Assignment -4

Boga. Vivek (21BEC2159)

1.Download the Employee Attrition Dataset https://www.kaggle.com/datasets/patelprashant/employee-attrition

2.Perfrom Data Preprocessing

3.Model Building using Logistic Regression and Decision Tree and Random Forest

4.Calculate Performance metrics

DATA PREPROCESSING

Import the libraries

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

Import the dataset.

In [5]:
```python
df=pd.read_csv("Employee_Attrition.csv")
```

In [6]:
```python
df.head()
```

Out[6]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...

5 rows × 35 columns

In [7]:
```python
df.shape
```

Out[7]:
```
(1470, 35)
```

In [8]:
```python
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 [9]:
```python
df.describe()
```

Out[9]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000

8 rows × 26 columns

In [10]:
```python
df.Attrition.value_counts()
```

Out[10]:
```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

Check for null values

In [11]:
```python
df.isnull().sum()
```

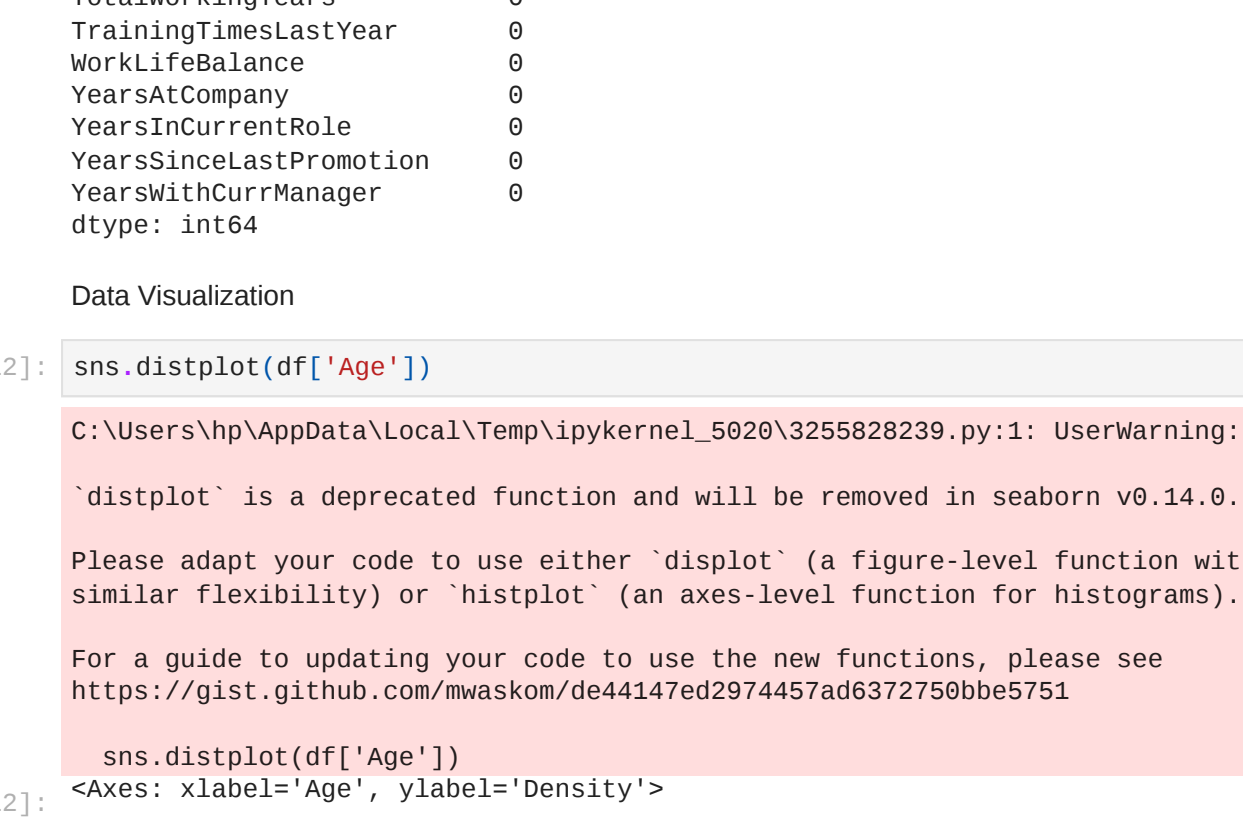
Out[11]:
```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              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 [12]:
```python
sns.distplot(df['Age'])
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_5020\325582829.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Age'])
```

Out[12]:
```
<Axes: xlabel='Age', ylabel='Density'>
```



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

```
C:\Users\hp\AppData\Local\Temp\ipykernel_5020\1134722465.py:1: FutureWarning: The default value of numeric_only i
n DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or speci
fy the value of numeric_only to silence this warning.
  df.corr()
```

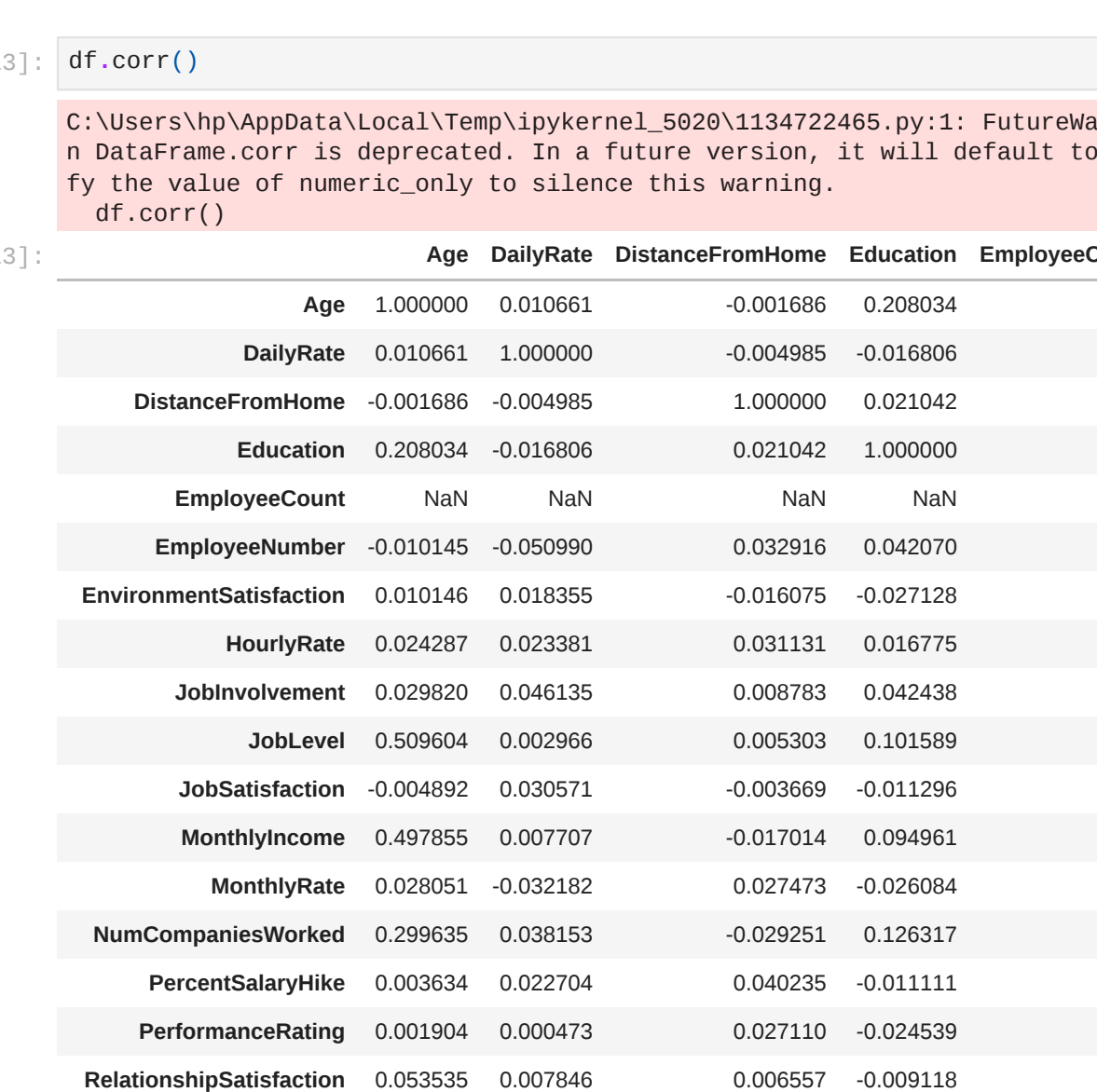
Out[13]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction
Age	1.000000	0.010661	-0.001686		NaN	-0.010145	0.010146
DailyRate	0.010661	1.000000	-0.004985		NaN	-0.050990	0.018355
DistanceFromHome	-0.001686	-0.004985	1.000000		NaN	0.032916	-0.016075
Education	0.208034	-0.016806	0.021042	1.000000	NaN	0.042070	-0.027128
EmployeeCount	NaN	NaN	NaN		NaN	NaN	NaN
EmployeeNumber	-0.010145	-0.050990	0.032916		NaN	1.000000	0.017621
EnvironmentSatisfaction	0.010146	0.018355	-0.016075		NaN	0.017621	1.000000
HourlyRate	0.024287	0.023381	0.031131		NaN	0.035179	-0.049857
JobInvolvement	0.029820	0.046135	0.008783		NaN	-0.006888	-0.008278
JobLevel	0.509604	0.002966	0.005303		NaN	-0.018519	0.001212
JobSatisfaction	-0.004892	0.030571	-0.003669		NaN	-0.046247	-0.006784
MonthlyIncome	0.497855	0.007707	-0.017014		NaN	-0.014829	-0.006259
MonthlyRate	0.028051	-0.032182	0.027473		NaN	0.012648	0.037600
NumCompaniesWorked	0.299635	0.038153	-0.029251		NaN	-0.001251	0.012594
PercentSalaryHike	0.003634	0.022704	0.040235		NaN	-0.012944	-0.031701
PerformanceRating	0.001904	0.000473	0.027110		NaN	-0.020359	-0.029548
RelationshipSatisfaction	0.053535	0.007846	0.006557		NaN	-0.069861	0.007665
StandardHours	NaN	NaN	NaN		NaN	NaN	NaN
StockOptionLevel	0.037510	0.042143	0.044872		NaN	0.062227	0.003432
TotalWorkingYears	0.680381	0.014515	0.004628		NaN	-0.014365	-0.002693
TrainingTimesLastYear	-0.019621	0.002453	-0.036942		NaN	0.023603	-0.019359
WorkLifeBalance	-0.021490	-0.037848	-0.026556		NaN	0.010309	0.027627
YearsAtCompany	0.311309	-0.034055	0.009508		NaN	-0.011240	0.001458
YearsInCurrentRole	0.212901	0.009932	0.018845		NaN	-0.008416	0.018007
YearsSinceLastPromotion	0.216513	-0.033229	0.010029		NaN	-0.009019	0.016194
YearsWithCurrManager	0.202089	-0.026363	0.014406		NaN	-0.009197	-0.004999

26 rows × 26 columns

In [14]:
```python
sns.boxplot(df.Age)
```

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



In [15]:
```python
df
```

Out[15]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7
...	...	...	...	...	...	...	...	...	...	...
1465	36	No	Travel_Frequently	884	Research & Development	23	2	Medical	1	2061
1466	39	No	Travel_Rarely	613	Research & Development	6	1	Medical	1	2062
1467	27	No	Travel_Rarely	155	Research & Development	4	3	Life Sciences	1	2064
1468	49	No	Travel_Frequently	1023	Sales	2	3	Medical	1	2065
1469	34	No	Travel_Rarely	628	Research & Development	8	3	Medical	1	2068

1470 rows × 35 columns

Splitting Dependent and Independent variables

Dependant - Attrition, Independant - All others

In [16]:
```python
x= df.drop(columns = ['Attrition'], axis=1)
x.head()
```

Out[16]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber
0	41	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1
1	49	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2
2	37	Travel_Rarely	1373	Research & Development	2	2	Other	1	4
3	33	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5
4	27	Travel_Rarely	591	Research & Development	2	1	Medical	1	7

5 rows × 34 columns

In [17]:
```python
y= df.Attrition
y.head
```

Out[17]:
```
<bound method NDFrame.head of 0       Yes
1        No
2       Yes
3        No
4        No
       ...
1465     No
1466     No
1467     No
1468     No
1469     No
Name: Attrition, Length: 1470, dtype: object>
```

Label Encoding

In [18]:
```python
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
x.BusinessTravel=le.fit_transform(x.BusinessTravel)
x.Department=le.fit_transform(x.Department)
x.EducationField=le.fit_transform(x.EducationField)
x.Gender=le.fit_transform(x.Gender)
x.JobRole=le.fit_transform(x.JobRole)
x.MaritalStatus=le.fit_transform(x.MaritalStatus)
x.Over18=le.fit_transform(x.Over18)
x.OverTime=le.fit_transform(x.OverTime)

x.head()
```

Out[18]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber
0	41	2	1102	2	1	2	1	1	1
1	49	1	279	1	8	1	1	1	2
2	37	2	1373	1	2	2	4	1	4
3	33	1	1392	1	3	4	1	1	5
4	27	2	591	1	2	1	3	1	7

5 rows × 34 columns

Feature Scaling using MINMAX

In [19]:
```python
from sklearn.preprocessing import MinMaxScaler
ms = MinMaxScaler()
x_Scaled= pd.DataFrame(ms.fit_transform(x),columns=x.columns)
x_Scaled
```

Out[19]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber
0	0.547619	1.0	0.715820	1.0	0.000000	0.25	0.2	0.0	0.000000
1	0.738095	0.5	0.126700	0.5	0.250000	0.00	0.2	0.0	0.000484
2	0.452381	1.0	0.909807	0.5	0.035714	0.25	0.8	0.0	0.001451
3	0.357143	0.5	0.923407	0.5	0.071429	0.75	0.2	0.0	0.001935
4	0.214286	1.0	0.350036	0.5	0.035714	0.00	0.6	0.0	0.002903
...	...	...	...	...	...	...	...	...	...
1465	0.428571	0.5	0.559771	0.5	0.785714	0.25	0.6	0.0	0.996613
1466	0.500000	1.0	0.365784	0.5	0.178571	0.00	0.6	0.0	0.997097
1467	0.214286	1.0	0.037938	0.5	0.107143	0.50	0.2	0.0	0.998065
1468	0.738095	0.5	0.659270	1.0	0.035714	0.50	0.6	0.0	0.998549
1469	0.380952	1.0	0.376521	0.5	0.250000	0.50	0.6	0.0	1.000000

1470 rows × 34 columns

Splitting Data into Train and Test.

In [20]:
```python
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,random_state=0)
```

In [21]:
```python
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[21]:
```
((1176, 34), (294, 34), (1176,), (294,))
```

In [22]:
```python
x_train.head()
```

Out[22]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber
1374	0.952381	1.0	0.360057	0.5	0.714286	0.50	0.2	0.0	0.937107
1092	0.642857	1.0	0.607015	0.5	0.964286	0.50	1.0	0.0	0.747460
768	0.523810	1.0	0.141732	1.0	0.892857	0.50	0.4	0.0	0.515239
569	0.428571	0.0	0.953472	1.0	0.250000	0.75	0.2	0.0	0.381229
911	0.166667	1.0	0.355762	1.0	0.821429	0.00	0.6	0.0	0.615385

5 rows × 34 columns

Model Building

LOGISTIC REGRESSION

In [23]:
```python
from sklearn.linear_model import LogisticRegression
modellr=LogisticRegression()
```

In [24]:
```python
modellr.fit(x_train,y_train)
```

Out[24]:
```
▾ LogisticRegression
LogisticRegression()
```

In [26]:
```python
pred=modellr.predict(x_test)
pred
```

Out[26]:
```
array([ [illegible]
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No'],
      dtype=object)
```

In [27]:
```python
y_test
```

Out[27]:
```
442     No
1091    No
981    Yes
785     No
1332   Yes
       ...
1439    No
481     No
124    Yes
198     No
1229    No
Name: Attrition, Length: 294, dtype: object
```

Evaluation of Classification Model

Accuracy score

In [28]:
```python
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_auc_score,roc_curve
```

In [29]:
```python
accuracy_score(y_test,pred)
```

Out[29]:
```
0.8843537414965986
```

In [30]:
```python
confusion_matrix(y_test,pred)
```

Out[30]:
```
array([[242,   3],
       [ 31,  18]], dtype=int64)
```

In [31]:
```python
pd.crosstab(y_test,pred)
```

Out[31]:

col_0	No	Yes
Attrition		
No	242	3
Yes	31	18

Performance Metrics:

In [32]:
```python
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

          No       0.89      0.99      0.93       245
         Yes       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 [33]:
```python
probability=modellr.predict_proba(x_test)[:,1]
probability
```

Out[33]:
```
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])
```

DECISION TREE

In [34]:
```python
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
```

In [35]:
```python
dtc.fit(x_train,y_train)
```

Out[35]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [37]:
```python
pred=dtc.predict(x_test)
pred
```

Out[37]:
```
array(['No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No',
       'No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No', 'Yes', 'Yes',
       'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'Yes', 'No', 'Yes',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'Yes', 'Yes', 'Yes',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes',
       'No', 'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No',
       'No', 'No', 'Yes', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No',
       'No', 'Yes', 'Yes', 'No', 'No', 'Yes', 'No', 'No', 'Yes', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No',
       'No', 'Yes', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'Yes', 'No', 'No', 'No',
       'No', 'No', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'No', 'No', 'No',
       'Yes', 'No', 'No', 'No', 'Yes', 'No', 'Yes', 'No', 'No', 'Yes',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'Yes', 'No', 'No',
       'No', 'No', 'Yes', 'No', 'No', 'Yes', 'No', 'No', 'Yes', 'No',
       'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'Yes', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'Yes', 'No',
       'No', 'No', 'No', 'Yes', 'Yes', 'No', 'No', 'Yes', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No',
       'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes',
       'No', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'No',
       'Yes', 'No', 'No', 'No', 'No', 'No'], dtype=object)
```

In [38]:
```python
y_test
```

Out[38]:
```
442     No
1091    No
981    Yes
785     No
1332   Yes
       ...
1439    No
481     No
124    Yes
198     No
1229    No
Name: Attrition, Length: 294, dtype: object
```

Evaluation of Classification Model

In [39]:
```python
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_auc_score,roc_curve
```

In [40]:
```python
accuracy_score(y_test,pred)
```

Out[40]:
```
0.7482993197278912
```

In [41]:
```python
confusion_matrix(y_test,pred)
```

Out[41]:
```
array([[204,  41],
       [ 33,  16]], dtype=int64)
```

In [42]:
```python
pd.crosstab(y_test,pred)
```

Out[42]:

col_0	No	Yes
Attrition		
No	204	41
Yes	33	16

Performance Metrics:

In [43]:
```python
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

          No       0.86      0.83      0.85       245
         Yes       0.28      0.33      0.30        49

    accuracy                           0.75       294
   macro avg       0.57      0.58      0.57       294
weighted avg       0.76      0.75      0.76       294
```

RANDOM FOREST

In [44]:
```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
rfc=RandomForestClassifier()
```

In [45]:
```python
forest_params = [{'max_depth': list(range(10, 15)), 'max_features': list(range(0,14))}]
```

In [46]:
```python
rfc_cv= GridSearchCV(rfc,param_grid=forest_params,cv=10,scoring="accuracy")
```

In [47]:
```python
rfc_cv.fit(x_train,y_train)
```

```
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.py:425: FitFailedWarning:
50 fits failed out of a total of 700.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
50 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\hp\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.py", line 732, in _fit_and_sc
ore
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\hp\anaconda3\Lib\site-packages\sklearn\base.py", line 1144, in wrapper
    estimator._validate_params()
  File "C:\Users\hp\anaconda3\Lib\site-packages\sklearn\base.py", line 637, in _validate_params
    validate_parameter_constraints(
  File "C:\Users\hp\anaconda3\Lib\site-packages\sklearn\utils\_param_validation.py", line 95, in validate_paramet
er_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter of RandomForestClassifier mus
t be an int in the range [1, inf), a float in the range (0.0, 1.0], a str among {'log2', 'sqrt'} or None. Got 0 i
nstead.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\hp\anaconda3\Lib\site-packages\sklearn\model_selection\_search.py:976: UserWarning: One or more of the t
est scores are non-finite: [        nan [illegible]]
  warnings.warn(
```

Out[47]:
```
▸        GridSearchCV
▸ estimator: RandomForestClassifier
   ▸ RandomForestClassifier
```

In [48]:
```python
pred=rfc_cv.predict(x_test)
```

In [49]:
```python
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

          No       0.85      0.98      0.91       245
         Yes       0.62      0.16      0.26        49

    accuracy                           0.84       294
   macro avg       0.73      0.57      0.59       294
weighted avg       0.81      0.84      0.80       294
```

In [50]:
```python
rfc_cv.best_params_
```

Out[50]:
```
{'max_depth': 13, 'max_features': 10}
```