In [1]:
```
print("Name: RAMISETTY PAVANI")
print("Registration Number: 21BCE9521")
print("Morning Batch")
```

```
Name: RAMISETTY PAVANI
Registration Number: 21BCE9521
Morning Batch
```

Data preprocessing-

import lib, import dataset, check null values, data visualixation, outlier detection, splitting dependent and independent, encoding, feature scaling, splitting into test and train

In [2]:
```
#importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
#importing the dataset
df=pd.read_csv("Employee-Attrition.csv")
df.head()
```

Out[3]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	E
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 × 35 columns

In [4]:
```
df.shape
```

Out[4]:
```
(1470, 35)
```

In [5]:
```
df.Attrition.value_counts()
```

Out[5]:
```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

In [6]:
```
df.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 [7]:
```
df.describe()
```

Out[7]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	Employe
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	147
mean	36.923810	802.485714	9.192517	2.912925	1.0	102
std	9.135373	403.509100	8.106864	1.024165	0.0	60
min	18.000000	102.000000	1.000000	1.000000	1.0	
25%	30.000000	465.000000	2.000000	2.000000	1.0	49
50%	36.000000	802.000000	7.000000	3.000000	1.0	102
75%	43.000000	1157.000000	14.000000	4.000000	1.0	155
max	60.000000	1499.000000	29.000000	5.000000	1.0	206

8 rows × 26 columns

In [8]:
```
#checking for null values
df.isnull().any()
```

Out[8]:
```
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
```

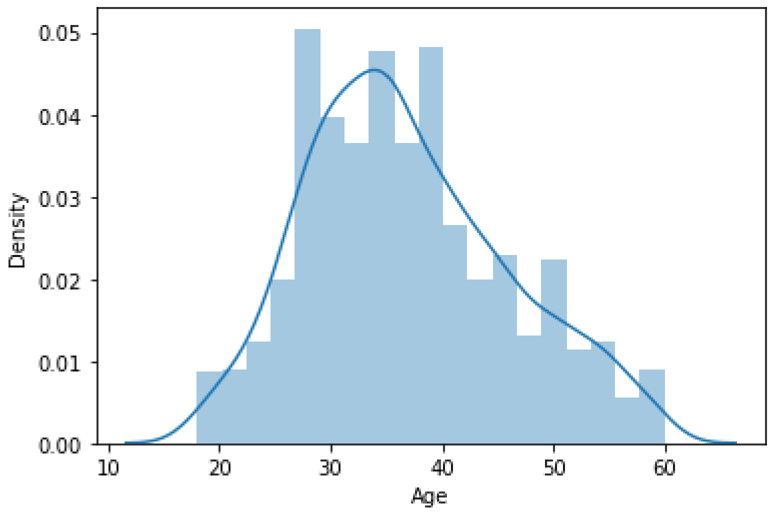
In [9]:
```
df.isnull().sum()
```

Out[9]:
```
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 [10]:
```
#data visualization
sns.distplot(df["Age"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[10]: <AxesSubplot:xlabel='Age', ylabel='Density'>



In [11]:
```
df.corr()
```

Out[11]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCou
Age	1.000000	0.010661	-0.001686	0.208034	Na
DailyRate	0.010661	1.000000	-0.004985	-0.016806	Na
DistanceFromHome	-0.001686	-0.004985	1.000000	0.021042	Na
Education	0.208034	-0.016806	0.021042	1.000000	Na
EmployeeCount	NaN	NaN	NaN	NaN	Na
EmployeeNumber	-0.010145	-0.050990	0.032916	0.042070	Na
EnvironmentSatisfaction	0.010146	0.018355	-0.016075	-0.027128	Na
HourlyRate	0.024287	0.023381	0.031131	0.016775	Na
JobInvolvement	0.029820	0.046135	0.008783	0.042438	Na
JobLevel	0.509604	0.002966	0.005303	0.101589	Na
JobSatisfaction	-0.004892	0.030571	-0.003669	-0.011296	Na
MonthlyIncome	0.497855	0.007707	-0.017014	0.094961	Na
MonthlyRate	0.028051	-0.032182	0.027473	-0.026084	Na
NumCompaniesWorked	0.299635	0.038153	-0.029251	0.126317	Na
PercentSalaryHike	0.003634	0.022704	0.040235	-0.011111	Na
PerformanceRating	0.001904	0.000473	0.027110	-0.024539	Na
RelationshipSatisfaction	0.053535	0.007846	0.006557	-0.009118	Na
StandardHours	NaN	NaN	NaN	NaN	Na
StockOptionLevel	0.037510	0.042143	0.044872	0.018422	Na
TotalWorkingYears	0.680381	0.014515	0.004628	0.148280	Na
TrainingTimesLastYear	-0.019621	0.002453	-0.036942	-0.025100	Na
WorkLifeBalance	-0.021490	-0.037848	-0.026556	0.009819	Na
YearsAtCompany	0.311309	-0.034055	0.009508	0.069114	Na
YearsInCurrentRole	0.212901	0.009932	0.018845	0.060236	Na
YearsSinceLastPromotion	0.216513	-0.033229	0.010029	0.054254	Na
YearsWithCurrManager	0.202089	-0.026363	0.014406	0.069065	Na

26 rows × 26 columns

In [12]:
```
df.head()
```

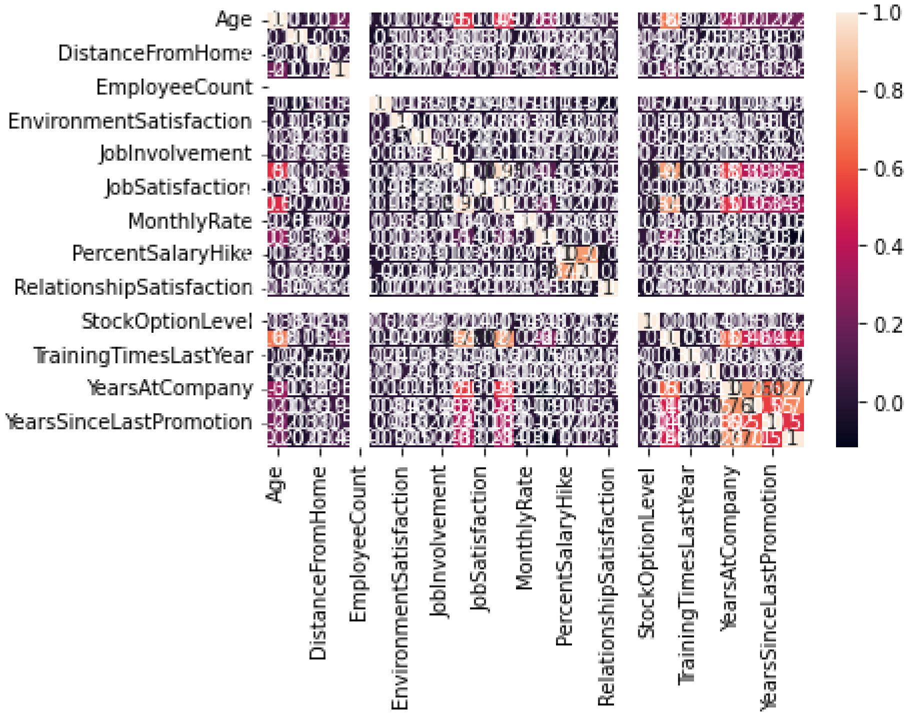
Out[12]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	E
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 × 35 columns

In [13]:
```
sns.heatmap(df.corr(),annot=True)
```

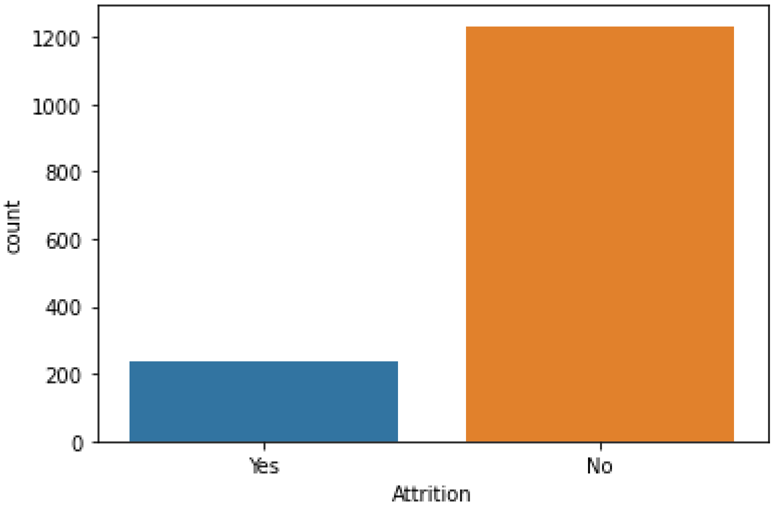
Out[13]: <AxesSubplot:>



In [14]:
```
sns.countplot(df.Attrition)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

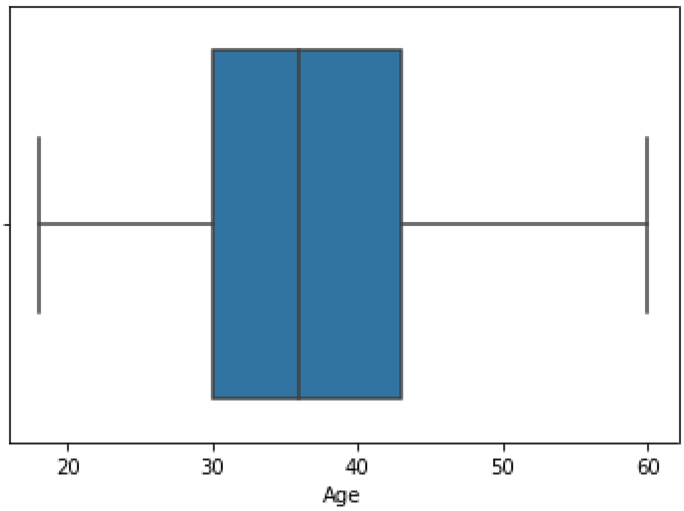
Out[14]:
```
<AxesSubplot:xlabel='Attrition', ylabel='count'>
```



In [15]:
```
#outlier detection
sns.boxplot(df['Age'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

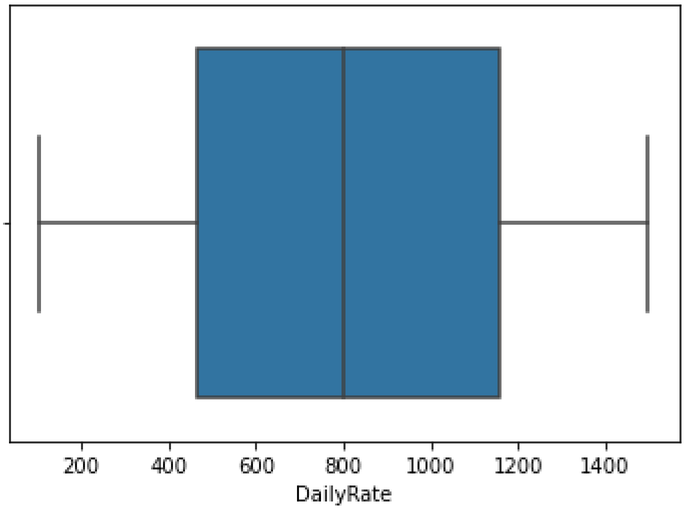
Out[15]:
```
<AxesSubplot:xlabel='Age'>
```



In [16]:
```
sns.boxplot(df['DailyRate'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

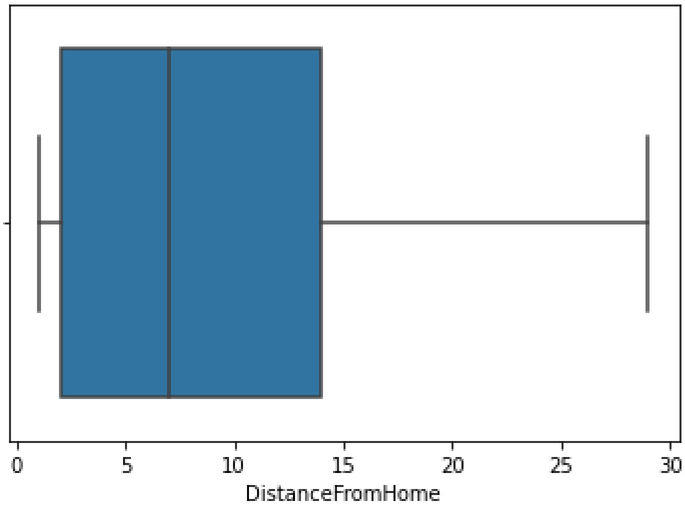
Out[16]: <AxesSubplot:xlabel='DailyRate'>



In [17]:
```
sns.boxplot(df['DistanceFromHome'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

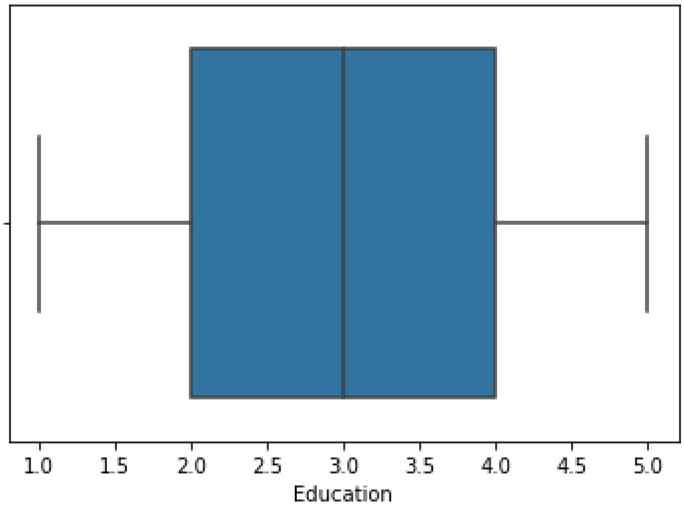
Out[17]: <AxesSubplot:xlabel='DistanceFromHome'>



In [18]:
```
sns.boxplot(df['Education'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

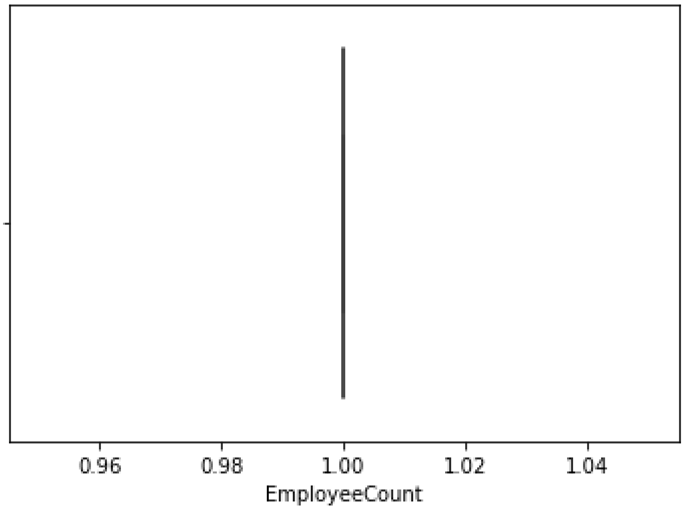
Out[18]: <AxesSubplot:xlabel='Education'>



In [19]:
```
sns.boxplot(df['EmployeeCount'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

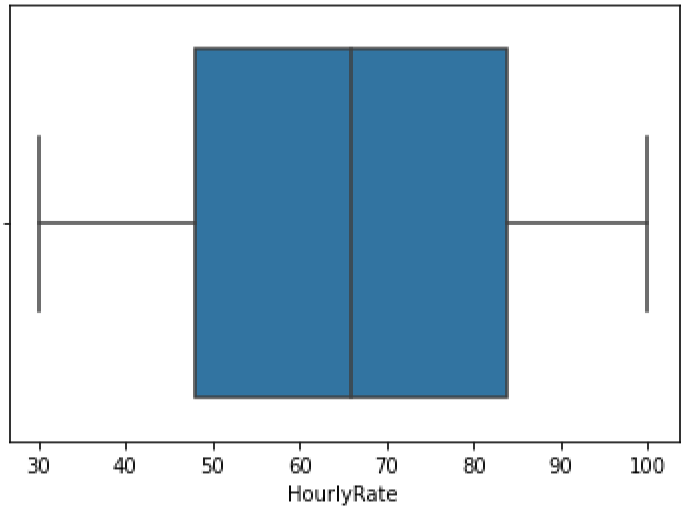
Out[19]: <AxesSubplot:xlabel='EmployeeCount'>



In [20]:
```
sns.boxplot(df['HourlyRate'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

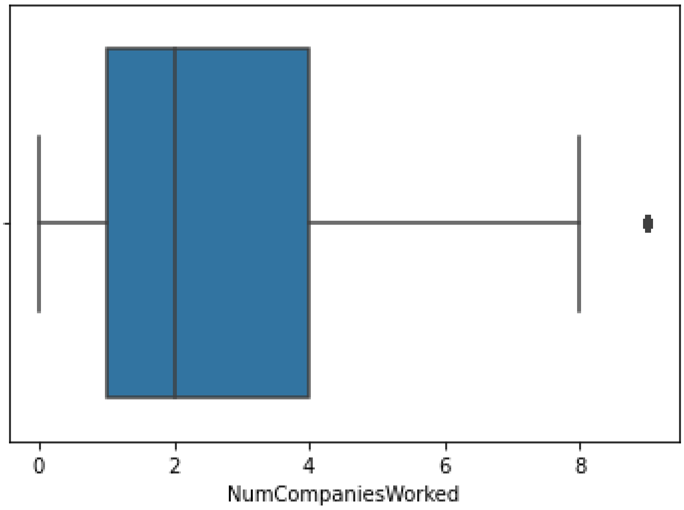
Out[20]: <AxesSubplot:xlabel='HourlyRate'>



In [21]:
```
sns.boxplot(df['NumCompaniesWorked'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

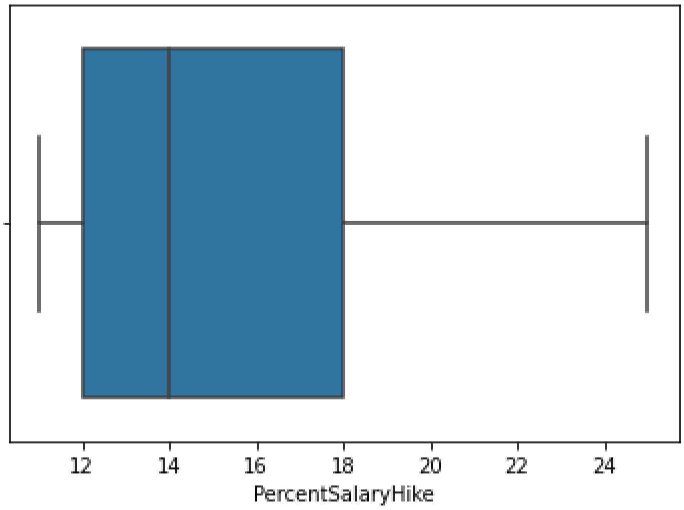
Out[21]: <AxesSubplot:xlabel='NumCompaniesWorked'>



In [22]:
```
sns.boxplot(df['PercentSalaryHike'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

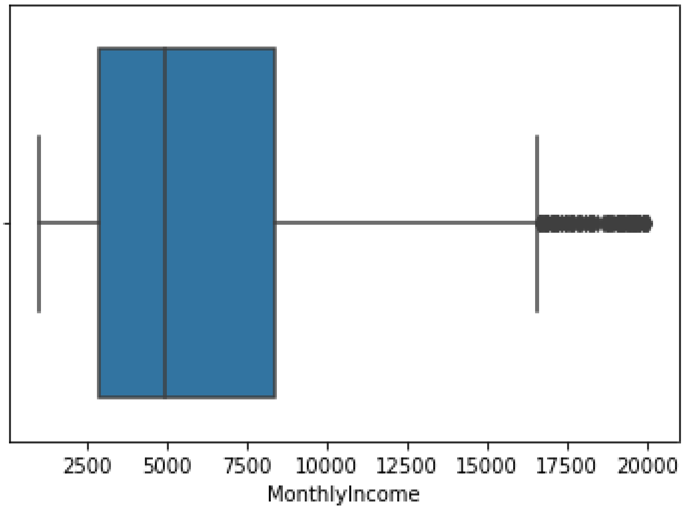
Out[22]: <AxesSubplot:xlabel='PercentSalaryHike'>



In [23]:
```
sns.boxplot(df['MonthlyIncome'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

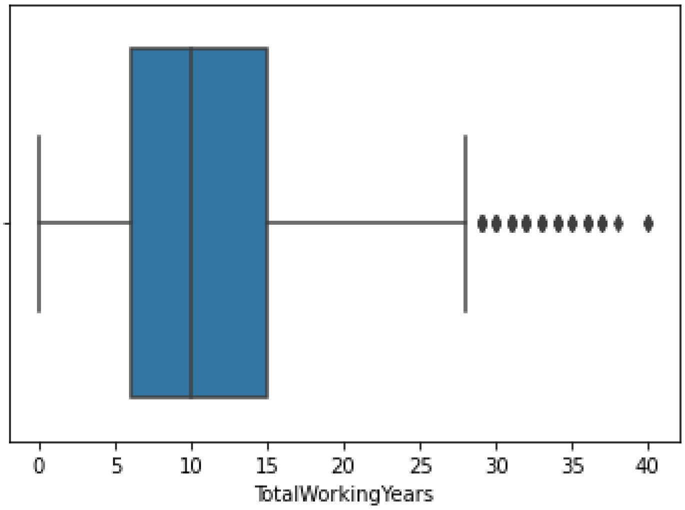
Out[23]: <AxesSubplot:xlabel='MonthlyIncome'>



In [24]:
```
sns.boxplot(df['TotalWorkingYears'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

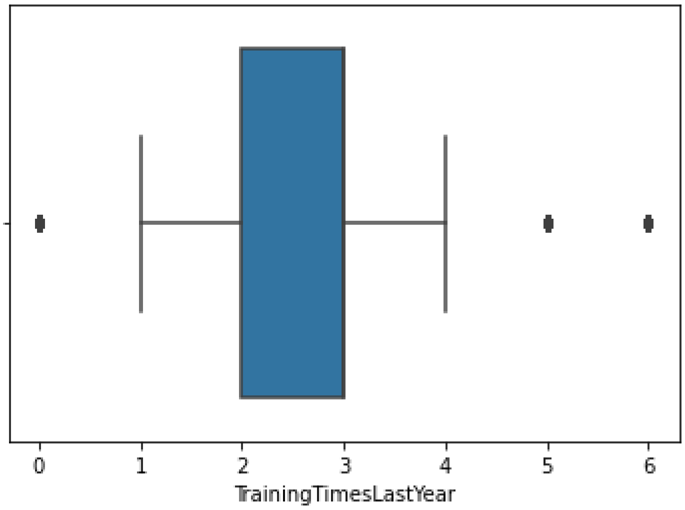
Out[24]: <AxesSubplot:xlabel='TotalWorkingYears'>



In [25]:
```
sns.boxplot(df['TrainingTimesLastYear'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

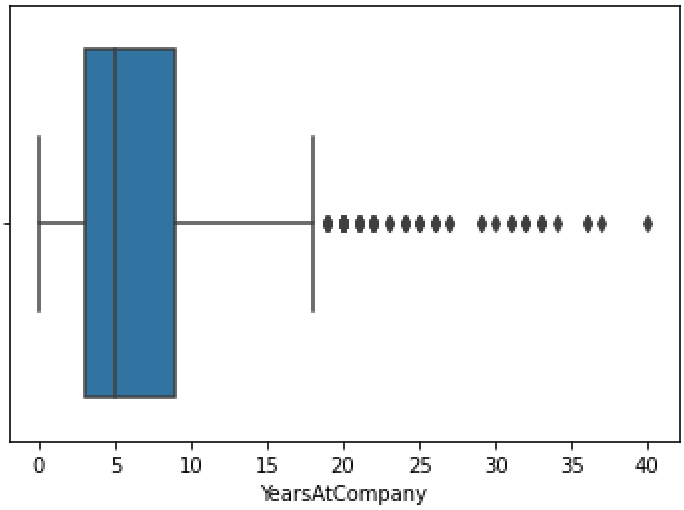
Out[25]: <AxesSubplot:xlabel='TrainingTimesLastYear'>



In [26]:
```
sns.boxplot(df['YearsAtCompany'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

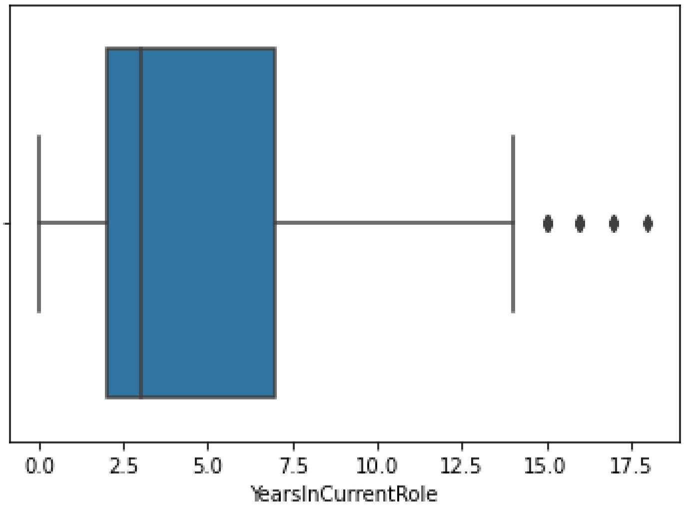
Out[26]: <AxesSubplot:xlabel='YearsAtCompany'>



In [27]:
```
sns.boxplot(df['YearsInCurrentRole'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

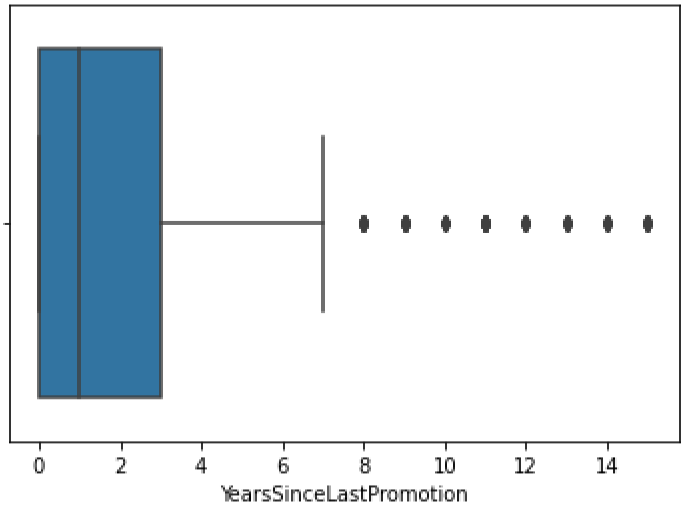
Out[27]: <AxesSubplot:xlabel='YearsInCurrentRole'>



In [28]:
```
sns.boxplot(df['YearsSinceLastPromotion'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

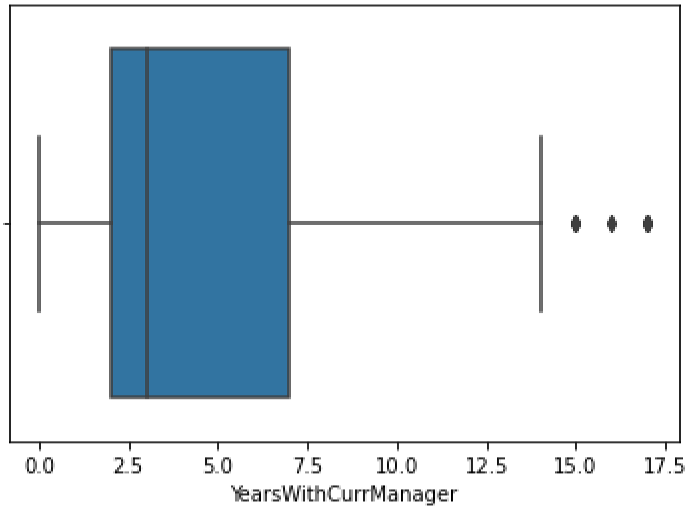
Out[28]: <AxesSubplot:xlabel='YearsSinceLastPromotion'>



In [29]:
```
sns.boxplot(df['YearsWithCurrManager'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

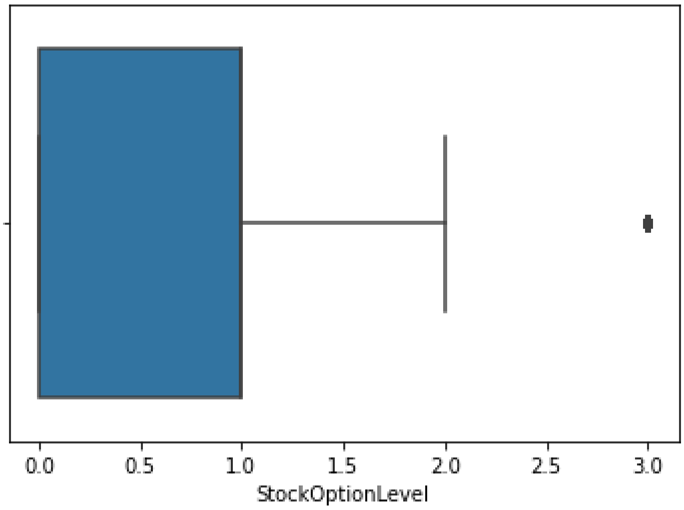
Out[29]: <AxesSubplot:xlabel='YearsWithCurrManager'>



In [30]:
```
sns.boxplot(df['StockOptionLevel'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

Out[30]: <AxesSubplot:xlabel='StockOptionLevel'>



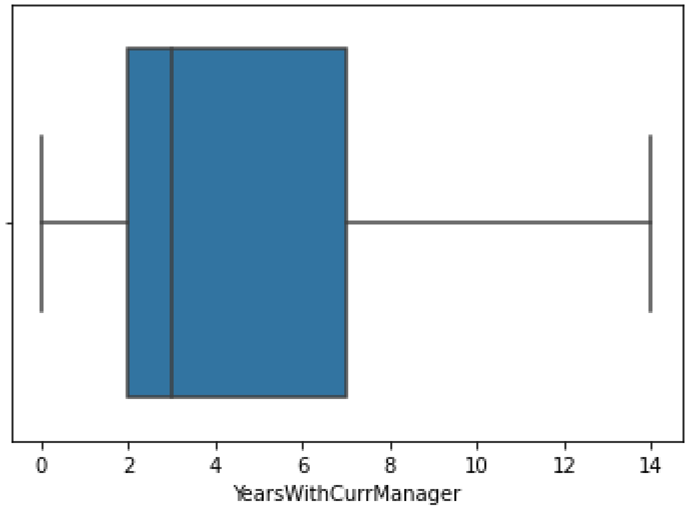
In [31]:
```
#YearsWithCurrManager
q1=df.YearsWithCurrManager.quantile(0.25)
q3=df.YearsWithCurrManager.quantile(0.75)
IQR=q3-q1
IQR
```

Out[31]: 5.0

In [32]:
```
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
df=df[ (df['YearsWithCurrManager']<upper_limit ) ]
sns.boxplot(df['YearsWithCurrManager'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

Out[32]: <AxesSubplot:xlabel='YearsWithCurrManager'>



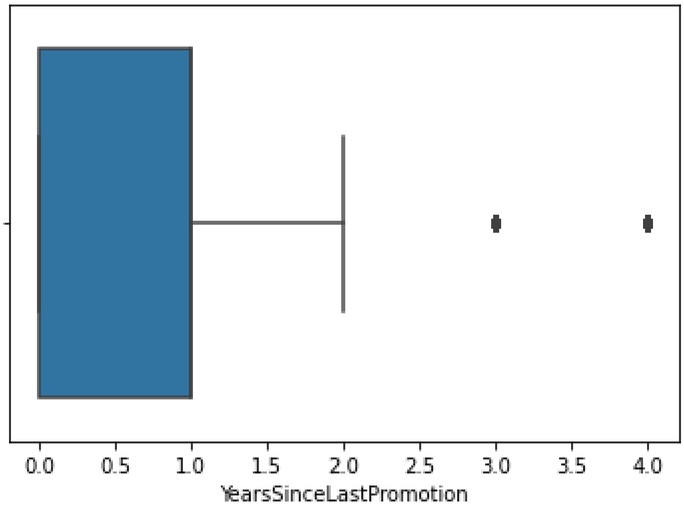
In [33]:
```
#YearsSinceLAstPromotion
q1=df.YearsSinceLastPromotion.quantile(0.25)
q3=df.YearsSinceLastPromotion.quantile(0.75)
IQR=q3-q1
IQR
```

Out[33]: 2.0

In [34]:
```
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
df=df[ (df['YearsSinceLastPromotion']<upper_limit ) ]
sns.boxplot(df['YearsSinceLastPromotion'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

Out[34]: <AxesSubplot:xlabel='YearsSinceLastPromotion'>



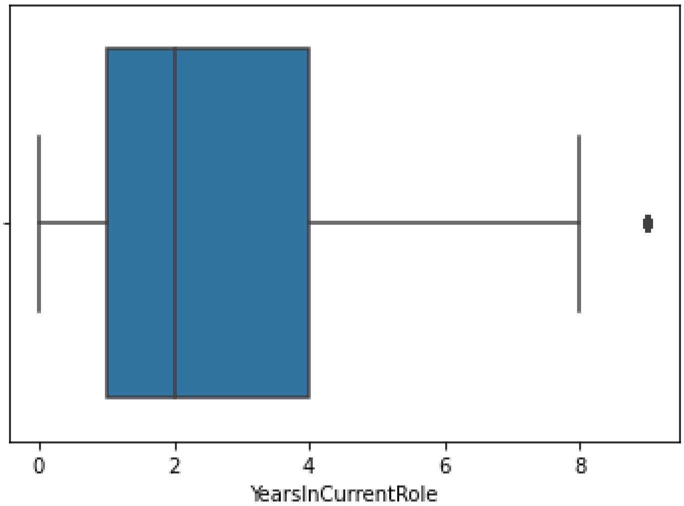
In [35]:
```
#YearsInCurrentRole
q1=df.YearsInCurrentRole.quantile(0.25)
q3=df.YearsInCurrentRole.quantile(0.75)
IQR=q3-q1
IQR
```

Out[35]: 3.0

In [36]:
```
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
df=df[ (df['YearsInCurrentRole']<upper_limit ) ]
sns.boxplot(df['YearsInCurrentRole'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[36]: <AxesSubplot:xlabel='YearsInCurrentRole'>



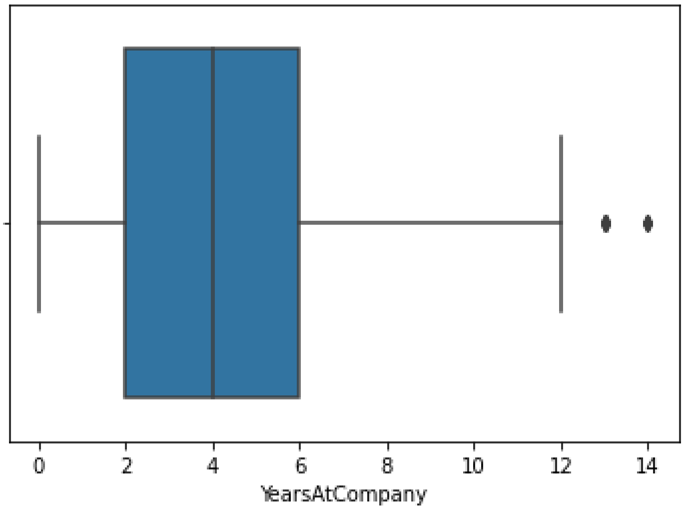
In [37]:
```
#YearsAtCompany
q1=df.YearsAtCompany.quantile(0.25)
q3=df.YearsAtCompany.quantile(0.75)
IQR=q3-q1
IQR
```

Out[37]: 5.0

In [38]:
```
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
df=df[ (df['YearsAtCompany']<upper_limit ) ]
sns.boxplot(df['YearsAtCompany'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[38]: <AxesSubplot:xlabel='YearsAtCompany'>



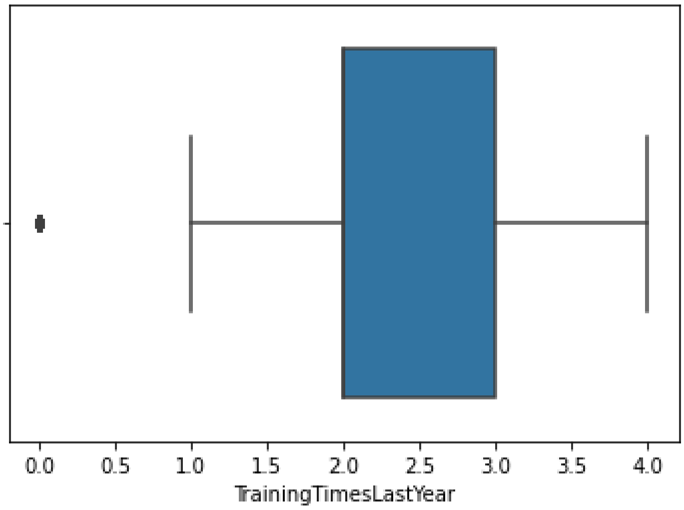
In [39]:
```
#TrainingTimesLastYear
q1=df.TrainingTimesLastYear.quantile(0.25)
q3=df.TrainingTimesLastYear.quantile(0.75)
IQR=q3-q1
IQR
```

Out[39]: 1.0

In [40]:
```
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
df=df[ (df['TrainingTimesLastYear']<upper_limit ) ]
sns.boxplot(df['TrainingTimesLastYear'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

Out[40]: <AxesSubplot:xlabel='TrainingTimesLastYear'>



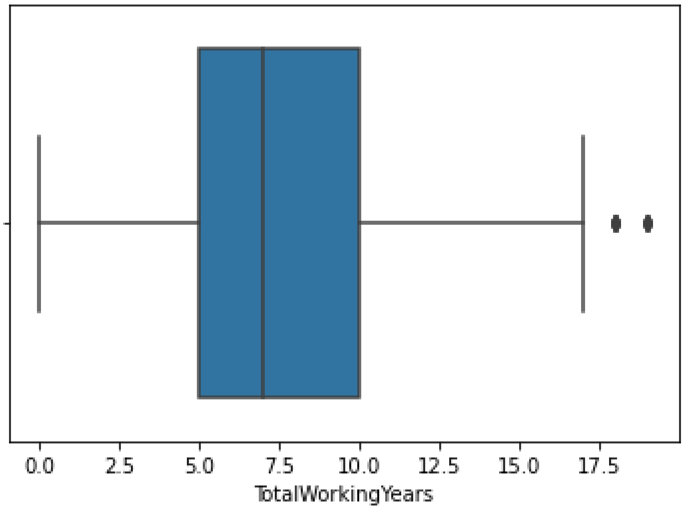
In [41]:
```
#TotalWorkingYears
q1=df.TotalWorkingYears.quantile(0.25)
q3=df.TotalWorkingYears.quantile(0.75)
IQR=q3-q1
IQR
```

Out[41]: 6.0

In [42]:
```
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
df=df[ (df['TotalWorkingYears']<upper_limit ) ]
sns.boxplot(df['TotalWorkingYears'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[42]: <AxesSubplot:xlabel='TotalWorkingYears'>



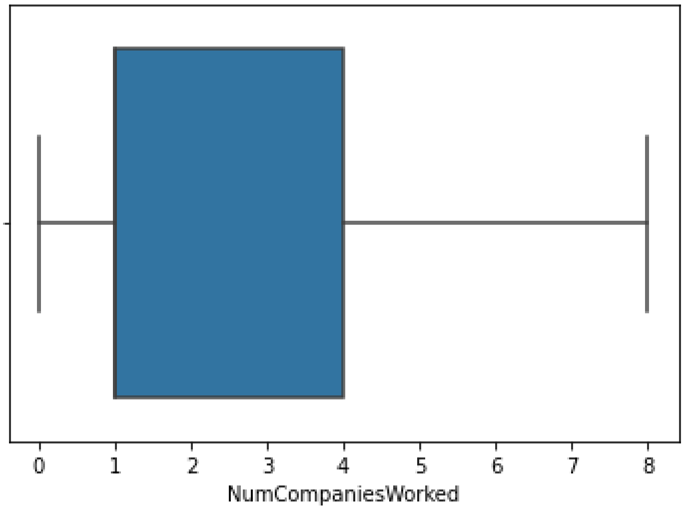
In [43]:
```
#NumCompaniesWorked
q1=df.NumCompaniesWorked.quantile(0.25)
q3=df.NumCompaniesWorked.quantile(0.75)
IQR=q3-q1
IQR
```

Out[43]: 3.0

In [44]:
```
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
df=df[ (df['NumCompaniesWorked']<upper_limit ) ]
sns.boxplot(df['NumCompaniesWorked'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(
```

Out[44]: <AxesSubplot:xlabel='NumCompaniesWorked'>



In [45]:
```
df.head()
```

Out[45]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	E
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	
5	32	No	Travel_Frequently	1005	Research & Development	2	2	

5 rows × 35 columns

In [46]:
```
#label encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df.Attrition=le.fit_transform(df.Attrition)
df.head()
```

Out[46]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	E
0	41	1	Travel_Rarely	1102	Sales	1	2	
1	49	0	Travel_Frequently	279	Research & Development	8	1	
2	37	1	Travel_Rarely	1373	Research & Development	2	2	
3	33	0	Travel_Frequently	1392	Research & Development	3	4	
5	32	0	Travel_Frequently	1005	Research & Development	2	2	

5 rows × 35 columns

In [47]:
```
#splitting dependent and independent variables
x=df.iloc[:,:]
x=x.drop(columns=["Attrition","EmployeeCount","StandardHours","Over18"],axi
x.head()
```

Out[47]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationF
0	41	Travel_Rarely	1102	Sales	1	2	Life Scier
1	49	Travel_Frequently	279	Research & Development	8	1	Life Scier
2	37	Travel_Rarely	1373	Research & Development	2	2	O
3	33	Travel_Frequently	1392	Research & Development	3	4	Life Scier
5	32	Travel_Frequently	1005	Research & Development	2	2	Life Scier

5 rows × 31 columns

In [48]:
```
y=df["Attrition"]
y.head()
```

Out[48]:
```
0    1
1    0
2    1
3    0
5    0
Name: Attrition, dtype: int32
```

In [49]:
```
#encoding
x_encoded=pd.get_dummies(x,columns=["Department","BusinessTravel","Educatio
```

In [50]:
```
x.drop(["Department","BusinessTravel","EducationField","Gender","OverTime",
x.head()
```

Out[50]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeNumber	EnvironmentSatisfaction
0	41	1102	1	2	1	2
1	49	279	8	1	2	3
2	37	1373	2	2	4	4
3	33	1392	3	4	5	4
5	32	1005	2	2	8	4

5 rows × 24 columns

In [51]:
```
x=pd.concat([x,x_encoded], axis=1)
x.head()
```

Out[51]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeNumber	EnvironmentSatisfaction
0	41	1102	1	2	1	2
1	49	279	8	1	2	3
2	37	1373	2	2	4	4
3	33	1392	3	4	5	4
5	32	1005	2	2	8	4

5 rows × 69 columns

In [52]:
```
#feature scaling
from sklearn.preprocessing import MinMaxScaler
ms=MinMaxScaler()
x_scaled=pd.DataFrame(ms.fit_transform(x), columns=x.columns)
x_scaled.head()
```

Out[52]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeNumber	EnvironmentSatisfa
0	0.547619	0.716129	0.000000	0.25	0.000000	0.33
1	0.738095	0.126165	0.250000	0.00	0.000484	0.66
2	0.452381	0.910394	0.035714	0.25	0.001451	1.00
3	0.357143	0.924014	0.071429	0.75	0.001935	1.00
4	0.333333	0.646595	0.035714	0.25	0.003387	1.00

5 rows × 69 columns

In [53]:
```
#splitting data into train and test
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(x_scaled,y,test_size=0.2,
```

In [54]:
```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[54]:
```
((687, 69), (172, 69), (687,), (172,))
```

model building-

import model builing lib, intialize the model, train and tst he model, evaluation of model, save the model

In [55]:
```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
```

In [56]:
```
model.fit(x_train,y_train)
```

Out[56]:
```
LogisticRegression()
```

In [57]:
```
y_pred=model.predict(x_test)
y_pred
```

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

In [58]:
```
y_test
```

Out[58]:
```
239     1
347     0
472     0
450     0
911     1
       ..
1246    1
255     0
342     0
1077    1
80      0
Name: Attrition, Length: 172, dtype: int32
```

In [59]:
```
df
```

Out[59]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educatior
0	41	1	Travel_Rarely	1102	Sales	1	2
1	49	0	Travel_Frequently	279	Research & Development	8	1
2	37	1	Travel_Rarely	1373	Research & Development	2	2
3	33	0	Travel_Frequently	1392	Research & Development	3	4
5	32	0	Travel_Frequently	1005	Research & Development	2	2
...	...	...	...	...	...	...	..
1464	26	0	Travel_Rarely	1167	Sales	5	3
1465	36	0	Travel_Frequently	884	Research & Development	23	2
1467	27	0	Travel_Rarely	155	Research & Development	4	3
1468	49	0	Travel_Frequently	1023	Sales	2	3
1469	34	0	Travel_Rarely	628	Research & Development	8	3

859 rows × 35 columns

evaluation of the classification model

In [61]:
```
#Accuracy core
from sklearn.metrics import accuracy_score, confusion_matrix, classificatio
```

In [62]:
```
accuracy_score(y_test, y_pred)
```

Out[62]:
```
0.8546511627906976
```

In [63]:
```
confusion_matrix(y_test, y_pred)
```

Out[63]:
```
array([[134,   5],
       [ 20,  13]], dtype=int64)
```

In [64]:
```
pd.crosstab(y_test,y_pred)
```

Out[64]:

col_0	0	1
Attrition		
0	134	5
1	20	13

In [65]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.96      0.91       139
           1       0.72      0.39      0.51        33

    accuracy                           0.85       172
   macro avg       0.80      0.68      0.71       172
weighted avg       0.84      0.85      0.84       172
```

In [66]:
```
#precision
p=(13/(13+5))
p
```

Out[66]:
```
0.7222222222222222
```

In [67]:
```
#recall
r=(13/(13+20))
r
```

Out[67]:
```
0.3939393939393939
```

In [68]:
```
#f1 score
f=2*((p*r)/(p+r))
f
```

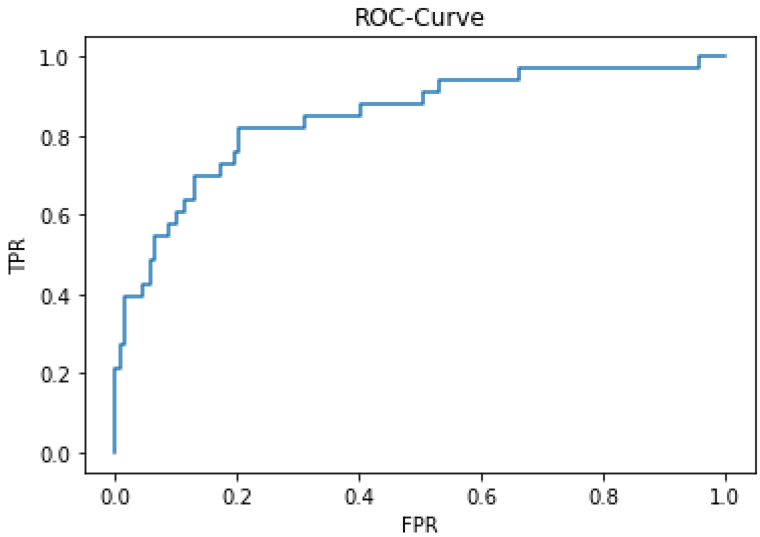
Out[68]:
```
0.5098039215686274
```

In [69]:
```
#ROC-AUC curve
probability=model.predict_proba(x_test)[:,1]
probability
```

Out[69]:
```
array([3.97281674e-01, 5.62736117e-01, 5.77506360e-02, 1.05521818e-01,
       9.26868534e-01, 1.55171106e-01, 9.45922656e-02, 1.46274892e-02,
       1.50861782e-02, 5.94576854e-02, 1.25660609e-01, 4.72841103e-02,
       6.55426277e-02, 6.30656817e-01, 3.09607430e-01, 3.38313637e-01,
       2.54332944e-01, 1.33713778e-01, 4.30317069e-02, 2.35945088e-02,
       1.41532406e-01, 2.22577704e-01, 3.38351888e-02, 1.27569054e-01,
       3.63912488e-03, 6.95357964e-01, 2.90566718e-01, 6.91645725e-02,
       1.63799321e-01, 1.22492053e-01, 1.84055099e-02, 3.32219071e-01,
       4.27330967e-01, 4.02784568e-01, 3.02454605e-03, 5.63018054e-02,
       5.86361846e-01, 3.89876300e-02, 5.87982682e-01, 7.86883409e-01,
       1.87614581e-03, 2.68437438e-02, 1.83211889e-02, 4.40489552e-01,
       1.52867340e-01, 3.99398040e-02, 3.23743980e-01, 1.95209463e-01,
       4.27518183e-02, 1.75277200e-02, 6.79584395e-02, 1.70080609e-01,
       3.62929988e-01, 9.24978482e-01, 1.84534400e-02, 8.65840329e-02,
       6.32057572e-01, 8.32945314e-02, 1.65912040e-01, 4.14436901e-02,
       2.31759441e-02, 4.18318692e-02, 1.55541626e-01, 9.68260452e-01,
       9.50186960e-02, 1.08336925e-01, 6.97722075e-03, 1.29891434e-01,
       1.28791017e-01, 6.67600507e-02, 8.68770425e-02, 6.48143264e-02,
       1.45526249e-01, 3.83166821e-01, 4.23005581e-02, 1.00872515e-01,
       2.41985748e-01, 4.22362070e-01, 1.53374315e-01, 1.71507447e-02,
       1.08112709e-01, 5.23689833e-02, 2.82197847e-01, 6.96227089e-02,
       6.98396463e-01, 4.67966151e-01, 3.81661487e-01, 1.45852186e-02,
       1.35743496e-01, 5.66578769e-02, 6.93205832e-02, 5.81181266e-01,
       8.04466445e-01, 7.42303626e-03, 1.17869178e-01, 8.19112312e-02,
       4.03301103e-02, 1.18970899e-02, 1.86672503e-01, 2.18902509e-01,
       1.01029593e-02, 9.75668335e-01, 1.34429147e-02, 6.55771760e-04,
       7.81083348e-01, 3.37872634e-03, 5.88362004e-03, 7.41987354e-02,
       1.55673018e-01, 8.64779602e-02, 7.05720803e-03, 6.72479367e-02,
       3.14209307e-02, 1.49256456e-01, 3.08521655e-02, 3.74359074e-01,
       1.08043895e-02, 1.21754270e-02, 2.14246803e-01, 9.09701360e-02,
       4.01616628e-03, 3.08745561e-02, 4.44834952e-02, 6.20974055e-01,
       4.29008834e-02, 4.44365331e-02, 5.28987817e-01, 1.63454889e-02,
       7.35315276e-02, 6.31320459e-02, 4.23504699e-01, 8.47554848e-02,
       2.38456058e-02, 9.48699397e-02, 1.51972390e-02, 1.91133963e-01,
       4.52529559e-02, 1.02301047e-02, 1.60954248e-01, 2.98014965e-01,
       1.87735301e-01, 4.79635343e-02, 1.92726720e-01, 2.33515061e-01,
       3.27244798e-01, 7.29524526e-02, 6.89244966e-02, 1.09552899e-02,
       3.61848390e-03, 8.36483697e-02, 1.66777314e-01, 3.79257107e-01,
       5.09733912e-04, 3.45589520e-01, 5.10086138e-03, 1.87080244e-01,
       3.31209165e-02, 4.57037448e-02, 6.69283201e-02, 1.15433771e-02,
       2.94771489e-01, 2.98370223e-01, 3.50690491e-01, 2.52902770e-01,
       3.50475690e-02, 3.57053899e-01, 5.76366859e-02, 4.55128139e-01,
       1.67233711e-02, 3.89386616e-02, 6.03699729e-01, 2.73236288e-02])
```

In [70]:
```
fpr,tpr,thresholds=roc_curve(y_test,probability)
```

In [71]:
```
plt.plot(fpr,tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC-Curve')
plt.show()
```



Decision Tree

In [72]:
```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
```

In [73]:
```
dtc.fit(x_train,y_train)
```

Out[73]:
```
DecisionTreeClassifier()
```

In [74]:
```
pred=dtc.predict(x_test)
pred
```

Out[74]:
```
array([0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 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, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1,
       0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1])
```

In [75]:
```
accuracy_score(y_test,pred)
```

Out[75]:
```
0.7325581395348837
```

In [76]:
```
confusion_matrix(y_test,pred)
```

Out[76]:
```
array([[111,  28],
       [ 18,  15]], dtype=int64)
```

In [77]:
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.80      0.83       139
           1       0.35      0.45      0.39        33

    accuracy                           0.73       172
   macro avg       0.60      0.63      0.61       172
weighted avg       0.76      0.73      0.75       172
```

In [78]:
```
pd.crosstab(y_test,pred)
```

Out[78]:

col_0	**0**	**1**
Attrition		
0	111	28
1	18	15

In [79]:
```
from sklearn import tree
plt.figure(figsize=(25,15))
tree.plot_tree(dtc,filled=True)
```

Out[79]:
```
[Text(725.5882457386364, 784.0384615384615, 'X[58] <= 0.5\ngini = 0.309
\nsamples = 687\nvalue = [556, 131]'),
 Text(369.6551846590909, 721.3153846153846, 'X[17] <= 0.132\ngini = 0.2
24\nsamples = 489\nvalue = [426, 63]'),
 Text(114.13636363636364, 658.5923076923077, 'X[30] <= 0.329\ngini = 0.
455\nsamples = 60\nvalue = [39, 21]'),
 Text(63.40909090909091, 595.8692307692307, 'X[6] <= 0.2\ngini = 0.472
\nsamples = 21\nvalue = [8, 13]'),
 Text(38.04545454545455, 533.1461538461538, 'X[53] <= 0.5\ngini = 0.463
\nsamples = 11\nvalue = [7, 4]'),
 Text(25.363636363636363, 470.4230769230769, 'X[42] <= 0.125\ngini = 0.
346\nsamples = 9\nvalue = [7, 2]'),
 Text(12.681818181818182, 407.7, 'gini = 0.0\nsamples = 1\nvalue = [0,
1]'),
 Text(38.04545454545455, 407.7, 'X[43] <= 0.833\ngini = 0.219\nsamples
= 8\nvalue = [7, 1]'),
 Text(25.363636363636363, 344.9769230769231, 'gini = 0.0\nsamples = 7\n
value = [7, 0]'),
 Text(50.72727272727273, 344.9769230769231, 'gini = 0.0\nsamples = 1\nv
```

In [80]:
```
from sklearn.model_selection import GridSearchCV
parameter={
 'criterion':['gini','entropy'],
  'splitter':['best'],
  'max_depth':[1,2,3,4,5],
  'max_features':['auto', 'sqrt', 'log2']

}
```

In [81]:
```
grid_search=GridSearchCV(estimator=dtc,param_grid=parameter,cv=5,scoring="a
```

In [82]:
```
grid_search.fit(x_train,y_train)
```

Out[82]:
```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [1, 2, 3, 4, 5],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'splitter': ['best']},
             scoring='accuracy')
```

In [83]:
```
grid_search.best_params_
```

Out[83]:
```
{'criterion': 'gini',
 'max_depth': 2,
 'max_features': 'auto',
 'splitter': 'best'}
```

In [84]:
```
dtc_cv=DecisionTreeClassifier(criterion= 'entropy',
 max_depth=3,
 max_features='sqrt',
 splitter='best')
dtc_cv.fit(x_train,y_train)
```

Out[84]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=3, max_features='sqr
t')
```

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

In [86]:
```
print(classification_report(y_test,predict))
```

```
              precision    recall  f1-score   support

           0       0.82      0.99      0.90       139
           1       0.67      0.06      0.11        33

    accuracy                           0.81       172
   macro avg       0.74      0.53      0.50       172
weighted avg       0.79      0.81      0.75       172
```

In [87]:
```
accuracy_score(y_test,predict)
```

Out[87]:
```
0.813953488372093
```

Random Forest

In [88]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [89]:
```
random_forest = RandomForestClassifier(n_estimators=100, random_state=42)
```

In [90]:
```
random_forest.fit(x_train, y_train)
```

Out[90]:
```
RandomForestClassifier(random_state=42)
```

In [91]:
```
RandomForestClassifier(random_state=42)
```

Out[91]:
```
RandomForestClassifier(random_state=42)
```

In [92]:
```
y_pred = random_forest.predict(x_test)
y_pred
```

Out[92]:
```
array([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, 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, 1, 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, 0, 0, 1, 0, 0, 0,
       0, 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, 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, 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])
```

In [93]:
```
print(accuracy_score(y_test, y_pred))
```

```
0.8430232558139535
```

In [94]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.99      0.91       139
           1       0.88      0.21      0.34        33

    accuracy                           0.84       172
   macro avg       0.86      0.60      0.63       172
weighted avg       0.85      0.84      0.80       172
```