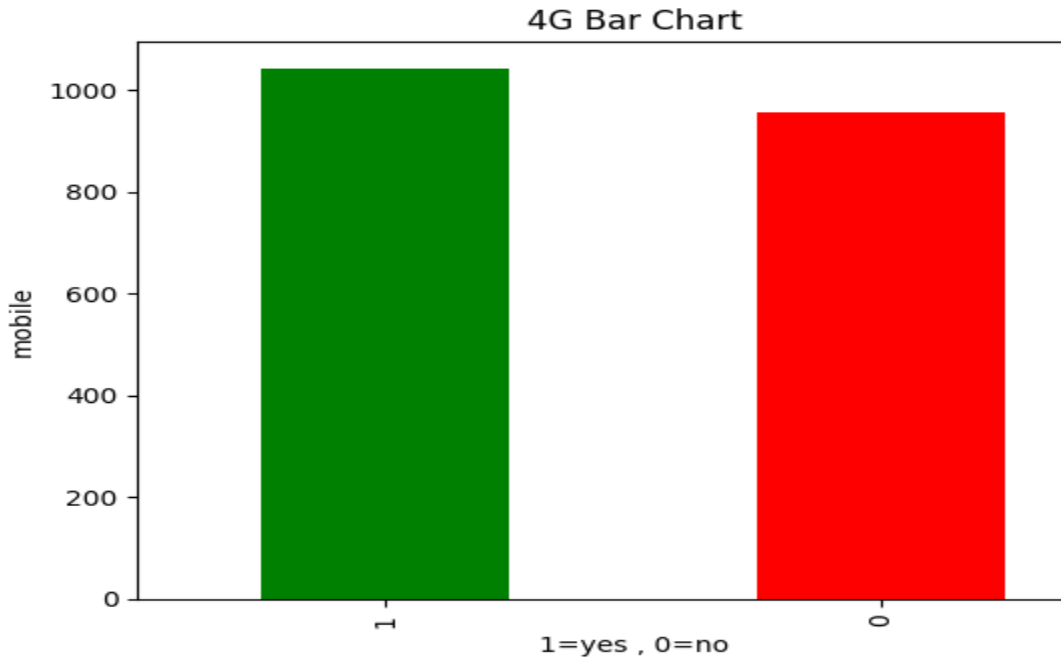


Now we draw the bar chart or pie chart for the categorical type data because of having Boolean values so we calculate the portion of mobile-price-train features by Bar chart

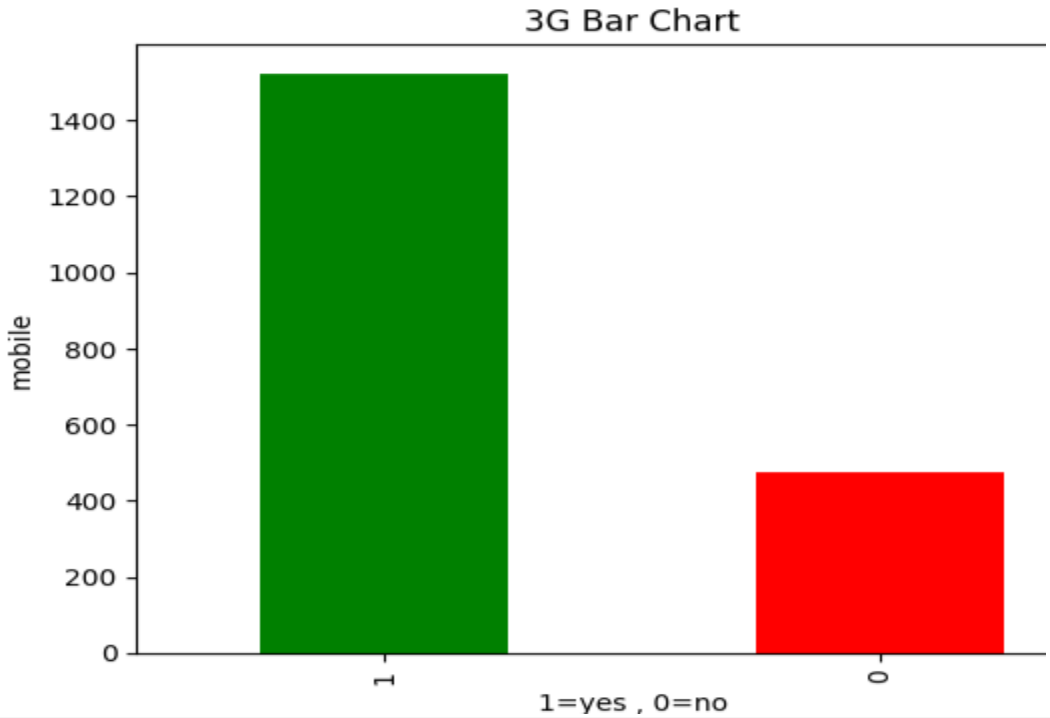
```
df["dual_sim"].value_counts().plot(kind='bar', color=['green','red'])
plt.title("Dual Sim Bar Chart")
plt.xlabel("1=yes , 0=no")
plt.ylabel("mobile")
plt.show()
```



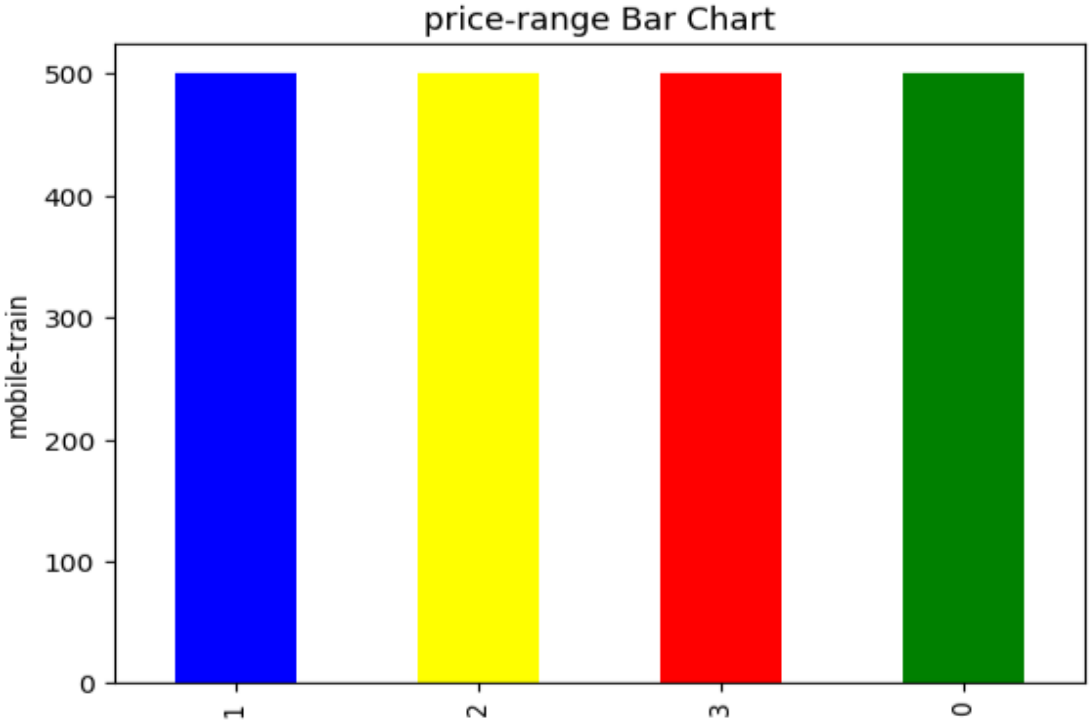

```
df["four_g"].value_counts().plot(kind='bar', color=['green','red'])
plt.title("4G Bar Chart")
plt.xlabel("1=yes , 0=no")
plt.ylabel("mobile")
plt.show()
```



```
df["three_g"].value_counts().plot(kind='bar', color=['green','red'])
plt.title("3G Bar Chart")
plt.xlabel("1=yes , 0=no")
plt.ylabel("mobile")
plt.show()
```



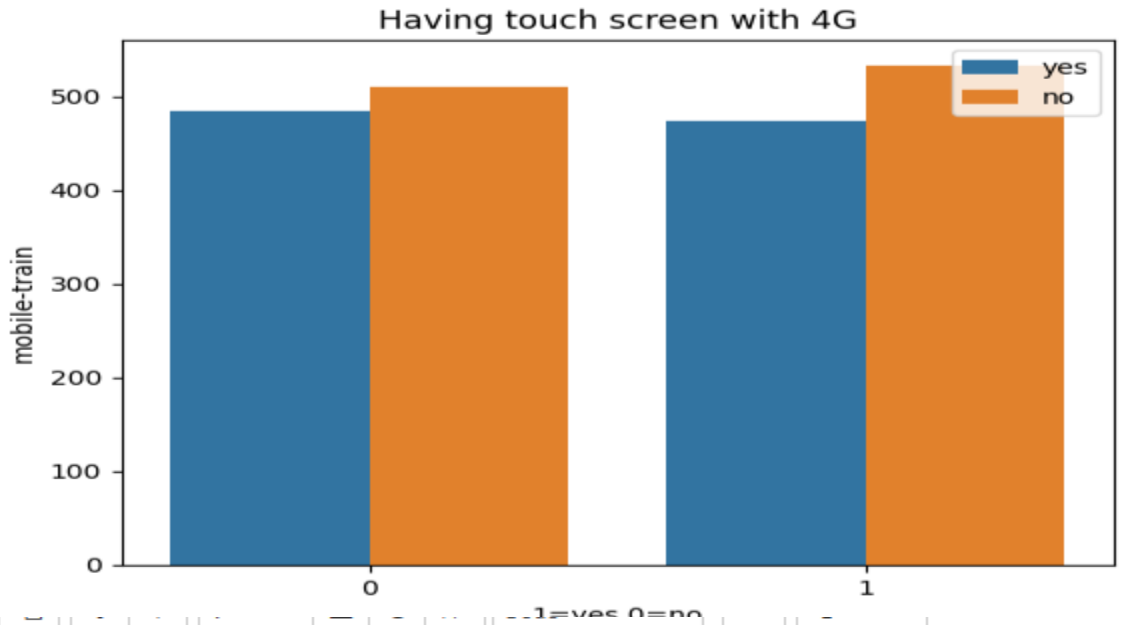

```
df["price_range"].value_counts().plot(kind='bar', color=['blue','yellow','red','green']
plt.title("price-range Bar Chart")
plt.ylabel("mobile-train")
plt.show()
```



Here, blue means 4G and orange means not 4G having touch screen at 1 and no touch screen at 0

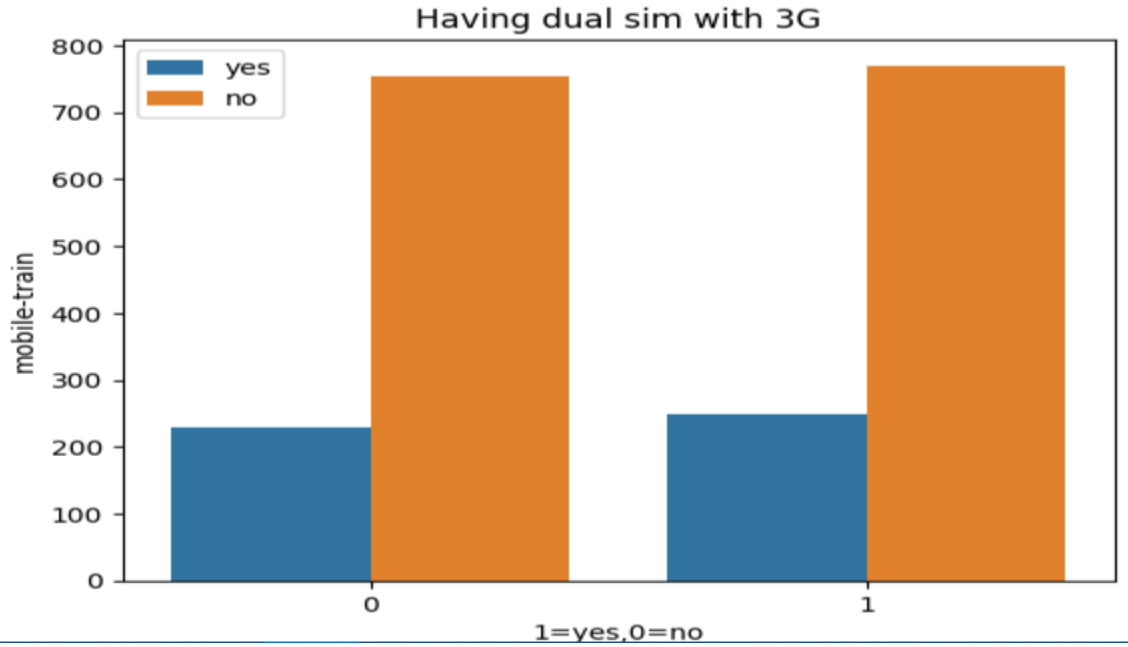
```
import seaborn as sns
sns.countplot(x='touch_screen', data=df, hue='four_g')
plt.title("Having touch screen with 4G")
plt.xlabel("1=yes,0=no")
plt.ylabel("mobile-train")
plt.legend(['yes','no'])
```

```
: <matplotlib.legend.Legend at 0x20eb745e588>
```



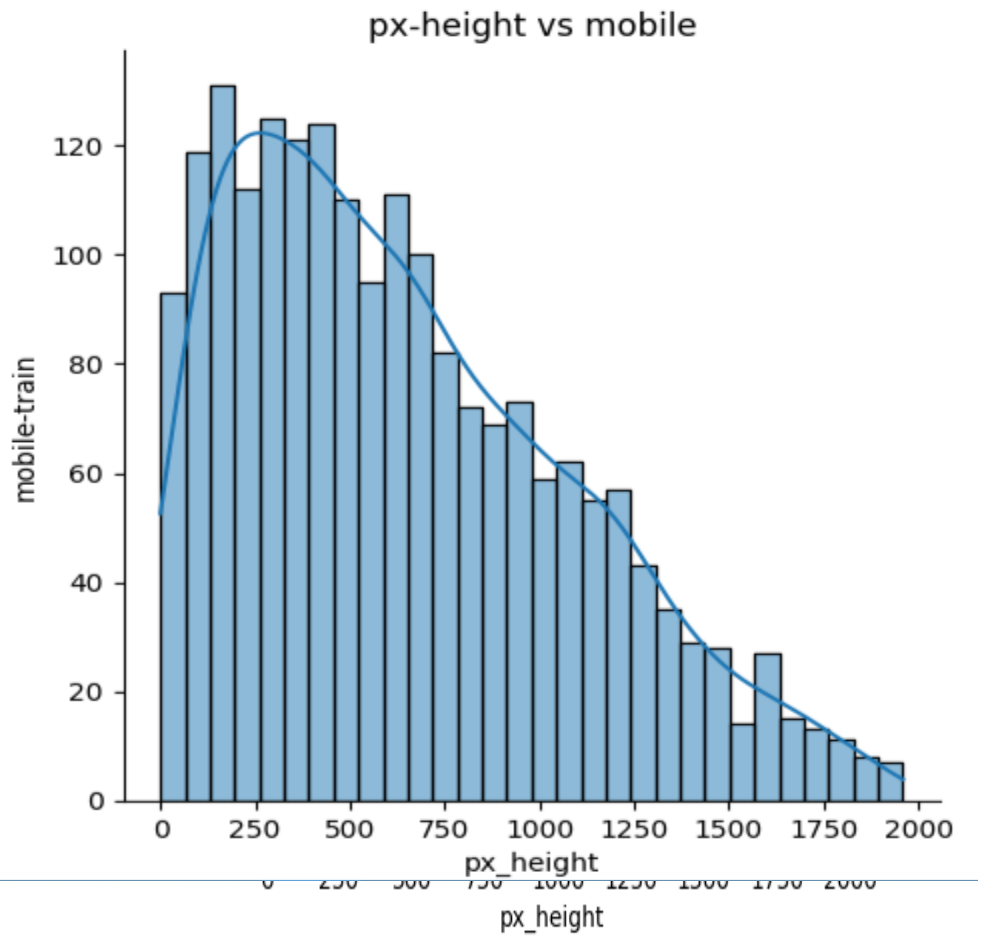

```
import seaborn as sns
sns.countplot(x='dual_sim', data=df, hue='three_g')
plt.title("Having dual sim with 3G")
plt.xlabel("1=yes,0=no")
plt.ylabel("mobile-train")
plt.legend(['yes','no'])
```

```
: <matplotlib.legend.Legend at 0x20eca1362c8>
```



Now we draw a continuous distribution curve and it is rightly skewed because we had outliers in this feature so normal distribution is not found here.

```
sns.displot(x='px_height',data=df,bins=30,kde=True)
plt.title('px-height vs mobile')
plt.ylabel("mobile-train")
plt.show()
```

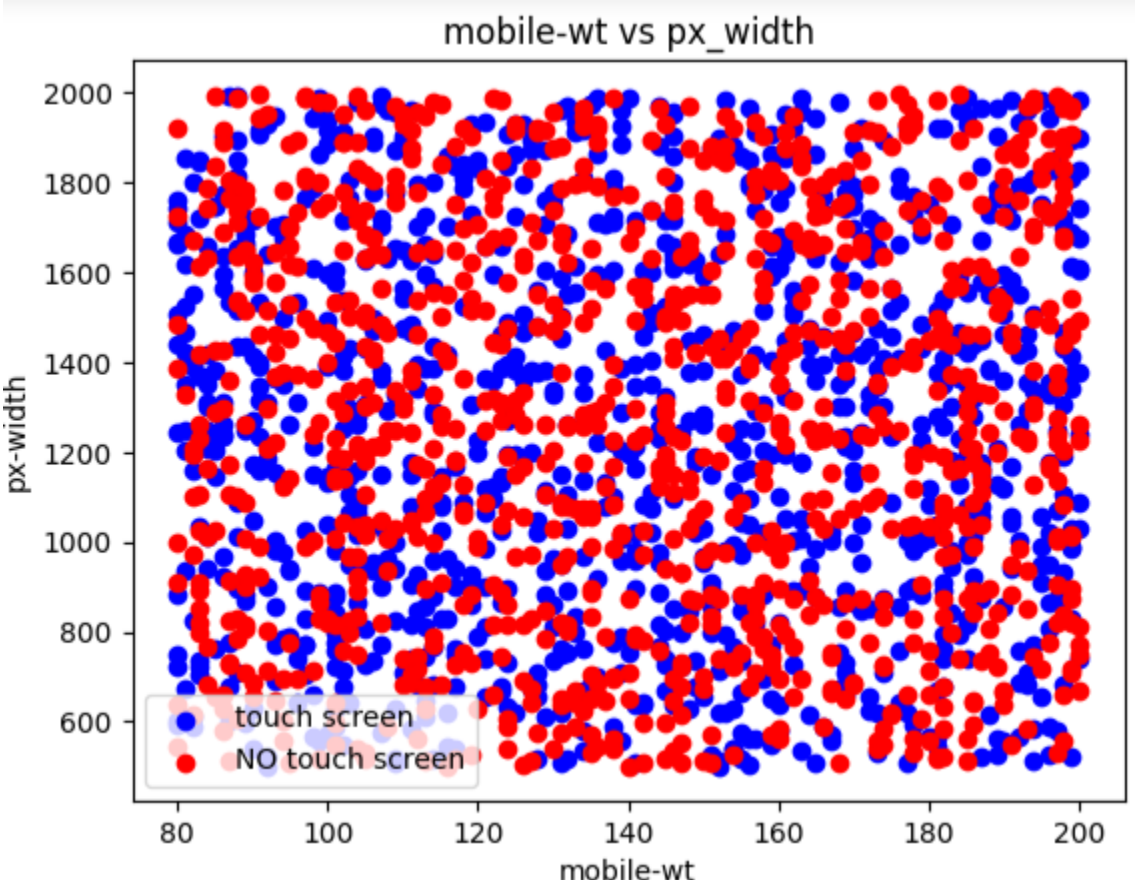


px_height

In [36]:
```
import matplotlib.pyplot as plt

plt.scatter(df.mobile_wt[df.touch_screen==1],
            df.px_width[df.touch_screen==1] , c='blue')
#no-touchscreen
plt.scatter(df.mobile_wt[df.touch_screen==0],
            df.px_width[df.touch_screen==0] , c='red')
plt.title('mobile-wt vs px_width')
plt.xlabel("mobile-wt")
plt.ylabel("px-width")
plt.legend(['touch screen','NO touch screen'])
```

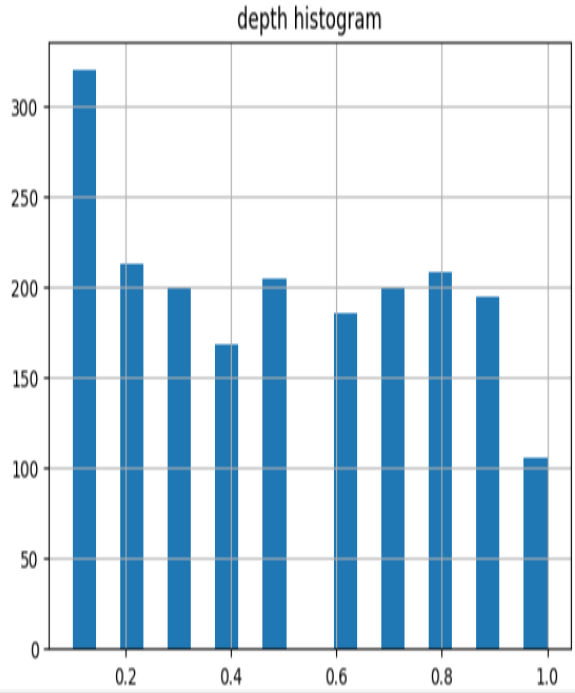
Out[36]: <matplotlib.legend.Legend at 0x20ecb91f948>

Here, we tried to find the correlation between mobile height and px width diameter having touch screen or having no touch screen but we could not find any relation from the below scatter plot.

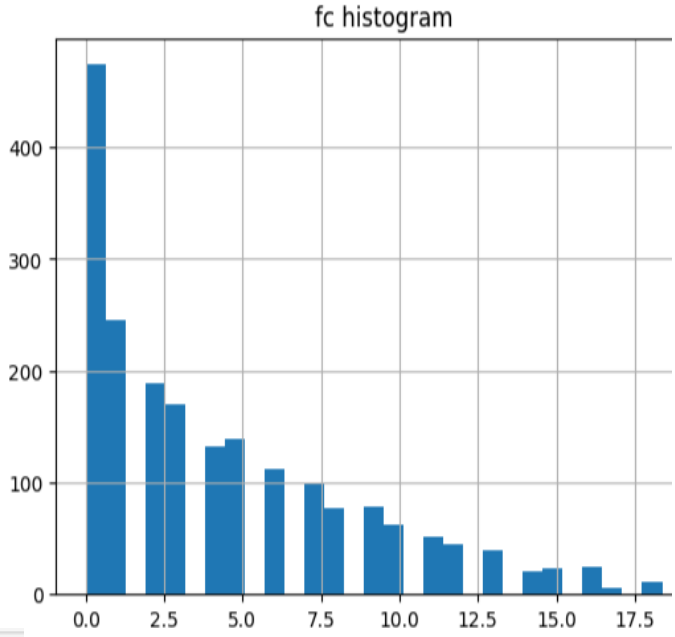


Now, we draw the histograms of some feature and we find: some plots are rightly skewed which says that the mean is greater than median so it is positively skewed and the more forward we get the lower size the data becomes which can cause outliers in some cases, Let’s see histogram of fc, m-dep and sc-w

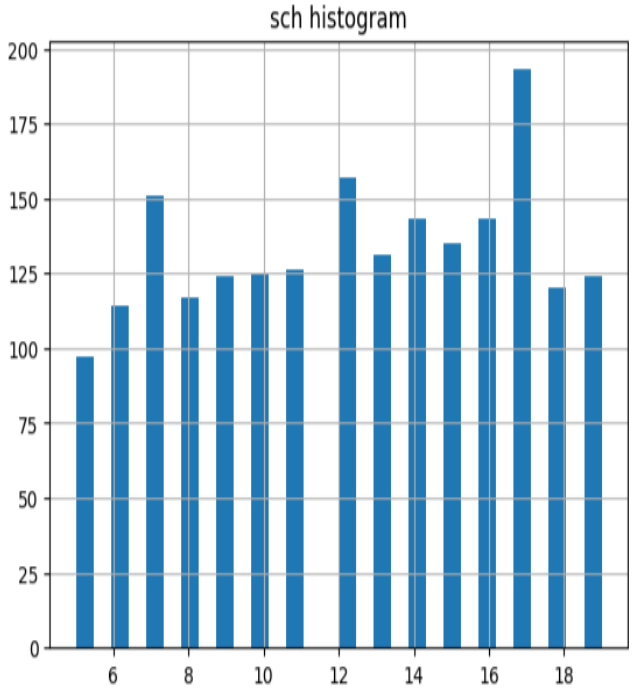
```
import matplotlib.pyplot as plt
df.hist(column='m_dep',bins=20)
plt.title('depth histogram')
plt.show()
```



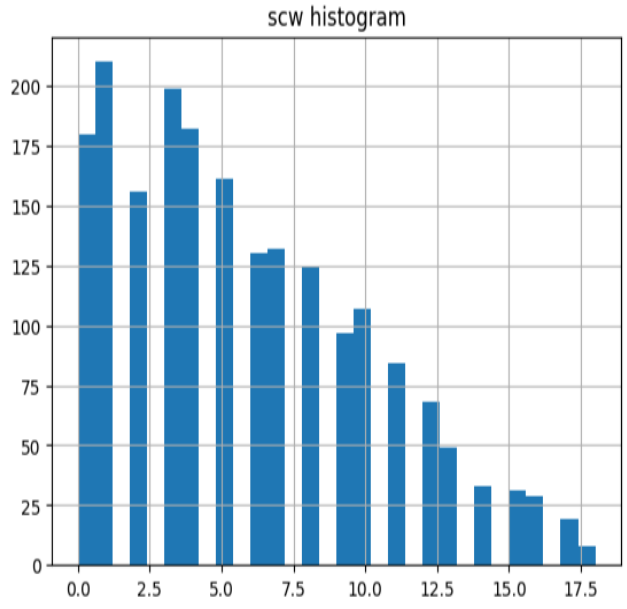

```
import matplotlib.pyplot as plt
df.hist(column='fc',bins=30)
plt.title('fc histogram')
plt.show()
```




```
import matplotlib.pyplot as plt
df.hist(column='sc_h',bins=30)
plt.title('sch histogram')
plt.show()
```

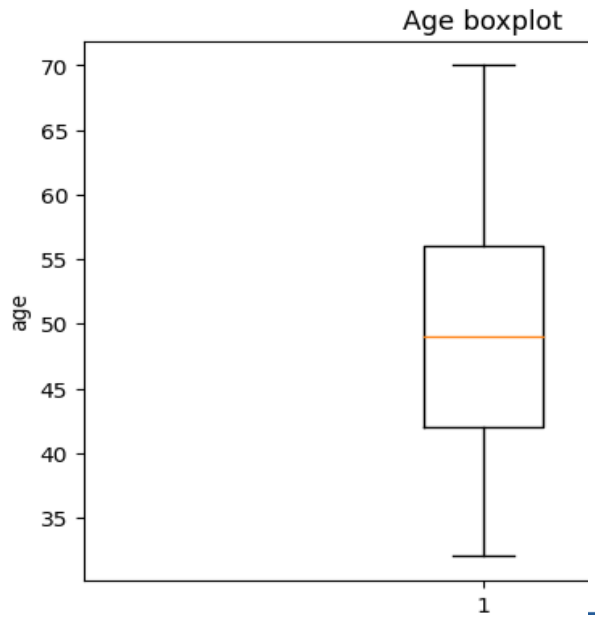



```
import matplotlib.pyplot as plt
df.hist(column='sc_w',bins=30)
plt.title('scw histogram')
plt.show()
```

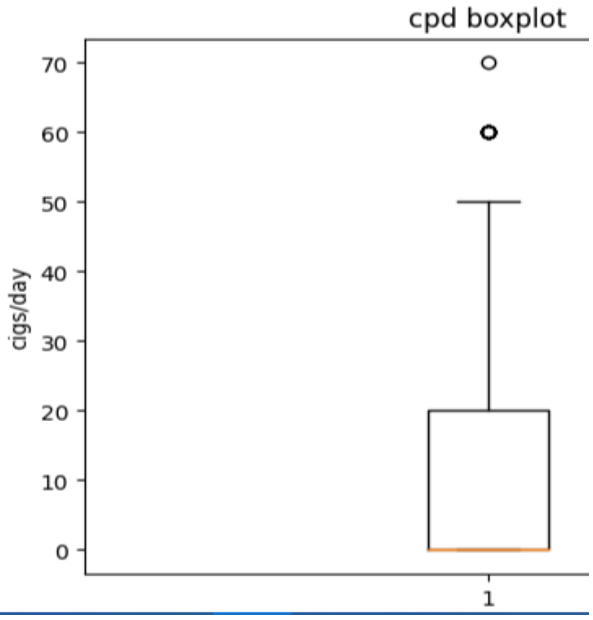


Secondly, we use heart_disease.csv dataset data set for EDA. First, Boxplots

```
import matplotlib.pyplot as plt

plt.boxplot(x="age", data=df)
plt.title("Age boxplot")
plt.ylabel("age")
plt.show()
```




```
import matplotlib.pyplot as plt

plt.boxplot(x="cigsPerDay", data=df)
plt.title("cpd boxplot")
plt.ylabel("cigs/day")
plt.show()
```

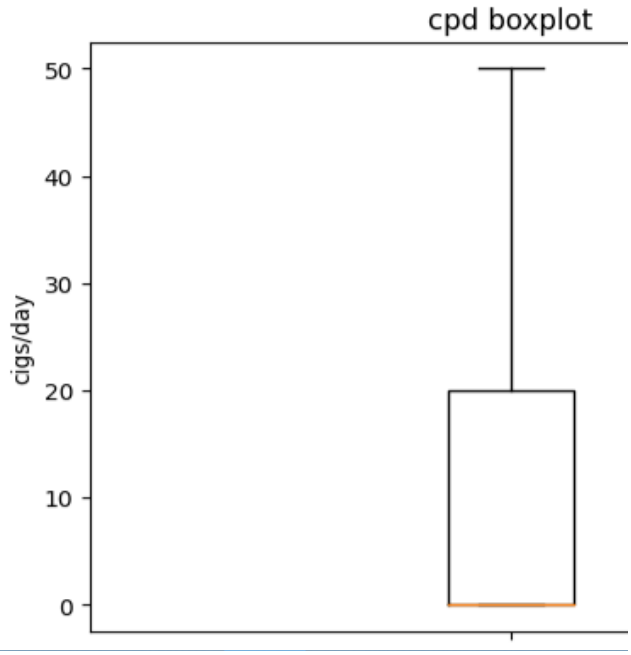


Run Code Voil

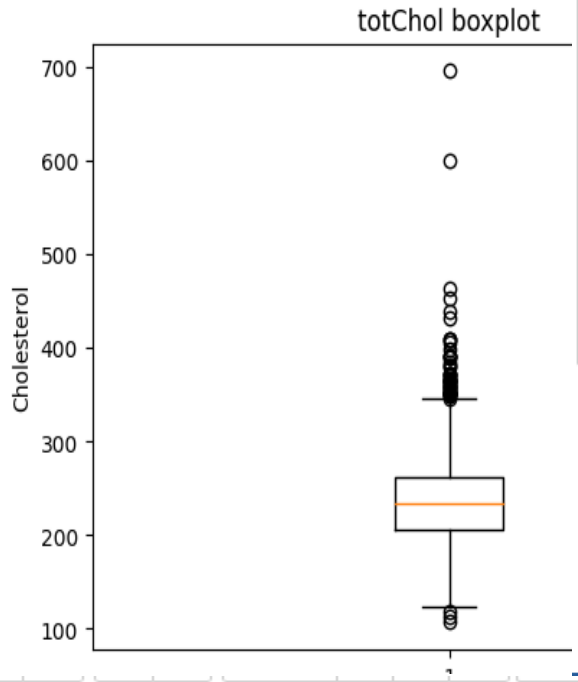
```
]:  C = df['cigsPerDay']
    print('For cpd: ')
    print('Q1: ', C.quantile(0.25))
    print('Q3: ', C.quantile(0.75))
    print('Median: ', C.quantile(0.5))

    IQR_C =  (C.quantile(0.75) - C.quantile(0.25))
    print('IQR for Amount: ', IQR_C)
    print('Lower fench: ', C.quantile(0.25)-(IQR_C * 1.5))
    print('Upper fench: ', C.quantile(0.75)+(IQR_C * 1.5))
    c_max = C.max()
    c_min = C.min()
    print('Minimum: ', c_min)
    print('Maximum: ', c_max)
    if (c_min < (C.quantile(0.25)-(IQR_C*1.5))):
        print("There are Outliers")
    elif(c_max > (C.quantile(0.75)+(IQR_C * 1.5))):
        print('There are right outliers ')
    else:
        print('There are no outliers in cpd. \n')
```

```
For cpd:
Q1:  0.0
Q3:  20.0
Median:  0.0
IQR for Amount:  20.0
Lower fench:  -30.0
Upper fench:  50.0
Minimum:  0.0
Maximum:  70.0
There are right outliers
```

```
]: for x in df.index:
     if df.loc[x, "cigsPerDay"] > 50:
       df.loc[x, "cigsPerDay"] = 20
   import matplotlib.pyplot as plt

   plt.boxplot(x="cigsPerDay", data=df)
   plt.title("cpd boxplot")
   plt.ylabel("cigs/day")
   plt.show()
```



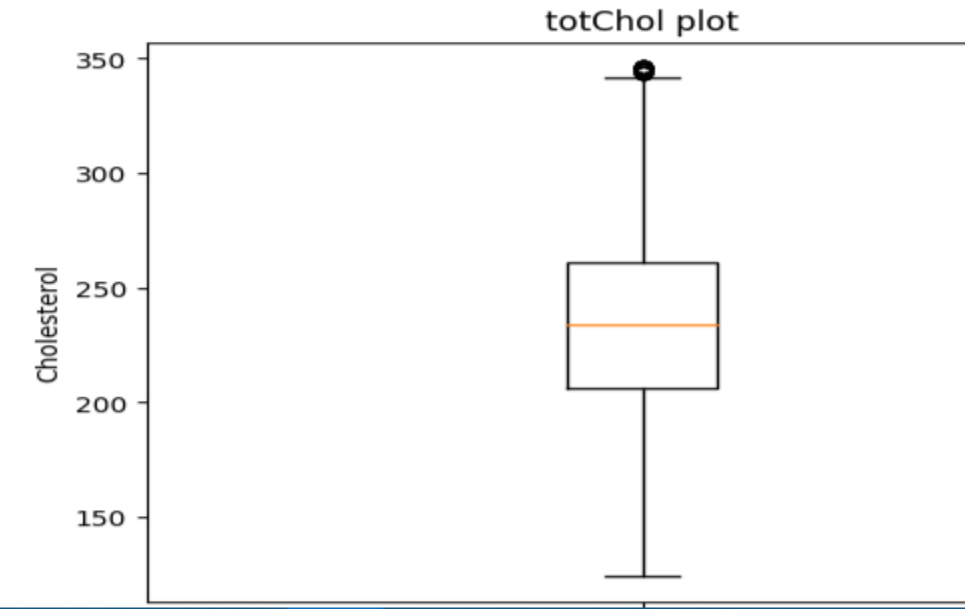
```
import matplotlib.pyplot as plt

plt.boxplot(x="totChol", data=df)
plt.title("totChol boxplot")
plt.ylabel("Cholesterol")
plt.show()
```




```
T = df['totChol']
print('For cholesterol: ')
print('Q1: ', T.quantile(0.25))
print('Q3: ', T.quantile(0.75))
print('Median: ', T.quantile(0.5))

IQR_T =  (T.quantile(0.75) - T.quantile(0.25))
print('IQR for mem: ', IQR_T)
print('Lower fench: ', T.quantile(0.25)-(IQR_T * 1.5))
print('Upper fench: ', T.quantile(0.75)+(IQR_T * 1.5))
t_max = T.max()
t_min = T.min()
print('Minimum: ', t_min)
print('Maximum: ', t_max)
if (t_min < (T.quantile(0.25)-(IQR_T*1.5))):
    print("There are Low Outliers")
if(t_max > (T.quantile(0.75)+(IQR_T * 1.5))):
    print('There are upr outliers ')
else:
    print('There are no outliers in totchol. \n')
```

```
For cholesterol:
Q1:  206.0
Q3:  262.0
Median:  234.0
IQR for mem:  56.0
Lower fench:  122.0
Upper fench:  346.0
Minimum:  107.0
Maximum:  696.0
There are Low Outliers
There are upr outliers
```

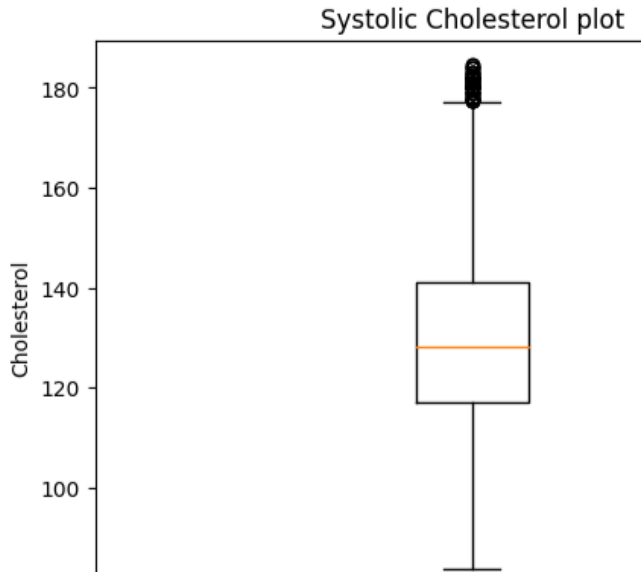
```
for x in df.index:
  if df.loc[x, "totChol"] > 346 or df.loc[x, "totChol"] < 122:
    df.loc[x, "totChol"] = 234
import matplotlib.pyplot as plt

plt.boxplot(x="totChol", data=df)
plt.title("totChol plot")
plt.ylabel("Cholesterol")
plt.show()
```



```
M = df['sysBP']
print('For SystBPP: ')
print('Q1: ', M.quantile(0.25))
print('Q3: ', M.quantile(0.75))
print('Median: ', M.quantile(0.5))

IQR_M =  (M.quantile(0.75) - M.quantile(0.25))
print('IQR for mem: ', IQR_M)
print('Lower fench: ', M.quantile(0.25)-(IQR_M * 1.5))
print('Upper fench: ', M.quantile(0.75)+(IQR_M * 1.5))
m_max = M.max()
m_min = M.min()
print('Minimum: ', m_min)
print('Maximum: ', m_max)
if (m_min < (M.quantile(0.25)-(IQR_M*1.5))):
    print("There are Outliers")
elif(m_max > (M.quantile(0.75)+(IQR_M * 1.5))):
    print('There are UPR outliers ')
else:
    print('There are no outliers in SBP. \n')
```

```
For SystBPP:
Q1:  117.0
Q3:  144.0
Median:  128.0
IQR for mem:  27.0
Lower fench:  76.5
Upper fench:  184.5
Minimum:  83.5
Maximum:  295.0
There are UPR outliers
```

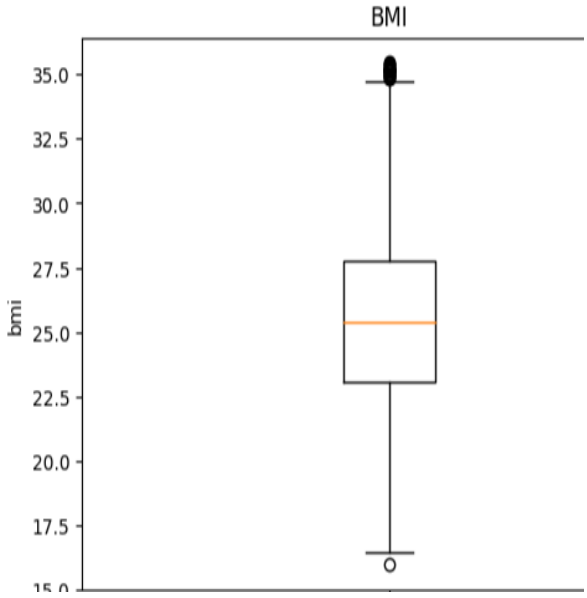
```
for x in df.index:
  if df.loc[x, "sysBP"] > 184.5:
    df.loc[x, "sysBP"] = 128
import matplotlib.pyplot as plt

plt.boxplot(x="sysBP", data=df)
plt.title("Systolic Cholesterol plot")
plt.ylabel("Cholesterol")
plt.show()
```




```
R = df['BMI']
print('For Bmi: ')
print('Q1: ', R.quantile(0.25))
print('Q3: ', R.quantile(0.75))
print('Median: ', R.quantile(0.5))

IQR_R =  (R.quantile(0.75) - R.quantile(0.25))
print('IQR for mem: ', IQR_R)
print('Lower fench: ', R.quantile(0.25)-(IQR_R * 1.5))
print('Upper fench: ', R.quantile(0.75)+(IQR_R * 1.5))
r_max = R.max()
r_min = R.min()
print('Minimum: ', r_min)
print('Maximum: ', r_max)
if (r_min < (R.quantile(0.25)-(IQR_R*1.5))):
    print("There are low Outliers")
if(r_max > (R.quantile(0.75)+(IQR_R * 1.5))):
    print('There are upr outliers ')
else:
    print('There are no outliers in bmi. \n')
```

```
For Bmi:
Q1:  23.08
Q3:  28.0375
Median:  25.41
IQR for mem:  4.957500000000003
Lower fench:  15.643749999999994
Upper fench:  35.47375000000001
Minimum:  15.54
Maximum:  56.8
There are low Outliers
There are upr outliers
```

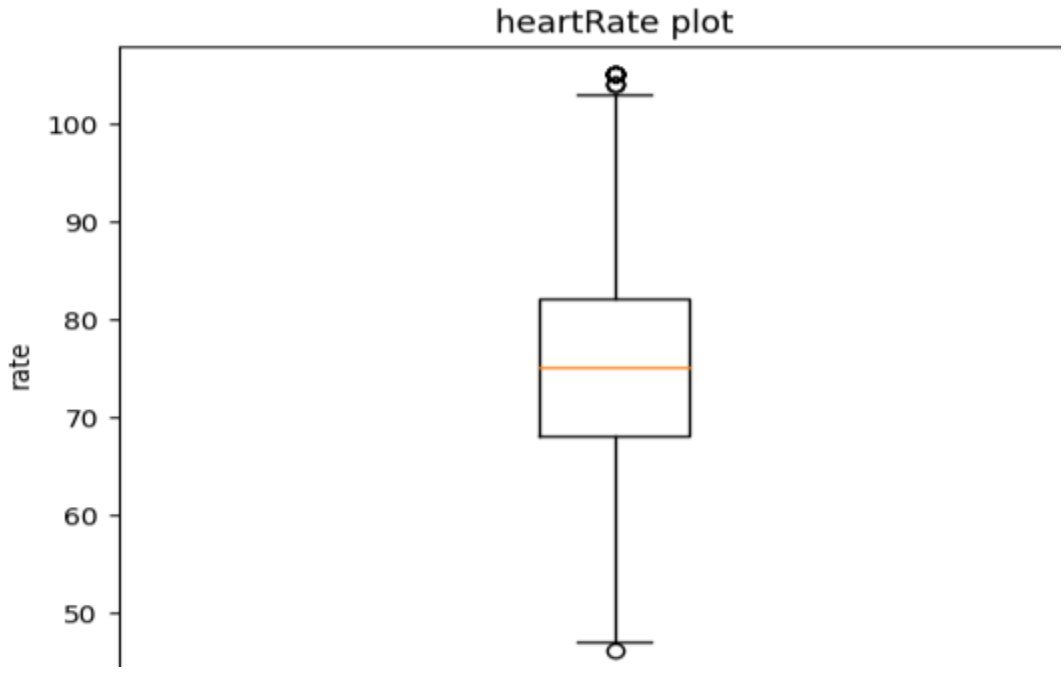
```
for x in df.index:
      if df.loc[x, "BMI"] > 35.5 or df.loc[x, "BMI"] < 15.65:
         df.loc[x, "BMI"] = 25.4
import matplotlib.pyplot as plt

plt.boxplot(x="BMI", data=df)
plt.title("BMI")
plt.ylabel("bmi")
plt.show()
```



```
N = df['heartRate']
print('For rate: ')
print('Q1: ', N.quantile(0.25))
print('Q3: ', N.quantile(0.75))
print('Median: ', N.quantile(0.5))

IQR_N =  (N.quantile(0.75) - N.quantile(0.25))
print('IQR for mem: ', IQR_N)
print('Lower fench: ', N.quantile(0.25)-(IQR_N * 1.5))
print('Upper fench: ', N.quantile(0.75)+(IQR_N * 1.5))
n_max = N.max()
n_min = N.min()
print('Minimum: ', n_min)
print('Maximum: ', n_max)
if (n_min < (N.quantile(0.25)-(IQR_N*1.5))):
    print("There are low Outliers")
if(n_max > (N.quantile(0.75)+(IQR_N * 1.5))):
    print('There are upr outliers ')
else:
    print('There are no outliers in core. \n')
```

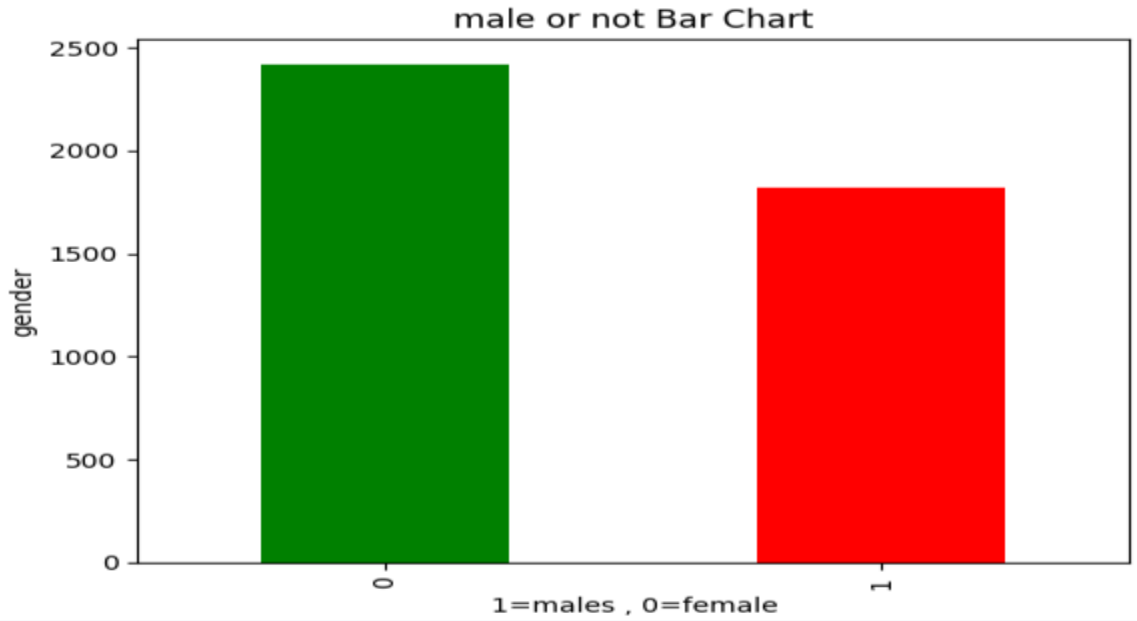
```
For rate:
Q1:  68.0
Q3:  83.0
Median:  75.0
IQR for mem:  15.0
Lower fench:  45.5
Upper fench:  105.5
Minimum:  44.0
Maximum:  143.0
There are low Outliers
There are upr outliers
```

```
for x in df.index:
    if df.loc[x, "heartRate"] > 105 or df.loc[x, "heartRate"] < 46:
        df.loc[x, "heartRate"] = 75
import matplotlib.pyplot as plt

plt.boxplot(x="heartRate", data=df)
plt.title("heartRate plot")
plt.ylabel("rate")
plt.show()
```

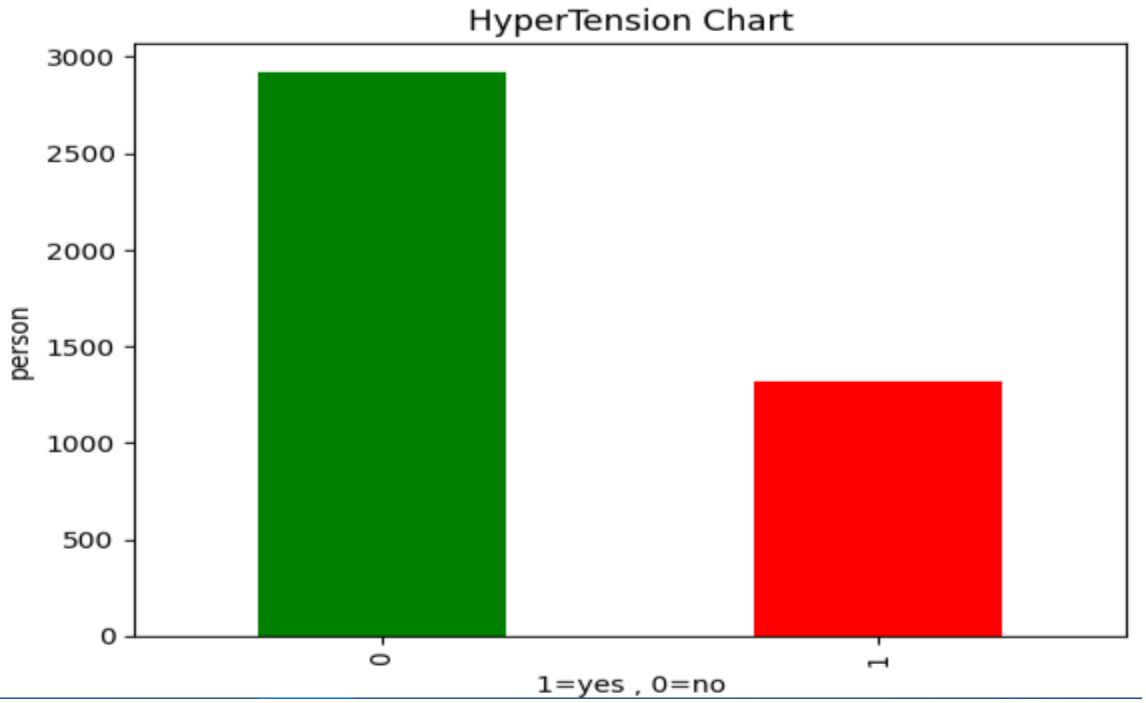


Now we draw the bar chart or pie chart for the categorical type data because of having Boolean values so we calculate the portion of the heart-disease features by Bar chart

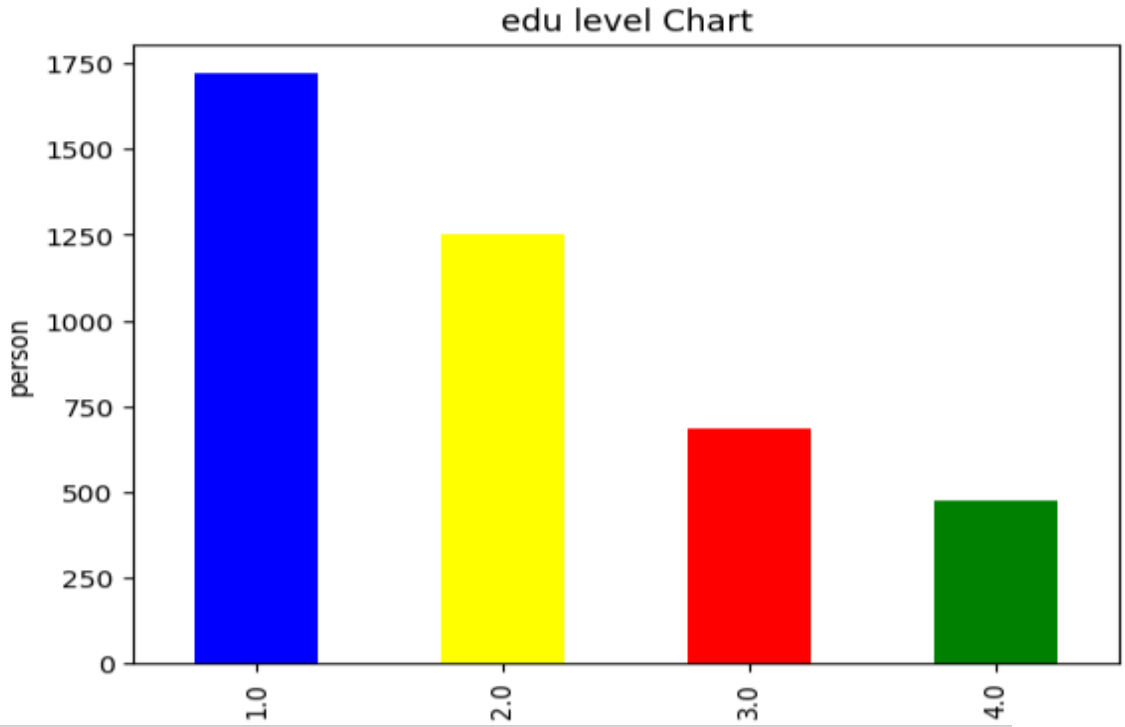
```
df["male"].value_counts().plot(kind='bar', color=['green','red'])
plt.title("male or not Bar Chart")
plt.xlabel("1=males , 0=female")
plt.ylabel("gender")
plt.show()
```



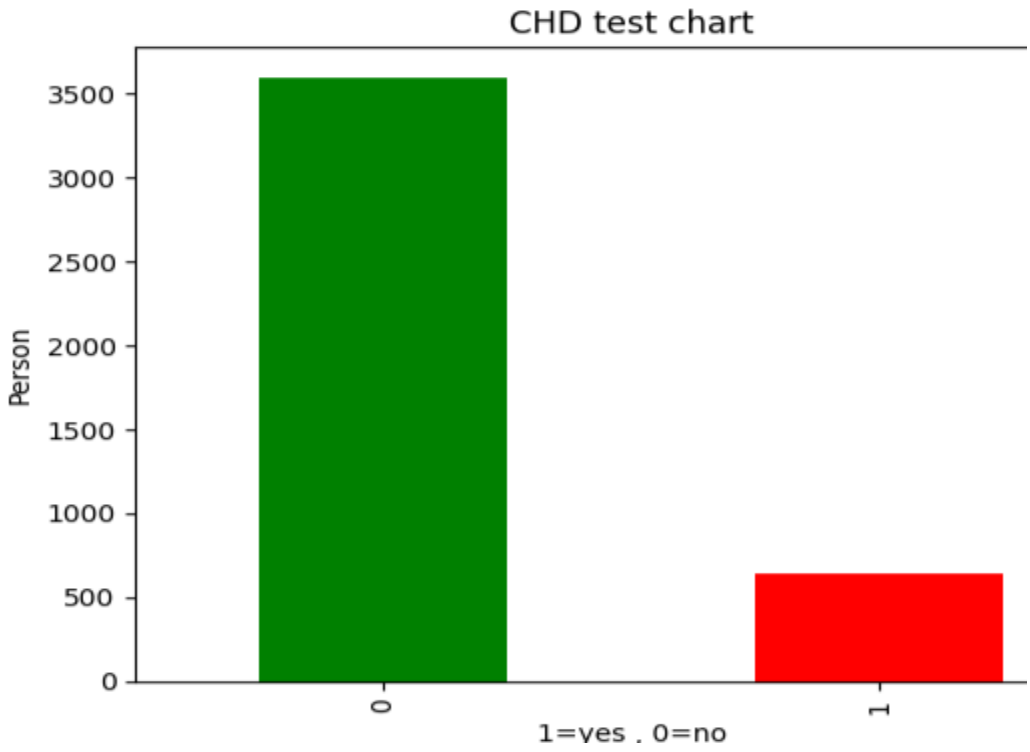

```
df["prevalentHyp"].value_counts().plot(kind='bar', color=['green','red'])
plt.title("HyperTension Chart")
plt.xlabel("1=yes , 0=no")
plt.ylabel("person")
plt.show()
```



```
df["education"].value_counts().plot(kind='bar', color=['blue','yellow','red','green'])
plt.title("edu level Chart")
plt.ylabel("person")
plt.show()
```

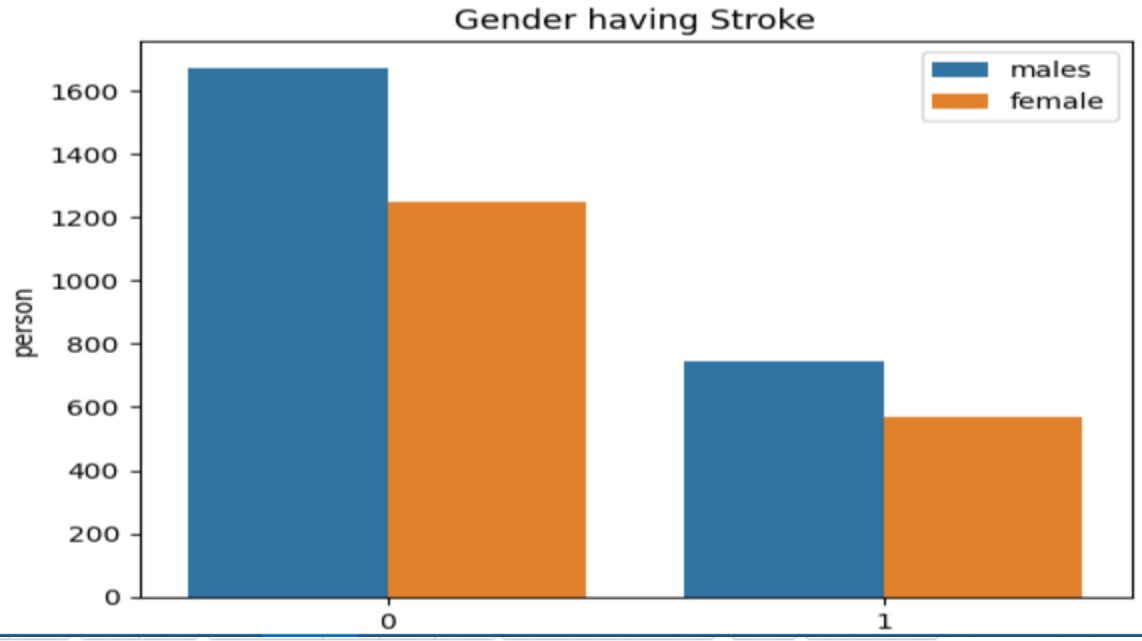



```
df["TenYearCHD"].value_counts().plot(kind='bar', color=['green','red
plt.title("CHD test chart")
plt.xlabel("1=yes , 0=no")
plt.ylabel("Person")
plt.show()
```



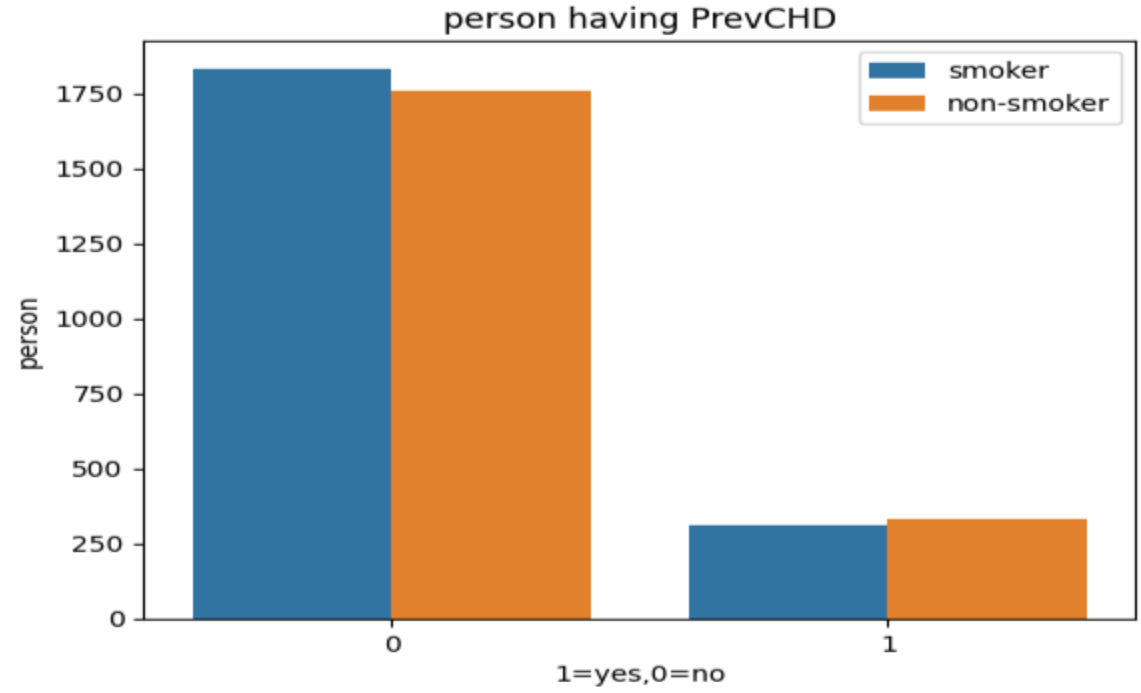
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.countplot(x='prevalentHyp', data=df, hue='male')
plt.title("Gender having Stroke")
plt.xlabel("1=yes,0=no")
plt.ylabel("person")
plt.legend(['males','female'])
```

```
<matplotlib.legend.Legend at 0x1d79d4f5848>
```

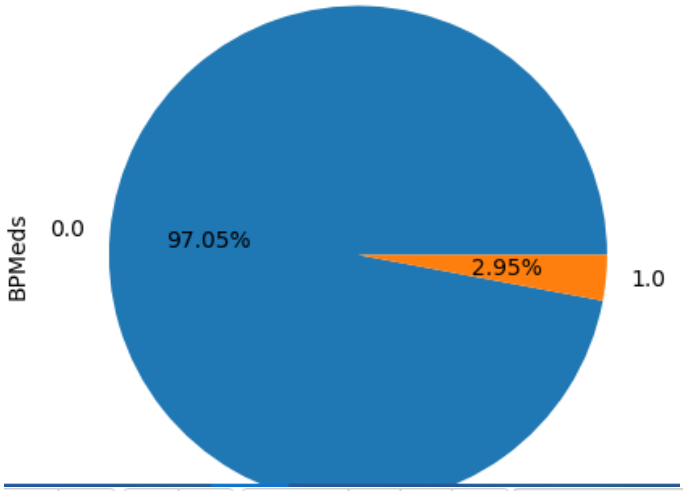



```
sns.countplot(x='TenYearCHD', data=df, hue='currentSmoker')
plt.title("person having PrevCHD")
plt.xlabel("1=yes,0=no")
plt.ylabel("person")
plt.legend(['smoker','non-smoker'])
```

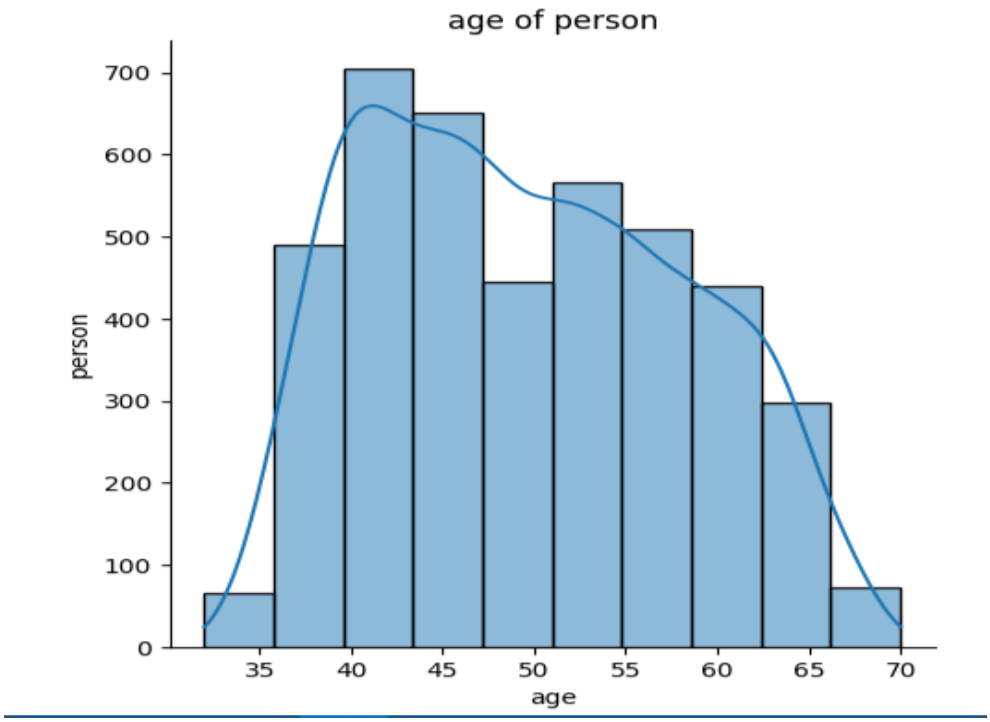
```
: <matplotlib.legend.Legend at 0x1d79d4d2908>
```



```
df.BPMeds.value_counts().plot(kind='pie',figsize=(10,5),autopct='%1.2f%%')
plt.legend([BPMeds, NoBPMeds])
```




```
sns.displot(x='age',data=df,bins=10,kde=True)
plt.title('age of person')
plt.ylabel("person")
plt.show()
```



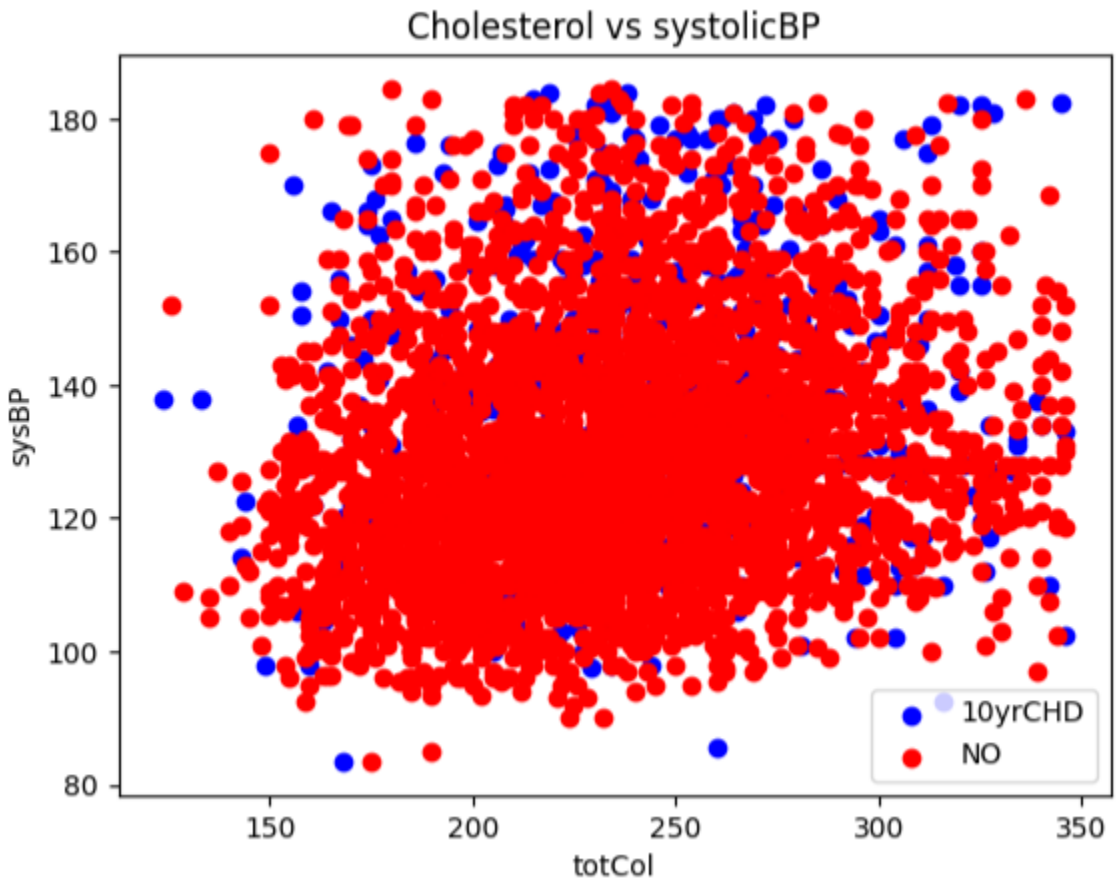
The continuous distribution for age is almost normally distributed because we did not see any outliers in the boxplot of age.

Hence, this is a good feature which can give good testing result

In [44]:
```
import matplotlib.pyplot as plt

plt.scatter(df.totChol[df.TenYearCHD==1],
            df.sysBP[df.TenYearCHD==1] , c='blue')
#no-10yrCHD
plt.scatter(df.totChol[df.TenYearCHD==0],
            df.sysBP[df.TenYearCHD==0] , c='red')
plt.title('Cholesterol vs systolicBP')
plt.xlabel("totCol")
plt.ylabel("sysBP")
plt.legend(['10yrCHD','NO'])
```

S d

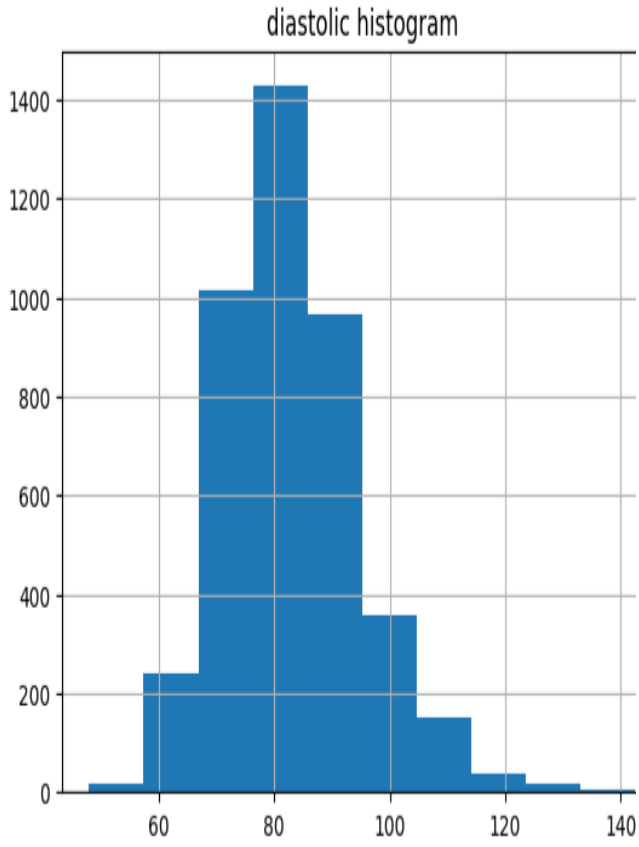
Out[44]: <matplotlib.legend.Legend at 0x1d79cd14e08>



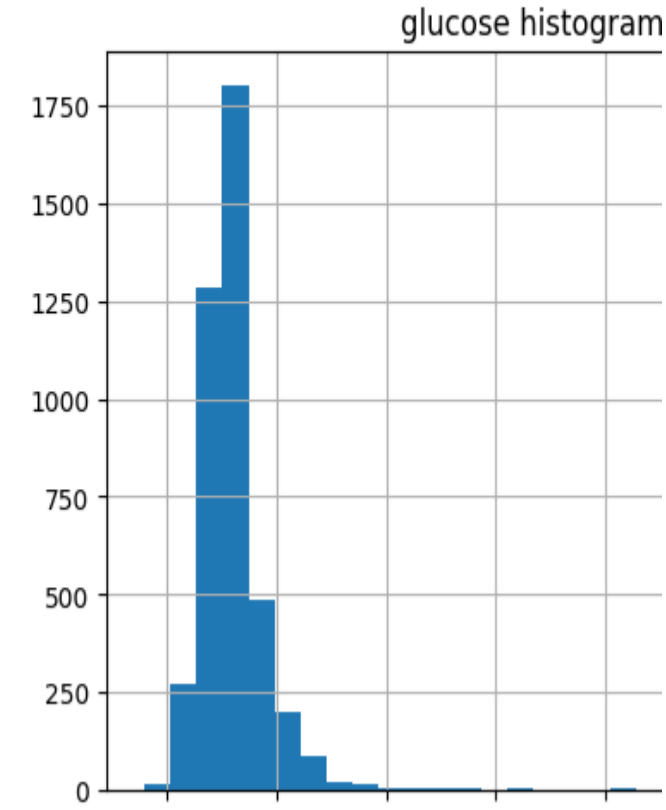
In this correlation scatter plot we can see that the systolic blood pressure has almost got positive correlation with total cholesterol having 10 year CHD or not.

Now, we draw the histograms of some feature and we find: some plots are rightly skewed which says that the mean is greater than median so it is positively skewed and the more forward we get the lower size the data becomes which can cause outliers in some cases, Let's see histogram of diaBP and glucose

```
import matplotlib.pyplot as plt
df.hist(column='diaBP',bins=10)
plt.title('diastolic histogram')
plt.show()
```




```
import matplotlib.pyplot as plt
df.hist(column='glucose',bins=30)
plt.title('glucose histogram')
plt.show()
```



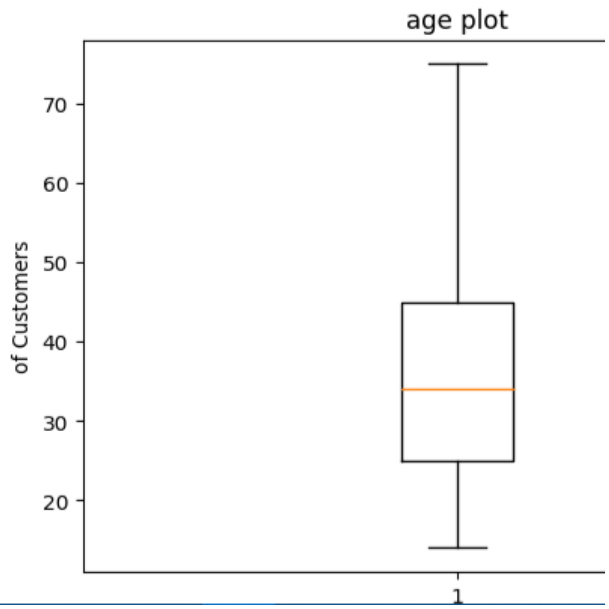
Finally we make the boxplots of the final dataset great_customers.csv features to find any possible outliers and standardize the data if needed

```
]:  A = numc['age']
    print('For age: ')
    print('Q1: ', A.quantile(0.25))
    print('Q3: ', A.quantile(0.75))
    print('Median: ', A.quantile(0.5))

    IQR_A =  (A.quantile(0.75) - A.quantile(0.25))
    print('IQR for Amount: ', IQR_A)
    print('Lower fench: ', A.quantile(0.25)-(IQR_A * 1.5))
    print('Upper fench: ', A.quantile(0.75)+(IQR_A * 1.5))
    a_max = A.max()
    a_min = A.min()
    print('Minimum: ', a_min)
    print('Maximum: ', a_max)
    if (a_min < (A.quantile(0.25)-(IQR_A*1.5))):
        print("There are low Outliers")
    if(a_max > (A.quantile(0.75)+(IQR_A * 1.5))):
        print('There are upr outliers ')
    else:
        print('There are no outliers in age. \n')
```

```
For age:
Q1:  25.0
Q3:  45.0
Median:  34.0
IQR for Amount:  20.0
Lower fench:  -5.0
Upper fench:  75.0
Minimum:  14.0
Maximum:  90.0
There are right outliers
```

```
6]:  for x in numc.index:
         if numc.loc[x, "age"] > 75:
             numc.loc[x, "age"] = 34
```

```
8]:  plt.boxplot(x="age", data=numc)
     plt.title("age plot")
     plt.ylabel("of Customers")
     plt.show()
```

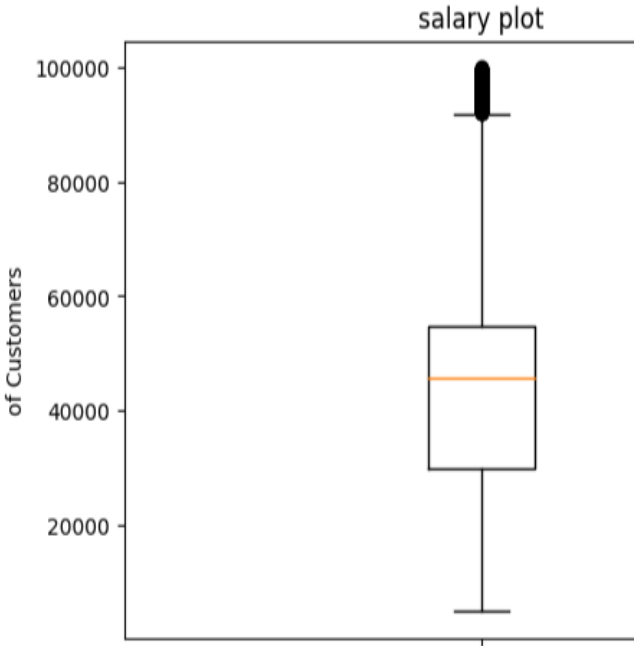



```
5]:  A = numc['salary']
     print('For salary: ')
     print('Q1: ', A.quantile(0.25))
     print('Q3: ', A.quantile(0.75))
     print('Median: ', A.quantile(0.5))
     IQR_A =  (A.quantile(0.75) - A.quantile(0.25))
     print('IQR for Amount: ', IQR_A)
     print('Lower fench: ', A.quantile(0.25)-(IQR_A * 1.5))
     print('Upper fench: ', A.quantile(0.75)+(IQR_A * 1.5))
     a_max = A.max()
     a_min = A.min()
     print('Minimum: ', a_min)
     print('Maximum: ', a_max)
     if (a_min < (A.quantile(0.25)-(IQR_A*1.5))):
         print("There are low Outliers")
     if(a_max > (A.quantile(0.75)+(IQR_A * 1.5))):
         print('There are upr outliers ')
     else:
         print('There are no outliers in salary. \n')
```

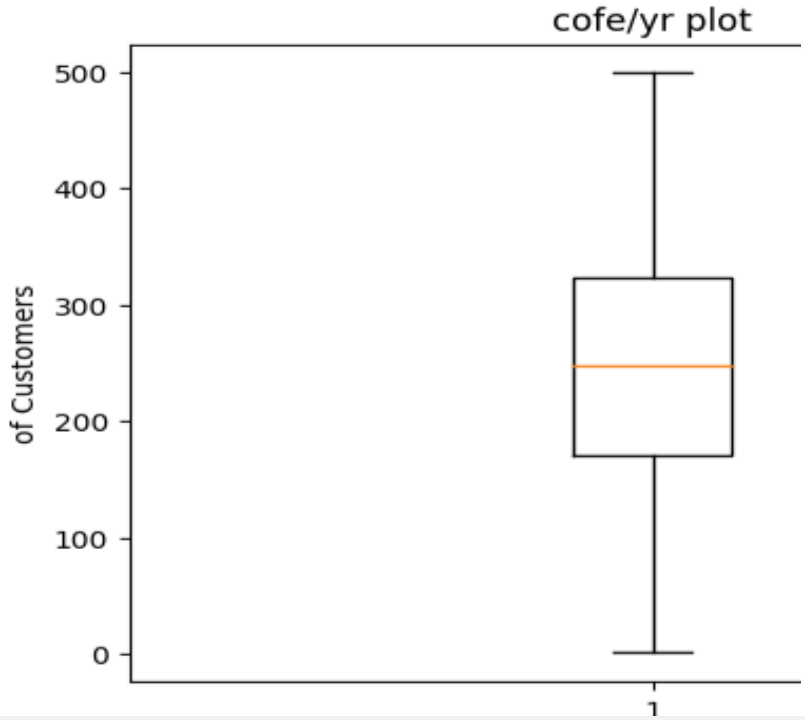
```
For salary:
Q1:  29967.0
Q3:  57953.625
Median:  45643.5
IQR for Amount:  27986.625
Lower fench:  -12012.9375
Upper fench:  99933.5625
Minimum:  4872.75
Maximum:  363858.75
There are upr outliers
```

```
7]:  for x in numc.index:
         if numc.loc[x, "salary"] > 99933.5625:
             numc.loc[x, "salary"] = 45643.5
```

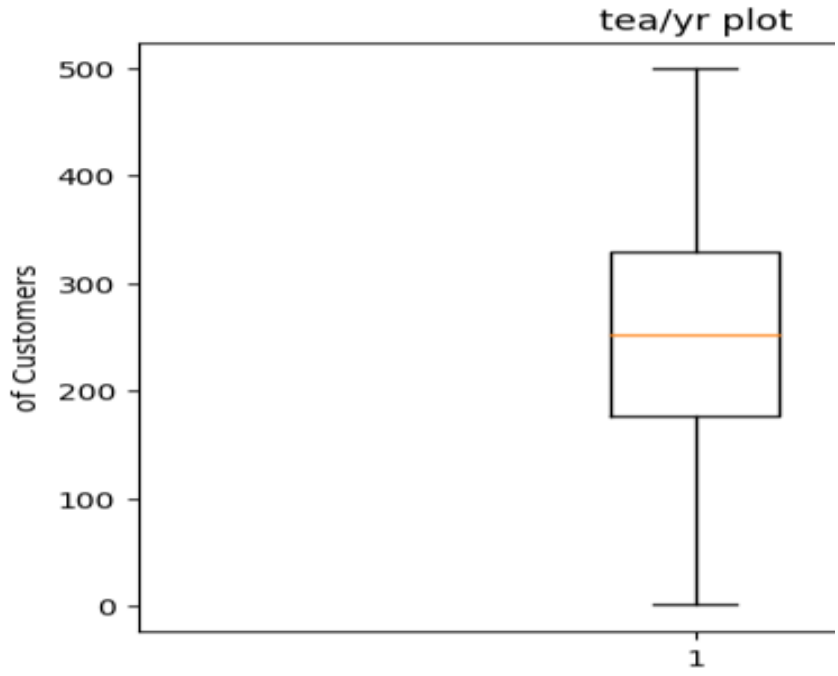
```
plt.boxplot(x="salary", data=numc)
plt.title("salary plot")
plt.ylabel("of Customers")
plt.show()
```



```
plt.boxplot(x="coffee_per_year", data=numc)
plt.title("cofe/yr plot")
plt.ylabel("of Customers")
plt.show()
```

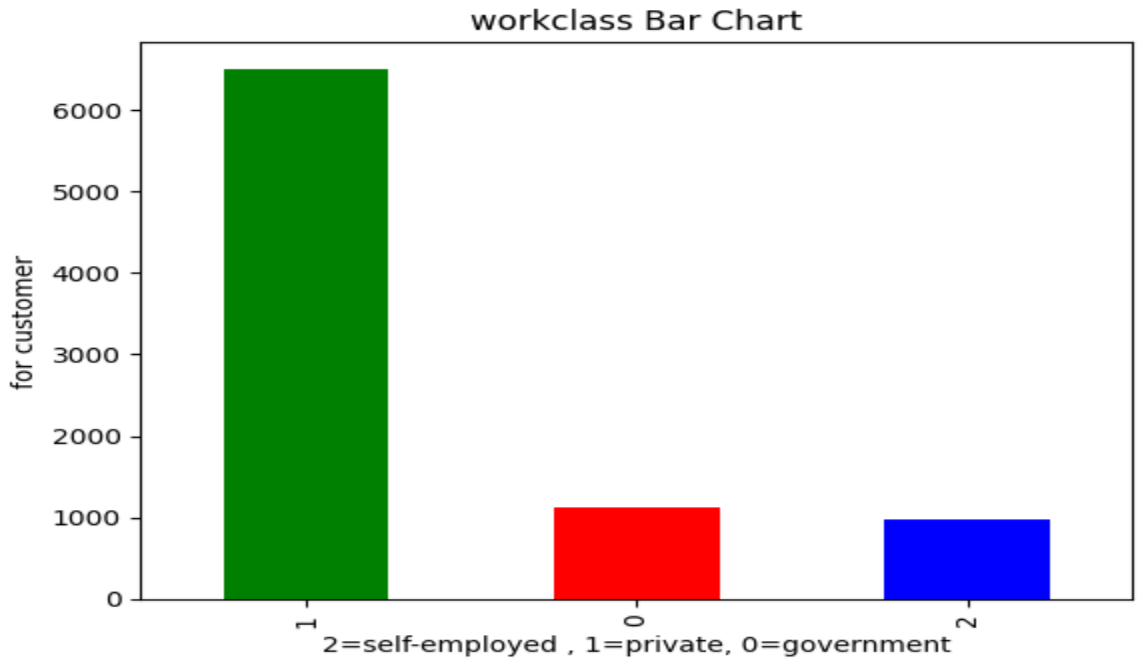



```
plt.boxplot(x="tea_per_year", data=numc)
plt.title("tea/yr plot")
plt.ylabel("of Customers")
plt.show()
```

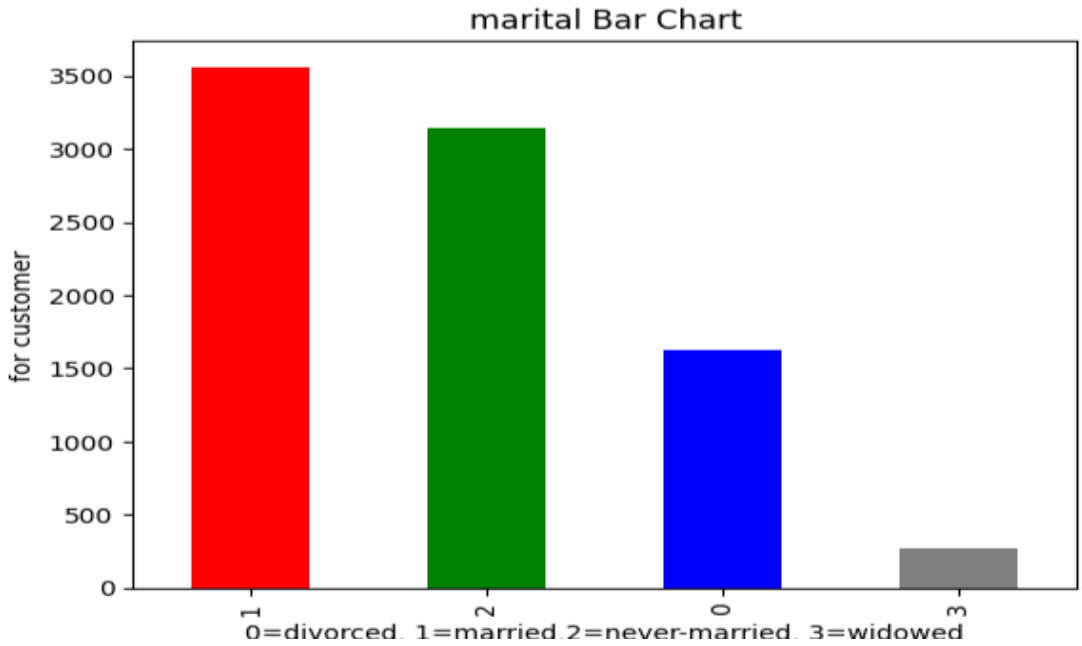


Now we draw the bar chart or pie chart for the categorical type data because of having Boolean values so we calculate the portion of the great-customers features by Bar chart

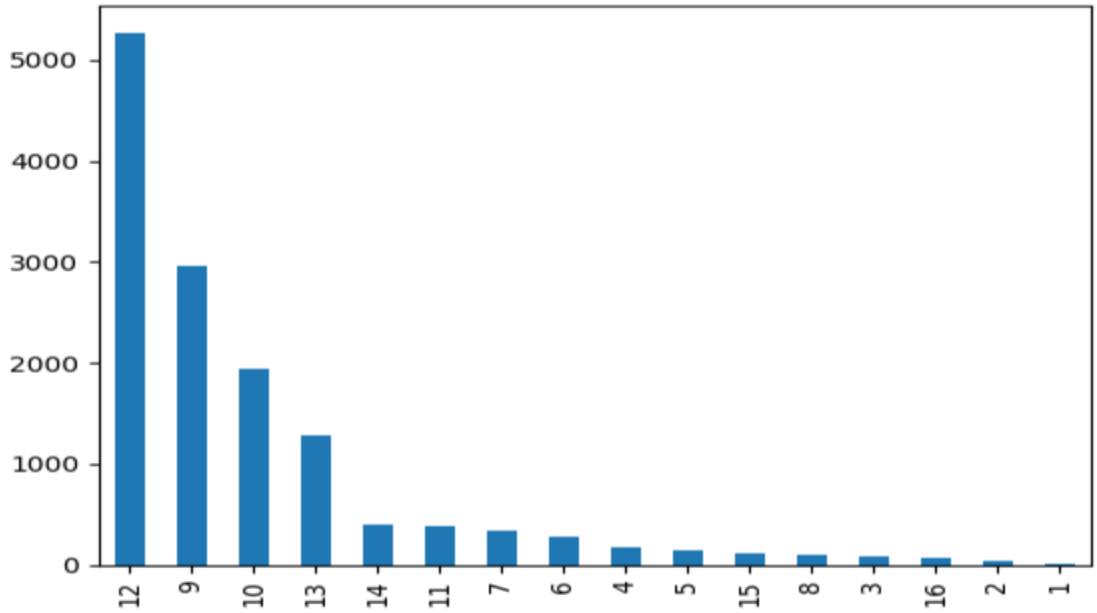
```
catg["workclass"].value_counts().plot(kind='bar', color=['green','red','blue']
plt.title("workclass Bar Chart")
plt.xlabel("2=self-employed , 1=private, 0=government")
plt.ylabel("for customer")
plt.show()
```



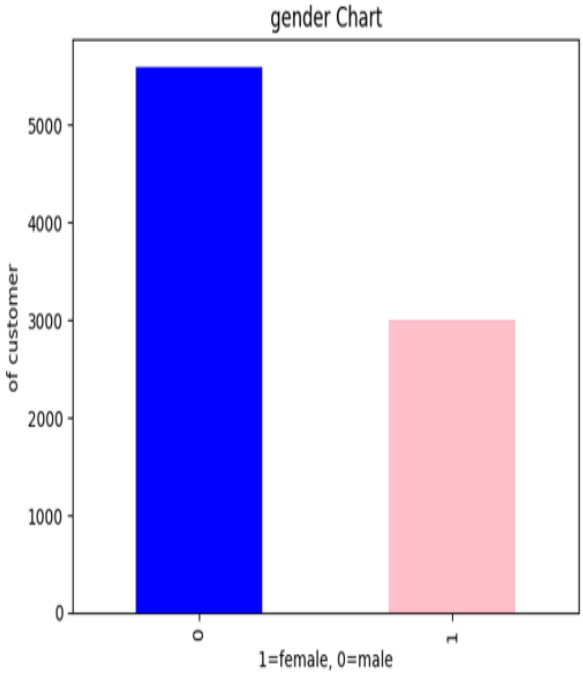

```
catg["marital-status"].value_counts().plot(kind='bar', color=['red','green','blue','grey']
plt.title("marital Bar Chart")
plt.xlabel("0=divorced, 1=married,2=never-married, 3=widowed")
plt.ylabel("for customer")
plt.show()
```



```
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\great_customers.csv'
df['education_rank'].value_counts().plot(kind='bar')
plt.show()
```

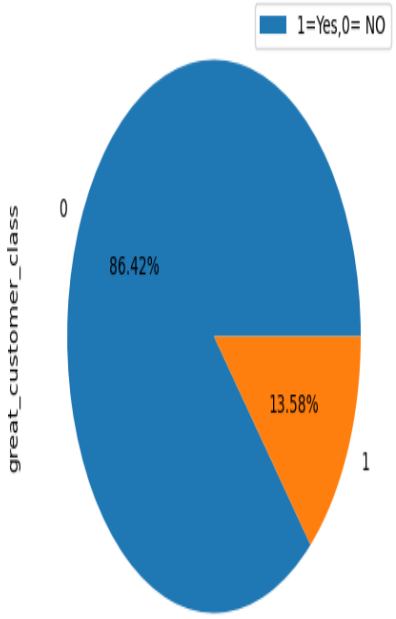



```
catg["sex"].value_counts().plot(kind='bar', color=['blue','pink'])
plt.title("gender Chart")
plt.xlabel("1=female, 0=male")
plt.ylabel("of customer")
plt.show()
```



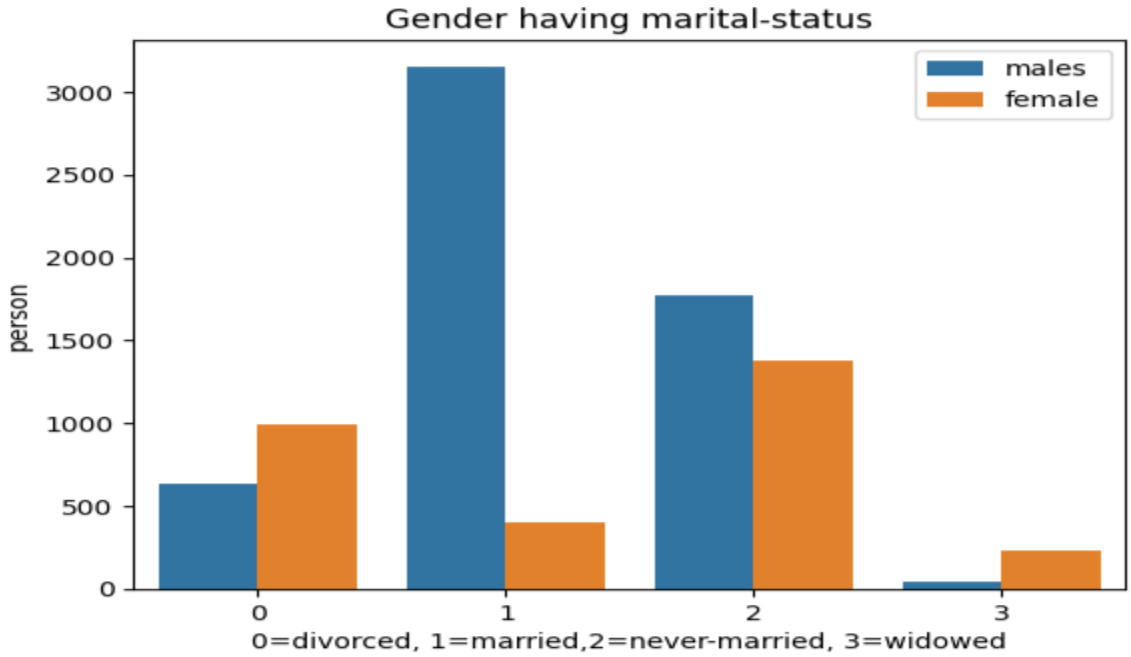

```
[46]: numc.great_customer_class.value_counts().plot(kind='pie',figsize=(10,5),autopct='%1.2f%%')
      plt.legend(["1=Yes,0= NO"])
```

```
[46]: <matplotlib.legend.Legend at 0x27cac708148>
```



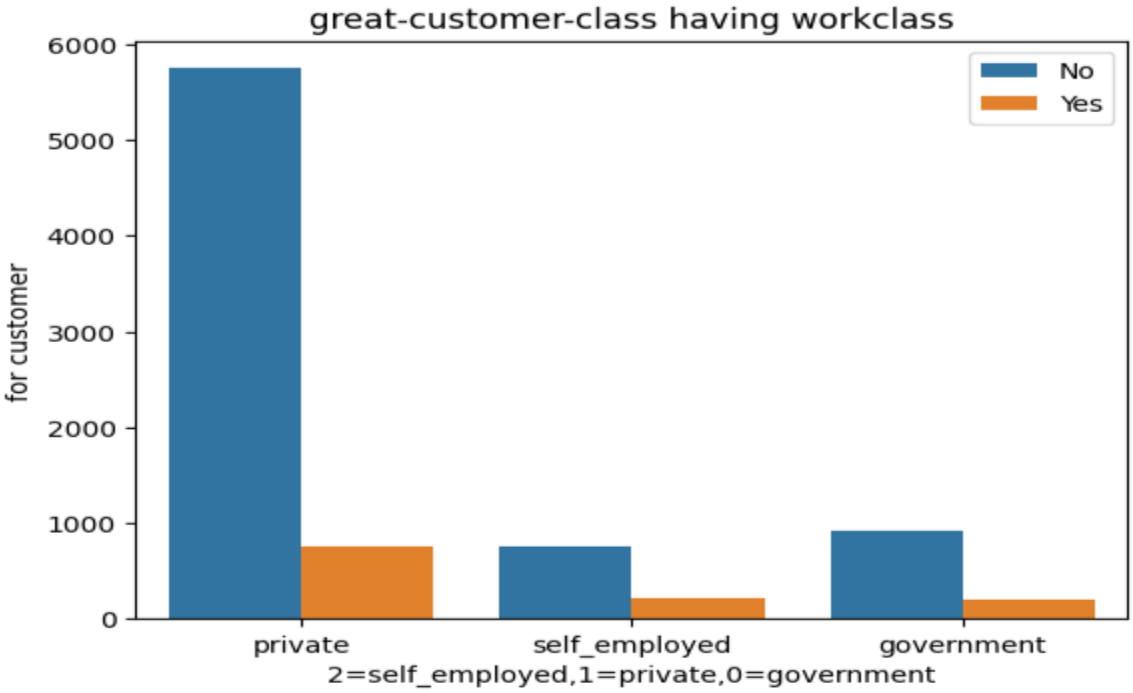
```
sns.countplot(x='marital-status', data=catg, hue='sex')
plt.title("Gender having marital-status")
plt.xlabel("0=divorced, 1=married,2=never-married, 3=widowed")
plt.ylabel("person")
plt.legend(['males','female'])
```

```
<matplotlib.legend.Legend at 0x27cac9b1dc8>
```

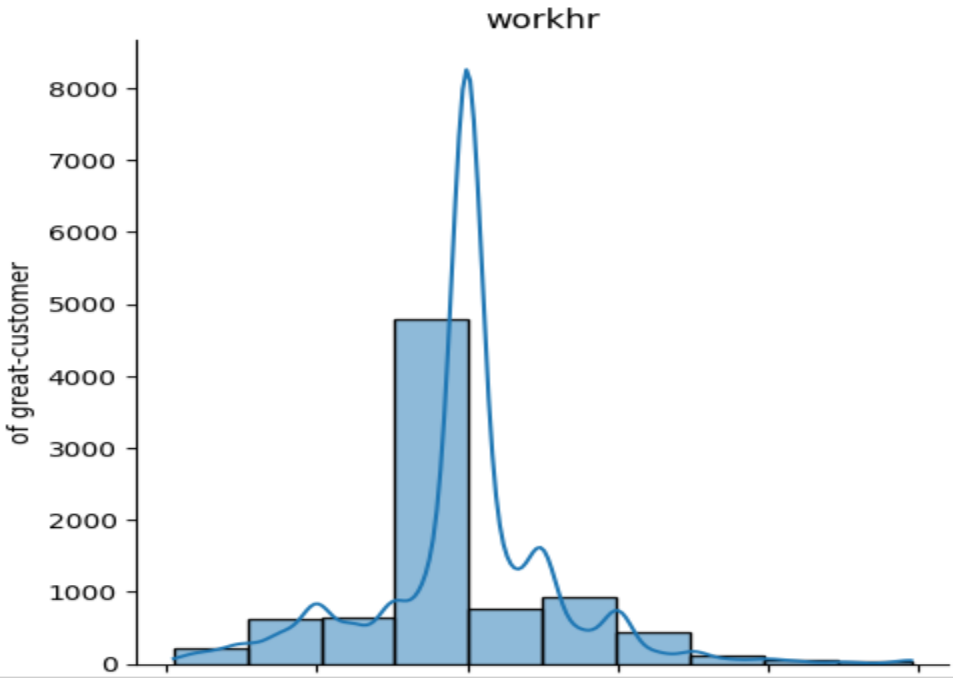



```
sns.countplot(x='workclass', data=df, hue='great_customer_class')
plt.title("great-customer-class having workclass")
plt.xlabel("2=self_employed,1=private,0=government")
plt.ylabel("for customer")
plt.legend(['No','Yes'])
```

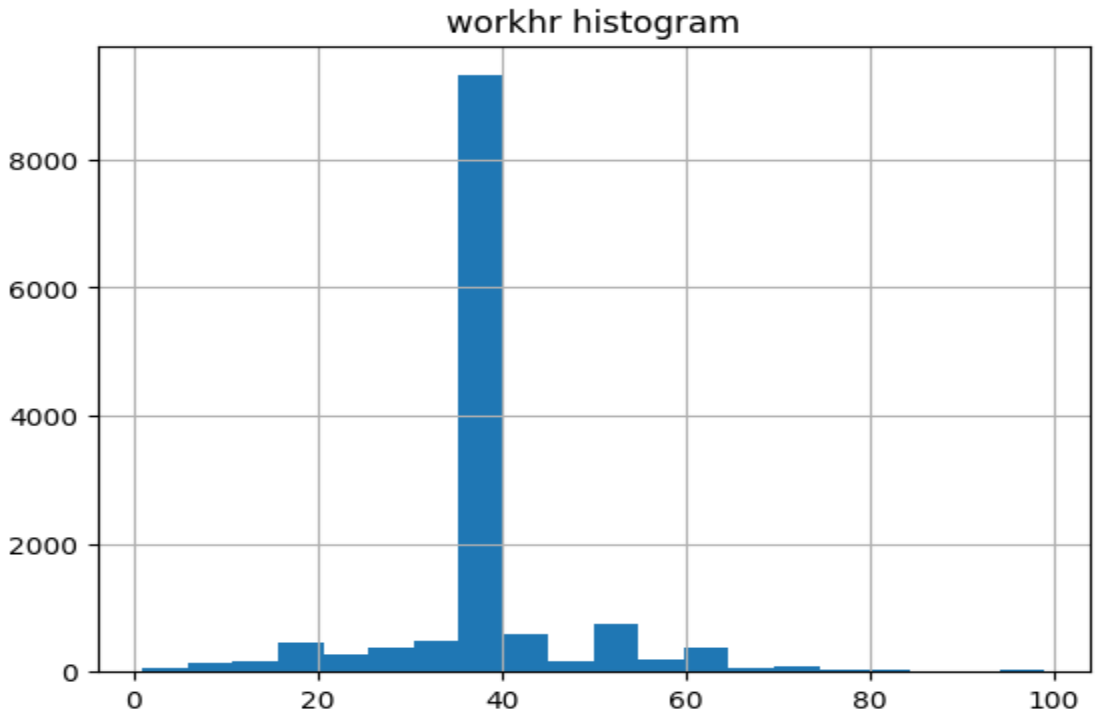
```
<matplotlib.legend.Legend at 0x27cae6a2308>
```



```
[64]: import seaborn as sns
      sns.displot(x='works_hours',data=df,bins=10,kde=True)
      plt.title('workhr')
      plt.ylabel("of great-customer")
      plt.show()
```




```
:  import matplotlib.pyplot as plt
   numc.hist(column='works_hours',bins=20)
   plt.title('workhr histogram')
   plt.show()
```



From both plots we can see the distribution of work-hour is Leptokartic because of having so much outliers hence we can not get fruitful result from this feature.

Hypothesis Testing : hypothesis testing means estimating some measurements and testing if our assumption is true or false, 1st we tested mobile-price-train dataset.

In [2]:
```
df['battery_power'].mean()
```

Out[2]:
```
1238.5185
```

In [18]:
```
import pandas as pd
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.csv')
#hypothesis testing for 1st 50 rows assuming the mean is 1000
df1 = df.loc[0:49,['battery_power']]
Z1 = (df1.mean()-1200)/(df1.std()/math.sqrt(50))
print("Z score value for")
print(Z1)
```

```
Z score value for
battery_power   -0.618399
dtype: float64
```

In [19]:
```
#assume 95% CI so significance level alpha = 0.05, now if P>alpha accept Null Hyp otherwise reject it
#for P(Z1<-0.618)=0.268,
P1=2*0.268
if(P1>0.05):
    print("accept Null Hypothesis because mean 1200 lies in 95% CI and is not in rejection region of significance")
else:
    print("Reject null hypothesis and accept alternative Hypothesis because mean 1200 does not lies in 95% CI and is in rejecti
```

```
accept Null Hypothesis because mean 1200 lies in 95% CI and is not in rejection region of significance
```

:
```
df['clock_speed'].mean()
```

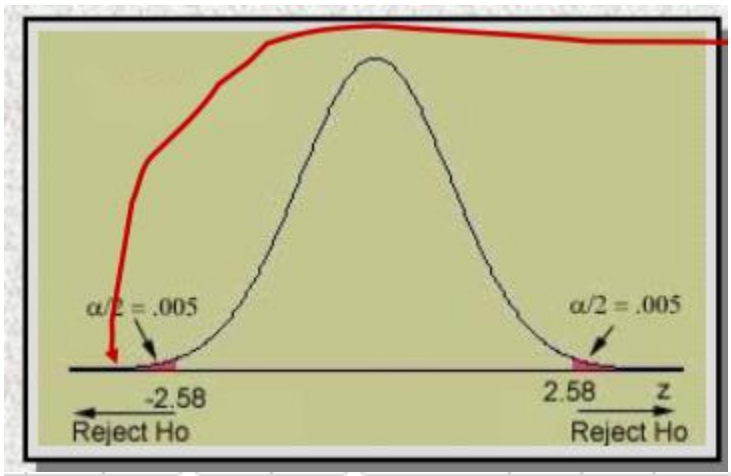
:
```
1.52225
```

:
```
#assume clock speed had been is 1.8
df2 = df.loc[0:49,['clock_speed']]
Z2 = (df2.mean()-1.8)/(df2.std()/math.sqrt(50))
print("Z score value for")
print(Z2)
```

```
Z score value for
clock_speed   -2.501534
dtype: float64
```

:
```
#assume 99% CI so significance level alpha = 0.01, now if P>alpha
#for P(Z1<-2.5)=0.268,
P2=2*0.0062
if(P2>0.01):
    print("accept Null Hypothesis because mean 1.8 lies in 99% CI
else:
    print("Reject null hypothesis and accept alternative Hypothes
```

```
accept Null Hypothesis because mean 1.8 lies in 99% CI and is not
```



8]:
```
import pandas as pd
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\m
#hypothesis testing for last 40 rows assuming the mean
df3 = df.loc[1960:1999,['int_memory']]
Z3 = (df3.mean()-30)/(df3.std()/math.sqrt(40))
print("Z score value for")
print(Z3)
```

```
Z score value for
int_memory    0.044201
dtype: float64
```

9]:
```
#assume 95% CI so significance level alpha = 0.05, now
#for P(Z1<0.0442)=0.0.136,
P1=2*0.48
if(P1>0.05):
    print("accept Null Hypothesis because mean 30 lies
else:
    print("Reject null hypothesis because mean 30 does
```

```
accept Null Hypothesis because mean 30 lies in 95% CI
```

Null Hypothesis: population mean is close to sample mean in a normal distribution curve with certain significance level alpha where alpha is the rejected region

Alternative Hypothesis: population mean varies a lot from sample mean.

```
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_pri
#hypothesis testing for last 35 rows assuming the mean is 30
df4 = df.loc[0:34,['m_dep']]
Z4 = (df4.mean()-0.7)/(df4.std()/math.sqrt(35))
print("Z score value for")
print(Z4)
```

```
Z score value for
m_dep   -1.90976
dtype: float64
```

```
#assume 94% CI so significance level alpha = 0.06, now if P>alp
#for P(Z1<-1.1)=0.028,
P1=2*0.028
if(P1>0.06):
    print("accept Null Hypothesis Ho because mean 30 lies in 95
else:
    print("Reject null hypothesis because mean 30 does not lies
```

```
Reject null hypothesis because mean 30 does not lies in 94% CI
```

n [16]:
```
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_pr
#hypothesis testing for last 50 rows assuming the mean is 150
df5 = df.loc[0:49,['mobile_wt']]
Z5 = (df5.mean()-150)/(df5.std()/math.sqrt(50))
print("Z score value for")
print(Z5)
```

```
Z score value for
mobile_wt   -1.138901
dtype: float64
```

n [17]:
```
#assume 95% CI so significance level alpha = 0.06, now if P>al
#for P(Z1<-1.14)=0.126,
P1=2*0.126
if(P1>0.05):
    print("accept Null Hypothesis Ho because mean 150 lies in
else:
    print("Reject null hypothesis because mean 150 does not li
```

```
accept Null Hypothesis Ho because mean 150 lies in 95% CI and
```


```
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_
#hypothesis testing for last 50 rows assuming the mean is 8
df6 = df.loc[0:49,['n_cores']]
Z6 = (df6.mean()-6)/(df6.std()/math.sqrt(50))
print("Z score value for")
print(Z6)
```

```
Z score value for
n_cores    -4.328103
dtype: float64
```

```
#assume 95% CI so significance level alpha = 0.06, now if P>
#for P(Z1<-4.33)=0.0000082,
P1=2*0.0000082
if(P1>0.05):
    print("accept Null Hypothesis Ho because mean 150 lies i
else:
    print("Reject null hypothesis because mean 150 does not
```

```
Reject null hypothesis because mean 150 does not lies in 95%
```

```
]: df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mob
#hypothesis testing for last 50 rows assuming the mean i
df7 = df.loc[0:49,['px_height']]
Z7 = (df7.mean()-800)/(df7.std()/math.sqrt(50))
print("Z score value for")
print(Z7)
```

```
Z score value for
px_height   -2.039256
dtype: float64
```

```
]: #assume 99% CI so significance level alpha = 0.06, now i
#for P(Z1<-2.04)=0.02,
P1=2*0.02
if(P1>0.01):
    print("accept Null Hypothesis Ho because mean 150 li
else:
    print("Reject null hypothesis because mean 150 does
```

```
accept Null Hypothesis Ho because mean 150 lies in 99% C
```

```
]: df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mob
#hypothesis testing for last 50 rows assuming the mean i
df8 = df.loc[0:49,['px_width']]
Z8 = (df8.mean()-1200)/(df8.std()/math.sqrt(50))
print("Z score value for")
print(Z8)
```

```
Z score value for
px_width    0.889525
dtype: float64
```

```
]: #assume 95% CI so significance level alpha = 0.06, now i
#for P(Z1<0.89)=0.8,
P1=2*0.8
if(P1>0.05):
    print("accept Null Hypothesis Ho because mean 150 li
else:
    print("Reject null hypothesis because mean 150 does
```

```
accept Null Hypothesis Ho because mean 150 lies in 95% C
```

```
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mc
#hypothesis testing for last 50 rows assuming the mean
df9 = df.loc[0:49,['talk_time']]
Z9 = (df9.mean()-12)/(df9.std()/math.sqrt(50))
print("Z score value for")
print(Z9)
```

```
Z score value for
talk_time   -1.427102
dtype: float64
```

```
#assume 95% CI so significance level alpha = 0.06, now
#for P(Z1<-1.427)=0.076,
P1=2*0.076
if(P1>0.05):
    print("accept Null Hypothesis Ho because mean 150 l
else:
    print("Reject null hypothesis because mean 150 does
```

```
accept Null Hypothesis Ho because mean 150 lies in 95%
```

Run | Code

[55]:
```
import pandas as pd
import math
dp=pd.read_csv(r'C:\Users\mdzid\Documents\project303
```

[77]:
```
dp1=dp.loc[0:99,['sc_h']]
dp2=dp.loc[0:99,['talk_time']]
```

[78]:
```
print(dp1.std()*dp1.std())
print(dp2.std()*dp2.std())
```

```
sc_h    17.460202
dtype: float64
talk_time    29.561515
dtype: float64
```

[79]:
```
print(dp1.mean())
```

```
sc_h    12.88
dtype: float64
```

[80]:
```
print(dp2.mean())
```

```
talk_time    10.79
dtype: float64
```

[83]:
```
Zp=(12.84-10.74)/math.sqrt((17.48/100)+(29.42/100))
print(Zp)
```

```
3.0664287167039874
```

Secondly, hypothesis testing for heart-disease dataset features estimation

In [9]:
```
d2=numc.loc[0:49,['salary']]
print(d2.mean())
```

```
salary    45654.713542
dtype: float64
```

In [10]:
```
#hypothesis test for 1st 50 samples with known mean of 2nd 50 s
df2 = numc.loc[50:99,['salary']]
Z2 = (df2.mean()-45654.7)/(df2.std()/math.sqrt(50))
print("Z score value for")
print(Z2)
```

```
Z score value for
salary   -0.585164
dtype: float64
```

In [11]:
```
#assume 95% CI so significance level alpha = 0.05, now if P>alp
#for P(Z1<-0.585)=0.279,
P1=2*0.279
if(P1>0.05):
    print("accept Null Hypothesis Ho because mean 45654.8 lies
else:
    print("Reject null hypothesis because mean does not lies in
```

```
accept Null Hypothesis Ho because mean 45654.8 lies in 95% CI a
```

[3]:
```
import math
#hypothesis testing for 1st 50 rows assuming the mean is 50
df1 = df.loc[0:49,['age']]
Z1 = (df1.mean()-50)/(df1.std()/math.sqrt(50))
print("Z score value for")
print(Z1)
```

```
Z score value for
age   -1.868074
dtype: float64
```

[4]:
```
#assume 95% CI so significance level alpha = 0.05, now if P>
#for P(Z1<-1.868)=0.03,
P1=2*0.03
if(P1>0.05):
    print("accept Null Hypothesis Ho because mean 50 lies in
else:
    print("Reject null hypothesis because mean 50 does not l
```

```
accept Null Hypothesis Ho because mean 50 lies in 95% CI and
```

[5]:
```
df['age'].mean()
```

[5]:
```
49.58494572911751
```

6]:
```
import math
#hypothesis testing for mid 50 rows assuming the mean is 50
df2 = df.loc[2000:2049,['cigsPerDay']]
Z2 = (df2.mean()-15)/(df2.std()/math.sqrt(50))
print("Z score value for")
print(Z2)
```

```
Z score value for
cigsPerDay   -4.890207
dtype: float64
```

7]:
```
#assume 95% CI so significance level alpha = 0.05, now if P>al
#for P(Z1<-4.89)=0.0000002,
P1=2*0.0000002
if(P1>0.05):
    print("accept Null Hypothesis Ho because mean 15 lies in 9
else:
    print("Reject null hypothesis because mean 15 does not lie
```

]:
```
import math
#hypothesis testing for mid 50 rows assuming the mean is 236.7
df3 = df.loc[1000:1049,['totChol']]
Z3 = (df3.mean()-236.75)/(df3.std()/math.sqrt(50))
print("Z score value for")
print(Z3)
```

```
Z score value for
totChol   -1.174179
dtype: float64
```

]:
```
#assume 95% CI so significance level alpha = 0.05, now if P>al
#for P(Z1<-1.17)=0.12,
P1=2*0.12
if(P1>0.05):
    print("accept Null Hypothesis Ho because mean 236.75 lies
else:
    print("Reject null hypothesis because mean 236.75 does not
```

```
accept Null Hypothesis Ho because mean 236.75 lies in 95% CI a
```

[13]:
```
import math
#hypothesis testing for mid 50 rows assuming the mean is 132
df4 = df.loc[0:49,['sysBP']]
Z4 = (df4.mean()-132.35)/(df4.std()/math.sqrt(50))
print("Z score value for")
print(Z4)
```

```
Z score value for
sysBP    0.38854
dtype: float64
```

[14]:
```
#assume 95% CI so significance level alpha = 0.05, now if P>
#for P(Z1<0.389)=0.65,
P1=2*0.65
if(P1>0.05):
    print("accept Null Hypothesis Ho because mean 132.35 lie
else:
    print("Reject null hypothesis because mean does not lie
```

```
accept Null Hypothesis Ho because mean 132.35 lies in 95% CI
```

[15]:
```
df['diaBP'].mean()
```

:[15]:
```
82.89346389806512
```

[18]:
```
import math
#hypothesis testing for mid 35 rows assuming the mean is 83
df5 = df.loc[3000:3049,['diaBP']]
Z5 = (df5.mean()-83)/(df5.std()/math.sqrt(35))
print("Z score value for")
print(Z5)
```

```
Z score value for
diaBP   -0.556361
dtype: float64
```

[19]:
```
#assume 95% CI so significance level alpha = 0.05, now if P>alp
#for P(Z1<-0.556)=0.289,
P1=2*0.289
if(P1>0.05):
    print("accept Null Hypothesis Ho because popultion mean 83
else:
    print("Reject null hypothesis because mean does not lie in
```

```
accept Null Hypothesis Ho because popultion mean 83 lies in 95%
```

```
: df['BMI'].mean()
```

```
: 25.802007584735716
```

```
: import math
#hypothesis testing for last 40 rows assuming the mean is 83
df6 = df.loc[4210:4249,['BMI']]
Z6 = (df6.mean()-25.8)/(df6.std()/math.sqrt(40))
print("Z score value for")
print(Z6)
```

```
Z score value for
BMI    -0.605662
dtype: float64
```

```
: #assume 95% CI so significance level alpha = 0.05, now if P>alpha
#for P(Z1<-0.6)=0.274,
P1=2*0.274
if(P1>0.05):
    print("accept Null Hypothesis Ho because popultion mean 25.8 l
else:
    print("Reject null hypothesis because mean does not lie in 95%
```

```
accept Null Hypothesis Ho because popultion mean 25.8 lies in 95%
```

```
29]: df['heartRate'].mean()
```

```
29]: 75.87871637564889
```

```
33]: import math
#hypothesis testing for 50 rows assuming the mean is 75.87
df7 = df.loc[150:199,['heartRate']]
Z7 = (df7.mean()-75.87)/(df7.std()/math.sqrt(50))
print("Z score value for")
print(Z7)
```

```
Z score value for
heartRate    -0.828881
dtype: float64
```

```
34]: #assume 95% CI so significance level alpha = 0.05, now if P>alpha
#for P(Z1<-0.828)=0.2,
P1=2*0.2
if(P1>0.05):
    print("accept Null Hypothesis Ho because popultion mean 75.8
else:
    print("Reject null hypothesis because mean does not lie in 95
```

```
accept Null Hypothesis Ho because popultion mean 75.8 lies in 95%
```

Finally, hypothesis testing for final dataset great-customers features estimation

In [9]:
```
d2=numc.loc[0:49,['salary']]
print(d2.mean())
```

```
salary    45654.713542
dtype: float64
```

In [10]:
```
#hypothesis test for 1st 50 samples with known mean of 2nd 50 s
df2 = numc.loc[50:99,['salary']]
Z2 = (df2.mean()-45654.7)/(df2.std()/math.sqrt(50))
print("Z score value for")
print(Z2)
```

```
Z score value for
salary   -0.585164
dtype: float64
```

In [11]:
```
#assume 95% CI so significance level alpha = 0.05, now if P>alp
#for P(Z1<-0.585)=0.279,
P1=2*0.279
if(P1>0.05):
    print("accept Null Hypothesis Ho because mean 45654.8 lies
else:
    print("Reject null hypothesis because mean does not lies in
```

```
accept Null Hypothesis Ho because mean 45654.8 lies in 95% CI a
```

]:
```
d3=numc.loc[7500:7549,['works_hours']]
print(d3.mean())
```

```
works_hours    40.0
dtype: float64
```

]:
```
#hypothesis test for 2nd 50 samples with known mean of mid 50
df3 = numc.loc[50:99,['works_hours']]
Z3 = (df3.mean()-40)/(df3.std()/math.sqrt(50))
print("Z score value for")
print(Z3)
```

```
Z score value for
works_hours    0.823931
dtype: float64
```

]:
```
#assume 95% CI so significance level alpha = 0.05, now if P>c
#for P(Z1<0.82)=0.79,
P1=2*0.79
if(P1>0.05):
    print("accept Null Hypothesis Ho because mean 40 lies in
else:
    print("Reject null hypothesis because mean does not lies
```

```
accept Null Hypothesis Ho because mean 40 lies in 95% CI and
```

5]:
```
import math
t = df["tea_per_year"].median()
df["tea_per_year"].fillna(t, inplace = True)
c = df["coffee_per_year"].median()
df["coffee_per_year"].fillna(c, inplace = True)
dp1=df.loc[0:49,['tea_per_year']]
dp2=df.loc[0:49,['coffee_per_year']]
m1=dp1.mean()
m2=dp2.mean()
print(m1)
print(m2)
s1=dp1.std()*dp1.std()
s2=dp2.std()*dp2.std()
print(s1/50)
print(s2/50)
```

```
tea_per_year    225.26
dtype: float64
coffee_per_year    308.94
dtype: float64
tea_per_year    312.429233
dtype: float64
coffee_per_year    477.65911
dtype: float64
```

6]:
```
Z = (225.26-308.94)/math.sqrt(312.4 + 477.6)
print(Z)
```

```
-2.977200792180807
```

]:
```
#for P(Z<-2.97)=0.00148 or 2p=0.00296 which is not greater than alpha 0.05
```

T test: T test also does estimation like hypothesis testing but the difference is here we have different significance level critical value because degree of freedom of different sized of features. So at first we do T test for mobile-price-train data set

```
import pandas as pd
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.csv')
df1 = df.loc[0:15,['clock_speed']]
df2 = df.loc[1985:2000,['clock_speed']]
D=(abs(df1.mean()-df2.mean()))
S=(df1.std()/math.sqrt(16)+df2.std()/math.sqrt(16))
print("T value for 16 samples of")
print(D/S)
```

```
T value for 16 samples of
clock_speed    0.846059
dtype: float64
```

```
#Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.697 and two tailed 2.042 critical value
#Here T value < Crtitcal value for both tailed so we accept Null Hypothesis that the means are equal for battery-power
```

```
import pandas as pd
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.csv')
df1 = df.loc[0:50,['battery_power']]
df2 = df.loc[51:101,['battery_power']]
T=(abs(df1.mean()-df2.mean()))
S=(df1.std()/math.sqrt(51)+df2.std()/math.sqrt(51))
print("T value for 51 samples of")
print(T/S)
```

```
T value for 51 samples of
battery_power    0.638628
dtype: float64
```

```
#Degree of Freedom = (n1+n2-2)=51+51-2=100 at 95% CI for one tailed 1.66 and two tailed 1.98 critical value
#Here T value < Crtitcal value for both tailed so we accept Null Hypothesis that the means are equal for battery-power
```

[17]:
```
import pandas as pd
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_pric
df1 = df.loc[1000:1020,['int_memory']]
df2 = df.loc[1500:1520,['int_memory']]
D=(abs(df1.mean()-df2.mean()))
S=(df1.std()/math.sqrt(21)+df2.std()/math.sqrt(21))
print("T value for 21 samples of")
print(D/S)
```

```
T value for 21 samples of
int_memory    1.791956
dtype: float64
```

[18]:
```
#Degree of Freedom = (n1+n2-2)=21+21-2=40 at 95% CI for one tail
#Here T value < Crtitcal value for two tailed so we accept Null
#but T value > Critical value for one tailed so we reject null h
```

:
```
import pandas as pd
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.
df1 = df.loc[0:15,['m_dep']]
df2 = df.loc[1500:1515,['m_dep']]
D=(abs(df1.mean()-df2.mean()))
S=(df1.std()/math.sqrt(16)+df2.std()/math.sqrt(16))
print("T value for 16 samples of")
print(D/S)
```

```
T value for 16 samples of
m_dep    1.493828
dtype: float64
```

:
```
#Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.69
#Here T value < Crtitcal value for both tailed so we accept Null Hypothe
```

```
import pandas as pd
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.
df1 = df.loc[0:15,['mobile_wt']]
df2 = df.loc[1000:1015,['mobile_wt']]
D=(abs(df1.mean()-df2.mean()))
S=(df1.std()/math.sqrt(16)+df2.std()/math.sqrt(16))
print("T value for 16 samples of")
print(D/S)
```

```
T value for 16 samples of
mobile_wt    1.224674
dtype: float64
```

```
#Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.697
#Here T value < Crtitcal value for both tailed so we accept Null Hypothe
```

```
import pandas as pd
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.
df1 = df.loc[0:15,['mobile_wt']]
df2 = df.loc[1000:1015,['mobile_wt']]
D=(abs(df1.mean()-df2.mean()))
S=(df1.std()/math.sqrt(16)+df2.std()/math.sqrt(16))
print("T value for 16 samples of")
print(D/S)
```

```
T value for 16 samples of
mobile_wt    1.224674
dtype: float64
```

```
#Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.697
#Here T value < Crtitcal value for both tailed so we accept Null Hypothe
```

```
]: import pandas as pd
   import math
   df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.
   df1 = df.loc[0:15,['n_cores']]
   df2 = df.loc[100:115,['n_cores']]
   D=(abs(df1.mean()-df2.mean()))
   S=(df1.std()/math.sqrt(16)+df2.std()/math.sqrt(16))
   print("T value for 16 samples of")
   print(D/S)
```

```
T value for 16 samples of
n_cores    0.801392
dtype: float64
```

```
]: #Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.69
   #Here T value < Crtitcal value for both tailed so we accept Null Hypothe
```

```
]: import pandas as pd
   import math
   df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.cs
   df1 = df.loc[0:15,['pc']]
   df2 = df.loc[200:215,['pc']]
   D=(abs(df1.mean()-df2.mean()))
   S=(df1.std()/math.sqrt(16)+df2.std()/math.sqrt(16))
   print("T value for 16 samples of")
   print(D/S)
```

```
T value for 16 samples of
pc    0.33397
dtype: float64
```

```
]: #Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.697 a
   #Here T value < Crtitcal value for both tailed so we accept Null Hypothesi
```

```
import pandas as pd
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.csv'
df1 = df.loc[0:15,['px_height']]
df2 = df.loc[300:315,['px_height']]
D=(abs(df1.mean()-df2.mean()))
S=(df1.std()/math.sqrt(16)+df2.std()/math.sqrt(16))
print("T value for 16 samples of")
print(D/S)
```

```
T value for 16 samples of
px_height    0.054062
dtype: float64
```

```
#Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.697 and
#Here T value < Crtitcal value for both tailed so we accept Null Hypothesis
```

```
import pandas as pd
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.csv')
df1 = df.loc[0:15,['px_width']]
df2 = df.loc[400:415,['px_width']]
D=(abs(df1.mean()-df2.mean()))
S=(df1.std()/math.sqrt(16)+df2.std()/math.sqrt(16))
print("T value for 16 samples of")
print(D/S)
```

```
T value for 16 samples of
px_width    0.400984
dtype: float64
```

```
#Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.697 and
#Here T value < Crtitcal value for both tailed so we accept Null Hypothesis t
```

```
]: import pandas as pd
   import math
   df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.csv')
   df1 = df.loc[0:15,['ram']]
   df2 = df.loc[400:415,['ram']]
   D=(abs(df1.mean()-df2.mean()))
   S=(df1.std()/math.sqrt(16)+df2.std()/math.sqrt(16))
   print("T value for 16 samples of")
   print(D/S)
```

```
T value for 16 samples of
ram    1.197085
dtype: float64
```

```
]: #Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.697 and t
   #Here T value < Crtitcal value for both tailed so we accept Null Hypothesis th
```

```
]: import pandas as pd
   import math
   df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.
   df1 = df.loc[0:15,['talk_time']]
   df2 = df.loc[500:515,['talk_time']]
   D=(abs(df1.mean()-df2.mean()))
   S=(df1.std()/math.sqrt(16)+df2.std()/math.sqrt(16))
   print("T value for 16 samples of")
   print(D/S)
```

```
T value for 16 samples of
talk_time    0.902705
dtype: float64
```

```
]: #Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.69
   #Here T value < Crtitcal value for both tailed so we accept Null Hypothe
```

In [48]:
```
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_pric
df1 = df.loc[0:15,['sc_h']]
df2 = df.loc[0:15,['sc_w']]
A1=df1.mean()
A2=df2.mean()
S1=df1.std()/math.sqrt(16)
S2=df2.std()/math.sqrt(16)
print(A1)
print(A2)
print(S1)
print(S2)
```

```
sc_h    13.625
dtype: float64
sc_w    5.0625
dtype: float64
sc_h    1.087332
dtype: float64
sc_w    1.093423
dtype: float64
```

In [47]:
```
print('T value')
print((13.625-5)/(1.087+1.09))
```

```
T value
3.9618741387230134
```

In [ ]:
```
#for 95% CI in both cases for 30DF T value > Critical value so i
#we reject that mean of sc_h and sc_w is eqaul
```

T test for heart disease dataset features estimation

4]:
```
import pandas as pd
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\heart_disease.csv')
df1 = df.loc[0:15,['cigsPerDay']]
df2 = df.loc[16:31,['cigsPerDay']]
D=(abs(df1.mean()-df2.mean()))
S=(df1.std()/math.sqrt(16)+df2.std()/math.sqrt(16))
print("T value of")
print(D/S)
```

```
T value of
cigsPerDay    0.624316
dtype: float64
```

]:
```
#Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.697
#Here T value < Crtitcal value for both tailed so we accept Null Hypothe
```

[2]:
```
import pandas as pd
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\heart_disease.csv
df1 = df.loc[100:115,['age']]
df2 = df.loc[200:215,['age']]
D=(abs(df1.mean()-df2.mean()))
S=(df1.std()/math.sqrt(16)+df2.std()/math.sqrt(16))
print("T value of")
print(D/S)
```

```
T value of
age    0.335052
dtype: float64
```

[ ]:
```
#Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.6
#Here T value < Crtitcal value for both tailed so we accept Null Hypot
```

7]:
```
import pandas as pd
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\heart_disease.csv')
df1 = df.loc[0:15,['totChol']]
df2 = df.loc[26:41,['totChol']]
D=(abs(df1.mean()-df2.mean()))
S=(df1.std()/math.sqrt(16)+df2.std()/math.sqrt(16))
print("T value of")
print(D/S)
```

```
T value of
totChol    1.138214
dtype: float64
```

]:
```
#Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.697 a
#Here T value < Crtitcal value for both tailed so we accept Null Hypothesi
```

9]:
```
import pandas as pd
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\heart_disease.csv')
df1 = df.loc[90:105,['heartRate']]
df2 = df.loc[150:165,['heartRate']]
D=(abs(df1.mean()-df2.mean()))
S=(df1.std()/math.sqrt(16)+df2.std()/math.sqrt(16))
print("T value of")
print(D/S)
```

```
T value of
heartRate    0.854853
dtype: float64
```

]:
```
#Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.697
#Here T value < Crtitcal value for both tailed so we accept Null Hypothes
```

[20]:
```
import pandas as pd
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\heart_disease.csv')
df1 = df.loc[40:55,['BMI']]
df2 = df.loc[70:85,['BMI']]
D=(abs(df1.mean()-df2.mean()))
S=(df1.std()/math.sqrt(16)+df2.std()/math.sqrt(16))
print("T value of")
print(D/S)
```

```
T value of
BMI    2.064284
dtype: float64
```

[ ]:
```
#Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.697 a
#Here T value > Crtitcal value for both tailed so we reject Null Hypothesi
```

:
```
import pandas as pd
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\heart_disease.csv')
df1 = df.loc[80:95,['glucose']]
df2 = df.loc[165:180,['glucose']]
D=(abs(df1.mean()-df2.mean()))
S=(df1.std()/math.sqrt(16)+df2.std()/math.sqrt(16))
print("T value of")
print(D/S)
```

```
T value of
glucose    1.553548
dtype: float64
```

:
```
#Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.697
#Here T value < Crtitcal value for both tailed so we accept Null Hypothe
```

```
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\heart_disease.csv')
df1 = df.loc[50:65,['sysBP']]
df2 = df.loc[70:85,['diaBP']]
A1=df1.mean()
A2=df2.mean()
S1=df1.std()/math.sqrt(16)
S2=df2.std()/math.sqrt(16)
print(A1)
print(A2)
print(S1)
print(S2)
```

```
sysBP    132.6875
dtype: float64
diaBP    86.5625
dtype: float64
sysBP    4.320705
dtype: float64
diaBP    3.137766
dtype: float64
```

```
T=(132.7-86.56)/(4.32+3.14)
print('T value for two different Unpaired samples is: ')
print(T)
```

```
T value for two different Unpaired samples is:
6.18498659517426
```

```
#Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.697 a
#Here T value > Crtitcal value for both tailed so we reject Null Hypothesi
```

Finally, T test for great-customers data set features estimation

```
import pandas as pd
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\great_customers.csv')
df1 = df.loc[0:15,['age']]
df2 = df.loc[16:31,['age']]
D=(abs(df1.mean()-df2.mean()))
S=(df1.std()/math.sqrt(16)+df2.std()/math.sqrt(16))
print("T value of")
print(D/S)
```

```
T value of
age    0.427445
dtype: float64
```

```
#Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.697 c
#Here T value < Crtitcal value for both tailed so we accept Null Hypothesi
```

]:
```
import pandas as pd
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\great_customers.csv
df1 = df.loc[25:40,['salary']]
df2 = df.loc[200:215,['salary']]
D=(abs(df1.mean()-df2.mean()))
S=(df1.std()/math.sqrt(16)+df2.std()/math.sqrt(16))
print("T value of")
print(D/S)
```

```
T value of
salary    0.356227
dtype: float64
```

]:
```
#Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.697
#Here T value < Crtitcal value for both tailed so we accept Null Hypothe
```

```
import pandas as pd
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\great_customers.csv
df1 = df.loc[5:20,['works_hours']]
df2 = df.loc[120:135,['works_hours']]
D=(abs(df1.mean()-df2.mean()))
S=(df1.std()/math.sqrt(16)+df2.std()/math.sqrt(16))
print("T value of")
print(D/S)
```

```
T value of
works_hours    0.875184
dtype: float64
```

```
#Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.697
#Here T value < Crtitcal value for both tailed so we accept Null Hypothe
```

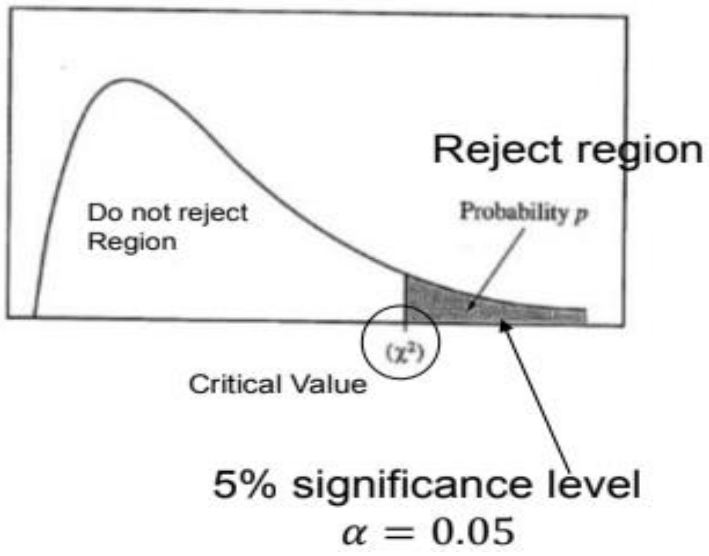
```
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\great_customers.csv
t = df["tea_per_year"].median()
df["tea_per_year"].fillna(t, inplace = True)
c = df["coffee_per_year"].median()
df["coffee_per_year"].fillna(c, inplace = True)
df1 = df.loc[500:515,['tea_per_year']]
df2 = df.loc[500:515,['coffee_per_year']]
A1=df1.mean()
A2=df2.mean()
S1=df1.std()/math.sqrt(16)
S2=df2.std()/math.sqrt(16)
print(A1)
print(A2)
print(S1)
print(S2)
```

```
tea_per_year    199.5625
dtype: float64
coffee_per_year    270.0625
dtype: float64
tea_per_year    19.398662
dtype: float64
coffee_per_year    44.009418
dtype: float64
```

```
T=(abs(199.56-270))/(19.4+44)
print(T) #print T value for different unpaired sample
```

```
1.1110410094637224
```

```
#Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.697
#Here T value < Crtitcal value for both tailed so we accept Null Hypothes
```



Chi Square Test : is the test of association in a feature consisting categorical values, to reveal the estimate we count the frequency of each categories then we select degree of Freedom accordingly and the expected values are always equally divided so if the frequencies are close to each other then we get good estmation from this test, here we do Chi square test for mobile-price-train dataset and we test the goodness of the features.
Null Hypothesis: Frequencies of features are Equal,
Alternative Hypothesis: At least one feature frequency differs from hypothesized value

```
]: import pandas as pd
   df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.csv')
   df['blue'].value_counts()
```

```
]: 0    1010
   1     990
   Name: blue, dtype: int64
```

```
]: exp = 1000
   ob1 = 1010
   ob2 = 990
   Cs = (ob1-exp)**2/exp +  (ob2-exp)**2/exp
   print('Chi sqaure value for blue')
   print(Cs)
```

```
   0.2
```

```
]: #here degree of freedom n-1=2-1=1 for 5% significance the crtitcal value is 3.841`
   #So ChiSq value < critical value so it is in the accepted region so its each proportion hypothesized value Ho
```

```
df['dual_sim'].value_counts()
```

```
1    1019
0     981
Name: dual_sim, dtype: int64
```

```
exp = 1000
ob1 = 1019
ob2 = 981
Cs = (ob1-exp)**2/exp +  (ob2-exp)**2/exp
print('Chi sqaure value for dual-sim')
print(Cs)
```

```
Chi sqaure value for dual-sim
0.722
```

```
#here degree of freedom n-1=2-1=1 for 5% significance the crtitcal value is 3.841`
#So ChiSq value < critical value so it is in the accepted region so its each propor
```

[10]:
```
df['four_g'].value_counts()
```

[10]:
```
1    1043
0     957
Name: four_g, dtype: int64
```

[11]:
```
exp = 1000
ob1 = 1043
ob2 = 957
Cs = (ob1-exp)**2/exp +  (ob2-exp)**2/exp
print('Chi sqaure value for 4G')
print(Cs)
```

```
Chi sqaure value for 4G
3.698
```

[ ]:
```
#here degree of freedom n-1=2-1=1 for 5% significance the cr
#So ChiSqr value < critical value so it is in the accepted r
```

In [13]:
```
df['three_g'].value_counts()
```

Out[13]:
```
1    1523
0     477
Name: three_g, dtype: int64
```

In [14]:
```
exp = 1000
ob1 = 1523
ob2 = 477
Cs = (ob1-exp)**2/exp +  (ob2-exp)**2/exp
print('Chi sqaure value for 3G')
print(Cs)
```

```
Chi sqaure value for 3G
547.058
```

In [ ]:
```
#here degree of freedom n-1=2-1=1 for 5% significance the crti
#So ChiSqr value > critical value so it is in the rejected reg
```

16]:
```
df['touch_screen'].value_counts()
```

16]:
```
1    1006
0     994
Name: touch_screen, dtype: int64
```

18]:
```
exp = 1000
ob1 = 1006
ob2 = 994
Cs = (ob1-exp)**2/exp + (ob2-exp)**2/exp
print('Chi sqaure value for touch-screen')
print(Cs)
```

```
Chi sqaure value for touch-screen
0.072
```

[ ]:
```
#here degree of freedom n-1=2-1=1 for 5% significance the crt
#So ChiSqr value < critical value so it is in the accepted re
```

```
0]: df['wifi'].value_counts()

0]: 1    1014
    0     986
    Name: wifi, dtype: int64
```

```
2]: exp = 1000
    ob1 = 1014
    ob2 = 986
    Cs = (ob1-exp)**2/exp + (ob2-exp)**2/exp
    print('Chi sqaure value for wi-fi')
    print(Cs)

    Chi sqaure value for wi-fi
    0.392
```

```
3]: #here degree of freedom n-1=2-1=1 for 5% significance the
    #So ChiSqr value < critical value so it is in the accepted
```

Chi square for heart-disease dataset estimation

```
3]: import pandas as pd
    df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\heart_disease.csv')
    df['male'].value_counts()
```

```
3]: 0    2419
    1    1819
    Name: male, dtype: int64
```

```
4]: exp = 2119
    ob1 = 2419
    ob2 = 1819
    Cs = (ob1-exp)**2/exp +  (ob2-exp)**2/exp
    print('Chi sqaure value for blue')
    print(Cs)
```

```
Chi sqaure value for blue
84.94572911750826
```

```
]: #here degree of freedom n-1=2-1=1 for 1% significance the crtitcal value is 6.635`
   #So ChiSqr value > critical value so it is in the rejected region and atleast one p
```

```
10]: df['currentSmoker'].value_counts()
```

```
10]: 0    2144
     1    2094
     Name: currentSmoker, dtype: int64
```

```
11]: exp = 2119
     ob1 = 2144
     ob2 = 2094
     Cs = (ob1-exp)**2/exp +  (ob2-exp)**2/exp
     print('Chi sqaure value for blue')
     print(Cs)
```

```
Chi sqaure value for blue
0.589900896649363
```

```
[ ]: #here degree of freedom n-1=2-1=1 for 5% significance the
     #So ChiSqr value < critical value so it is in the accepted
```

16]:
```
df['prevalentStroke'].value_counts()
```

16]:
```
0    4213
1      25
Name: prevalentStroke, dtype: int64
```

18]:
```
df['prevalentHyp'].value_counts()
```

18]:
```
0    2922
1    1316
Name: prevalentHyp, dtype: int64
```

19]:
```
exp = 2119
ob1 = 2922
ob2 = 1316
Cs = (ob1-exp)**2/exp +  (ob2-exp)**2/exp
print('Chi sqaure value for blue')
print(Cs)
```

```
Chi sqaure value for blue
608.5974516281265
```

21]:
```
#here degree of freedom n-1=2-1=1 for 5% significance the c
#So ChiSqr value > critical value so it is in the rejected
df['diabetes'].value_counts()
```

21]:
```
0    4129
1     109
Name: diabetes, dtype: int64
```

]:
```
df['education'] = df['education'].fillna(df['education'].mode()[0])
df['education'].value_counts()
```

]:
```
1.0    1825
2.0    1253
3.0     687
4.0     473
Name: education, dtype: int64
```

]:
```
exp = 1059.5
ob1 = 1825
ob2 = 1253
ob3 = 687
ob4 = 473
Cs = (ob1-exp)**2/exp + (ob2-exp)**2/exp + (ob3-exp)**2/exp +  (ob4-exp)**2/exp
print('Chi sqaure value for blue')
print(Cs)
```

```
Chi sqaure value for blue
1044.0500235960358
```

]:
```
#here degree of freedom n-1=4-1=3 for 5% significance the crtitcal value is 7.815`
#So ChiSqr value > critical value so it is in the rejected region and atleast one proportion differ
```

```
0    4129
1     109
Name: diabetes, dtype: int64
```

```
df['TenYearCHD'].value_counts()
```

```
0    3594
1     644
Name: TenYearCHD, dtype: int64
```

```
exp = 2119
ob1 = 3594
ob2 = 644
Cs = (ob1-exp)**2/exp +  (ob2-exp)**2/exp
print('Chi sqaure value for blue')
print(Cs)
```

```
Chi sqaure value for blue
2053.4450212364322
```

```
#here degree of freedom n-1=2-1=1 for 5% significance the cr
#So ChiSqr value > critical value so it is in the rejected r
```

Chi square test for great-customers dataset feature estimations.

```
import pandas as pd
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\great_customers.csv')
df.drop_duplicates(inplace=True)
df['workclass'] = df['workclass'].fillna(df['workclass'].mode()[0])
df['workclass'].value_counts()
```

```
private          6500
government       1127
self_employed     972
Name: workclass, dtype: int64
```

```
exp = 2866.33
ob1 = 6500
ob2 = 1127
ob3 = 972
Cs = (ob1-exp)**2/exp + (ob2-exp)**2/exp + (ob3-exp)**2/exp
print('Chi sqaure value for blue')
print(Cs)
```

```
Chi sqaure value for blue
6913.828019348784
```

```
#here degree of freedom n-1=2-1=1 for 5% significance the crtitcal value is 3.841
#So ChiSqr value > critical value so it is in the rejected region and atleast one
```

```
import pandas as pd
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\great_customers.csv')
df.drop_duplicates(inplace=True)
df['marital-status'] = df['marital-status'].fillna(df['marital-status'].mode()[0]
df2 = df.loc[0:99,['marital-status']]
df2['marital-status'].value_counts()
```

```
Never-married    41
Married          38
Divorced         19
Widowed           2
Name: marital-status, dtype: int64
```

```
exp = 25
ob1 = 41
ob2 = 38
ob3 = 19
ob4 = 2
Cs = (ob1-exp)**2/exp + (ob2-exp)**2/exp + (ob3-exp)**2/exp +  (ob4-exp)**2/exp
print('Chi sqaure value for first 100 samples of marital-status')
print(Cs)
```

```
Chi sqaure value for first 100 samples of marital-status
39.6
```

```
#here degree of freedom n-1=4-1=3 for 5% significance the crtitcal value is 7.815
#So ChiSqr value > critical value so it is in the rejected region and atleast one
```

```
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\great_cust
df.drop_duplicates(inplace=True)
df['sex'].value_counts()
```

```
Male      5597
Female    3002
Name: sex, dtype: int64
```

```
exp = 4299.5
ob1 = 5597
ob2 = 3002
Cs = (ob1-exp)**2/exp +  (ob2-exp)**2/exp
print('Chi sqaure value for blue')
print(Cs)
```

```
Chi sqaure value for blue
783.1172229328992
```

```
#here degree of freedom n-1=2-1=1 for 5% significance the crtit
#So ChiSqr value > critical value so it is in the rejected regi
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\great_cust
df.drop_duplicates(inplace=True)
df['great_customer_class'].value_counts()
```

```
0    7431
1    1168
Name: great_customer_class, dtype: int64
```

```
#too much variance from expected value 4299.5 means greater Chi
```

```
df['occupation'].value_counts()
```

t[39]:

```
craft            1657
clerical          999
professional      984
service           973
executive         953
sales             920
factory           555
trucker           414
cleaner           399
farm              301
tech              234
lawenf            161
estate_agent       47
soldier             2
Name: occupation, dtype: int64
```

[40]:

```
df.drop_duplicates(inplace=True)
df['race'].value_counts()
```

t[40]:

```
caucasian        7313
not_caucasian    1286
Name: race, dtype: int64
```

[41]:

```
exp = 4299.5
ob1 = 7313
ob2 = 1286
Cs = (ob1-exp)**2/exp +  (ob2-exp)**2/exp
print('Chi sqaure value for blue')
print(Cs) # chi sqr value cant be this huge
```

```
Chi sqaure value for blue
4224.2968949877895
```

Discussion :

From above, we found out at preprocessing that mobile-price-train dataset had the best data hence it produced the best outcomes and heavy preprocessing was needed in heart disease and great customers dataset before doing data analysis and hypothesis testing.

While, plotting the histograms of outlier features the shape is positively skewed means mean > median also it is not normally distributed hence we mostly don't find suitable result, also in scatter plot we can not find correlation

Also while plotting the bar charts for the features we see the charts are almost on same height gave best estimation in Chi Square test. And the normally distributed features has the best estimation in T test and here most of the features passed null hypothesis which states that our data set become good after preprocessing means our way of data analysis and hypothesis testing is correct hopefully.

Conclusion :

in the end, it was a great experience to analyze the dataset with different techniques that our honorable teachers taught us, we feel really happy to be able to complete this project although it is not possible to complete the project perfectly because of time limitations for which there may be some mistakes possible but on the contrary we hope our teachers will like our project and value our efforts. Thank you.

Necessary python codes used for the project:

```
 import pandas as pd
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.csv')

print(df.info())

print('Num of Columns: ', df.shape[1])
print('Num of Rows: ', df.shape[0])
#%%
import pandas as pd
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.csv')

print("Showing amount of Null values")
print(df.isnull().sum())
print(df.isnull().any())

print("Showing amount of duplicate values")
print(df.duplicated().sum())
df.drop_duplicates(inplace=True)
#%%
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\heart_disease.csv')

print(df.info())

print('Num of Columns: ', df.shape[1])
print('Num of Rows: ', df.shape[0])
#%%
import pandas as pd

df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\heart_disease.csv')

print("Showing amount of Null values")
print(df.isnull().sum())
sns.heatmap(df.isna())
plt.show()

print("Showing amount of duplicate values")
print(df.duplicated().sum())
df.drop_duplicates(inplace=True)
#%%
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\heart_disease.csv')

df['education'] = df['education'].fillna(df['education'].mode()[0])
```

```
c = df["cigsPerDay"].median()
df["cigsPerDay"].fillna(c, inplace = True)

b = df["BPMeds"].fillna(method='ffill')
df["BPMeds"].fillna(b, inplace = True)

t = df["totChol"].median()
df["totChol"].fillna(t, inplace = True)

bm = df["BMI"].mean()
df["BMI"].fillna(bm, inplace = True)

h = df["heartRate"].median()
df["heartRate"].fillna(h, inplace = True)

g = df["glucose"].median()
df["glucose"].fillna(g, inplace = True)

#%%
import pandas as pd
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\great_customers.csv')

print(df.info())

print('Num of Columns: ', df.shape[1])
print('Num of Rows: ', df.shape[0])
#%%
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\great_customers.csv')

print("Showing amount of Null values")
print(df.isnull().sum())
sns.heatmap(df.isna())
plt.show()

print("Showing amount of duplicate values")
print(df.duplicated().sum())
df.drop_duplicates(inplace=True)
#%%
import pandas as pd
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\great_customers.csv')

g = df["age"].median()
df["age"].fillna(g, inplace = True)

df['workclass'] = df['workclass'].fillna(df['workclass'].mode()[0])

s = df["salary"].mean()
```

```
df["salary"].fillna(s, inplace = True)

df['occupation'] = df['occupation'].fillna(df['occupation'].mode()[0])

df["mins_beerdrinking_year"].fillna(0, inplace = True)
df["mins_exercising_year"].fillna(0,inplace=True)

t = df["tea_per_year"].median()
df["tea_per_year"].fillna(t, inplace = True)

c = df["coffee_per_year"].median()
df["coffee_per_year"].fillna(c, inplace = True)
print("amount of null rows")
print(df.isnull().sum())
#%%
import pandas as pd
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\great_customers.csv')
catg = df.select_dtypes(object)
numc = df.select_dtypes(["int64","float64"])
print(catg.info())
print(numc.info())

catg["workclass"].replace("government",0,inplace=True)
catg["workclass"].replace("private",1,inplace=True)
catg["workclass"].replace("self_employed",2,inplace=True)

catg["marital-status"].replace("Divorced",0,inplace=True)
catg["marital-status"].replace("Married",1,inplace=True)
catg["marital-status"].replace("Never-married",2,inplace=True)
catg["marital-status"].replace("Widowed",3,inplace=True)

catg["race"].replace("caucasian",0,inplace=True)
catg["race"].replace("not_caucasian",1,inplace=True)

catg["sex"].replace("Male",0,inplace=True)
catg["sex"].replace("Female",1,inplace=True)

from sklearn.preprocessing import LabelEncoder
ob = LabelEncoder()
ob.fit_transform(catg["occupation"])

for col in catg:
    ob=LabelEncoder()
    catg[col]=ob.fit_transform(catg[col])
print(catg.info())
print(catg["occupation"].value_counts())
print(catg.head())
#%%
import matplotlib.pyplot as plt
numc.hist(column='sysBP',bins=10)
plt.title('edurank histogram')
```

```
plt.show()

eda:
import pandas as pd
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.csv')
df2 = df.loc[0:9,['battery_power']]
print(df2)
#%%
   A = df['battery_power']
   print('For bp: ')
   print('Q1: ', A.quantile(0.25))
   print('Q3: ', A.quantile(0.75))
   print('Median: ', A.quantile(0.5))

   IQR_A =  (A.quantile(0.75) - A.quantile(0.25))
   print('IQR for Amount: ', IQR_A)
   print('Lower fench: ', A.quantile(0.25)-(IQR_A * 1.5))
   print('Upper fench: ', A.quantile(0.75)+(IQR_A * 1.5))
   a_max = A.max()
   a_min = A.min()
   print('Minimum: ', a_min)
   print('Maximum: ', a_max)
   if (a_min < (A.quantile(0.25)-(IQR_A*1.5))):
      print("There are Outliers")
   elif(a_max > (A.quantile(0.75)+(IQR_A * 1.5))):
      print('There are right outliers ')
   else:
      print('There are no outliers in bp. \n')

import matplotlib.pyplot as plt

plt.boxplot(x="battery_power", data=df)
plt.title("battery-power boxplot")
plt.ylabel("battery-power")
plt.show()
#%%
  C = df['clock_speed']
   print('For bp: ')
   print('Q1: ', C.quantile(0.25))
   print('Q3: ', C.quantile(0.75))
   print('Median: ', C.quantile(0.5))

   IQR_C =  (C.quantile(0.75) - C.quantile(0.25))
   print('IQR for Amount: ', IQR_C)
   print('Lower fench: ', C.quantile(0.25)-(IQR_C * 1.5))
   print('Upper fench: ', C.quantile(0.75)+(IQR_C * 1.5))
   c_max = C.max()
   c_min = C.min()
   print('Minimum: ', c_min)
   print('Maximum: ', c_max)
   if (c_min < (C.quantile(0.25)-(IQR_C*1.5))):
```

```
    print("There are Outliers")
  elif(c_max > (C.quantile(0.75)+(IQR_C * 1.5))):
    print('There are right outliers ')
  else:
    print('There are no outliers in clock. \n')

import matplotlib.pyplot as plt

plt.boxplot(x="clock_speed", data=df)
plt.title("clock-speed plot")
plt.ylabel("clock-speed")
plt.show()
#%%
  I = df['int_memory']
  print('For mem: ')
  print('Q1: ', I.quantile(0.25))
  print('Q3: ', I.quantile(0.75))
  print('Median: ', I.quantile(0.5))

  IQR_I =  (I.quantile(0.75) - I.quantile(0.25))
  print('IQR for mem: ', IQR_I)
  print('Lower fench: ', I.quantile(0.25)-(IQR_I * 1.5))
  print('Upper fench: ', I.quantile(0.75)+(IQR_I * 1.5))
  i_max = I.max()
  i_min = I.min()
  print('Minimum: ', i_min)
  print('Maximum: ', i_max)
  if (i_min < (I.quantile(0.25)-(IQR_I*1.5))):
    print("There are Outliers")
  elif(i_max > (I.quantile(0.75)+(IQR_I * 1.5))):
    print('There are right outliers ')
  else:
    print('There are no outliers in mem. \n')

import matplotlib.pyplot as plt

plt.boxplot(x="int_memory", data=df)
plt.title("int-memory plot")
plt.ylabel("int-memory")
plt.show()
#%%
  M = df['mobile_wt']
  print('For mwt: ')
  print('Q1: ', M.quantile(0.25))
  print('Q3: ', M.quantile(0.75))
  print('Median: ', M.quantile(0.5))

  IQR_M =  (M.quantile(0.75) - M.quantile(0.25))
  print('IQR for mem: ', IQR_M)
  print('Lower fench: ', M.quantile(0.25)-(IQR_M * 1.5))
  print('Upper fench: ', M.quantile(0.75)+(IQR_M * 1.5))
```

```
    m_max = M.max()
    m_min = M.min()
    print('Minimum: ', m_min)
    print('Maximum: ', m_max)
    if (m_min < (M.quantile(0.25)-(IQR_M*1.5))):
        print("There are Outliers")
    elif(m_max > (M.quantile(0.75)+(IQR_M * 1.5))):
        print('There are right outliers ')
    else:
        print('There are no outliers in mem. \n')

plt.boxplot(x="mobile_wt", data=df)
plt.title("mobile-weight plot")
plt.ylabel("mobwt")
plt.show()
#%%
    N = df['n_cores']
    print('For core: ')
    print('Q1: ', N.quantile(0.25))
    print('Q3: ', N.quantile(0.75))
    print('Median: ', N.quantile(0.5))

    IQR_N =  (N.quantile(0.75) - N.quantile(0.25))
    print('IQR for mem: ', IQR_N)
    print('Lower fench: ', N.quantile(0.25)-(IQR_N * 1.5))
    print('Upper fench: ', N.quantile(0.75)+(IQR_N * 1.5))
    n_max = N.max()
    n_min = N.min()
    print('Minimum: ', n_min)
    print('Maximum: ', n_max)
    if (n_min < (N.quantile(0.25)-(IQR_N*1.5))):
        print("There are Outliers")
    elif(n_max > (N.quantile(0.75)+(IQR_N * 1.5))):
        print('There are right outliers ')
    else:
        print('There are no outliers in core. \n')

import matplotlib.pyplot as plt

plt.boxplot(x="n_cores", data=df)
plt.title("ncore plot")
plt.ylabel("core")
plt.show()
#%%
    P = df['px_height']
    print('For height: ')
    print('Q1: ', P.quantile(0.25))
    print('Q3: ', P.quantile(0.75))
    print('Median: ', P.quantile(0.5))

    IQR_P =  (P.quantile(0.75) - P.quantile(0.25))
```

```
    print('IQR for mem: ', IQR_P)
    print('Lower fench: ', P.quantile(0.25)-(IQR_P * 1.5))
    print('Upper fench: ', P.quantile(0.75)+(IQR_P * 1.5))
    p_max = P.max()
    p_min = P.min()
    print('Minimum: ', p_min)
    print('Maximum: ', p_max)
    if (p_min < (P.quantile(0.25)-(IQR_P*1.5))):
        print("There are Outliers")
    elif(p_max > (P.quantile(0.75)+(IQR_P * 1.5))):
        print('There are right outliers ')
    else:
        print('There are no outliers in core. \n')

import matplotlib.pyplot as plt

plt.boxplot(x="px_height", data=df)
plt.title("px-height plot")
plt.ylabel("px-height")
plt.show()
#%%
#now that there are outler so data preprocessing needed
for x in df.index:
  if df.loc[x, "px_height"] > 1944:
    df.loc[x, "px_height"] = 564
for x in df.index:
  if df.loc[x, "px_height"]<-714:
    df.loc[x, "px_height"] = 564

import matplotlib.pyplot as plt

plt.boxplot(x="px_height", data=df)
plt.title("px-height plot")
plt.ylabel("px-height")
plt.show()
#%%
    R = df['ram']
    print('For ram: ')
    print('Q1: ', R.quantile(0.25))
    print('Q3: ', R.quantile(0.75))
    print('Median: ', R.quantile(0.5))

    IQR_R =  (R.quantile(0.75) - R.quantile(0.25))
    print('IQR for mem: ', IQR_R)
    print('Lower fench: ', R.quantile(0.25)-(IQR_R * 1.5))
    print('Upper fench: ', R.quantile(0.75)+(IQR_R * 1.5))
    r_max = R.max()
    r_min = R.min()
    print('Minimum: ', r_min)
    print('Maximum: ', r_max)
    if (r_min < (R.quantile(0.25)-(IQR_R*1.5))):
```

```
    print("There are Outliers")
  elif(r_max > (R.quantile(0.75)+(IQR_R * 1.5))):
    print('There are right outliers ')
  else:
    print('There are no outliers in ram. \n')

import matplotlib.pyplot as plt

plt.boxplot(x="ram", data=df)
plt.title("ram-plot")
plt.ylabel("ram")
plt.show()
#%%
  T = df['talk_time']
  print('For talk: ')
  print('Q1: ', T.quantile(0.25))
  print('Q3: ', T.quantile(0.75))
  print('Median: ', T.quantile(0.5))

  IQR_T =  (T.quantile(0.75) - T.quantile(0.25))
  print('IQR for mem: ', IQR_T)
  print('Lower fench: ', T.quantile(0.25)-(IQR_T * 1.5))
  print('Upper fench: ', T.quantile(0.75)+(IQR_T * 1.5))
  t_max = T.max()
  t_min = T.min()
  print('Minimum: ', t_min)
  print('Maximum: ', t_max)
  if (t_min < (T.quantile(0.25)-(IQR_T*1.5))):
    print("There are Outliers")
  elif(t_max > (T.quantile(0.75)+(IQR_T * 1.5))):
    print('There are right outliers ')
  else:
    print('There are no outliers in talktime. \n')

import matplotlib.pyplot as plt

plt.boxplot(x="talk_time", data=df)
plt.title("talktime plot")
plt.ylabel("talktime")
plt.show()
#%%
import matplotlib.pyplot as plt

df["dual_sim"].value_counts().plot(kind='bar', color=['green','red'])
plt.title("Dual Sim Bar Chart")
plt.xlabel("1=yes , 0=no")
plt.ylabel("mobile")
plt.show()

import matplotlib.pyplot as plt
```

```
df["four_g"].value_counts().plot(kind='bar', color=['green','red'])
plt.title("4G Bar Chart")
plt.xlabel("1=yes , 0=no")
plt.ylabel("mobile")
plt.show()

import matplotlib.pyplot as plt

df["three_g"].value_counts().plot(kind='bar', color=['green','red'])
plt.title("3G Bar Chart")
plt.xlabel("1=yes , 0=no")
plt.ylabel("mobile")
plt.show()

import matplotlib.pyplot as plt

df["price_range"].value_counts().plot(kind='bar', color=['blue','yellow','red','green'])
plt.title("price-range Bar Chart")
plt.ylabel("mobile-train")
plt.show()
#%%
import matplotlib.pyplot as plt
import seaborn as sns
sns.countplot(x='touch_screen', data=df, hue='four_g')
plt.title("Having touch screen with 4G")
plt.xlabel("1=yes,0=no")
plt.ylabel("mobile-train")
plt.legend(['yes','no'])

import matplotlib.pyplot as plt
import seaborn as sns
sns.countplot(x='dual_sim', data=df, hue='three_g')
plt.title("Having dual sim with 3G")
plt.xlabel("1=yes,0=no")
plt.ylabel("mobile-train")
plt.legend(['yes','no'])
#%%
import seaborn as sns
sns.displot(x='px_height',data=df,bins=30,kde=True)
plt.title('px-height vs mobile')
plt.ylabel("mobile-train")
plt.show()
#%%
import matplotlib.pyplot as plt

plt.scatter(df.mobile_wt[df.touch_screen==1],
        df.px_width[df.touch_screen==1] , c='blue')
#no-touchscreen
plt.scatter(df.mobile_wt[df.touch_screen==0],
        df.px_width[df.touch_screen==0] , c='red')
plt.title('mobile-wt vs px_width')
```

```
plt.xlabel("mobile-wt")
plt.ylabel("px-width")
plt.legend(['touch screen','NO touch screen'])
#%%
import matplotlib.pyplot as plt
df.hist(column='fc',bins=30)
plt.title('fc histogram')
plt.show()

eda2
import pandas as pd
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\heart_disease.csv')
#%%
import matplotlib.pyplot as plt

plt.boxplot(x="age", data=df)
plt.title("Age boxplot")
plt.ylabel("age")
plt.show()
#%%
  C = df['cigsPerDay']
    print('For cpd: ')
    print('Q1: ', C.quantile(0.25))
    print('Q3: ', C.quantile(0.75))
    print('Median: ', C.quantile(0.5))

    IQR_C =  (C.quantile(0.75) - C.quantile(0.25))
    print('IQR for Amount: ', IQR_C)
    print('Lower fench: ', C.quantile(0.25)-(IQR_C * 1.5))
    print('Upper fench: ', C.quantile(0.75)+(IQR_C * 1.5))
    c_max = C.max()
    c_min = C.min()
    print('Minimum: ', c_min)
    print('Maximum: ', c_max)
    if (c_min < (C.quantile(0.25)-(IQR_C*1.5))):
       print("There are Outliers")
    elif(c_max > (C.quantile(0.75)+(IQR_C * 1.5))):
       print('There are right outliers ')
    else:
       print('There are no outliers in cpd. \n')

for x in df.index:
  if df.loc[x, "cigsPerDay"] > 50:
    df.loc[x, "cigsPerDay"] = 20
import matplotlib.pyplot as plt

plt.boxplot(x="cigsPerDay", data=df)
plt.title("cpd plot")
plt.ylabel("cigs/day")
plt.show()
```

```
#%%
   M = df['sysBP']
   print('For SystBPP: ')
   print('Q1: ', M.quantile(0.25))
   print('Q3: ', M.quantile(0.75))
   print('Median: ', M.quantile(0.5))

   IQR_M =  (M.quantile(0.75) - M.quantile(0.25))
   print('IQR for mem: ', IQR_M)
   print('Lower fench: ', M.quantile(0.25)-(IQR_M * 1.5))
   print('Upper fench: ', M.quantile(0.75)+(IQR_M * 1.5))
   m_max = M.max()
   m_min = M.min()
   print('Minimum: ', m_min)
   print('Maximum: ', m_max)
   if (m_min < (M.quantile(0.25)-(IQR_M*1.5))):
      print("There are Outliers")
   elif(m_max > (M.quantile(0.75)+(IQR_M * 1.5))):
      print('There are UPR outliers ')
   else:
      print('There are no outliers in SBP. \n')

for x in df.index:
  if df.loc[x, "sysBP"] > 184.5:
    df.loc[x, "sysBP"] = 128
import matplotlib.pyplot as plt

plt.boxplot(x="sysBP", data=df)
plt.title("Systolic Cholesterol plot")
plt.ylabel("Cholesterol")
plt.show()
#%%
   N = df['heartRate']
   print('For rate: ')
   print('Q1: ', N.quantile(0.25))
   print('Q3: ', N.quantile(0.75))
   print('Median: ', N.quantile(0.5))

   IQR_N =  (N.quantile(0.75) - N.quantile(0.25))
   print('IQR for mem: ', IQR_N)
   print('Lower fench: ', N.quantile(0.25)-(IQR_N * 1.5))
   print('Upper fench: ', N.quantile(0.75)+(IQR_N * 1.5))
   n_max = N.max()
   n_min = N.min()
   print('Minimum: ', n_min)
   print('Maximum: ', n_max)
   if (n_min < (N.quantile(0.25)-(IQR_N*1.5))):
      print("There are low Outliers")
   elif(n_max > (N.quantile(0.75)+(IQR_N * 1.5))):
      print('There are upr outliers ')
   else:
```

```
      print('There are no outliers in core. \n')

 for x in df.index:
     if df.loc[x, "heartRate"] > 105 or df.loc[x, "heartRate"] < 46:
       df.loc[x, "heartRate"] = 75
import matplotlib.pyplot as plt

plt.boxplot(x="heartRate", data=df)
plt.title("heartRate plot")
plt.ylabel("rate")
plt.show()
#%%
   R = df['BMI']
   print('For Bmi: ')
   print('Q1: ', R.quantile(0.25))
   print('Q3: ', R.quantile(0.75))
   print('Median: ', R.quantile(0.5))

   IQR_R =  (R.quantile(0.75) - R.quantile(0.25))
   print('IQR for mem: ', IQR_R)
   print('Lower fench: ', R.quantile(0.25)-(IQR_R * 1.5))
   print('Upper fench: ', R.quantile(0.75)+(IQR_R * 1.5))
   r_max = R.max()
   r_min = R.min()
   print('Minimum: ', r_min)
   print('Maximum: ', r_max)
   if (r_min < (R.quantile(0.25)-(IQR_R*1.5))):
      print("There are Outliers")
   elif(r_max > (R.quantile(0.75)+(IQR_R * 1.5))):
      print('There are right outliers ')
   else:
      print('There are no outliers in bmi. \n')

 for x in df.index:
     if df.loc[x, "BMI"] > 35.5 or df.loc[x, "BMI"] < 15.65:
       df.loc[x, "BMI"] = 25.4
import matplotlib.pyplot as plt

plt.boxplot(x="BMI", data=df)
plt.title("BMI-plot")
plt.ylabel("bmi")
plt.show()
#%%
   T = df['totChol']
   print('For cholesterol: ')
   print('Q1: ', T.quantile(0.25))
   print('Q3: ', T.quantile(0.75))
   print('Median: ', T.quantile(0.5))

   IQR_T =  (T.quantile(0.75) - T.quantile(0.25))
   print('IQR for mem: ', IQR_T)
```

```
    print('Lower fench: ', T.quantile(0.25)-(IQR_T * 1.5))
    print('Upper fench: ', T.quantile(0.75)+(IQR_T * 1.5))
    t_max = T.max()
    t_min = T.min()
    print('Minimum: ', t_min)
    print('Maximum: ', t_max)
    if (t_min < (T.quantile(0.25)-(IQR_T*1.5))):
        print("There are Outliers")
    elif(t_max > (T.quantile(0.75)+(IQR_T * 1.5))):
        print('There are right outliers ')
    else:
        print('There are no outliers in totchol. \n')

  for x in df.index:
      if df.loc[x, "totChol"] > 346 or df.loc[x, "totChol"] < 122:
          df.loc[x, "totChol"] = 234

import matplotlib.pyplot as plt

plt.boxplot(x="totChol", data=df)
plt.title("totChol plot")
plt.ylabel("cholesterol")
plt.show()
#%%
import matplotlib.pyplot as plt

df["male"].value_counts().plot(kind='bar', color=['green','red'])
plt.title("male or not Bar Chart")
plt.xlabel("1=males , 0=female")
plt.ylabel("gender")
plt.show()

import matplotlib.pyplot as plt

df["prevalentHyp"].value_counts().plot(kind='bar', color=['green','red'])
plt.title("HyperTension Chart")
plt.xlabel("1=yes , 0=no")
plt.ylabel("person")
plt.show()

import matplotlib.pyplot as plt

df["TenYearCHD"].value_counts().plot(kind='bar', color=['green','red'])
plt.title("CHD test chart")
plt.xlabel("1=yes , 0=no")
plt.ylabel("Person")
plt.show()

import matplotlib.pyplot as plt

df["education"].value_counts().plot(kind='bar', color=['blue','yellow','red','green'])
```

```
plt.title("edu level Chart")
plt.ylabel("person")
plt.show()

df.BPMeds.value_Counts().plot(kind='pie',figsize=(10,5),autopct='%1.2f%%')
plt.legend([BPMeds, NoBPMeds])
#%%
import matplotlib.pyplot as plt
import seaborn as sns
sns.countplot(x='prevalentHyp', data=df, hue='male')
plt.title("Gender having Stroke")
plt.xlabel("1=yes,0=no")
plt.ylabel("person")
plt.legend(['males','female'])

import matplotlib.pyplot as plt
import seaborn as sns
sns.countplot(x='TenYearCHD', data=df, hue='currentSmoker')
plt.title("person having PrevCHD")
plt.xlabel("1=yes,0=no")
plt.ylabel("person")
plt.legend(['smoker','non-smoker'])
#%%
import seaborn as sns
sns.displot(x='age',data=df,bins=30,kde=True)
plt.title('age of person')
plt.ylabel("person")
plt.show()
#%%
import matplotlib.pyplot as plt

plt.scatter(df.totChol[df.TenYearCHD==1],
        df.sysBP[df.TenYearCHD==1] , c='blue')
#no-touchscreen
plt.scatter(df.totChol[df.TenYearCHD==0],
        df.sysBP[df.TenYearCHD==0] , c='red')
plt.title('totcol vs systolicBP')
plt.xlabel("cholesterol")
plt.ylabel("sysBP")
plt.legend(['10yrCHD','NO'])
#%%
import matplotlib.pyplot as plt
df.hist(column='glucose',bins=30)
plt.title('glucose histogram')
plt.show()
eda3
import pandas as pd
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\great_customers.csv')

  A = numc['age']
  print('For age: ')
```

```
  print('Q1: ', A.quantile(0.25))
  print('Q3: ', A.quantile(0.75))
  print('Median: ', A.quantile(0.5))

  IQR_A =  (A.quantile(0.75) - A.quantile(0.25))
  print('IQR for Amount: ', IQR_A)
  print('Lower fench: ', A.quantile(0.25)-(IQR_A * 1.5))
  print('Upper fench: ', A.quantile(0.75)+(IQR_A * 1.5))
  a_max = A.max()
  a_min = A.min()
  print('Minimum: ', a_min)
  print('Maximum: ', a_max)
  if (a_min < (A.quantile(0.25)-(IQR_A*1.5))):
     print("There are Outliers")
  elif(a_max > (A.quantile(0.75)+(IQR_A * 1.5))):
     print('There are right outliers ')
  else:
     print('There are no outliers in age. \n')

import matplotlib.pyplot as plt

plt.boxplot(x="age", data=numc)
plt.title("Systolic Cholesterol plot")
plt.ylabel("Cholesterol")
plt.show()
#%%
   A = numc['salary']
   print('For salary: ')
   print('Q1: ', A.quantile(0.25))
   print('Q3: ', A.quantile(0.75))
   print('Median: ', A.quantile(0.5))

   IQR_A =  (A.quantile(0.75) - A.quantile(0.25))
   print('IQR for Amount: ', IQR_A)
   print('Lower fench: ', A.quantile(0.25)-(IQR_A * 1.5))
   print('Upper fench: ', A.quantile(0.75)+(IQR_A * 1.5))
   a_max = A.max()
   a_min = A.min()
   print('Minimum: ', a_min)
   print('Maximum: ', a_max)
   if (a_min < (A.quantile(0.25)-(IQR_A*1.5))):
      print("There are low Outliers")
   elif(a_max > (A.quantile(0.75)+(IQR_A * 1.5))):
      print('There are upr outliers ')
   else:
      print('There are no outliers in salary. \n')
   for x in numc.index:
      if numc.loc[x, "salary"] > 70940.5:
         numc.loc[x, "salary"] = 21000

plt.boxplot(x="salary", data=numc)
```

```
plt.title("Systolic Cholesterol plot")
plt.ylabel("Cholesterol")
plt.show()
#%%
  A = numc['coffee_per_year']
  print('For education: ')
  print('Q1: ', A.quantile(0.25))
  print('Q3: ', A.quantile(0.75))
  print('Median: ', A.quantile(0.5))

  IQR_A =  (A.quantile(0.75) - A.quantile(0.25))
  print('IQR for Amount: ', IQR_A)
  print('Lower fench: ', A.quantile(0.25)-(IQR_A * 1.5))
  print('Upper fench: ', A.quantile(0.75)+(IQR_A * 1.5))
  a_max = A.max()
  a_min = A.min()
  print('Minimum: ', a_min)
  print('Maximum: ', a_max)
  if (a_min < (A.quantile(0.25)-(IQR_A*1.5))):
     print("There are low Outliers")
  elif(a_max > (A.quantile(0.75)+(IQR_A * 1.5))):
     print('There are upr outliers ')
  else:
     print('There are no outliers in coffee. \n')
#%%
  A = numc['works_hours']
  print('For working-hour: ')
  print('Q1: ', A.quantile(0.25))
  print('Q3: ', A.quantile(0.75))
  print('Median: ', A.quantile(0.5))

  IQR_A =  (A.quantile(0.75) - A.quantile(0.25))
  print('IQR for Amount: ', IQR_A)
  print('Lower fench: ', A.quantile(0.25)-(IQR_A * 1.5))
  print('Upper fench: ', A.quantile(0.75)+(IQR_A * 1.5))
  a_max = A.max()
  a_min = A.min()
  print('Minimum: ', a_min)
  print('Maximum: ', a_max)
  if (a_min < (A.quantile(0.25)-(IQR_A*1.5))):
     print("There are low Outliers")
  if(a_max > (A.quantile(0.75)+(IQR_A * 1.5))):
     print('There are upr outliers ')
  else:
     print('There are no outliers in whr. \n')

plt.boxplot(x="works_hours", data=numc)
plt.title("Systolic Cholesterol plot")
plt.ylabel("Cholesterol")
plt.show()
#%%
```

```
import matplotlib.pyplot as plt

catg["workclass"].value_counts().plot(kind='bar', color=['green','red','blue'])
plt.title("workclass Bar Chart")
plt.xlabel("2=self-employed , 1=private, 0=government")
plt.ylabel("for customer")
plt.show()

import matplotlib.pyplot as plt

catg["marital-status"].value_counts().plot(kind='bar', color=['red','green','blue','grey'])
plt.title("marital Bar Chart")
plt.xlabel("3=divorced, 2=married,1=never-married, 0=widowed")
plt.ylabel("for customer")
plt.show()

catg["sex"].value_counts().plot(kind='bar', color=['blue','pink'])
plt.title("gender Chart")
plt.xlabel("1=female, 0=male")
plt.ylabel("of customer")
plt.show()

numc.great_customer_class.value_counts().plot(kind='pie',figsize=(10,5),autopct='%1.2f%%')
plt.legend(["1=Yes,0= NO"])

import matplotlib.pyplot as plt
import seaborn as sns
sns.countplot(x='marital-status', data=catg, hue='sex')
plt.title("Gender having marital-status")
plt.xlabel("0=divorced, 1=married,2=never-married, 3=widowed")
plt.ylabel("person")
plt.legend(['males','female'])

import matplotlib.pyplot as plt
import seaborn as sns
sns.countplot(x='workclass', data=df, hue='great_customer_class')
plt.title("great-customer-class having workclass")
plt.xlabel("2=self_employed,1=private,0=government")
plt.ylabel("for customer")
plt.legend(['No','Yes'])

#%%
plt.scatter(numc.tea_per_year[numc.great_customer_class==1],
        numc.coffee_per_year[numc.great_customer_class==1] , c='blue')
plt.scatter(numc.tea_per_year[numc.great_customer_class==0],
        numc.coffee_per_year[numc.great_customer_class==0] , c='red')
plt.title('tea vs coffee')
plt.xlabel("tea")
plt.ylabel("coffee")
plt.legend(['greaterClass','NO'])
#%%
```

```
import seaborn as sns
sns.displot(x='works_hours',data=df,bins=10,kde=True)
plt.title('workhr')
plt.ylabel("of great-customer")
plt.show()

hyp test
import pandas as pd
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.csv')
#hypothesis testing for 1st 50 rows assuming the mean is 1000
df1 = df.loc[0:49,['battery_power']]
Z1 = (df1.mean()-1200)/(df1.std()/math.sqrt(50))
print("Z score value for")
print(Z1)
#%%
#assume 95% CI so significance level alpha = 0.05, now if P>alpha accept Null Hyp otherwise reject it
#for P(Z1<-0.618)=0.268,
P1=2*0.268
if(P1>0.05):
   print("accept Null Hypothesis because mean 1200 lies in 95% CI and is not in rejection region of significance")
else:
   print("Reject null hypothesis and accept alternative Hypothesis because mean 1200 does not lies in 95% CI and is in
rejection region of significance")

#%%
#assume clock speed had been is 1.8
df2 = df.loc[0:49,['clock_speed']]
Z2 = (df2.mean()-1.8)/(df2.std()/math.sqrt(50))
print("Z score value for")
print(Z2)
#%%
#assume 99% CI so significance level alpha = 0.01, now if P>alpha accept Null Hyp otherwise reject it
#for P(Z1<-2.5)=0.268,
P2=2*0.0062
if(P2>0.01):
   print("accept Null Hypothesis because mean 1.8 lies in 99% CI and is not in rejection region of significance")
else:
   print("Reject null hypothesis and accept alternative Hypothesis because mean 1.8 does not lies in 99% CI and is in
rejection region of significance")
#%%
import pandas as pd
import math
dp=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.csv')
#hypothesis testing for last 50 rows assuming the mean is 15
dp1=dp.loc[0:49,['sc_h']]
dp2=dp.loc[0:49,['talk_time']]
r1=dp1.std()*dp1.std()
r2=dp2.std()*dp2.std()
d=dp2.mean()
r3=r1/50
```

```
r4=r2/50
r5=r3+r4
print(r5)
#%%
import math
#hypothesis testing for 100 rows assuming the mean is 81.7
df8 = df.loc[0:99,['glucose']]
Z8 = (df8.mean()-81.6)/(df8.std()/math.sqrt(50))
print("Z score value for")
print(Z8)

#assume 95% CI so significance level alpha = 0.05, now if P>alpha accept Null Hyp otherwise reject it
#for P(Z1<0.658)=0.8,
P1=2*0.8
if(P1>0.05):
   print("accept Null Hypothesis Ho because popultion mean 81.8 lies in 95% CI and is not in rejection region of
significance")
else:
   print("Reject null hypothesis because mean does not lie in 95% CI and is in rejection region of significance accept Ha")

#%%
import math
t = df["tea_per_year"].median()
df["tea_per_year"].fillna(t, inplace = True)
c = df["coffee_per_year"].median()
df["coffee_per_year"].fillna(c, inplace = True)
dp1=df.loc[0:49,['tea_per_year']]
dp2=df.loc[0:49,['coffee_per_year']]
m1=dp1.mean()
m2=dp2.mean()
print(m1)
print(m2)
s1=dp1.std()*dp1.std()
s2=dp2.std()*dp2.std()
print(s1/50)
print(s2/50)
T test:
import pandas as pd
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.csv')
df2 = df.loc[0:9,['battery_power']]
print(df2)

import pandas as pd
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.csv')
df1 = df.loc[0:50,['battery_power']]
df2 = df.loc[51:101,['battery_power']]
D=(abs(df1.mean()-df2.mean()))
S=(df1.std()/math.sqrt(51)+df2.std()/math.sqrt(51))
print(D/S)
#%%
```

```
import math
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.csv')
df1 = df.loc[0:15,['sc_h']]
df2 = df.loc[0:15,['sc_w']]
A1=df1.mean()
A2=df2.mean()
S1=df1.std()/math.sqrt(16)
S2=df2.std()/math.sqrt(16)
print(A1)
print(A2)
print(S1)
print(S2)
print(abs(A1-A2))
#Degree of Freedom = (n1+n2-2)=16+16-2=30 at 95% CI for one tailed 1.697 and two tailed 2.042 critical value
#Here T value < Crtitcal value for both tailed so we accept Null Hypothesis that the means are equal
Chi square
import pandas as pd
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\mobile_price_train.csv')

exp = 1000
ob1 = 1010
ob2 = 990
Cs = (ob1-exp)**2/exp +  (ob2-exp)**2/exp
print('Chi sqaure value for blue')
print(Cs)

#here degree of freedom n-1=2-1=1 for 5% significance the crtitcal value is 3.841`
#So ChiSqr value < critical value so it is in the accepted region so its each proportion hypothesized value Ho

#here degree of freedom n-1=2-1=1 for 5% significance the crtitcal value is 3.841`
#So ChiSqr value > critical value so it is in the rejected region and atleast one proportion differ from hypothesized value Ha
import pandas as pd
df=pd.read_csv(r'C:\Users\mdzid\Documents\project303\great_customers.csv')
df.drop_duplicates(inplace=True)
df['workclass'] = df['workclass'].fillna(df['workclass'].mode()[0])
df['workclass'].value_counts()

exp = 2866.33
ob1 = 6500
ob2 = 1127
ob3 = 972
Cs = (ob1-exp)**2/exp + (ob2-exp)**2/exp + (ob3-exp)**2/exp
print('Chi sqaure value for blue')
print(Cs)
```