ASSIGNMENT 4

21BCE8842

ANIKET SAXENA

1.Download the Employee Attrition Dataset https://www.kaggle.com/datasets/patelprashant/employee-attrition 2.Perform Data Preprocessing 3.Model Building using Logistic Regression and Decision Tree 4.Calculate Performance metrics

Data Collection.

o Collect the dataset or Create the dataset

Data Preprocessing.

o Import the Libraries. o Importing the dataset. o Checking for Null Values. o Data Visualization. o Outlier Detection o Splitting Dependent and Independent variables o Encoding o Feature Scaling. o Splitting Data into Train and Test.

Model Building

o Import the model building Libraries o Initializing the model o Training and testing the model o Evaluation of Model o Save the Model

Data Preprocessing

o Import the Libraries o Importing the Dataset o Handling Null values o Data Visualization o Outlier Detection o Splitting Dependent and Independent variable o Encoding o Feature Scaling o Splitting Data in Train and Test

Import the Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing the Dataset

In [2]:
```
df=pd.read_csv('WA_Fn-UseC_-HR-Employee-Attrition.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education
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.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]:
```
df.describe()
```

Out[6]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	Employ
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1
mean	36.923810	802.485714	9.192517	2.912925	1.0	1
std	9.135373	403.509100	8.106864	1.024165	0.0	
min	18.000000	102.000000	1.000000	1.000000	1.0	
25%	30.000000	465.000000	2.000000	2.000000	1.0	
50%	36.000000	802.000000	7.000000	3.000000	1.0	1
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1
max	60.000000	1499.000000	29.000000	5.000000	1.0	2

8 rows × 26 columns

Handling Nulll values

In [7]:
```
df.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]:
```
df.isnull().sum() # no null values found
```

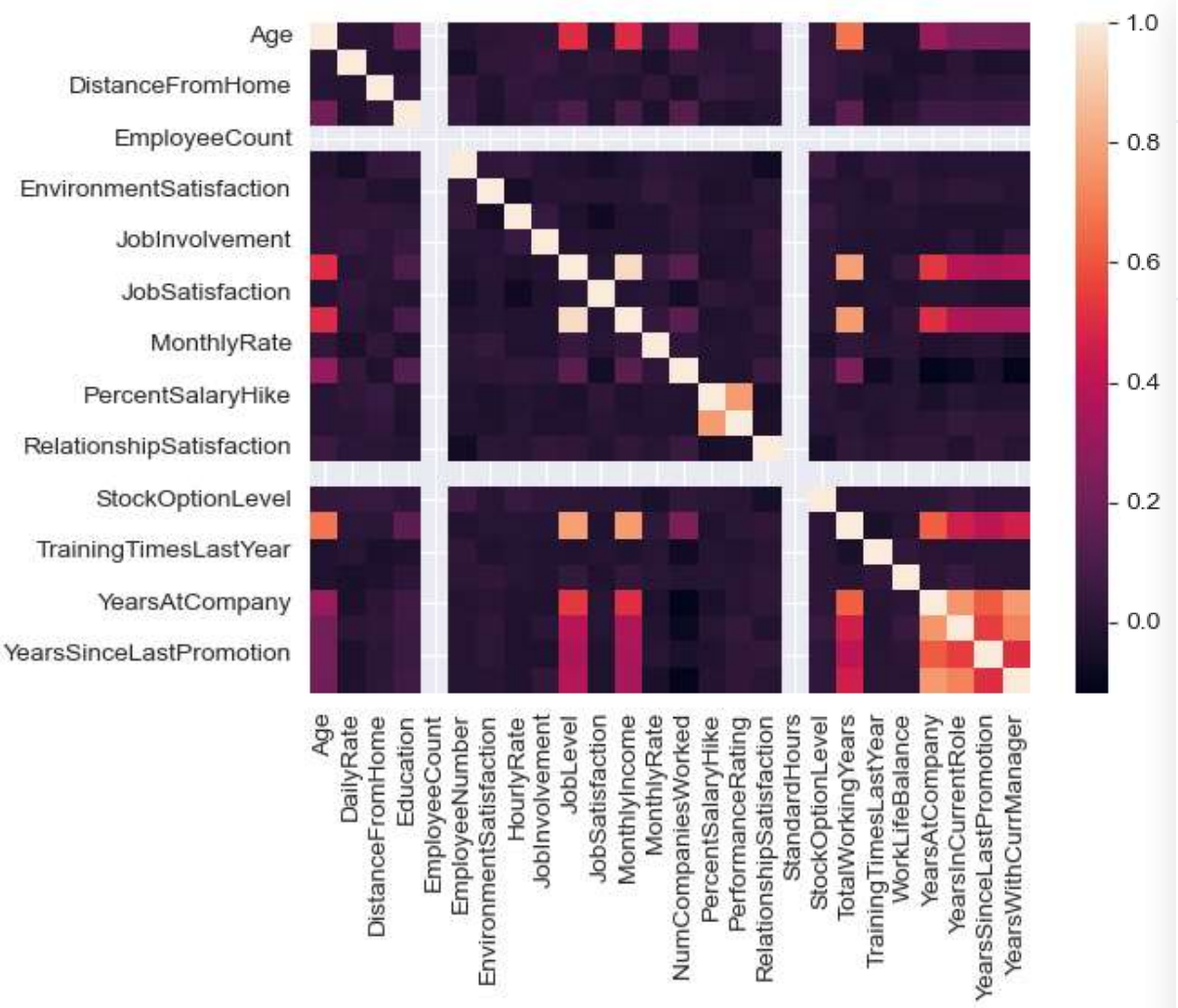
Out[8]:
```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              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
```

Data Visualization

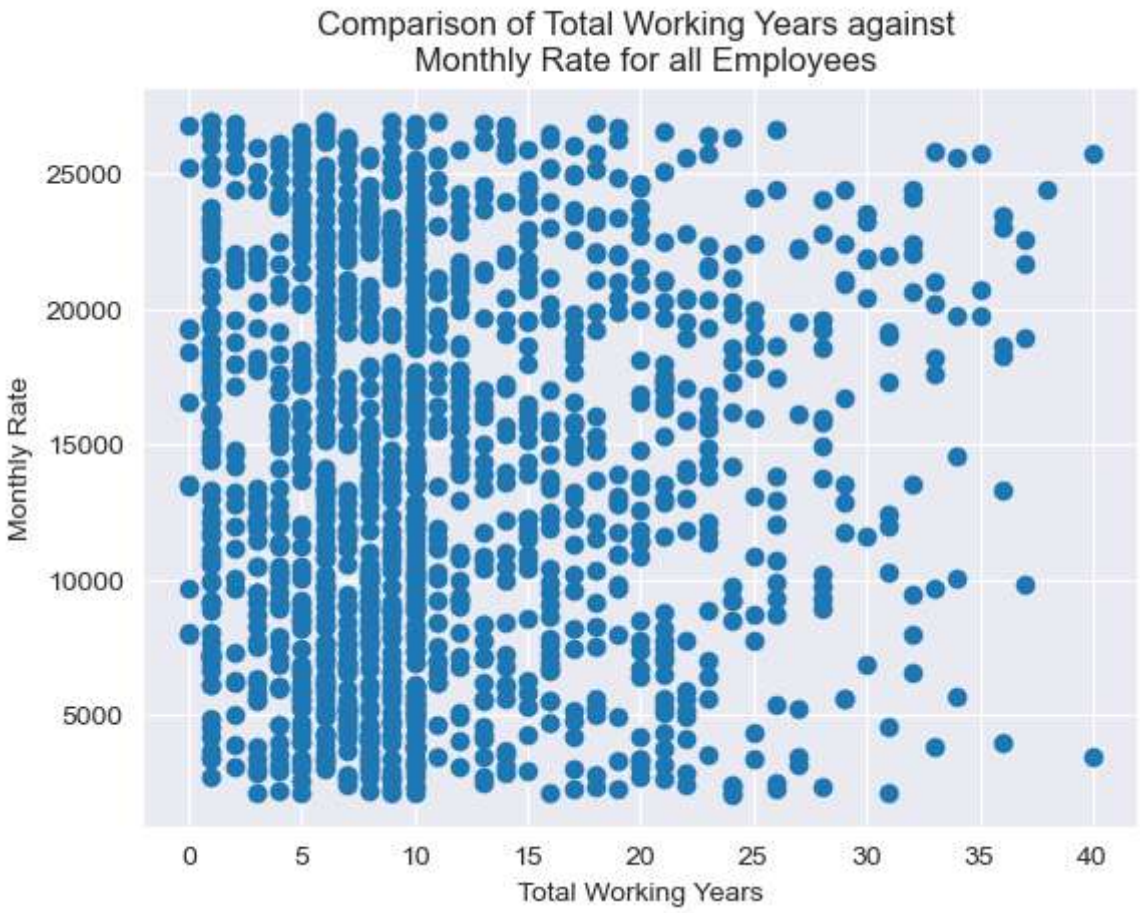
In [9]:
```
sns.heatmap(df.corr())
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_2984\1072140413.py:1: FutureWarning: The
default value of numeric_only in DataFrame.corr is deprecated. In a future version,
it will default to False. Select only valid columns or specify the value of numeric_
only to silence this warning.
  sns.heatmap(df.corr())
```

Out[9]:
```
<Axes: >
```



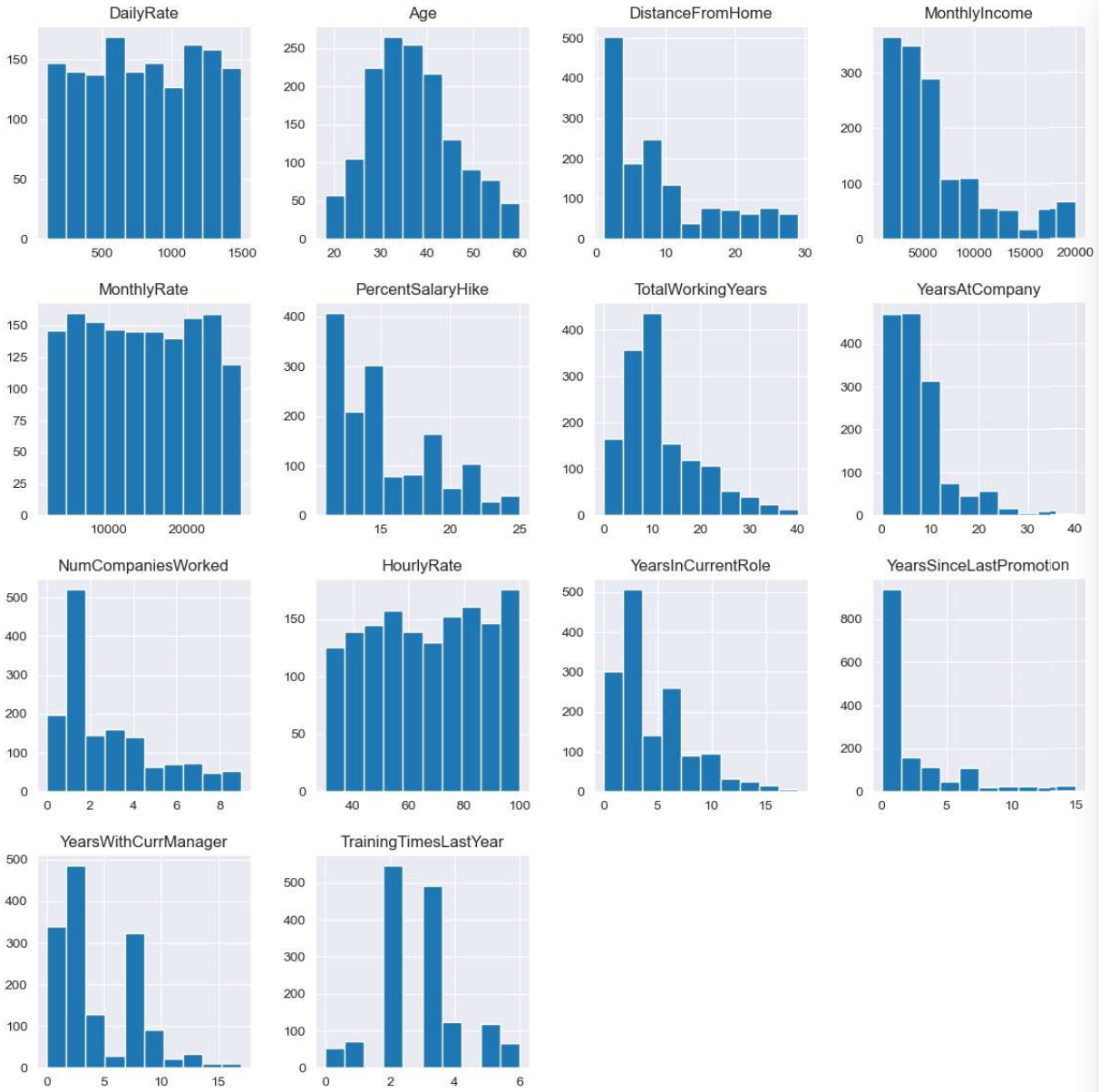
In [10]:
```
plt.scatter(df['TotalWorkingYears'], df['MonthlyRate'])
plt.title("Comparison of Total Working Years against \n Monthly Rate for all Employ
plt.xlabel("Total Working Years")
plt.ylabel("Monthly Rate")
plt.show()
```



In [11]:
```
numeric_columns = ['DailyRate','Age','DistanceFromHome','MonthlyIncome','MonthlyRat
                   'YearsAtCompany','NumCompaniesWorked','HourlyRate',
                   'YearsInCurrentRole','YearsSinceLastPromotion','YearsWithCurrMan
```

In [12]:
```
df[numeric_columns].hist(figsize=(14, 14))
```

Out[12]:
```
array([[<Axes: title={'center': 'DailyRate'}>,
        <Axes: title={'center': 'Age'}>,
        <Axes: title={'center': 'DistanceFromHome'}>,
        <Axes: title={'center': 'MonthlyIncome'}>],
       [<Axes: title={'center': 'MonthlyRate'}>,
        <Axes: title={'center': 'PercentSalaryHike'}>,
        <Axes: title={'center': 'TotalWorkingYears'}>,
        <Axes: title={'center': 'YearsAtCompany'}>],
       [<Axes: title={'center': 'NumCompaniesWorked'}>,
        <Axes: title={'center': 'HourlyRate'}>,
        <Axes: title={'center': 'YearsInCurrentRole'}>,
        <Axes: title={'center': 'YearsSinceLastPromotion'}>],
       [<Axes: title={'center': 'YearsWithCurrManager'}>,
        <Axes: title={'center': 'TrainingTimesLastYear'}>, <Axes: >,
        <Axes: >]], dtype=object)
```



Outlier detection

In [13]:
```
df.corr()
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_2984\592014893.py:1: FutureWarning: The d
efault value of numeric_only in DataFrame.corr is deprecated. In a future version, i
t will default to False. Select only valid columns or specify the value of numeric_o
nly to silence this warning.
  df.corr()
```

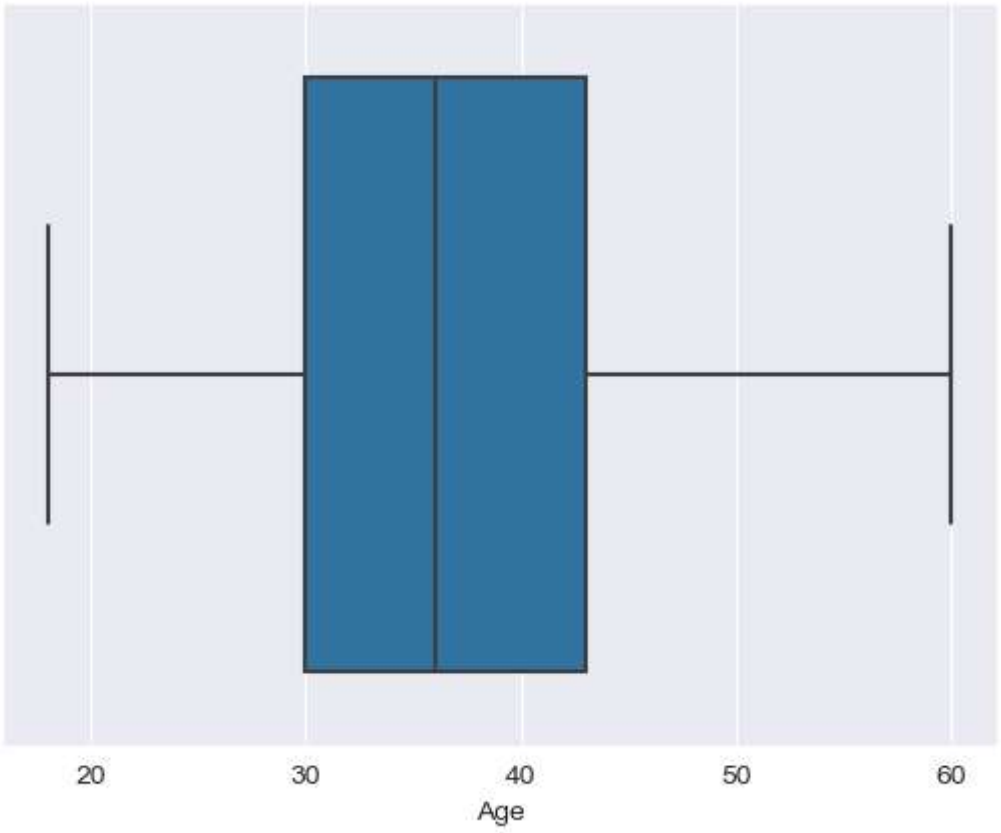
Out[13]:

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

26 rows × 26 columns

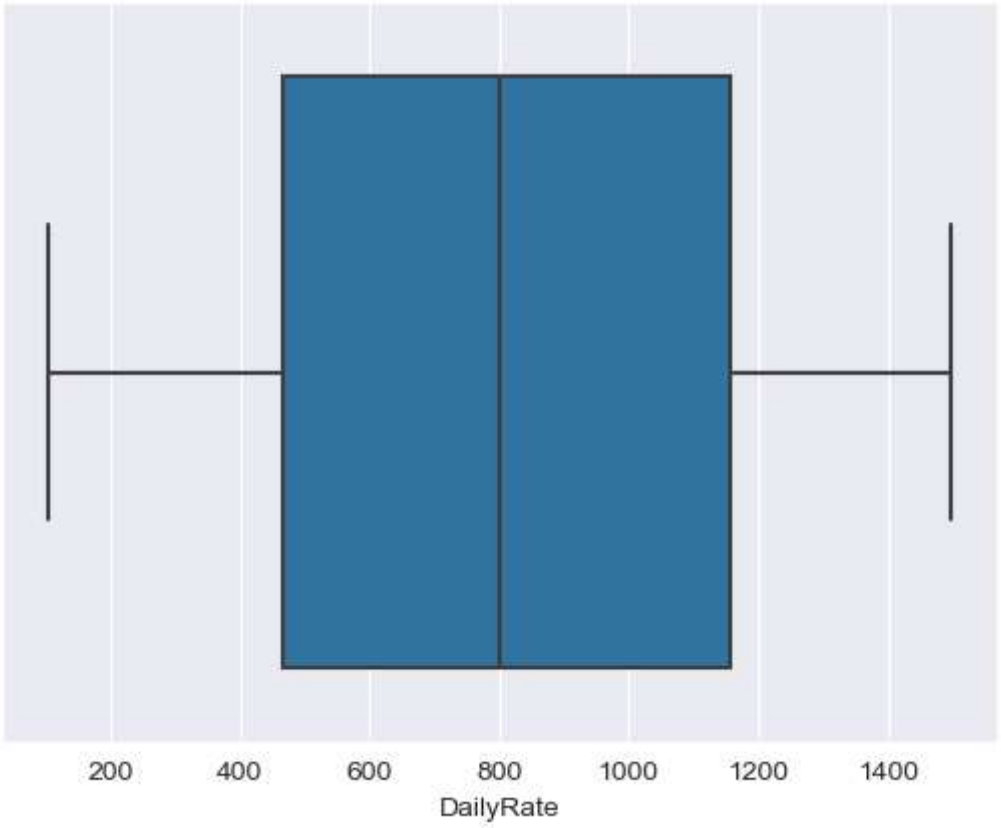
In [14]:
```
sns.boxplot(x='Age',data=df)
```

Out[14]: <Axes: xlabel='Age'>



In [15]:
```
sns.boxplot(x='DailyRate',data=df)
```

Out[15]: <Axes: xlabel='DailyRate'>



Label Encoding

In [16]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [17]:
```
columns = ['Attrition','OverTime','BusinessTravel', 'Department','Education',
                        'EducationField','JobSatisfaction','EnvironmentSatisfaction'
                        'StockOptionLevel','Gender', 'PerformanceRating', 'JobInvolv
                        'JobLevel', 'JobRole', 'MaritalStatus','RelationshipSatisfac
```

In [18]:
```
df1=df.copy()
```

In [19]:
```
df1[columns]=df1[columns].apply(le.fit_transform)
```

In [20]:
```
df1.head()
```

Out[20]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education
0	41	1	2	1102	2	1	1
1	49	0	1	279	1	8	0
2	37	1	2	1373	1	2	1
3	33	0	1	1392	1	3	3
4	27	0	2	591	1	2	0

5 rows × 35 columns

In [21]:
```
df1.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   int32
 2   BusinessTravel            1470 non-null   int32
 3   DailyRate                 1470 non-null   int64
 4   Department                1470 non-null   int32
 5   DistanceFromHome          1470 non-null   int64
 6   Education                 1470 non-null   int64
 7   EducationField            1470 non-null   int32
 8   EmployeeCount             1470 non-null   int64
 9   EmployeeNumber            1470 non-null   int64
 10  EnvironmentSatisfaction   1470 non-null   int64
 11  Gender                    1470 non-null   int32
 12  HourlyRate                1470 non-null   int64
 13  JobInvolvement            1470 non-null   int64
 14  JobLevel                  1470 non-null   int64
 15  JobRole                   1470 non-null   int32
 16  JobSatisfaction           1470 non-null   int64
 17  MaritalStatus             1470 non-null   int32
 18  MonthlyIncome             1470 non-null   int64
 19  MonthlyRate               1470 non-null   int64
 20  NumCompaniesWorked        1470 non-null   int64
 21  Over18                    1470 non-null   int32
 22  OverTime                  1470 non-null   int32
 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: int32(9), int64(26)
memory usage: 350.4 KB
```

Spliting independent and dependent variable

In [22]:
```
df1.corr()
```

Out[22]:

	Age	Attrition	BusinessTravel	DailyRate	Department	Dist
Age	1.000000	-0.159205	0.024751	0.010661	-0.031882	
Attrition	-0.159205	1.000000	0.000074	-0.056652	0.063991	
BusinessTravel	0.024751	0.000074	1.000000	-0.004086	-0.009044	
DailyRate	0.010661	-0.056652	-0.004086	1.000000	0.007109	
Department	-0.031882	0.063991	-0.009044	0.007109	1.000000	
DistanceFromHome	-0.001686	0.077924	-0.024469	-0.004985	0.017225	
Education	0.208034	-0.031373	0.000757	-0.016806	0.007996	
EducationField	-0.040873	0.026846	0.023724	0.037709	0.013720	
EmployeeCount	NaN	NaN	NaN	NaN	NaN	
EmployeeNumber	-0.010145	-0.010577	-0.015578	-0.050990	-0.010895	
EnvironmentSatisfaction	0.010146	-0.103369	0.004174	0.018355	-0.019395	
Gender	-0.036311	0.029453	-0.032981	-0.011716	-0.041583	
HourlyRate	0.024287	-0.006846	0.026528	0.023381	-0.004144	
JobInvolvement	0.029820	-0.130016	0.039062	0.046135	-0.024586	
JobLevel	0.509604	-0.169105	0.019311	0.002966	0.101963	
JobRole	-0.122427	0.067151	0.002724	-0.009472	0.662431	
JobSatisfaction	-0.004892	-0.103481	-0.033962	0.030571	0.021001	
MaritalStatus	-0.095029	0.162070	0.024001	-0.069586	0.056073	
MonthlyIncome	0.497855	-0.159840	0.034319	0.007707	0.053130	
MonthlyRate	0.028051	0.015170	-0.014107	-0.032182	0.023642	
NumCompaniesWorked	0.299635	0.043494	0.020875	0.038153	-0.035882	
Over18	NaN	NaN	NaN	NaN	NaN	
OverTime	0.028062	0.246118	0.016543	0.009135	0.007481	
PercentSalaryHike	0.003634	-0.013478	-0.029377	0.022704	-0.007840	
PerformanceRating	0.001904	0.002889	-0.026341	0.000473	-0.024604	
RelationshipSatisfaction	0.053535	-0.045872	-0.035986	0.007846	-0.022414	
StandardHours	NaN	NaN	NaN	NaN	NaN	
StockOptionLevel	0.037510	-0.137145	-0.016727	0.042143	-0.012193	
TotalWorkingYears	0.680381	-0.171063	0.034226	0.014515	-0.015762	
TrainingTimesLastYear	-0.019621	-0.059478	0.015240	0.002453	0.036875	

	Age	Attrition	BusinessTravel	DailyRate	Department	Dist
WorkLifeBalance	-0.021490	-0.063939	-0.011256	-0.037848	0.026383	
YearsAtCompany	0.311309	-0.134392	-0.014575	-0.034055	0.022920	
YearsInCurrentRole	0.212901	-0.160545	-0.011497	0.009932	0.056315	
YearsSinceLastPromotion	0.216513	-0.033019	-0.032591	-0.033229	0.040061	
YearsWithCurrManager	0.202089	-0.156199	-0.022636	-0.026363	0.034282	

35 rows × 35 columns

In [23]:
```
x = df1.drop(columns='Attrition')
```

In [24]:
```
y = df1['Attrition']
```

In [25]:
```
x.head()
```

Out[25]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationF
0	41	2	1102	2	1	1	
1	49	1	279	1	8	0	
2	37	2	1373	1	2	1	
3	33	1	1392	1	3	3	
4	27	2	591	1	2	0	

5 rows × 34 columns

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

Out[26]:
```
0    1
1    0
2    1
3    0
4    0
Name: Attrition, dtype: int32
```

Feature Scaling

In [27]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [28]:
```
ms=MinMaxScaler()
ms
```

Out[28]:
```
MinMaxScaler()
```

In [29]:
```
x_scaled=pd.DataFrame(ms.fit_transform(x),columns=x.columns)
```

In [30]:
```
x_scaled
```

Out[30]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Ed
0	0.547619	1.0	0.715820	1.0	0.000000	0.25	
1	0.738095	0.5	0.126700	0.5	0.250000	0.00	
2	0.452381	1.0	0.909807	0.5	0.035714	0.25	
3	0.357143	0.5	0.923407	0.5	0.071429	0.75	
4	0.214286	1.0	0.350036	0.5	0.035714	0.00	
...	...	...	...	...	...	...	
1465	0.428571	0.5	0.559771	0.5	0.785714	0.25	
1466	0.500000	1.0	0.365784	0.5	0.178571	0.00	
1467	0.214286	1.0	0.037938	0.5	0.107143	0.50	
1468	0.738095	0.5	0.659270	1.0	0.035714	0.50	
1469	0.380952	1.0	0.376521	0.5	0.250000	0.50	

1470 rows × 34 columns

Splitting Data into Train and test Split

In [31]:
```
from sklearn.model_selection import train_test_split
```

In [32]:
```
x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.2,random_stat
```

In [33]:
```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[33]:
```
((1176, 34), (294, 34), (1176,), (294,))
```

- # Model Building

  - Import the model building Libraries
  - Initializing the model

- Training and testing the model
- Evaluation of Model
- Save the Model

LOGISTIC REGRESSION

In [34]:
```
from sklearn.linear_model import LogisticRegression
```

In [35]:
```
model_logistic=LogisticRegression()
model_logistic
```

Out[35]:
```
LogisticRegression()
```

In [36]:
```
model_logistic.fit(x_train,y_train)
```

Out[36]:
```
LogisticRegression()
```

In [37]:
```
pred_logistic=model_logistic.predict(x_test)
```

In [38]:
```
pred_logistic
```

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

In [39]:
```
y_test
```

Out[39]:
```
442     0
1091    0
981     1
785     0
1332    1
       ..
1439    0
481     0
124     1
198     0
1229    0
Name: Attrition, Length: 294, dtype: int32
```

DECISION TREE CLASSIFIER

In [41]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [42]:
```
model_decision_tree=DecisionTreeClassifier()
model_decision_tree
```

Out[42]:
```
DecisionTreeClassifier
DecisionTreeClassifier()
```

In [43]:
```
model_decision_tree.fit(x_train,y_train)
```

Out[43]:
```
DecisionTreeClassifier
DecisionTreeClassifier()
```

In [44]:
```
pred_decision_tree=model_decision_tree.predict(x_test)
```

In [45]:
```
pred_decision_tree
```

Out[45]:
```
array([0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 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, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 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, 0, 0, 1,
       0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 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, 1, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0])
```

In [46]:
```
y_test
```

Out[46]:
```
442     0
1091    0
981     1
785     0
1332    1
       ..
1439    0
481     0
124     1
198     0
1229    0
Name: Attrition, Length: 294, dtype: int32
```

Evaluation of Classificaton model

For Logistic Regression Model

In [47]:
```
#Accuracy score
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,r
```

In [48]:
```
accuracy_score_logistic=accuracy_score(y_test,pred_logistic)
accuracy_score_logistic
```

Out[48]:
```
0.8843537414965986
```

In [49]:
```
confusion_matrix_logistic=confusion_matrix(y_test,pred_logistic)
confusion_matrix_logistic
```

Out[49]:
```
array([[242,   3],
       [ 31,  18]], dtype=int64)
```

In [50]:
```
crosstab_logistic=pd.crosstab(y_test,pred_logistic)
crosstab_logistic
```

Out[50]:

col_0	0	1
Attrition		
0	242	3
1	31	18

In [51]:
```
print(classification_report(y_test,pred_logistic))
```

	precision	recall	f1-score	support
0	0.89	0.99	0.93	245
1	0.86	0.37	0.51	49
accuracy			0.88	294
macro avg	0.87	0.68	0.72	294
weighted avg	0.88	0.88	0.86	294

In [52]:
```
probability_logistic=model_logistic.predict_proba(x_test)[:,1]
```

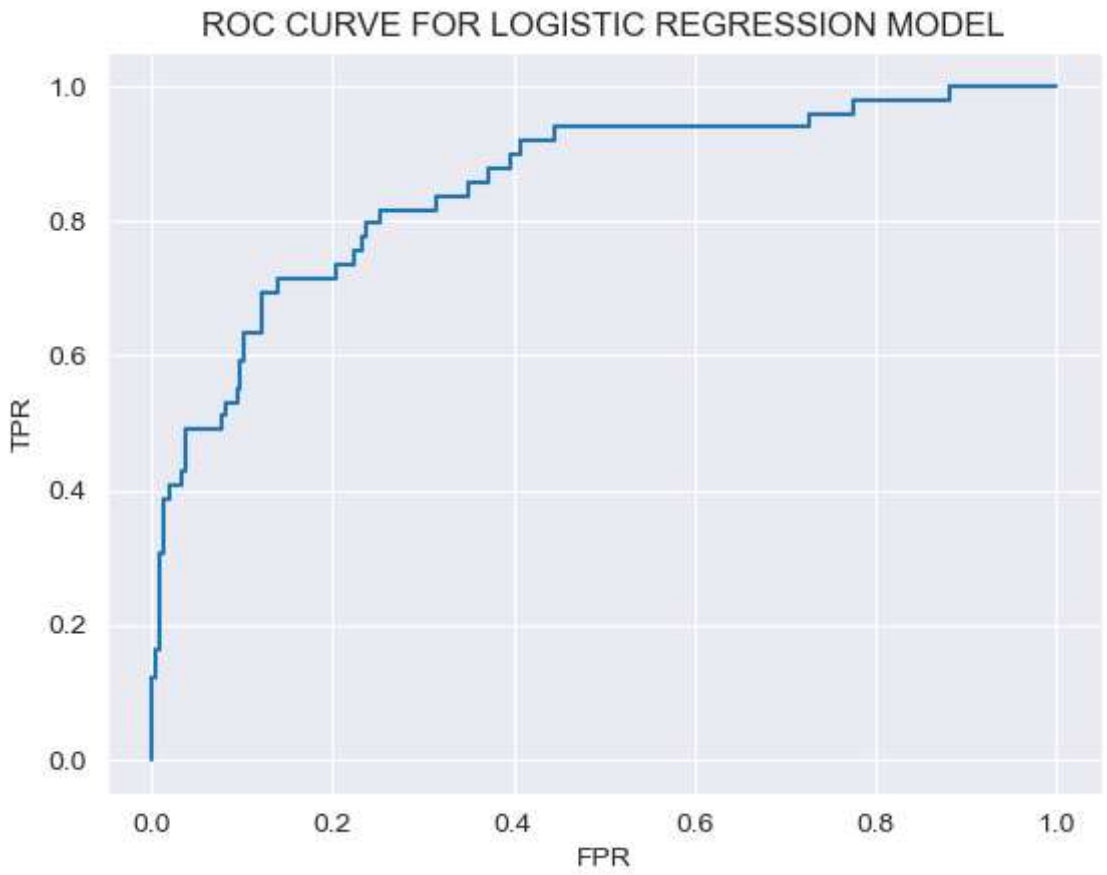
In [53]:
```
probability_logistic
```

Out[53]:
```
array([0.16000127, 0.20600667, 0.31532384, 0.09242886, 0.63667551,
       0.06153061, 0.61819432, 0.0757087 , 0.00841372, 0.3912069 ,
       0.05398439, 0.33293123, 0.02020698, 0.67215483, 0.19786547,
       0.03454902, 0.11043981, 0.17101703, 0.04477777, 0.22783614,
       0.2335018 , 0.01553905, 0.06464492, 0.05029956, 0.58792413,
       0.44849464, 0.07412714, 0.04460935, 0.67666632, 0.0584383 ,
       0.01599026, 0.03521098, 0.06963085, 0.17397462, 0.07830857,
       0.04288032, 0.08150424, 0.07106342, 0.03622137, 0.05223965,
       0.04862098, 0.02091497, 0.01819361, 0.01362467, 0.02873997,
       0.50236969, 0.41553218, 0.00306874, 0.73976412, 0.51382382,
       0.09637213, 0.48845516, 0.08036228, 0.25757243, 0.66516772,
       0.26308027, 0.01964858, 0.30198497, 0.02919946, 0.16038964,
       0.02102747, 0.21670232, 0.13981568, 0.0358316 , 0.37208403,
       0.03002317, 0.29091186, 0.16041142, 0.10437497, 0.08695177,
       0.08217589, 0.30984518, 0.08531362, 0.07420689, 0.12268651,
       0.06192552, 0.04640904, 0.07624712, 0.19738483, 0.03236316,
       0.00884439, 0.0244108 , 0.13635803, 0.0260104 , 0.03341008,
       0.08186888, 0.00499397, 0.03474852, 0.03858027, 0.14602694,
       0.26167665, 0.16667357, 0.27400109, 0.24159565, 0.02160421,
       0.17748606, 0.34076078, 0.28022482, 0.06914126, 0.05003806,
       0.24437761, 0.74698271, 0.35438567, 0.01920627, 0.08778845,
       0.03255847, 0.05461351, 0.15123251, 0.06843702, 0.13752637,
       0.09584388, 0.04669882, 0.02493091, 0.15383171, 0.07081259,
       0.03089296, 0.0537667 , 0.11554316, 0.00881616, 0.01263271,
       0.17552253, 0.05045234, 0.08823238, 0.82995757, 0.03017756,
       0.0236819 , 0.0087012 , 0.1349589 , 0.16474801, 0.05202613,
       0.01524549, 0.29278083, 0.54767448, 0.34275448, 0.04629541,
       0.38966344, 0.61333366, 0.14552367, 0.07402366, 0.24143471,
       0.09418418, 0.0689069 , 0.10061956, 0.19346327, 0.20026293,
       0.03004939, 0.14900424, 0.00348846, 0.11225149, 0.15843155,
       0.06047573, 0.18601882, 0.06085869, 0.12221317, 0.03280184,
       0.02738799, 0.06356425, 0.08302382, 0.01541716, 0.014665  ,
       0.38517822, 0.01264231, 0.14961974, 0.80508787, 0.11598661,
       0.2842811 , 0.17020143, 0.1530583 , 0.02764153, 0.00613226,
       0.04191632, 0.09782393, 0.11551417, 0.10377982, 0.01779313,
       0.14371315, 0.10615435, 0.10298963, 0.05132621, 0.09061081,
       0.02897383, 0.09924087, 0.00512032, 0.75108423, 0.04296968,
       0.04062134, 0.37518972, 0.04563128, 0.7251816 , 0.10671665,
       0.36949086, 0.38146941, 0.32095493, 0.05266802, 0.08172004,
       0.13947833, 0.04334317, 0.01469593, 0.26413988, 0.06330966,
       0.1614747 , 0.15380517, 0.67152357, 0.05840793, 0.27891823,
       0.04512564, 0.46033865, 0.00348431, 0.14068967, 0.02747401,
       0.12714133, 0.17284246, 0.07341066, 0.10099827, 0.16870885,
       0.02560842, 0.01824031, 0.08670796, 0.02834237, 0.13710215,
       0.08778935, 0.2200061 , 0.73401148, 0.15938978, 0.4095449 ,
       0.01513845, 0.11306309, 0.21497506, 0.32337575, 0.03409266,
       0.04256318, 0.32157531, 0.05454465, 0.02348479, 0.16423352,
       0.32696147, 0.22892063, 0.00877159, 0.08198819, 0.01156361,
       0.1408691 , 0.29235147, 0.01270305, 0.17329916, 0.04081391,
       0.04094165, 0.42771425, 0.34958286, 0.03766772, 0.12025286,
       0.37698923, 0.3192629 , 0.79559338, 0.05385659, 0.21597037,
       0.06383728, 0.00570991, 0.66018187, 0.35855286, 0.37783606,
       0.36781398, 0.03554512, 0.21718203, 0.05943622, 0.06554485,
       0.10081475, 0.00818713, 0.26591316, 0.42809675, 0.06542835,
       0.09296803, 0.01259826, 0.14226651, 0.05072662, 0.02372258,
       0.02586923, 0.06760427, 0.24315648, 0.26961432, 0.19831733,
```

```
       0.2652296 , 0.0165923 , 0.15784236, 0.08398982, 0.02711775,
       0.18750547, 0.00783535, 0.2844239 , 0.00270742, 0.02484969,
       0.22585745, 0.72775605, 0.07691547, 0.26304359])
```

In [54]:
```
# Roc curve
fpr_logistic,tpr_logistic,threshsholds_logistic = roc_curve(y_test,probability_logi
```

In [55]:
```
plt.plot(fpr_logistic,tpr_logistic)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC CURVE FOR LOGISTIC REGRESSION MODEL')
plt.show()
```



For Decision Tree Model

In [56]:
```
accuracy_score_decision=accuracy_score(y_test,pred_decision_tree)
accuracy_score_decision
```

Out[56]:
```
0.7551020408163265
```

In [57]:
```
confusion_matrix_decision=confusion_matrix(y_test,pred_decision_tree)
confusion_matrix_decision
```

Out[57]:
```
array([[208,  37],
       [ 35,  14]], dtype=int64)
```

In [58]:
```
crosstab_decision=pd.crosstab(y_test,pred_decision_tree)
crosstab_decision
```

Out[58]:

col_0	0	1
Attrition		
0	208	37
1	35	14

In [59]:
```
print(classification_report(y_test,pred_decision_tree))
```

```
              precision    recall  f1-score   support

           0       0.86      0.85      0.85       245
           1       0.27      0.29      0.28        49

    accuracy                           0.76       294
   macro avg       0.57      0.57      0.57       294
weighted avg       0.76      0.76      0.76       294
```

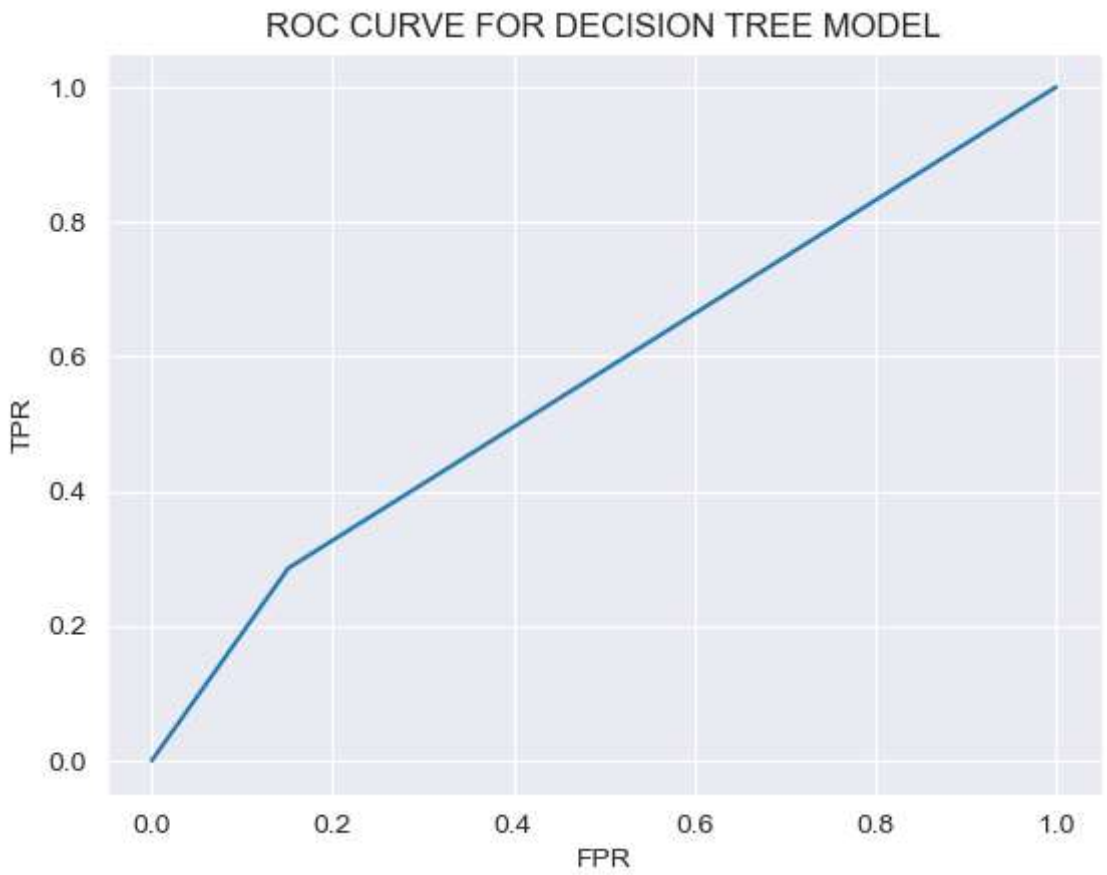
In [60]:
```
probability_decision=model_decision_tree.predict_proba(x_test)[:,1]
```

In [61]:
```
probability_decision
```

Out[61]:
```
array([0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
       0., 0., 0., 1., 0., 0., 0., 1., 0., 0., 0., 1., 0., 1., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1.,
       1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 0., 0.,
       0., 0., 0., 1., 0., 0., 0., 1., 1., 0., 0., 0., 1., 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., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 0., 0.,
       0., 0., 1., 0., 0., 0., 1., 0., 1., 0., 0., 1., 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., 0., 0., 1., 0., 1., 1., 0., 0., 0., 0., 1., 0., 0., 1.,
       0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.,
       1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.,
       0., 1., 0., 0., 0., 0., 1., 1., 0., 0., 1., 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., 1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 1., 1., 0., 1., 0., 0., 0.,
       0., 0., 0., 0., 0.])
```

In [62]:
```
fpr_decision,tpr_decision,threshsholds_decision = roc_curve(y_test,probability_deci
```

In [63]:
```
plt.plot(fpr_decision,tpr_decision)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC CURVE FOR DECISION TREE MODEL')
plt.show()
```


ROC CURVE FOR DECISION TREE MODEL
TPR
FPR
0.0
0.2
0.4
0.6
0.8
1.0