In [ ]:

```
Name: N.S.Sai Parasanth
Regno: 21BCE8305
Date: 22nd septmber, 2023
```

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
data=pd.read_csv("Employee-Attrition.csv")
```

In [3]:

```
data.head()
```

Out[3]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educ
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Lif
1	49	No	Travel_Frequently	279	Research & Development	8	1	Lif
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Lif
4	27	No	Travel_Rarely	591	Research & Development	2	1	

5 rows × 35 columns

In [4]:

```
data.tail()
```

Out[4]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	E
1465	36	No	Travel_Frequently	884	Research & Development	23	2	
1466	39	No	Travel_Rarely	613	Research & Development	6	1	
1467	27	No	Travel_Rarely	155	Research & Development	4	3	
1468	49	No	Travel_Frequently	1023	Sales	2	3	
1469	34	No	Travel_Rarely	628	Research & Development	8	3	

5 rows × 35 columns

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1470 non-null   int64 
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64 
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64 
 6   Education                 1470 non-null   int64 
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64 
 9   EmployeeNumber            1470 non-null   int64 
 10  EnvironmentSatisfaction   1470 non-null   int64 
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64 
 13  JobInvolvement            1470 non-null   int64 
 14  JobLevel                  1470 non-null   int64 
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64 
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64 
 19  MonthlyRate               1470 non-null   int64 
 20  NumCompaniesWorked        1470 non-null   int64 
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64 
 24  PerformanceRating         1470 non-null   int64 
 25  RelationshipSatisfaction  1470 non-null   int64 
 26  StandardHours             1470 non-null   int64 
 27  StockOptionLevel          1470 non-null   int64 
 28  TotalWorkingYears         1470 non-null   int64 
 29  TrainingTimesLastYear     1470 non-null   int64 
 30  WorkLifeBalance           1470 non-null   int64 
 31  YearsAtCompany            1470 non-null   int64 
 32  YearsInCurrentRole        1470 non-null   int64 
 33  YearsSinceLastPromotion   1470 non-null   int64 
 34  YearsWithCurrManager      1470 non-null   int64 
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

In [6]:
```
data.describe()
```

Out[6]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNu
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.00
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.86
std	9.135373	403.509100	8.106864	1.024165	0.0	602.02
min	18.000000	102.000000	1.000000	1.000000	1.0	1.00
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.25
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.50
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.75
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.00

8 rows × 26 columns

Handling the null values

In [7]:
```
data.isnull().any()
```

Out[7]:
```
Age                         False
Attrition                   False
BusinessTravel              False
DailyRate                   False
Department                  False
DistanceFromHome            False
Education                   False
EducationField              False
EmployeeCount               False
EmployeeNumber              False
EnvironmentSatisfaction     False
Gender                      False
HourlyRate                  False
JobInvolvement              False
JobLevel                    False
JobRole                     False
JobSatisfaction             False
MaritalStatus               False
MonthlyIncome               False
MonthlyRate                 False
NumCompaniesWorked          False
Over18                      False
OverTime                    False
PercentSalaryHike           False
PerformanceRating           False
RelationshipSatisfaction    False
StandardHours               False
StockOptionLevel            False
TotalWorkingYears           False
TrainingTimesLastYear       False
WorkLifeBalance             False
YearsAtCompany              False
YearsInCurrentRole          False
YearsSinceLastPromotion     False
YearsWithCurrManager        False
dtype: bool
```

In [8]:
```
data.isnull().sum()
```

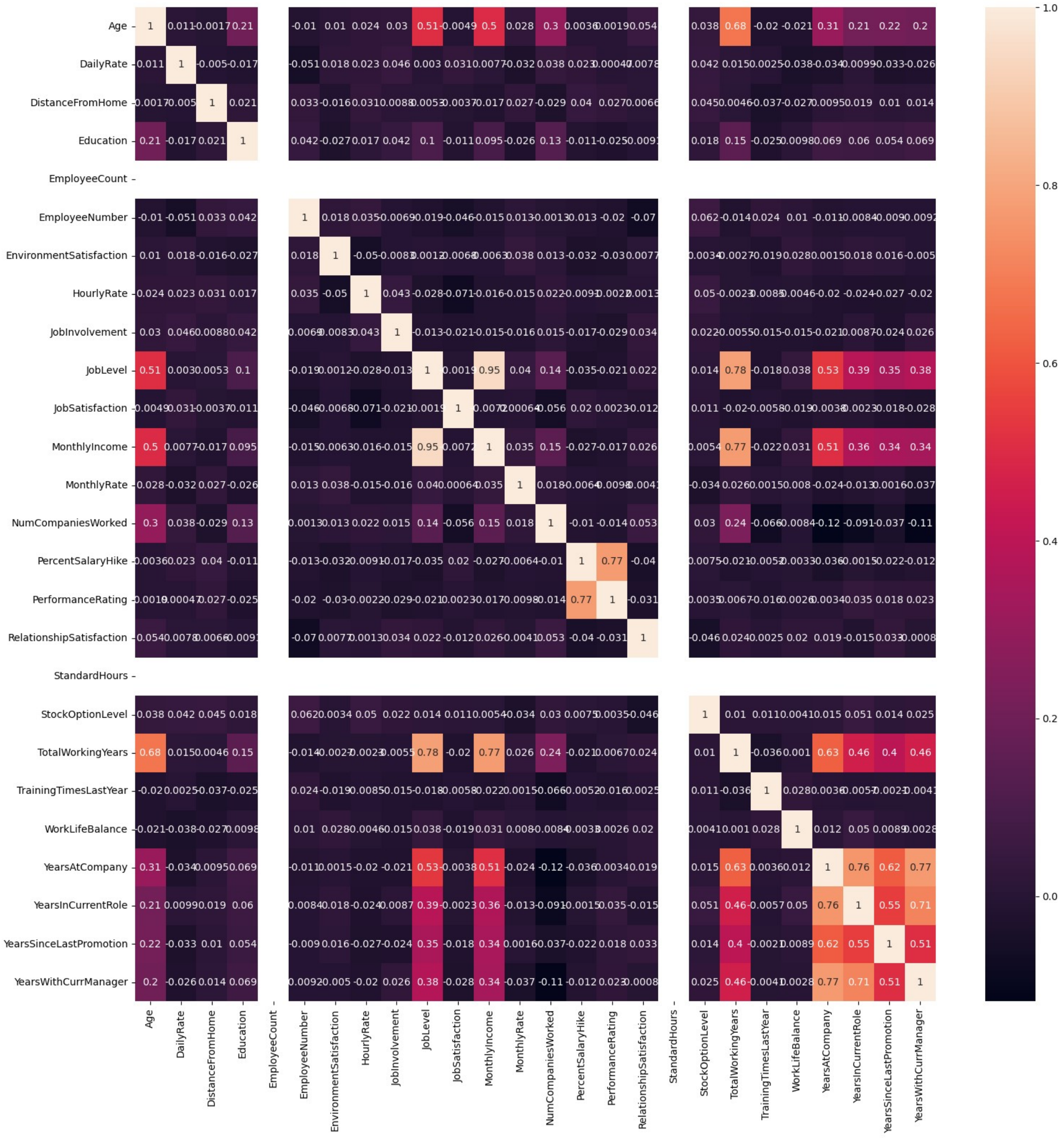
Out[8]:
```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

In [9]:
```
cor=data.corr()
```

```
C:\Users\Prasanth Nimmala\AppData\Local\Temp\ipykernel_8884\1426905697.py:1:
FutureWarning: The default value of numeric_only in DataFrame.corr is depreca
ted. In a future version, it will default to False. Select only valid columns
or specify the value of numeric_only to silence this warning.
  cor=data.corr()
```

In [10]:
```
fig=plt.figure(figsize=(18,18))
sns.heatmap(cor,annot=True)
```

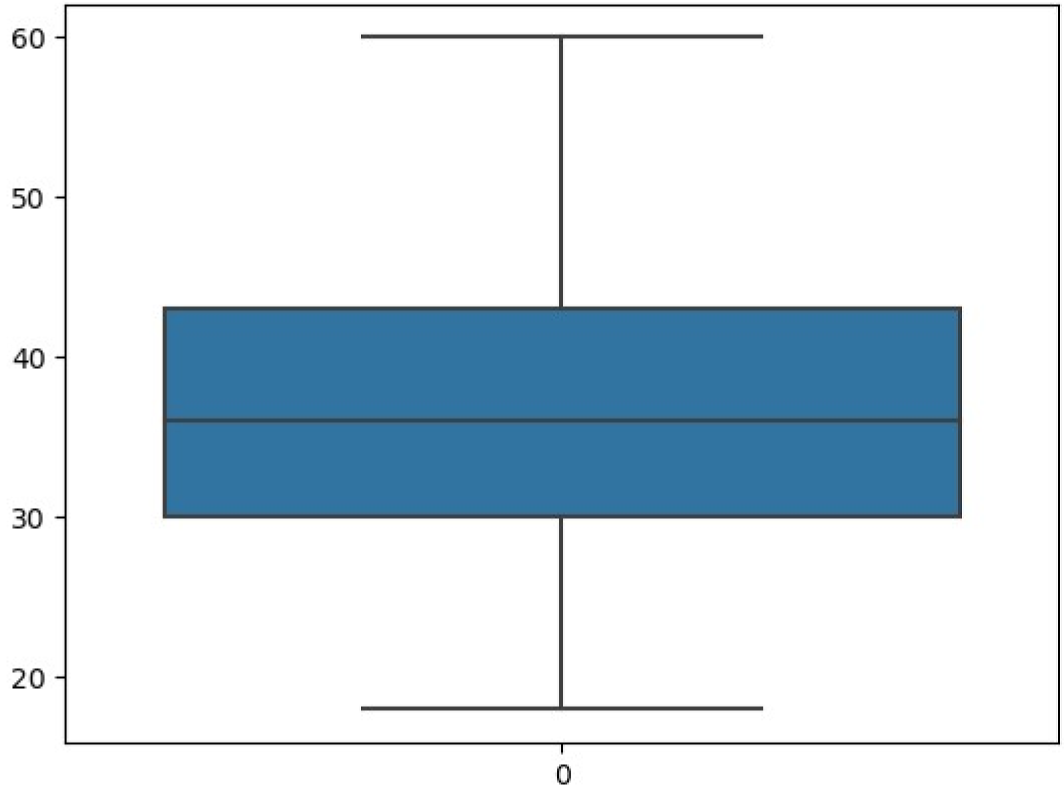
Out[10]:
```
<Axes: >
```



Outliers

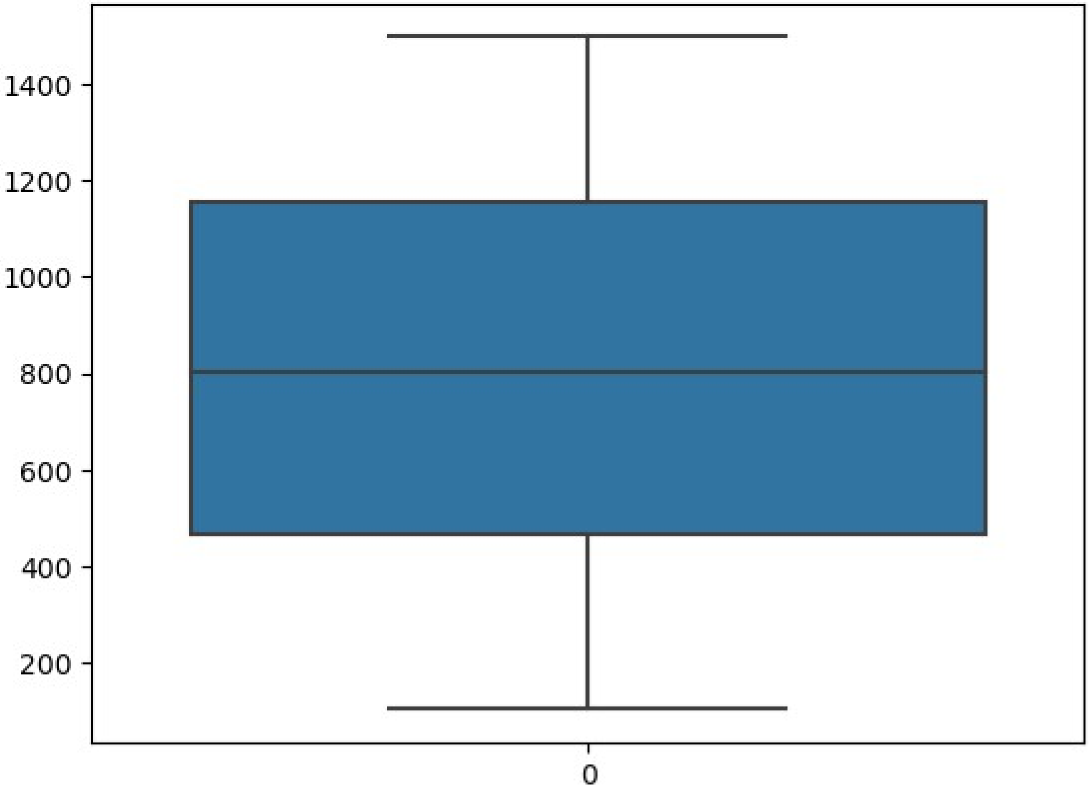
```
In [11]: sns.boxplot(data["Age"])
```

```
Out[11]: <Axes: >
```



```
In [12]: sns.boxplot(data["DailyRate"])
```

```
Out[12]: <Axes: >
```



```
In [29]: data.describe()
```

Out[29]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNu
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.00
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.86
std	9.135373	403.509100	8.106864	1.024165	0.0	602.02
min	18.000000	102.000000	1.000000	1.000000	1.0	1.00
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.25
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.50
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.75
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.00

8 rows × 26 columns

In [30]:
```
data.head()
```

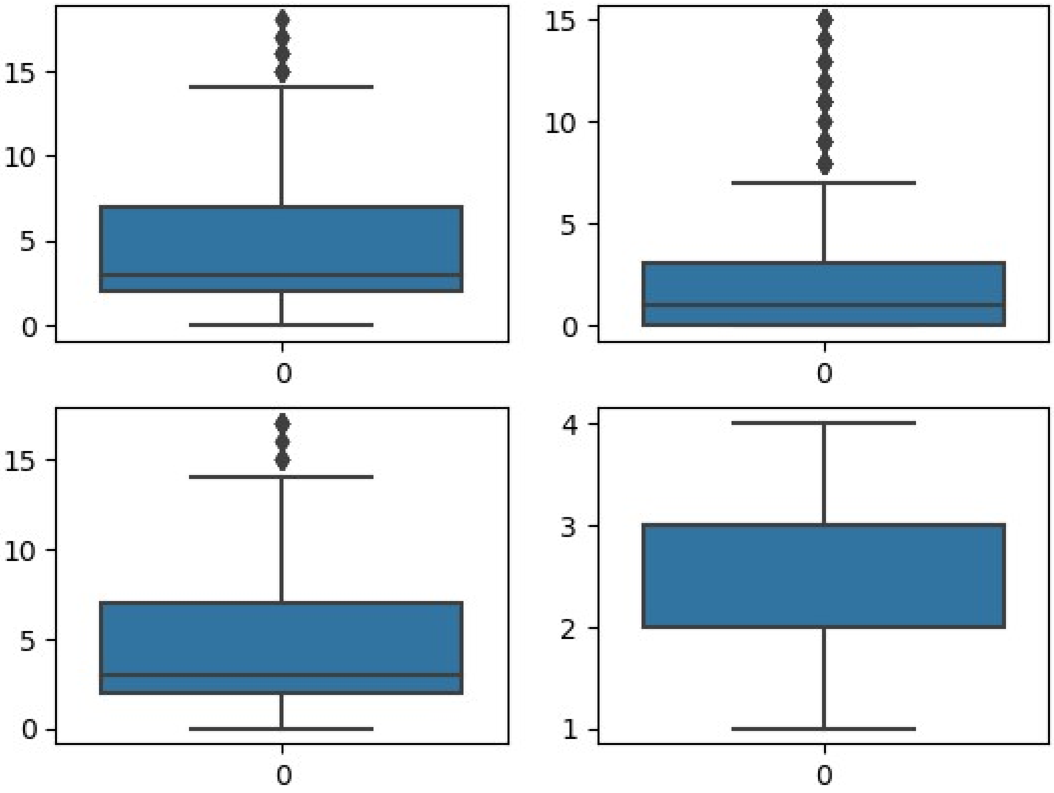
Out[30]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EmployeeCoun
0	41	Travel_Rarely	1102	Sales	1	2	
1	49	Travel_Frequently	279	Research & Development	8	1	
2	37	Travel_Rarely	1373	Research & Development	2	2	
3	33	Travel_Frequently	1392	Research & Development	3	4	
4	27	Travel_Rarely	591	Research & Development	2	1	

5 rows × 33 columns

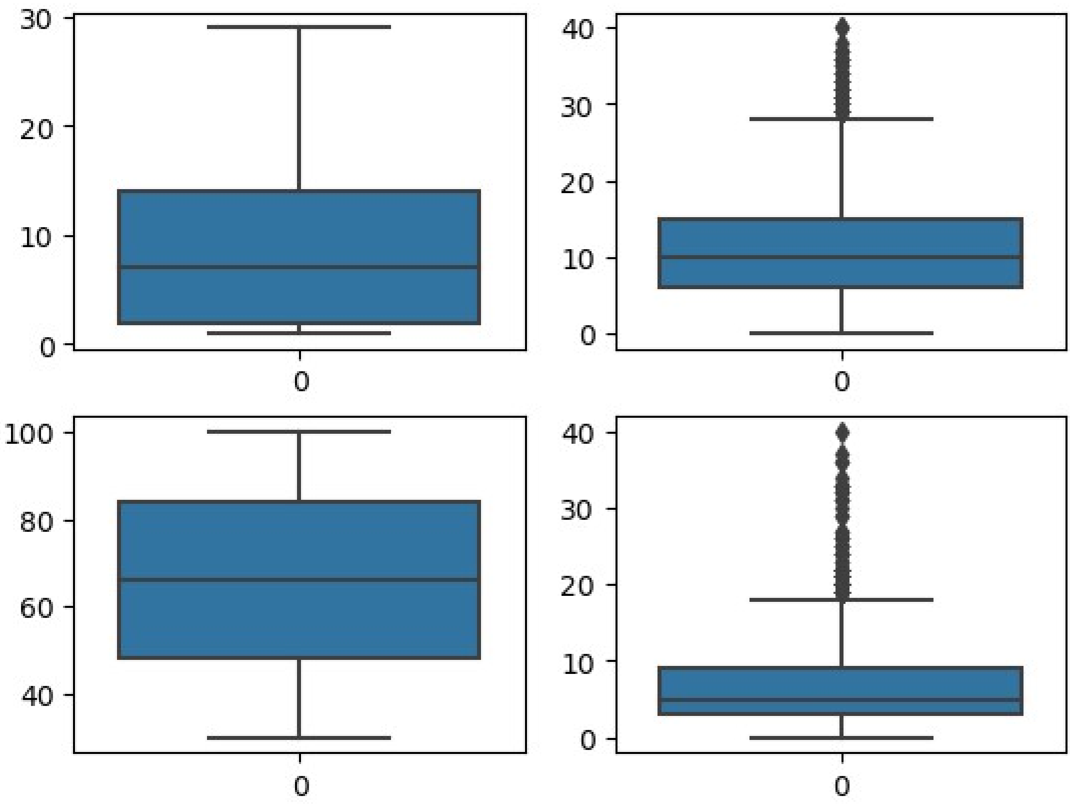
In [14]:
```
fig, axes = plt.subplots(2,2)
sns.boxplot(data=data["YearsInCurrentRole"],ax=axes[0,0])
sns.boxplot(data=data["YearsSinceLastPromotion"],ax=axes[0,1])
sns.boxplot(data=data["YearsWithCurrManager"],ax=axes[1,0])
sns.boxplot(data=data["WorkLifeBalance"],ax=axes[1,1])
```

Out[14]:
```
<Axes: >
```



In [15]:
```
fig, axes = plt.subplots(2,2)
sns.boxplot(data=data["DistanceFromHome"],ax=axes[0,0])
sns.boxplot(data=data["TotalWorkingYears"],ax=axes[0,1])
sns.boxplot(data=data["HourlyRate"],ax=axes[1,0])
sns.boxplot(data=data["YearsAtCompany"],ax=axes[1,1])
```

Out[15]:
```
<Axes: >
```



Handling the outliers

In [16]:
```
YearsInCurrentRole_q1 = data.YearsInCurrentRole.quantile(0.25)
YearsInCurrentRole_q3 = data.YearsInCurrentRole.quantile(0.75)
IQR_YearsInCurrentRole=YearsInCurrentRole_q3-YearsInCurrentRole_q1
upperlimit_YearsInCurrentRole=YearsInCurrentRole_q3+1.5*IQR_YearsInCurrentRole
lower_limit_YearsInCurrentRole =YearsInCurrentRole_q1-1.5*IQR_YearsInCurrentRo
median_YearsInCurrentRole=data["YearsInCurrentRole"].median()
data['YearsInCurrentRole'] = np.where(
    (data['YearsInCurrentRole'] > upperlimit_YearsInCurrentRole),
    median_YearsInCurrentRole,
    data['YearsInCurrentRole']
)
```

In [17]:

```
YearsSinceLastPromotion_q1 = data.YearsSinceLastPromotion.quantile(0.25)
YearsSinceLastPromotion_q3 = data.YearsSinceLastPromotion.quantile(0.75)
IQR_YearsSinceLastPromotion=YearsSinceLastPromotion_q3-YearsSinceLastPromotion
upperlimit_YearsSinceLastPromotion=YearsSinceLastPromotion_q3+1.5*IQR_YearsSin
lower_limit_YearsSinceLastPromotion =YearsSinceLastPromotion_q1-1.5*IQR_YearsS
median_YearsSinceLastPromotion=data["YearsSinceLastPromotion"].median()
data['YearsSinceLastPromotion'] = np.where(
    (data['YearsSinceLastPromotion'] > upperlimit_YearsSinceLastPromotion),
    median_YearsSinceLastPromotion,
    data['YearsSinceLastPromotion']
)
```

In [18]:

```
YearsWithCurrManager_q1 = data.YearsWithCurrManager.quantile(0.25)
YearsWithCurrManager_q3 = data.YearsWithCurrManager.quantile(0.75)
IQR_YearsWithCurrManager=YearsWithCurrManager_q3-YearsWithCurrManager_q1
upperlimit_YearsWithCurrManager=YearsWithCurrManager_q3+1.5*IQR_YearsWithCurrM
lower_limit_YearsWithCurrManager =YearsWithCurrManager_q1-1.5*IQR_YearsWithCur
median_YearsWithCurrManager=data["YearsWithCurrManager"].median()
data['YearsWithCurrManager'] = np.where(
    (data['YearsWithCurrManager'] > upperlimit_YearsWithCurrManager),
    median_YearsWithCurrManager,
    data['YearsWithCurrManager']
)
```

In [19]:

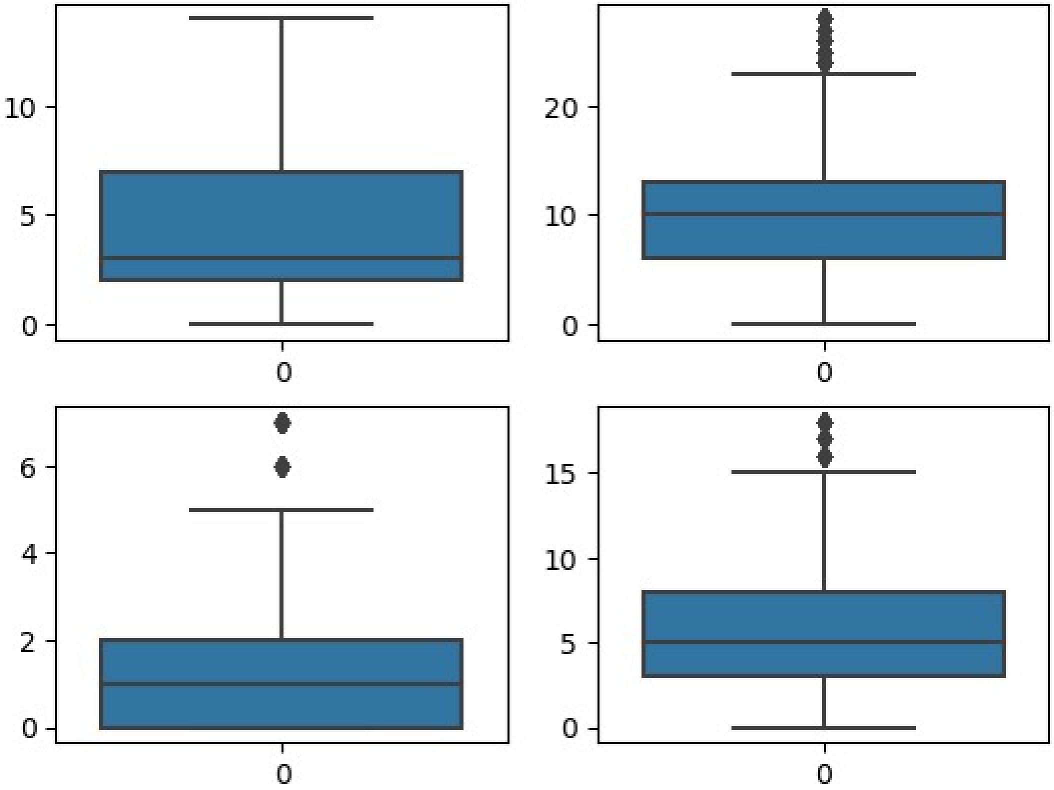
```
TotalWorkingYears_q1 = data.TotalWorkingYears.quantile(0.25)
TotalWorkingYears_q3 = data.TotalWorkingYears.quantile(0.75)
IQR_TotalWorkingYears=TotalWorkingYears_q3-TotalWorkingYears_q1
upperlimit_TotalWorkingYears=TotalWorkingYears_q3+1.5*IQR_TotalWorkingYears
lower_limit_TotalWorkingYears=TotalWorkingYears_q1-1.5*IQR_TotalWorkingYears
median_TotalWorkingYears=data["TotalWorkingYears"].median()
data['TotalWorkingYears'] = np.where(
    (data['TotalWorkingYears'] > upperlimit_TotalWorkingYears),
    median_TotalWorkingYears,
    data['TotalWorkingYears']
)
```

In [20]:

```
YearsAtCompany_q1 = data.YearsAtCompany.quantile(0.25)
YearsAtCompany_q3 = data.YearsAtCompany.quantile(0.75)
IQR_YearsAtCompany=YearsAtCompany_q3-YearsAtCompany_q1
upperlimit_YearsAtCompany=YearsAtCompany_q3+1.5*IQR_YearsAtCompany
lower_limit_YearsAtCompany=YearsAtCompany_q1-1.5*IQR_YearsAtCompany
median_YearsAtCompany=data["YearsAtCompany"].median()
data['YearsAtCompany'] = np.where(
    (data['YearsAtCompany'] > upperlimit_YearsAtCompany),
    median_YearsAtCompany,
    data['YearsAtCompany']
)
```

In [21]:
```
fig, axes = plt.subplots(2,2)
sns.boxplot(data=data["YearsWithCurrManager"],ax=axes[0,0])
sns.boxplot(data=data["TotalWorkingYears"],ax=axes[0,1])
sns.boxplot(data=data["YearsSinceLastPromotion"],ax=axes[1,0])
sns.boxplot(data=data["YearsAtCompany"],ax=axes[1,1])
```

Out[21]:
```
<Axes: >
```



In [31]:
```
data.head()
```

Out[31]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EmployeeCoun
0	41	Travel_Rarely	1102	Sales	1	2	1
1	49	Travel_Frequently	279	Research & Development	8	1	1
2	37	Travel_Rarely	1373	Research & Development	2	2	1
3	33	Travel_Frequently	1392	Research & Development	3	4	1
4	27	Travel_Rarely	591	Research & Development	2	1	1

5 rows × 33 columns

In [22]:
```
data.drop("EducationField",axis=1,inplace=True)
```

In [23]:
```
data.head(2)
```

Out[23]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Empl
0	41	Yes	Travel_Rarely	1102	Sales	1	2	
1	49	No	Travel_Frequently	279	Research & Development	8	1	

2 rows × 34 columns

In [24]:
```
data["BusinessTravel"].unique()
```

Out[24]:
```
array(['Travel_Rarely', 'Travel_Frequently', 'Non-Travel'], dtype=object)
```

Splitting the data

In [25]:
```
y=data["Attrition"]
```

In [26]:
```
y.head()
```

Out[26]:
```
0     Yes
1      No
2     Yes
3      No
4      No
Name: Attrition, dtype: object
```

In [27]:
```
data.drop("Attrition",axis=1,inplace=True)
```

In [28]:
```
data.head()
```

Out[28]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EmployeeCoun
0	41	Travel_Rarely	1102	Sales	1	2	
1	49	Travel_Frequently	279	Research & Development	8	1	
2	37	Travel_Rarely	1373	Research & Development	2	2	
3	33	Travel_Frequently	1392	Research & Development	3	4	
4	27	Travel_Rarely	591	Research & Development	2	1	

5 rows × 33 columns

Encoding

In [32]:
```
from sklearn.preprocessing import LabelEncoder
```

In [33]:
```
le=LabelEncoder()
```

In [34]:
```
data["BusinessTravel"]=le.fit_transform(data["BusinessTravel"])
```

In [35]:
```
data["Department"]=le.fit_transform(data["Department"])
```

In [36]:
```
data["Gender"]=le.fit_transform(data["Gender"])
```

In [37]:
```
y=le.fit_transform(y)
```

In [38]:
```
y
```

Out[38]:
```
array([1, 0, 1, ..., 0, 0, 0])
```

In [39]:
```
data["JobRole"]=le.fit_transform(data["JobRole"])
```

In [40]:
```
data["Over18"]=le.fit_transform(data["Over18"])
```

In [41]:
```
data["MaritalStatus"]=le.fit_transform(data["MaritalStatus"])
```

In [42]:
```
data["OverTime"]=le.fit_transform(data["OverTime"])
```

In [43]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 33 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Age                       1470 non-null   int64  
 1   BusinessTravel            1470 non-null   int32  
 2   DailyRate                 1470 non-null   int64  
 3   Department                1470 non-null   int32  
 4   DistanceFromHome          1470 non-null   int64  
 5   Education                 1470 non-null   int64  
 6   EmployeeCount             1470 non-null   int64  
 7   EmployeeNumber            1470 non-null   int64  
 8   EnvironmentSatisfaction   1470 non-null   int64  
 9   Gender                    1470 non-null   int32  
 10  HourlyRate                1470 non-null   int64  
 11  JobInvolvement            1470 non-null   int64  
 12  JobLevel                  1470 non-null   int64  
 13  JobRole                   1470 non-null   int32  
 14  JobSatisfaction           1470 non-null   int64  
 15  MaritalStatus             1470 non-null   int32  
 16  MonthlyIncome             1470 non-null   int64  
 17  MonthlyRate               1470 non-null   int64  
 18  NumCompaniesWorked        1470 non-null   int64  
 19  Over18                    1470 non-null   int32  
 20  OverTime                  1470 non-null   int32  
 21  PercentSalaryHike         1470 non-null   int64  
 22  PerformanceRating         1470 non-null   int64  
 23  RelationshipSatisfaction  1470 non-null   int64  
 24  StandardHours             1470 non-null   int64  
 25  StockOptionLevel          1470 non-null   int64  
 26  TotalWorkingYears         1470 non-null   float64
 27  TrainingTimesLastYear     1470 non-null   int64  
 28  WorkLifeBalance           1470 non-null   int64  
 29  YearsAtCompany            1470 non-null   float64
 30  YearsInCurrentRole        1470 non-null   float64
 31  YearsSinceLastPromotion   1470 non-null   float64
 32  YearsWithCurrManager      1470 non-null   float64
dtypes: float64(5), int32(7), int64(21)
memory usage: 338.9 KB
```

train test split

In [44]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(data,y,test_size=0.3,random_sta
```

In [45]:
```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[45]:
```
((1029, 33), (441, 33), (1029,), (441,))
```

Feature Scaling

```
In [46]: from sklearn.preprocessing import StandardScaler
```

```
In [47]: sc=StandardScaler()
```

```
In [48]: x_train=sc.fit_transform(x_train)
```

```
In [49]: x_test=sc.fit_transform(x_test)
```

Building the model

Multi-Linear Regression

```
In [50]: from sklearn.linear_model import LinearRegression
```

```
In [51]: lr = LinearRegression()
```

```
In [52]: lr.fit(x_train,y_train)
```

```
Out[52]: LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [53]: lr.coef_  #slope(m)
```

```
Out[53]: array([-3.54940447e-02,  7.88352347e-05, -1.70825038e-02,  3.46389690e-02,
                 2.44612841e-02,  3.65668214e-03,  4.16333634e-17, -9.46820520e-03,
                -4.11203734e-02,  1.06338881e-02, -2.97662154e-03, -3.84864283e-02,
                -1.52927977e-02, -1.57839139e-02, -3.67252862e-02,  3.35765928e-02,
                -5.90043558e-03,  5.81099165e-03,  3.78471890e-02, -6.93889390e-18,
                 9.55263279e-02, -2.55800078e-02,  2.01844797e-02, -2.64773510e-02,
                -1.21430643e-17, -1.79286106e-02, -3.30529386e-02, -1.09247807e-02,
                -3.10631611e-02, -2.47887717e-02, -1.10177742e-02,  2.11897289e-02,
                -6.60823991e-03])
```

```
In [54]: lr.intercept_  #(c)
```

```
Out[54]: 0.16229348882410102
```

```
In [55]: y_pred = lr.predict(x_test)
```

In [56]:
```
y_pred
```

Out[56]:
```
array([ 1.30302477e-01,  2.17626230e-01,  3.46282415e-01,  5.41382549e-03,
        4.99292896e-01,  1.01628868e-01,  3.44742777e-01,  1.23994945e-01,
       -1.60694945e-01,  4.02435622e-01,  1.44159172e-01,  2.67416840e-01,
       -4.62559536e-02,  5.58671849e-01,  2.81858700e-01,  1.53537792e-02,
        1.78573363e-01,  2.77532834e-01,  9.37121052e-02,  2.17571624e-01,
        2.65936178e-01,  1.41499184e-02,  8.36251186e-02,  9.58849826e-02,
        5.09869963e-01,  2.94764240e-01,  7.85819529e-02,  1.26647773e-01,
        5.05518902e-01,  8.48456917e-02, -7.97229275e-02,  2.15516993e-02,
        1.08079105e-01,  3.65998400e-01,  1.24517362e-01,  5.13682786e-02,
        1.06749689e-01,  6.07640778e-02,  6.66425313e-02,  4.81312859e-02,
       -1.16761425e-02, -2.97852924e-02,  5.25135582e-02, -1.59076817e-02,
       -1.71522795e-02,  4.17777714e-01,  3.67341564e-01, -2.14569245e-01,
        5.47964121e-01,  4.40723777e-01,  1.96701754e-01,  4.42415223e-01,
        1.45760263e-01,  3.75821843e-01,  4.92762622e-01,  2.95885645e-01,
       -4.62363391e-02,  3.16337190e-01, -7.90813313e-03,  2.52644685e-01,
       -3.18239329e-02,  2.83907645e-01,  9.03615010e-02,  1.26934391e-01,
        3.58670014e-01,  2.40923530e-02,  3.55890111e-01,  1.95961225e-01,
        1.28554515e-01,  1.18806226e-01, -2.86217094e-02,  3.17635336e-01,
        1.08017895e-01,  1.25723940e-01,  2.30183307e-01,  9.84315444e-02,
```

In [57]:
```
y_test
```

Out[57]:
```
array([0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0])
```

Logistic Regression

In [58]:
```
from sklearn.linear_model import LogisticRegression
```

In [59]:
```
lg=LogisticRegression()
```

In [60]:
```
lg.fit(x_train,y_train)
```

Out[60]:
```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [61]:
```
y_pred_lg=lg.predict(x_test)
```

In [62]:
```
y_pred
```

Out[62]:
```
array([ 1.30302477e-01,  2.17626230e-01,  3.46282415e-01,  5.41382549e-03,
        4.99292896e-01,  1.01628868e-01,  3.44742777e-01,  1.23994945e-01,
       -1.60694945e-01,  4.02435622e-01,  1.44159172e-01,  2.67416840e-01,
       -4.62559536e-02,  5.58671849e-01,  2.81858700e-01,  1.53537792e-02,
        1.78573363e-01,  2.77532834e-01,  9.37121052e-02,  2.17571624e-01,
        2.65936178e-01,  1.41499184e-02,  8.36251186e-02,  9.58849826e-02,
        5.09869963e-01,  2.94764240e-01,  7.85819529e-02,  1.26647773e-01,
        5.05518902e-01,  8.48456917e-02, -7.97229275e-02,  2.15516993e-02,
        1.08079105e-01,  3.65998400e-01,  1.24517362e-01,  5.13682786e-02,
        1.06749689e-01,  6.07640778e-02,  6.66425313e-02,  4.81312859e-02,
       -1.16761425e-02, -2.97852924e-02,  5.25135582e-02, -1.59076817e-02,
       -1.71522795e-02,  4.17777714e-01,  3.67341564e-01, -2.14569245e-01,
        5.47964121e-01,  4.40723777e-01,  1.96701754e-01,  4.42415223e-01,
        1.45760263e-01,  3.75821843e-01,  4.92762622e-01,  2.95885645e-01,
       -4.62363391e-02,  3.16337190e-01, -7.90813313e-03,  2.52644685e-01,
       -3.18239329e-02,  2.83907645e-01,  9.03615010e-02,  1.26934391e-01,
        3.58670014e-01,  2.40923530e-02,  3.55890111e-01,  1.95961225e-01,
        1.28554515e-01,  1.18806226e-01, -2.86217094e-02,  3.17635336e-01,
        1.08017895e-01,  1.25723940e-01,  2.30183307e-01,  9.84315444e-02,
```

In [63]:
```
y_test
```

Out[63]:
```
array([0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0])
```

In [64]:
```
score = lg.score(x_test, y_test)
print(score)
```

```
0.8820861678004536
```

Confusion matrix

In [65]:
```
from sklearn import metrics
cm = metrics.confusion_matrix(y_test,y_pred_lg)
print(cm)
```

```
[[366   5]
 [ 47  23]]
```

Ridge and Lasso

In [66]:
```
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
```

In [67]:
```
rg=Ridge()
```

```
In [68]: parametres={"alpha":[1,2,3,5,10,20,30,40,60,70,80,90]}
         ridgecv=GridSearchCV(rg,parametres,scoring="neg_mean_squared_error",cv=5)
         ridgecv.fit(x_train,y_train)
```

```
Out[68]: GridSearchCV(cv=5, estimator=Ridge(),
                      param_grid={'alpha': [1, 2, 3, 5, 10, 20, 30, 40, 60, 70, 80, 9
         0]},
                      scoring='neg_mean_squared_error')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [69]: print(ridgecv.best_params_)

         {'alpha': 90}
```

```
In [70]: print(ridgecv.best_score_)

         -0.11390621139234183
```

```
In [71]: y_pred_rg=ridgecv.predict(x_test)
```

```
In [72]: y_pred_rg
```

```
Out[72]: array([ 1.34413485e-01,  2.22561818e-01,  3.41692977e-01,  3.88209867e-03,
                 4.84617338e-01,  1.16361483e-01,  3.30449743e-01,  1.27358807e-01,
                -1.34442619e-01,  3.77692888e-01,  1.33001445e-01,  2.69898751e-01,
                -2.54707392e-02,  5.25771894e-01,  2.67543514e-01,  2.78725024e-02,
                 1.82233111e-01,  2.78896415e-01,  9.12689699e-02,  2.11494641e-01,
                 2.70103341e-01,  8.44922044e-03,  8.74746722e-02,  1.05348798e-01,
                 4.87749940e-01,  2.83080512e-01,  8.80556209e-02,  1.23817268e-01,
                 4.82185624e-01,  9.34824523e-02, -7.16448509e-02,  4.07003104e-02,
                 1.08437994e-01,  3.42151399e-01,  1.22270929e-01,  6.85889862e-02,
                 1.06690533e-01,  7.08689637e-02,  7.51570276e-02,  6.05829413e-02,
                 1.08782897e-02, -6.91368661e-03,  5.83191600e-02, -1.54680056e-02,
                -4.02267475e-03,  4.08010612e-01,  3.43668700e-01, -1.83519405e-01,
                 5.29536511e-01,  4.27646098e-01,  1.95234877e-01,  4.25012930e-01,
                 1.40754410e-01,  3.52173952e-01,  4.70372694e-01,  2.89240343e-01,
                -3.11642726e-02,  3.04206456e-01,  9.89337674e-03,  2.44569884e-01,
                -1.40249115e-02,  2.75133912e-01,  8.64669565e-02,  1.24214885e-01,
                 3.48994545e-01,  3.41026778e-02,  3.40548051e-01,  1.95847356e-01,
                 1.30040885e-01,  1.32259137e-01, -2.34680143e-02,  3.04595468e-01,
                 1.12452197e-01,  1.30525275e-01,  2.19329505e-01,  9.44722098e-02,
                 9.08185783e-02,  3.60943486e-01,  2.51475715e-01,  4.59939918e-02,
```

In [73]:
```
y_test
```

Out[73]:
```
array([0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0])
```

In [74]:
```
from sklearn import metrics
print(metrics.r2_score(y_test,y_pred_rg))
print(metrics.r2_score(y_train,ridgecv.predict(x_train)))
```

```
0.21073458438815906
0.206156721028510‌9
```

Lasso

In [75]:
```
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV
```

In [76]:
```
la=Ridge()
```

In [77]:
```
parametres={"alpha":[1,2,3,5,10,20,30,40,60,70,80,90]}
ridgecv=GridSearchCV(la,parametres,scoring="neg_mean_squared_error",cv=5)
ridgecv.fit(x_train,y_train)
```

Out[77]:
```
GridSearchCV(cv=5, estimator=Ridge(),
             param_grid={'alpha': [1, 2, 3, 5, 10, 20, 30, 40, 60, 70, 80, 9
0]},
             scoring='neg_mean_squared_error')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [78]: print(ridgecv.best_params_)

         {'alpha': 90}
```

```
In [79]: print(ridgecv.best_score_)

         -0.11390621139234183
```

```
In [80]: y_pred_la=ridgecv.predict(x_test)
```

```
In [81]: y_pred_la

Out[81]: array([ 1.34413485e-01,  2.22561818e-01,  3.41692977e-01,  3.88209867e-03,
                 4.84617338e-01,  1.16361483e-01,  3.30449743e-01,  1.27358807e-01,
                -1.34442619e-01,  3.77692888e-01,  1.33001445e-01,  2.69898751e-01,
                -2.54707392e-02,  5.25771894e-01,  2.67543514e-01,  2.78725024e-02,
                 1.82233111e-01,  2.78896415e-01,  9.12689699e-02,  2.11494641e-01,
                 2.70103341e-01,  8.44922044e-03,  8.74746722e-02,  1.05348798e-01,
                 4.87749940e-01,  2.83080512e-01,  8.80556209e-02,  1.23817268e-01,
                 4.82185624e-01,  9.34824523e-02, -7.16448509e-02,  4.07003104e-02,
                 1.08437994e-01,  3.42151399e-01,  1.22270929e-01,  6.85889862e-02,
                 1.06690533e-01,  7.08689637e-02,  7.51570276e-02,  6.05829413e-02,
                 1.08782897e-02, -6.91368661e-03,  5.83191600e-02, -1.54680056e-02,
                -4.02267475e-03,  4.08010612e-01,  3.43668700e-01, -1.83519405e-01,
                 5.29536511e-01,  4.27646098e-01,  1.95234877e-01,  4.25012930e-01,
                 1.40754410e-01,  3.52173952e-01,  4.70372694e-01,  2.89240343e-01,
                -3.11642726e-02,  3.04206456e-01,  9.89337674e-03,  2.44569884e-01,
                -1.40249115e-02,  2.75133912e-01,  8.64669565e-02,  1.24214885e-01,
                 3.48994545e-01,  3.41026778e-02,  3.40548051e-01,  1.95847356e-01,
                 1.30040885e-01,  1.32259137e-01, -2.34680143e-02,  3.04595468e-01,
                 1.12452197e-01,  1.30525275e-01,  2.19329505e-01,  9.44722098e-02,
```

```
In [82]: from sklearn import metrics
         print(metrics.r2_score(y_test,y_pred_la))
         print(metrics.r2_score(y_train,ridgecv.predict(x_train)))

         0.21073458438815906
         0.2061567210285109
```

Decision Tree

```
In [83]: from sklearn.tree import DecisionTreeClassifier
         dtc=DecisionTreeClassifier()
```

```
In [84]: dtc.fit(x_train,y_train)

Out[84]: DecisionTreeClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [85]:
```
pred=dtc.predict(x_test)
```

In [86]:
```
pred
```

Out[86]:
```
array([0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0])
```

In [87]:
```
y_test
```

Out[87]:
```
array([0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0])
```

In [88]:
```
#Accuracy score
from sklearn.metrics import accuracy_score,confusion_matrix,classification_rep
```

In [89]:
```
accuracy_score(y_test,pred)
```

Out[89]:
```
0.7664399092970522
```

In [90]:
```
confusion_matrix(y_test,pred)
```

Out[90]:
```
array([[316,  55],
       [ 48,  22]], dtype=int64)
```

In [91]:
```
pd.crosstab(y_test,pred)
```

Out[91]:

col_0	0	1
row_0		
0	316	55
1	48	22

In [92]:
```
print(classification_report(y_test,pred))
```

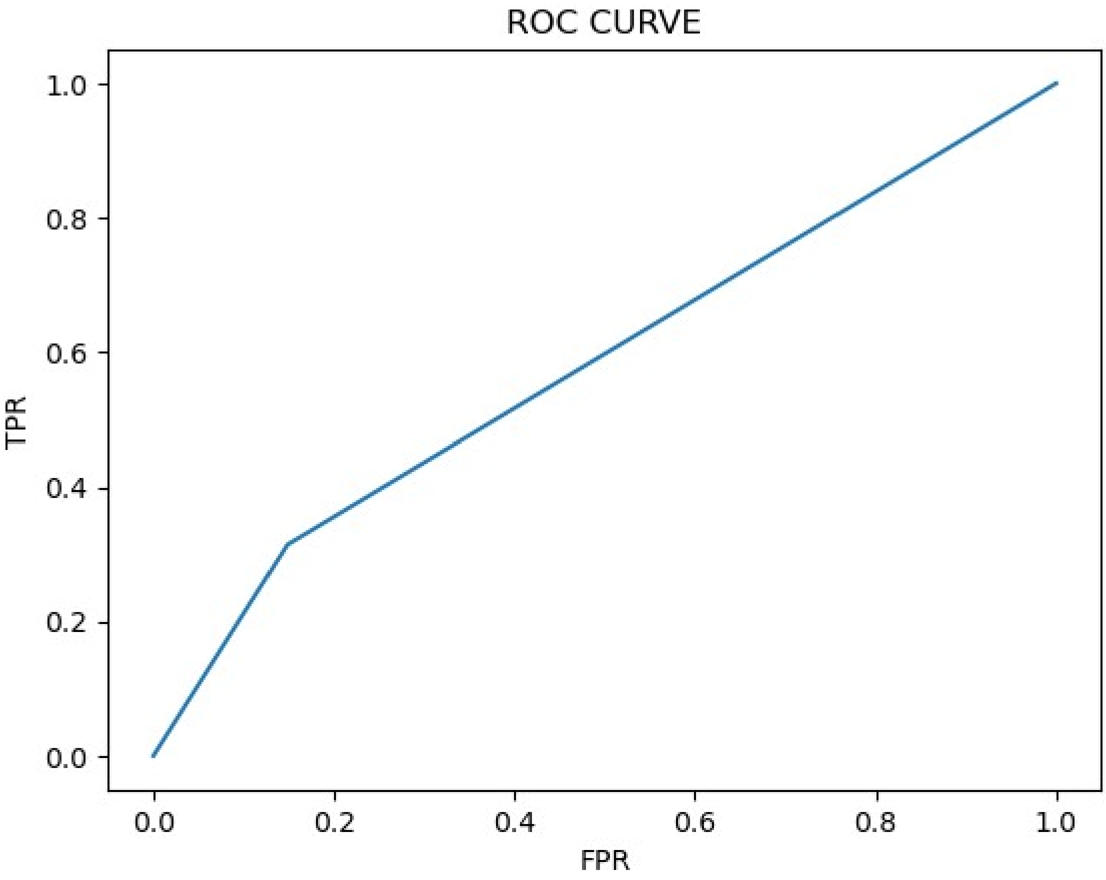
```
              precision    recall  f1-score   support

           0       0.87      0.85      0.86       371
           1       0.29      0.31      0.30        70

    accuracy                           0.77       441
   macro avg       0.58      0.58      0.58       441
weighted avg       0.78      0.77      0.77       441
```

In [93]:
```
probability=dtc.predict_proba(x_test)[:,1]
```

In [94]:
```
# roc_curve
fpr,tpr,threshsholds = roc_curve(y_test,probability)
```

In [95]:

```
plt.plot(fpr,tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC CURVE')
plt.show()
```



Random Forest

In [96]:

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
```

In [97]:

```
forest_params = [{'max_depth': list(range(10, 15)), 'max_features': list(range
```

In [98]:

```
from sklearn.model_selection import GridSearchCV
```

In [99]:

```
rfc_cv= GridSearchCV(rfc,param_grid=forest_params,cv=10,scoring="accuracy")
```

In [100]:
```
rfc_cv.fit(x_train,y_train)
```

```
C:\Users\Prasanth Nimmala\anaconda3\lib\site-packages\sklearn\model_selectio
n\_validation.py:378: FitFailedWarning:
50 fits failed out of a total of 700.
The score on these train-test partitions for these parameters will be set to
nan.
If these failures are not expected, you can try to debug them by setting erro
r_score='raise'.

Below are more details about the failures:
-----------------------------------------------------------------------------
---
50 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\Prasanth Nimmala\anaconda3\lib\site-packages\sklearn\model_s
election\_validation.py", line 686, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\Prasanth Nimmala\anaconda3\lib\site-packages\sklearn\ensembl
e\_forest.py", line 340, in fit
    self._validate_params()
  File "C:\Users\Prasanth Nimmala\anaconda3\lib\site-packages\sklearn\base.p
y", line 581, in _validate_params
    validate_parameter_constraints(
  File "C:\Users\Prasanth Nimmala\anaconda3\lib\site-packages\sklearn\utils\_
param_validation.py", line 97, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' par
ameter of RandomForestClassifier must be an int in the range [1, inf), a floa
t in the range (0.0, 1.0], a str among {'log2', 'sqrt', 'auto' (deprecated)}
or None. Got 0 instead.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\Prasanth Nimmala\anaconda3\lib\site-packages\sklearn\model_selectio
n\_search.py:952: UserWarning: One or more of the test scores are non-finite:
[       nan 0.84353703 0.84840091 0.8483914  0.85325528 0.85033314
 0.85421664 0.85033314 0.85422616 0.84644013 0.85517799 0.85519703
 0.85033314 0.84449838        nan 0.8445079  0.84935275 0.85031411
 0.85421664 0.84936227 0.85516848 0.85032362 0.84934323 0.8512945
 0.84935275 0.84934323 0.85322673 0.85032362        nan 0.8445079
 0.84936227 0.85324576 0.85033314 0.85033314 0.85324576 0.85810013
 0.85711974 0.84935275 0.85225585 0.8483914  0.85131354 0.85324576
        nan 0.84546926 0.84937179 0.84936227 0.85325528 0.85324576
 0.85615839 0.85324576 0.85520655 0.85615839 0.85517799 0.85324576
 0.8512945  0.85030459        nan 0.84547877 0.84644965 0.84546926
 0.85518751 0.84353703 0.84937179 0.85615839 0.85031411 0.8561679
 0.85713878 0.84838188 0.85227489 0.84643061]
  warnings.warn(
```

Out[100]:
```
GridSearchCV(cv=10, estimator=RandomForestClassifier(),
             param_grid=[{'max_depth': [10, 11, 12, 13, 14],
                          'max_features': [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
11,
                                           12, 13]}],
             scoring='accuracy')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [101]:
```
pred=rfc_cv.predict(x_test)
```

In [102]:
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.99      0.93       371
           1       0.84      0.23      0.36        70

    accuracy                           0.87       441
   macro avg       0.86      0.61      0.64       441
weighted avg       0.87      0.87      0.84       441
```

In [103]:
```
rfc_cv.best_params_
```

Out[103]:
```
{'max_depth': 12, 'max_features': 7}
```

In [104]:
```
rfc_cv.best_score_
```

Out[104]:
```
0.8581001332571864
```