# GUDIVADA VENKATA SESHA SAI DEEPAK - 21BCE9822

In [102]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [103]:
```
data=pd.read_csv("Employee-Attrition.csv")
```

In [104]:
```
data.head()
```

Out[104]:

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

5 rows × 35 columns

In [105]:
```
data.tail()
```

Out[105]:

| | 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 [106]:
```
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 [107]:
```
data.describe()
```

Out[107]:

| | 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 Null Values

In [108]:
```
data.isnull().any()
```

Out[108]:
```
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
```

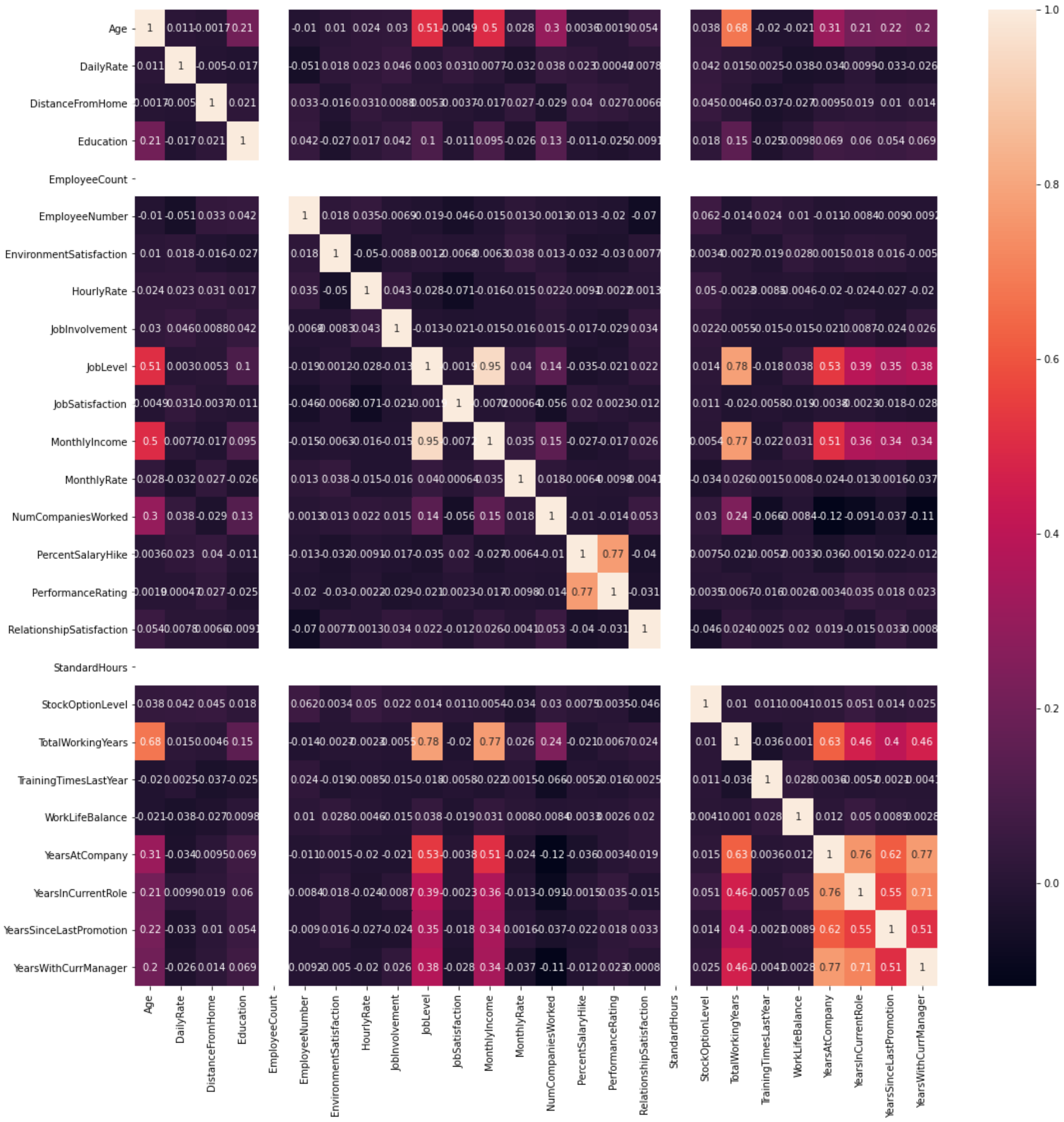
In [109]:
```
data.isnull().sum()
```

Out[109]:
```
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 [110]:
```
cor=data.corr()
```

In [111]:
```
fig=plt.figure(figsize=(18,18))
sns.heatmap(cor,annot=True)
```

Out[111]: <AxesSubplot:>

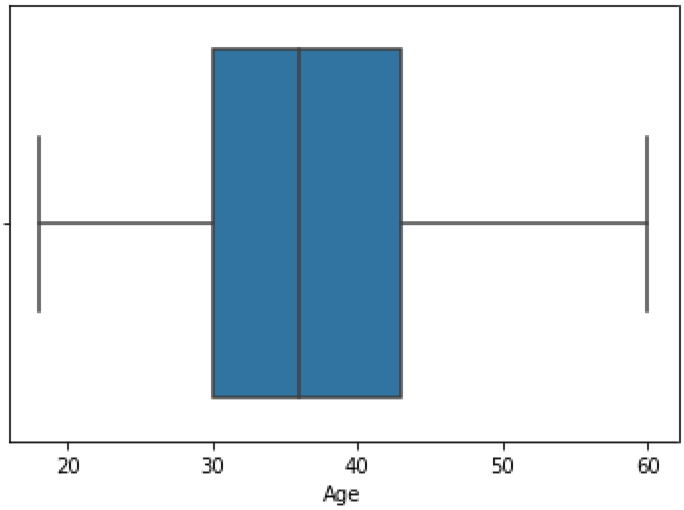


# Outliers

In [112]:
```
sns.boxplot(data["Age"])
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From v
ersion 0.12, the only valid positional argument will be `data`, and pa
ssing other arguments without an explicit keyword will result in an er
ror or misinterpretation.
  warnings.warn(
```

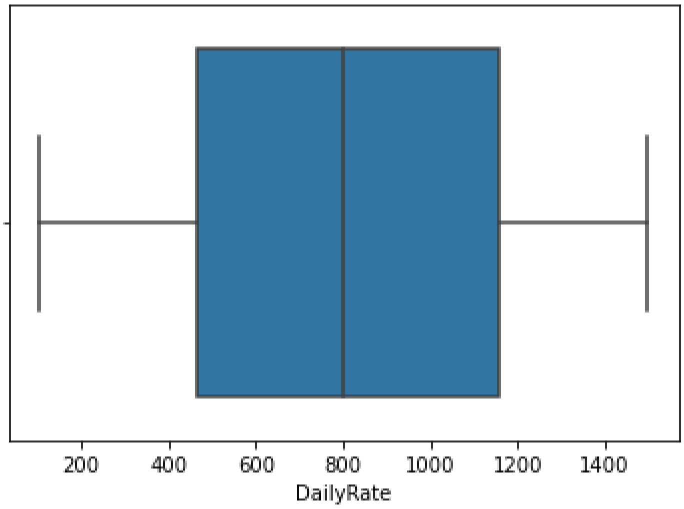
Out[112]: <AxesSubplot:xlabel='Age'>




In [113]:
```
sns.boxplot(data["DailyRate"])
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From v
ersion 0.12, the only valid positional argument will be `data`, and pa
ssing other arguments without an explicit keyword will result in an er
ror or misinterpretation.
  warnings.warn(
```

Out[113]: <AxesSubplot:xlabel='DailyRate'>



In [114]:
```
data.describe()
```

Out[114]:

| | Age | DailyRate | DistanceFromHome | Education | EmployeeCount | EmployeeNu |
|---|---|---|---|---|---|---|
| **count** | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.0 | 1470.0( |
| **mean** | 36.923810 | 802.485714 | 9.192517 | 2.912925 | 1.0 | 1024.8( |
| **std** | 9.135373 | 403.509100 | 8.106864 | 1.024165 | 0.0 | 602.02 |
| **min** | 18.000000 | 102.000000 | 1.000000 | 1.000000 | 1.0 | 1.0( |
| **25%** | 30.000000 | 465.000000 | 2.000000 | 2.000000 | 1.0 | 491.2 |
| **50%** | 36.000000 | 802.000000 | 7.000000 | 3.000000 | 1.0 | 1020.5( |
| **75%** | 43.000000 | 1157.000000 | 14.000000 | 4.000000 | 1.0 | 1555.7 |
| **max** | 60.000000 | 1499.000000 | 29.000000 | 5.000000 | 1.0 | 2068.0( |

8 rows × 26 columns

In [115]:
```
data.head()
```

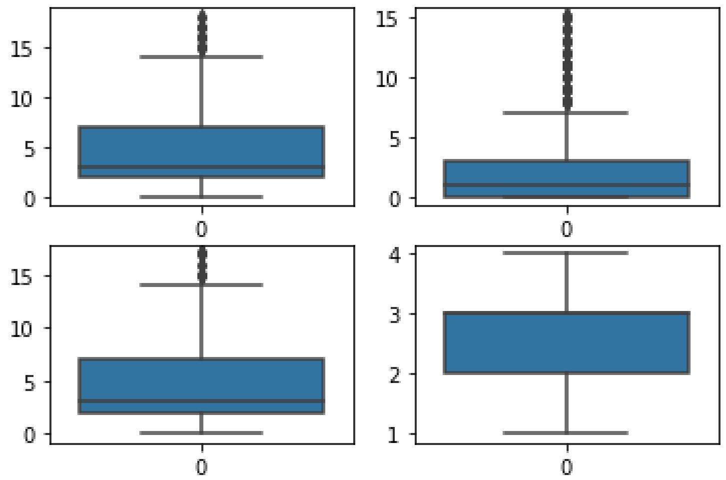
Out[115]:

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

5 rows × 35 columns

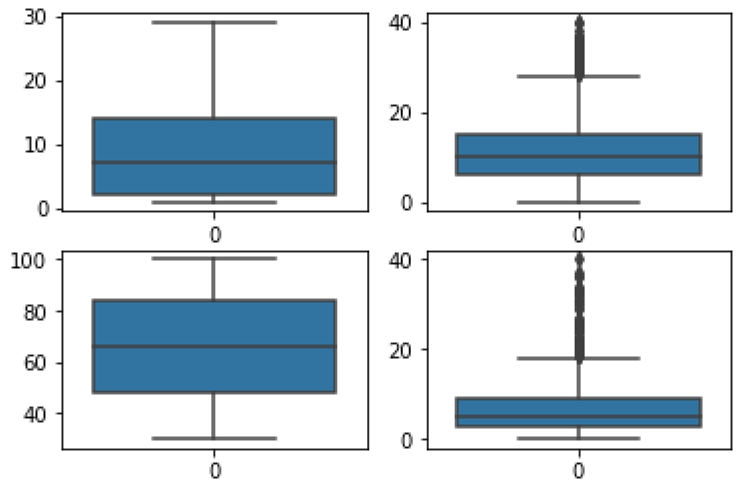
In [116]:
```
fig, axes = plt.subplots(2,2)
sns.boxplot(data=data["YearsInCurrentRole"],ax=axes[0,0])
sns.boxplot(data=data["YearsSinceLastPromotion"],ax=axes[0,1])
sns.boxplot(data=data["YearsWithCurrManager"],ax=axes[1,0])
sns.boxplot(data=data["WorkLifeBalance"],ax=axes[1,1])
```

Out[116]: <AxesSubplot:>



In [117]:
```
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[117]: <AxesSubplot:>



## Handling the Outliers

In [118]:
```
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_YearsInCurre
lower_limit_YearsInCurrentRole =YearsInCurrentRole_q1-1.5*IQR_YearsInCur
median_YearsInCurrentRole=data["YearsInCurrentRole"].median()
data['YearsInCurrentRole'] = np.where(
    (data['YearsInCurrentRole'] > upperlimit_YearsInCurrentRole),
    median_YearsInCurrentRole,
    data['YearsInCurrentRole']
)
```

In [119]:
```
YearsSinceLastPromotion_q1 = data.YearsSinceLastPromotion.quantile(0.25)
YearsSinceLastPromotion_q3 = data.YearsSinceLastPromotion.quantile(0.75)
IQR_YearsSinceLastPromotion=YearsSinceLastPromotion_q3-YearsSinceLastPro
upperlimit_YearsSinceLastPromotion=YearsSinceLastPromotion_q3+1.5*IQR_Ye
lower_limit_YearsSinceLastPromotion =YearsSinceLastPromotion_q1-1.5*IQR_
median_YearsSinceLastPromotion=data["YearsSinceLastPromotion"].median()
data['YearsSinceLastPromotion'] = np.where(
    (data['YearsSinceLastPromotion'] > upperlimit_YearsSinceLastPromotio
    median_YearsSinceLastPromotion,
    data['YearsSinceLastPromotion']
)
```

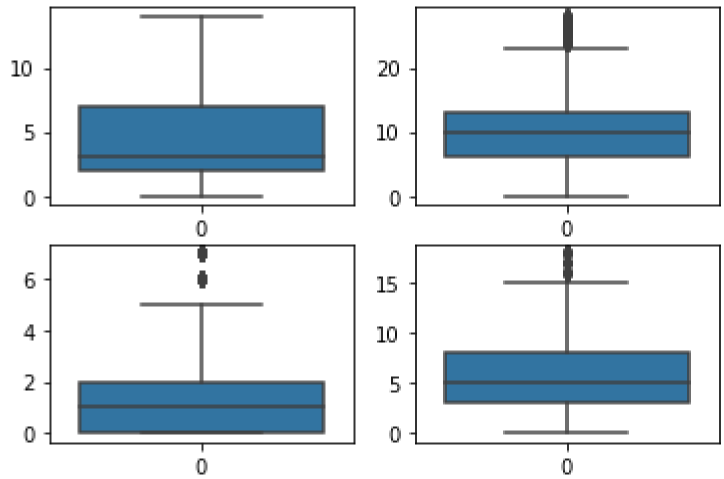
In [120]:
```
YearsWithCurrManager_q1 = data.YearsWithCurrManager.quantile(0.25)
YearsWithCurrManager_q3 = data.YearsWithCurrManager.quantile(0.75)
IQR_YearsWithCurrManager=YearsWithCurrManager_q3-YearsWithCurrManager_q1
upperlimit_YearsWithCurrManager=YearsWithCurrManager_q3+1.5*IQR_YearsWit
lower_limit_YearsWithCurrManager =YearsWithCurrManager_q1-1.5*IQR_YearsW
median_YearsWithCurrManager=data["YearsWithCurrManager"].median()
data['YearsWithCurrManager'] = np.where(
    (data['YearsWithCurrManager'] > upperlimit_YearsWithCurrManager),
    median_YearsWithCurrManager,
    data['YearsWithCurrManager']
)
```

In [121]:
```
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_TotalWorkingYe
lower_limit_TotalWorkingYears=TotalWorkingYears_q1-1.5*IQR_TotalWorkingY
median_TotalWorkingYears=data["TotalWorkingYears"].median()
data['TotalWorkingYears'] = np.where(
    (data['TotalWorkingYears'] > upperlimit_TotalWorkingYears),
    median_TotalWorkingYears,
    data['TotalWorkingYears']
)
```

In [122]:
```
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 [123]:
```
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[123]: <AxesSubplot:>




In [124]:
```
data.head()
```

Out[124]:

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

5 rows × 35 columns

In [125]:
```
data.drop("EducationField",axis=1,inplace=True)
```

In [126]:
```
data.head()
```

Out[126]:

| | 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** | 37 | Yes | Travel_Rarely | 1373 | Research & Development | 2 | 2 | |
| **3** | 33 | No | Travel_Frequently | 1392 | Research & Development | 3 | 4 | |
| **4** | 27 | No | Travel_Rarely | 591 | Research & Development | 2 | 1 | |

5 rows × 34 columns

In [127]:
```
data["BusinessTravel"].unique()
```

Out[127]:
```
array(['Travel_Rarely', 'Travel_Frequently', 'Non-Travel'], dtype=obje
ct)
```

# Splitting the data

In [128]:
```
y=data["Attrition"]
```

In [129]:
```
y.head()
```

Out[129]:
```
0    Yes
1     No
2    Yes
3     No
4     No
Name: Attrition, dtype: object
```

In [130]:
```
data.drop("Attrition",axis=1,inplace=True)
```

In [131]:
```
data.head()
```

Out[131]:

| | 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

## Encoding

In [132]:
```
from sklearn.preprocessing import LabelEncoder
```

In [133]:
```
le=LabelEncoder()
```

In [134]:
```
data["BusinessTravel"]=le.fit_transform(data["BusinessTravel"])
```

In [135]:
```
data["Department"]=le.fit_transform(data["Department"])
```

In [136]:
```
data["Gender"]=le.fit_transform(data["Gender"])
```

In [137]:
```
y=le.fit_transform(y)
```

In [138]:
```
y
```

Out[138]:
```
array([1, 0, 1, ..., 0, 0, 0])
```

In [139]:
```
data["JobRole"]=le.fit_transform(data["JobRole"])
```

In [140]:
```
data["Over18"]=le.fit_transform(data["Over18"])
```

In [141]:
```
data["MaritalStatus"]=le.fit_transform(data["MaritalStatus"])
```

In [142]:
```
data["OverTime"]=le.fit_transform(data["OverTime"])
```

In [143]:
```
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   int64  
 2   DailyRate                 1470 non-null   int64  
 3   Department                1470 non-null   int64  
 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   int64  
 10  HourlyRate                1470 non-null   int64  
 11  JobInvolvement            1470 non-null   int64  
 12  JobLevel                  1470 non-null   int64  
 13  JobRole                   1470 non-null   int64  
 14  JobSatisfaction           1470 non-null   int64  
 15  MaritalStatus             1470 non-null   int64  
 16  MonthlyIncome             1470 non-null   int64  
 17  MonthlyRate               1470 non-null   int64  
 18  NumCompaniesWorked        1470 non-null   int64  
 19  Over18                    1470 non-null   int64  
 20  OverTime                  1470 non-null   int64  
 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), int64(28)
memory usage: 379.1 KB
```

# Train Test Split

In [144]:
```
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,rand
```

In [145]:
```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[145]:
```
((1029, 33), (441, 33), (1029,), (441,))
```

# Featuring Scaling

In [146]:
```
from sklearn.preprocessing import StandardScaler
```

In [147]:
```
sc=StandardScaler()
```

In [148]:
```
x_train=sc.fit_transform(x_train)
```

In [149]:
```
x_test=sc.fit_transform(x_test)
```

# Building the model

## Multi Linear Regression

In [150]:
```
from sklearn.linear_model import LinearRegression
```

In [151]:
```
lr = LinearRegression()
```

In [152]:
```
lr.fit(x_train,y_train)
```

Out[152]:
```
LinearRegression()
```

In [153]:
```
lr.coef_  #slope(m)
```

Out[153]:
```
array([-3.54940447e-02,  7.88352347e-05, -1.70825038e-02,  3.46389690e
-02,
        2.44612841e-02,  3.65668214e-03, -2.50667542e-16, -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,
        2.60208521e-18, -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 [154]:
```
lr.intercept_  #(c)
```

Out[154]:
```
0.16229348882410102
```

In [155]:
```
y_pred = lr.predict(x_test)
```

In [156]:
```
y_pred
```

```
-01,
        6.66728668e-02,  4.49620331e-02,  3.30502696e-01,  9.74393000e
-02,
        5.51447175e-01,  1.52212203e-01,  3.58819339e-01,  3.66371593e
-01,
        2.47091987e-01,  5.86970935e-02,  1.28678988e-01,  2.80584025e
-01,
        7.21059443e-02, -8.07006907e-02,  3.39791632e-01,  8.25270203e
-02,
        2.20338157e-01,  2.47703594e-01,  4.97067397e-01,  1.36010592e
-01,
        2.88153807e-01,  4.61306498e-02,  4.52544344e-01, -8.24037634e
-02,
        2.26796295e-01,  1.42129836e-02,  1.62111340e-01,  2.32246950e
-01,
        9.12503556e-02,  1.18866795e-01,  2.12735292e-01, -2.69559828e
-02,
        4.53611463e-02,  1.09618223e-01,  2.64436901e-02,  2.32180310e
-01,
        1.63285101e-01,  2.42669261e-01,  5.44757533e-01,  1.25881866e
```

In [157]:
```
y_test
```

Out[157]:
```
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 [158]:
```
from sklearn.linear_model import LogisticRegression
```

In [159]:
```
lg=LogisticRegression()
```

In [160]:
```
lg.fit(x_train,y_train)
```

Out[160]:
```
LogisticRegression()
```

In [161]:
```
y_pred_lg=lg.predict(x_test)
```

In [162]:
```
y_pred
```

Out[162]:
```
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
```

In [163]:
```
y_test
```

Out[163]:
```
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 [164]:
```
score = lg.score(x_test, y_test)
print(score)
```

```
0.8820861678004536
```

# Confusion Matrix

In [165]:
```
from sklearn import metrics
cm = metrics.confusion_matrix(y_test,y_pred_lg)
print(cm)
```

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

## Ridge and Lasso

In [166]:
```
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
```

In [167]:
```
rg=Ridge()
```

In [168]:
```
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[168]:
```
GridSearchCV(cv=5, estimator=Ridge(),
             param_grid={'alpha': [1, 2, 3, 5, 10, 20, 30, 40, 60, 70,
80, 90]},
             scoring='neg_mean_squared_error')
```

In [169]:
```
print(ridgecv.best_params_)
```

```
{'alpha': 90}
```

In [170]:
```
print(ridgecv.best_score_)
```

```
-0.11390621139234183
```

In [171]:
```
y_pred_rg=ridgecv.predict(x_test)
```

In [172]:
```
y_pred_rg
```

Out[172]:
```
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
```

In [173]:
```
y_test
```

Out[173]:
```
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 [174]:
```
from sklearn import metrics
print(metrics.r2_score(y_test,y_pred_rg))
print(metrics.r2_score(y_train,ridgecv.predict(x_train)))
```

```
0.21073458438815873
0.2061567210285108
```

# Lasso

In [175]:
```
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV
```

In [176]:
```
la=Ridge()
```

In [177]:
```
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[177]:
```
GridSearchCV(cv=5, estimator=Ridge(),
             param_grid={'alpha': [1, 2, 3, 5, 10, 20, 30, 40, 60, 70,
80, 90]},
             scoring='neg_mean_squared_error')
```

In [178]:
```
print(ridgecv.best_params_)
```

```
{'alpha': 90}
```

In [179]:
```
print(ridgecv.best_score_)
```

```
-0.11390621139234183
```

In [180]:
```
y_pred_la=ridgecv.predict(x_test)
```

In [181]:
```
y_pred_la
```

```
        7.50317322e-02,  1.67646673e-01,  1.16585544e-01,  1.07157808e
-01,
       -1.84689359e-02,  1.86217544e-01,  1.16586463e-01,  4.67201201e
-02,
        1.11060472e-01,  2.27053971e-01, -7.00247692e-02, -5.81070776e
-02,
        2.03141688e-01,  4.69029664e-02,  1.31525768e-01,  5.66738022e
-01,
        2.41883060e-02, -3.41250985e-02, -1.13904557e-01,  2.18572744e
-01,
        2.60568042e-01,  1.65533667e-01, -5.94078459e-05,  2.60009384e
-01,
        4.20709666e-01,  3.71031267e-01,  1.70250288e-01,  4.03052216e
-01,
        4.67312765e-01,  1.98845366e-01,  1.55005619e-01,  3.41505080e
-01,
        2.20024496e-01,  1.40989758e-01,  1.97796963e-01,  2.57841889e
-01,
        2.99122317e-01,  9.24907038e-03,  1.39162817e-01, -1.13916709e
-01,
```

In [182]:
```
from sklearn import metrics
print(metrics.r2_score(y_test,y_pred_la))
print(metrics.r2_score(y_train,ridgecv.predict(x_train)))
```

```
0.21073458438815873
0.2061567210285108
```

## Decision Tree

In [183]:
```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
```

In [184]:
```
dtc.fit(x_train,y_train)
```

Out[184]:
```
DecisionTreeClassifier()
```

In [185]:
```
pred=dtc.predict(x_test)
```

In [186]:
```
pred
```

Out[186]:
```
array([0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
0,
       0, 0, 1, 0, 0, 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, 1, 0, 1, 0, 0, 0, 0, 1, 0, 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, 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, 1, 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, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 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, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 1, 0, 0, 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, 1, 1, 0,
1,
       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, 1,
1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 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, 0, 1, 0, 1, 1, 0,
0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 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, 0, 0, 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, 1, 1, 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])
```

In [187]:
```
y_test
```

Out[187]:
```
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 [188]:
```
#Accuracy score
from sklearn.metrics import accuracy_score,confusion_matrix,classificati
```

In [189]:
```
accuracy_score(y_test,pred)
```

Out[189]:
```
0.7664399092970522
```

In [190]:
```
confusion_matrix(y_test,pred)
```

Out[190]:
```
array([[320,  51],
       [ 52,  18]])
```

In [191]:
```
pd.crosstab(y_test,pred)
```

Out[191]:

| col_0 | 0 | 1 |
|---|---|---|
| **row_0** | | |
| **0** | 320 | 51 |
| **1** | 52 | 18 |

In [192]:
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.86      0.86       371
           1       0.26      0.26      0.26        70

    accuracy                           0.77       441
   macro avg       0.56      0.56      0.56       441
weighted avg       0.77      0.77      0.77       441
```

In [193]:
```
probability=dtc.predict_proba(x_test)[:,1]
```

In [194]:
```
# roc_curve
fpr,tpr,threshsholds = roc_curve(y_test,probability)
```

In [195]:
```
plt.plot(fpr,tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC CURVE')
plt.show()
```