Import Libraries

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

In [2]:
```
df=pd.read_csv("HR_Employee_Attrition.csv")
df
```

Out[2]:

	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	
...	...	...	...	...	...	...	...	
1465	36	No	Travel_Frequently	884	Research & Development	23	2	
1466	39	No	Travel_Rarely	613	Research & Development	6	1	
1467	27	No	Travel_Rarely	155	Research & Development	4	3	
1468	49	No	Travel_Frequently	1023	Sales	2	3	
1469	34	No	Travel_Rarely	628	Research & Development	8	3	

1470 rows × 33 columns

In [3]:
```
df.shape
```

Out[3]:
```
(1470, 33)
```

In [4]:
```
df.Attrition.value_counts()
```

Out[4]:
```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 33 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1470 non-null   int64 
 1   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   EmployeeNumber            1470 non-null   int64 
 9   EnvironmentSatisfaction   1470 non-null   int64 
 10  Gender                    1470 non-null   object
 11  HourlyRate                1470 non-null   int64 
 12  JobInvolvement            1470 non-null   int64 
 13  JobLevel                  1470 non-null   int64 
 14  JobRole                   1470 non-null   object
 15  JobSatisfaction           1470 non-null   int64 
 16  MaritalStatus             1470 non-null   object
 17  MonthlyIncome             1470 non-null   int64 
 18  MonthlyRate               1470 non-null   int64 
 19  NumCompaniesWorked        1470 non-null   int64 
 20  Over18                    1470 non-null   object
 21  OverTime                  1470 non-null   object
 22  PercentSalaryHike         1470 non-null   int64 
 23  PerformanceRating         1470 non-null   int64 
 24  RelationshipSatisfaction  1470 non-null   int64 
 25  StockOptionLevel          1470 non-null   int64 
 26  TotalWorkingYears         1470 non-null   int64 
 27  TrainingTimesLastYear     1470 non-null   int64 
 28  WorkLifeBalance           1470 non-null   int64 
 29  YearsAtCompany            1470 non-null   int64 
 30  YearsInCurrentRole        1470 non-null   int64 
 31  YearsSinceLastPromotion   1470 non-null   int64 
 32  YearsWithCurrManager      1470 non-null   int64 
dtypes: int64(24), object(9)
memory usage: 379.1+ KB
```

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

Out[6]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeNumber	Environmer
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	
mean	36.923810	802.485714	9.192517	2.912925	1024.865306	
std	9.135373	403.509100	8.106864	1.024165	602.024335	
min	18.000000	102.000000	1.000000	1.000000	1.000000	
25%	30.000000	465.000000	2.000000	2.000000	491.250000	
50%	36.000000	802.000000	7.000000	3.000000	1020.500000	
75%	43.000000	1157.000000	14.000000	4.000000	1555.750000	
max	60.000000	1499.000000	29.000000	5.000000	2068.000000	

8 rows × 24 columns

Checking for Null Values.

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

Out[7]:
```
Age                         False
Attrition                   False
BusinessTravel              False
DailyRate                   False
Department                  False
DistanceFromHome            False
Education                   False
EducationField              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
StockOptionLevel            False
TotalWorkingYears           False
TrainingTimesLastYear       False
WorkLifeBalance             False
YearsAtCompany              False
YearsInCurrentRole          False
YearsSinceLastPromotion     False
YearsWithCurrManager        False
dtype: bool
```

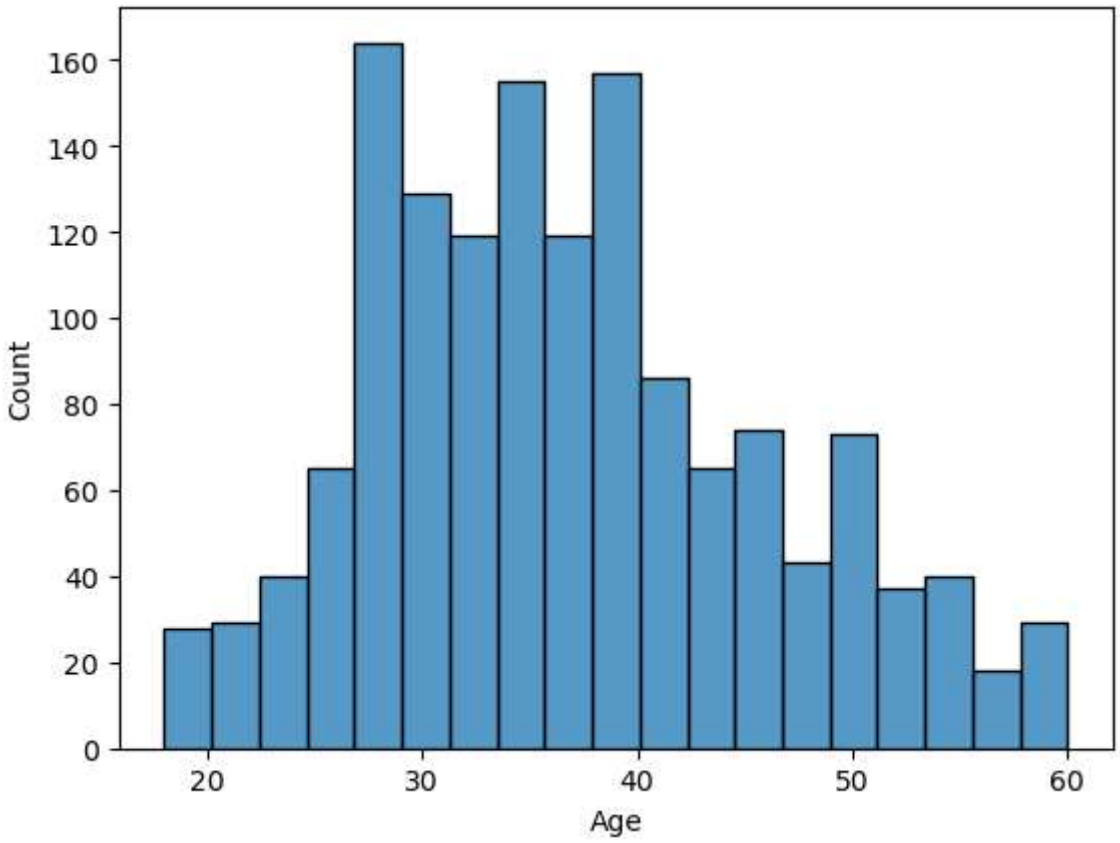
In [8]:
```
df.isnull().sum().sum()
```

Out[8]:
```
0
```

Data Visualization

In [9]:
```
sns.histplot(df["Age"])
```

Out[9]:
```
<Axes: xlabel='Age', ylabel='Count'>
```



In [10]:
```
df.corr()
```

```
C:\Users\Mishra\AppData\Local\Temp\ipykernel_26656\1134722465.py:1: FutureWar
ning: 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 specif
y the value of numeric_only to silence this warning.
  df.corr()
```

Out[10]:

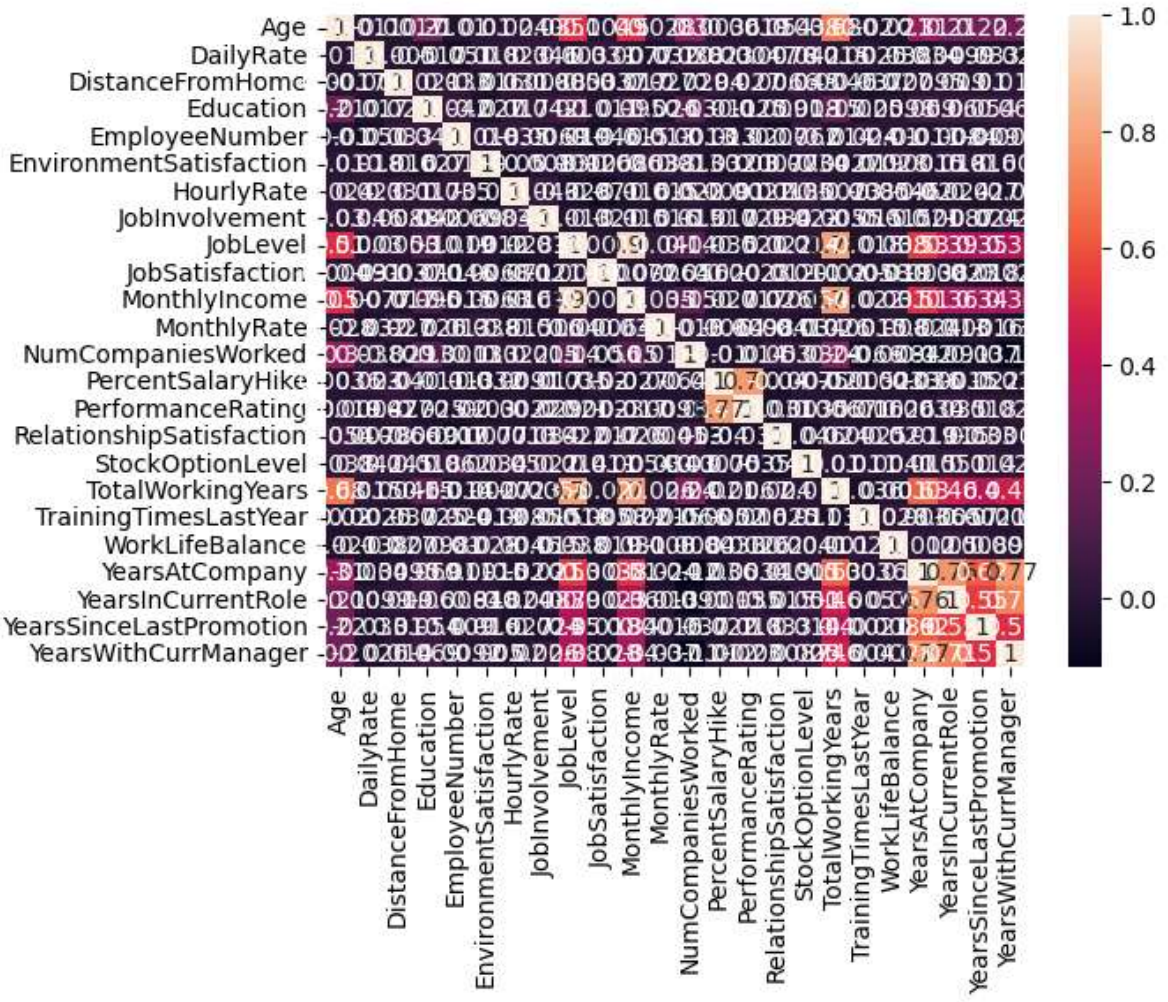
	Age	DailyRate	DistanceFromHome	Education	EmployeeNumber
Age	1.000000	0.010661	-0.001686	0.208034	-0.010145
DailyRate	0.010661	1.000000	-0.004985	-0.016806	-0.050990
DistanceFromHome	-0.001686	-0.004985	1.000000	0.021042	0.032916
Education	0.208034	-0.016806	0.021042	1.000000	0.042070
EmployeeNumber	-0.010145	-0.050990	0.032916	0.042070	1.000000
EnvironmentSatisfaction	0.010146	0.018355	-0.016075	-0.027128	0.017621
HourlyRate	0.024287	0.023381	0.031131	0.016775	0.035179
JobInvolvement	0.029820	0.046135	0.008783	0.042438	-0.006888
JobLevel	0.509604	0.002966	0.005303	0.101589	-0.018519
JobSatisfaction	-0.004892	0.030571	-0.003669	-0.011296	-0.046247
MonthlyIncome	0.497855	0.007707	-0.017014	0.094961	-0.014829
MonthlyRate	0.028051	-0.032182	0.027473	-0.026084	0.012648
NumCompaniesWorked	0.299635	0.038153	-0.029251	0.126317	-0.001251
PercentSalaryHike	0.003634	0.022704	0.040235	-0.011111	-0.012944
PerformanceRating	0.001904	0.000473	0.027110	-0.024539	-0.020359
RelationshipSatisfaction	0.053535	0.007846	0.006557	-0.009118	-0.069861
StockOptionLevel	0.037510	0.042143	0.044872	0.018422	0.062227
TotalWorkingYears	0.680381	0.014515	0.004628	0.148280	-0.014365
TrainingTimesLastYear	-0.019621	0.002453	-0.036942	-0.025100	0.023603
WorkLifeBalance	-0.021490	-0.037848	-0.026556	0.009819	0.010309
YearsAtCompany	0.311309	-0.034055	0.009508	0.069114	-0.011240
YearsInCurrentRole	0.212901	0.009932	0.018845	0.060236	-0.008416
YearsSinceLastPromotion	0.216513	-0.033229	0.010029	0.054254	-0.009019
YearsWithCurrManager	0.202089	-0.026363	0.014406	0.069065	-0.009197

24 rows × 24 columns

In [11]:
```
sns.heatmap(df.corr(),annot=True)
```

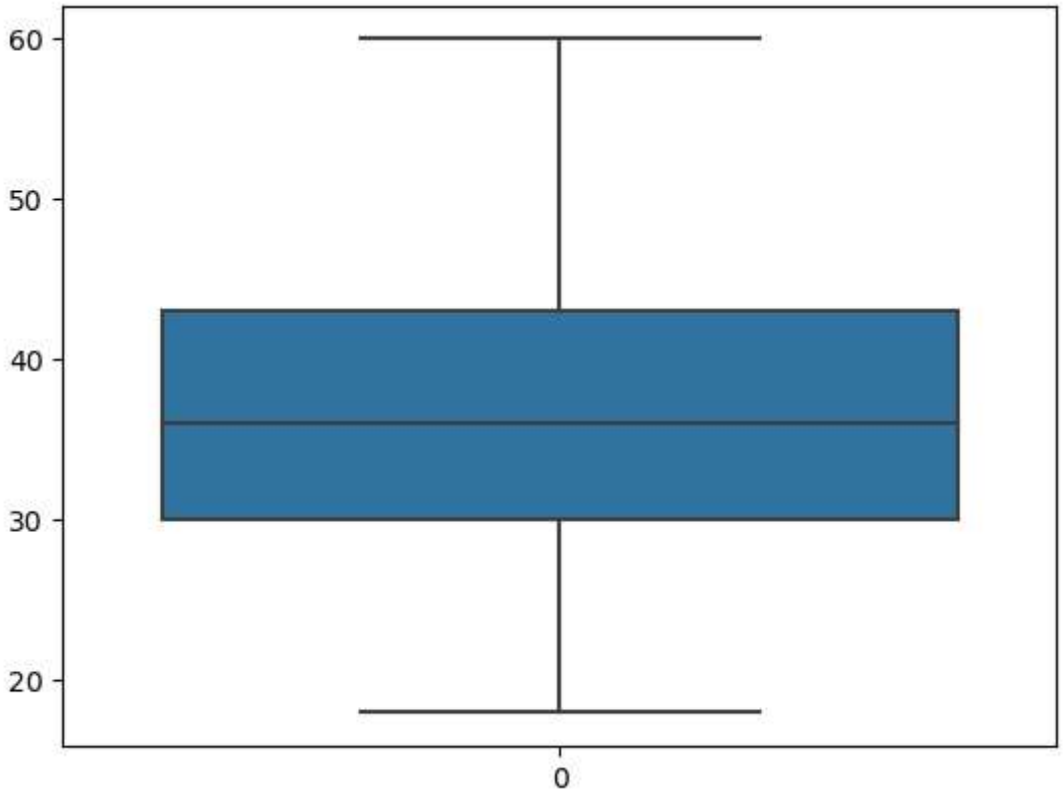
```
C:\Users\Mishra\AppData\Local\Temp\ipykernel_26656\4277794465.py:1: FutureWar
ning: The default value of numeric_only in DataFrame.corr is deprecated. In a
future version, it will default to False. Select only valid columns or specif
y the value of numeric_only to silence this warning.
  sns.heatmap(df.corr(),annot=True)
```

Out[11]: <Axes: >



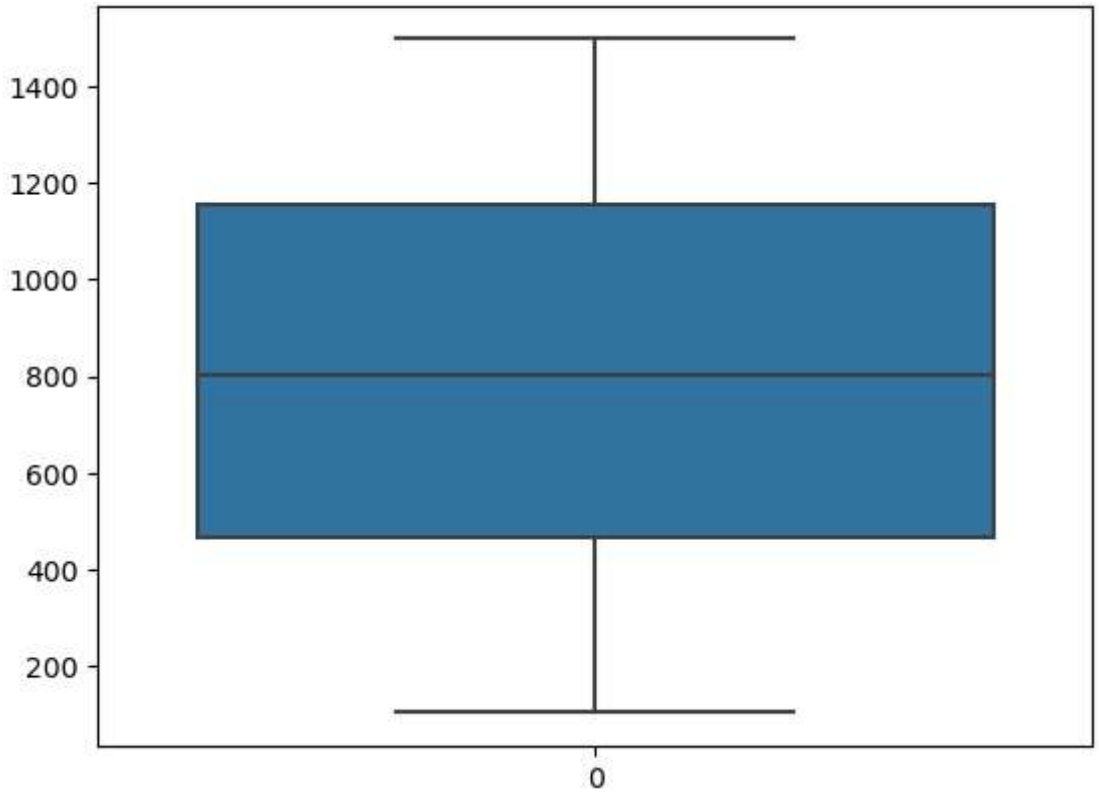
In [12]:
```
sns.boxplot(df.Age)
```

Out[12]: <Axes: >



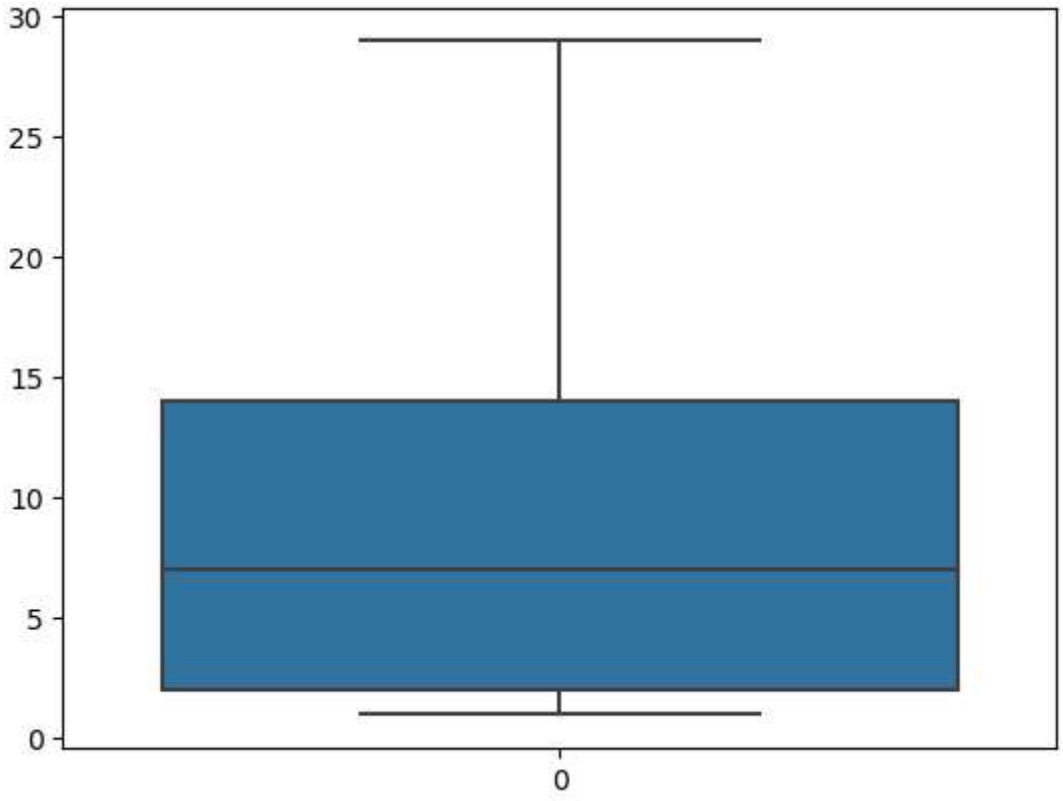
In [13]:
```
sns.boxplot(df.DailyRate)
```

Out[13]: <Axes: >



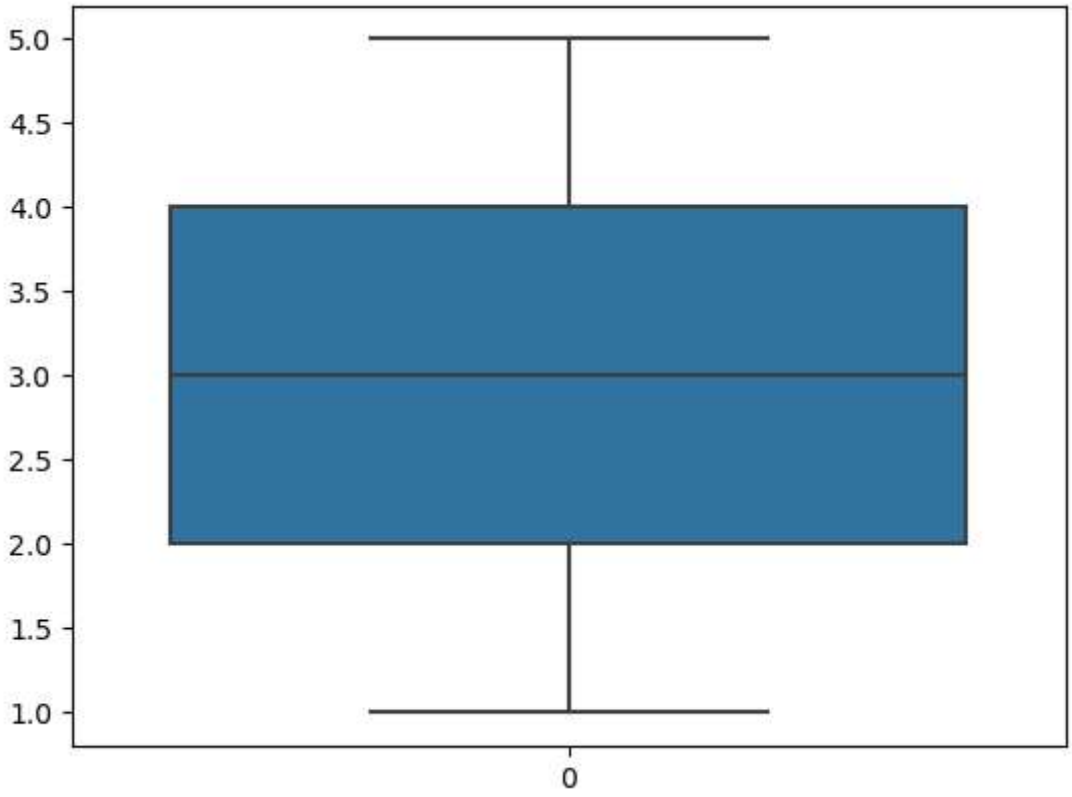
In [14]:
```
sns.boxplot(df.DistanceFromHome)
```

Out[14]: <Axes: >



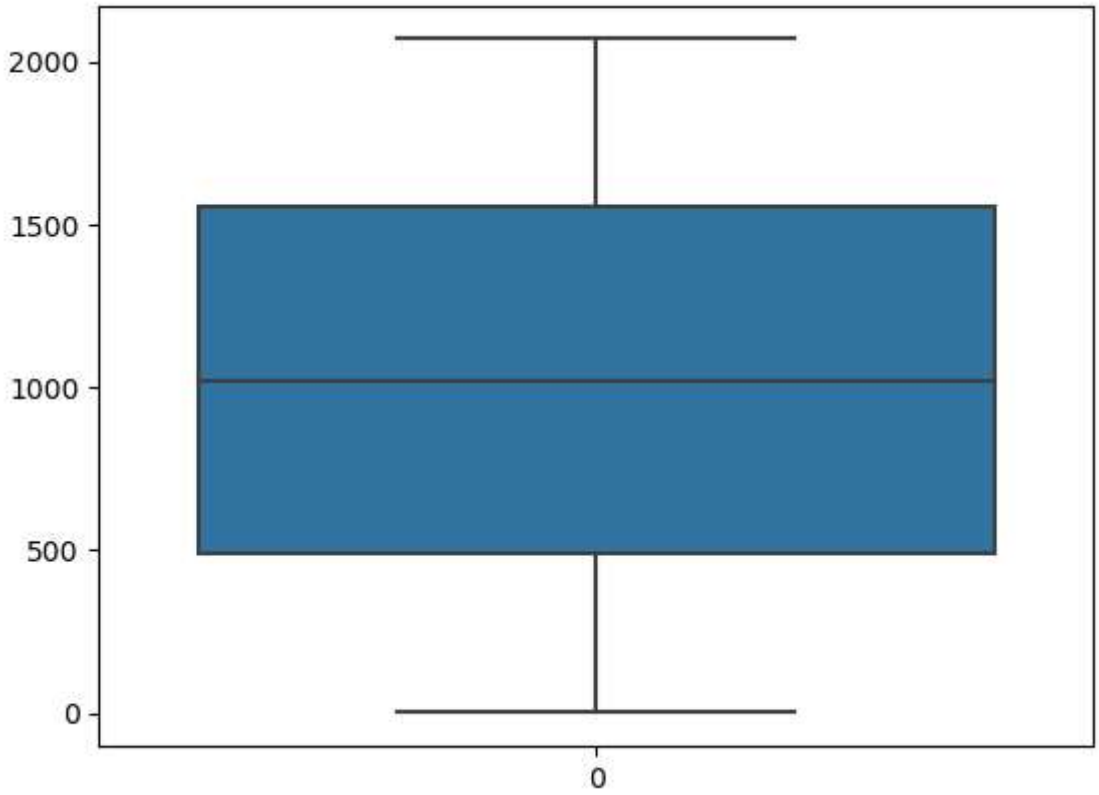
In [15]:
```
sns.boxplot(df.Education)
```

Out[15]: <Axes: >



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

Out[16]: <Axes: >



Splitting Dependent and Independent variables

In [17]:
```
x=df.iloc[:,2:]
y=df.iloc[:,1]
```

Label Encoding

In [18]:
```
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]:

	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	Emplo
0	2	1102	2	1	2	1	
1	1	279	1	8	1	1	
2	2	1373	1	2	2	4	
3	1	1392	1	3	4	1	
4	2	591	1	2	1	3	

5 rows × 31 columns

Feature Scaling

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

Out[19]:

	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	Em
0	1.0	0.715820	1.0	0.000000	0.25	0.2	
1	0.5	0.126700	0.5	0.250000	0.00	0.2	
2	1.0	0.909807	0.5	0.035714	0.25	0.8	
3	0.5	0.923407	0.5	0.071429	0.75	0.2	
4	1.0	0.350036	0.5	0.035714	0.00	0.6	
...	...	...	...	...	...	...	
1465	0.5	0.559771	0.5	0.785714	0.25	0.6	
1466	1.0	0.365784	0.5	0.178571	0.00	0.6	
1467	1.0	0.037938	0.5	0.107143	0.50	0.2	
1468	0.5	0.659270	1.0	0.035714	0.50	0.6	
1469	1.0	0.376521	0.5	0.250000	0.50	0.6	

1470 rows × 31 columns

Splitting Data into Train and Test.

In [20]:
```
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
```

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

Out[21]:
```
((1176, 31), (294, 31), (1176,), (294,))
```

Logisitic Regression Model Building

In [22]:
```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
```

In [23]:
```
model.fit(x_train,y_train)
```

Out[23]:
```
LogisticRegression
LogisticRegression()
```

In [24]:
```
pred=model.predict(x_test)
pred
```

Out[24]:
```
array(['No', 'No', 'No', 'No', 'Yes', 'No', 'Yes', 'No', 'No', 'Yes',
       'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'Yes', 'No', 'No', 'Yes', 'No', 'No', 'Yes', 'No',
       'No', 'Yes', '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', 'No', 'No', 'No',
       'No', 'No', 'No', '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', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'Yes', 'No', 'No', 'No', 'Yes', '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', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No',
       'No', 'Yes', 'No', 'Yes', 'No', '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', 'No',
       'No', 'No', 'No', 'No', 'Yes', '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',
       'No', 'Yes', 'No', 'No', 'No', 'No', 'Yes', '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', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No'],
      dtype=object)
```

In [25]:
```
y_test
```

Out[25]:
```
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
```

Accuracy Score

In [26]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_rep
```

In [27]:
```
confusion_matrix(y_test,pred)
```

Out[27]:
```
array([[241,   4],
       [ 30,  19]], dtype=int64)
```

In [28]:
```
pd.crosstab(y_test,pred)
```

Out[28]:

col_0	No	Yes
Attrition		
No	241	4
Yes	30	19

In [29]:
```
Accuracy = (241+19)/(241+4+30+19)
Accuracy
```

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

In [30]:
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

          No       0.89      0.98      0.93       245
         Yes       0.83      0.39      0.53        49

    accuracy                           0.88       294
   macro avg       0.86      0.69      0.73       294
weighted avg       0.88      0.88      0.87       294
```

In [31]:
```
probability=model.predict_proba(x_test)[:,1]
probability
```

Out[31]:
```
array([0.15891475, 0.21511559, 0.32432557, 0.08886681, 0.63303258,
       0.06182676, 0.60116073, 0.06129281, 0.01244633, 0.52894224,
       0.05911797, 0.40055503, 0.01774956, 0.61600177, 0.19536204,
       0.03097475, 0.11993564, 0.14259998, 0.0441882 , 0.28487654,
       0.18435044, 0.01360069, 0.06054637, 0.0644042 , 0.50468314,
       0.43046476, 0.10822989, 0.05258899, 0.63461419, 0.08664196,
       0.01485371, 0.03713133, 0.06832962, 0.20850356, 0.09852004,
       0.03286342, 0.082464  , 0.05914568, 0.05256949, 0.05318322,
       0.05700568, 0.01903842, 0.01641415, 0.01302266, 0.02503352,
       0.50677165, 0.36259837, 0.00234831, 0.66839676, 0.44671953,
       0.13405863, 0.56997014, 0.07936646, 0.28134011, 0.69621889,
       0.24937791, 0.01621117, 0.38833096, 0.02564579, 0.17550708,
       0.02883122, 0.18284739, 0.14299095, 0.02734075, 0.34548398,
       0.04414777, 0.31497096, 0.14558263, 0.1235461 , 0.09541113,
       0.09102041, 0.2608112 , 0.07637309, 0.07676458, 0.10931979,
       0.05017179, 0.08388532, 0.10813308, 0.1900822 , 0.03545992,
       0.0091634 , 0.02462897, 0.16628409, 0.02543071, 0.03139766,
       0.07830403, 0.00499672, 0.07289363, 0.03522334, 0.12782832,
       0.1997292 , 0.14301624, 0.2646213 , 0.24404641, 0.01720617,
       0.20455338, 0.34599494, 0.25017011, 0.09201517, 0.05121543,
       0.2112655 , 0.72467912, 0.35414797, 0.02786452, 0.09955845,
       0.04508169, 0.06873754, 0.15215574, 0.10096503, 0.15594135,
       0.08245439, 0.04400721, 0.04334864, 0.14834368, 0.05975021,
       0.04272249, 0.04574552, 0.11551546, 0.00941756, 0.01223489,
       0.22613438, 0.04843507, 0.08376676, 0.80373244, 0.04366118,
       0.027391  , 0.01291323, 0.13356578, 0.17716949, 0.04168438,
       0.01438738, 0.30332401, 0.56809177, 0.26727437, 0.05807149,
       0.42124429, 0.56577335, 0.24697458, 0.06163264, 0.22610041,
       0.08386132, 0.07842809, 0.08930405, 0.17701088, 0.29890668,
       0.03919743, 0.13828096, 0.0033842 , 0.11208064, 0.13953154,
       0.05557145, 0.14898315, 0.05451647, 0.11730045, 0.0341553 ,
       0.04390226, 0.06912775, 0.07821587, 0.01381096, 0.01241026,
       0.38855565, 0.01307225, 0.11239813, 0.80343597, 0.1942669 ,
       0.33130457, 0.16264036, 0.13382165, 0.03038525, 0.00542577,
       0.03733729, 0.17353554, 0.17097854, 0.08239189, 0.0161542 ,
       0.11497677, 0.09675853, 0.09017036, 0.04375561, 0.09275858,
       0.02416675, 0.11140631, 0.00530973, 0.81022589, 0.06321252,
       0.04128112, 0.53764442, 0.04502352, 0.73399774, 0.0824389 ,
       0.34750978, 0.32974373, 0.31554156, 0.05230788, 0.07749644,
       0.21347688, 0.04652234, 0.01956346, 0.25828489, 0.05695346,
       0.15536   , 0.17308441, 0.63309302, 0.0554643 , 0.23351402,
       0.041376  , 0.4338105 , 0.00331211, 0.12265664, 0.02913167,
       0.11640111, 0.18765739, 0.09235799, 0.08987611, 0.24930837,
       0.0231433 , 0.01520841, 0.08704603, 0.0228926 , 0.12615554,
       0.09957933, 0.23980335, 0.67402732, 0.18515948, 0.35788124,
       0.02958548, 0.15886055, 0.16351833, 0.28564987, 0.02851941,
       0.03820667, 0.35638525, 0.05565143, 0.02953751, 0.16095279,
       0.28425   , 0.20774611, 0.01027417, 0.07141752, 0.01208132,
       0.19008834, 0.26995358, 0.01436142, 0.16016645, 0.05334459,
       0.03607947, 0.40769009, 0.4200565 , 0.0318672 , 0.10361636,
       0.4059028 , 0.35236182, 0.73182365, 0.04756796, 0.2262603 ,
       0.0848035 , 0.00518662, 0.62543897, 0.3014179 , 0.35547649,
       0.35354735, 0.0335931 , 0.19246384, 0.05074979, 0.05413504,
       0.15366914, 0.00781302, 0.212933  , 0.37675336, 0.06984907,
       0.10237479, 0.00992434, 0.1386785 , 0.05692345, 0.03202143,
       0.03304906, 0.06063555, 0.35044495, 0.35817183, 0.17578989,
       0.20989673, 0.01532338, 0.12868418, 0.08311898, 0.03062858,
```

```
0.21635223, 0.00706043, 0.24320165, 0.00281585, 0.0253744 ,
0.24037253, 0.67469459, 0.06700778, 0.26604659])
```

Decision Tree and RandomForest Classifier

In [32]:
```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
```

In [33]:
```
dtc.fit(x_train,y_train)
```

Out[33]:
```
DecisionTreeClassifier

DecisionTreeClassifier()
```

In [34]:
```
pred=dtc.predict(x_test)
pred
```

Out[34]:
```
array(['No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No',
       'No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No',
       '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', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'Yes', 'No',
       'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No',
       'Yes', 'Yes', 'No', 'No', 'No', 'Yes', 'No', 'No', 'Yes', 'No',
       'No', 'Yes', 'No', 'No', 'No', 'Yes', 'No', 'No', 'Yes', 'No',
       'No', 'No', 'No', 'Yes', 'No', 'No', 'Yes', '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', 'No', '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', 'Yes', 'Yes',
       'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'Yes', 'No', 'No', 'No',
       'No', 'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'Yes', 'No', 'Yes', 'No', 'No', 'Yes', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No',
       'Yes', 'No', 'No', 'Yes', 'No', 'No', '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', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'Yes', 'Yes', 'No', 'No', 'No', 'Yes', 'No', 'No', 'Yes',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No'], dtype=object)
```

In [35]:
```
y_test
```

Out[35]:
```
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
```

Accuracy Score

In [36]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_rep
```

In [37]:
```
accuracy_score(y_test,pred)
```

Out[37]:
```
0.7551020408163265
```

In [38]:
```
confusion_matrix(y_test,pred)
```

Out[38]:
```
array([[206,  39],
       [ 33,  16]], dtype=int64)
```

In [39]:
```
pd.crosstab(y_test,pred)
```

Out[39]:

col_0	No	Yes
Attrition		
No	206	39
Yes	33	16

In [40]:
```
Accuracy = (212+17)/(212+33+32+17)
Accuracy
```

Out[40]:
```
0.7789115646258503
```

In [41]:
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

          No       0.86      0.84      0.85       245
         Yes       0.29      0.33      0.31        49

    accuracy                           0.76       294
   macro avg       0.58      0.58      0.58       294
weighted avg       0.77      0.76      0.76       294
```

In [42]:
```
probability=model.predict_proba(x_test)[:,1]
probability
```

Out[42]:
```
array([0.15891475, 0.21511559, 0.32432557, 0.08886681, 0.63303258,
       0.06182676, 0.60116073, 0.06129281, 0.01244633, 0.52894224,
       0.05911797, 0.40055503, 0.01774956, 0.61600177, 0.19536204,
       0.03097475, 0.11993564, 0.14259998, 0.0441882 , 0.28487654,
       0.18435044, 0.01360069, 0.06054637, 0.0644042 , 0.50468314,
       0.43046476, 0.10822989, 0.05258899, 0.63461419, 0.08664196,
       0.01485371, 0.03713133, 0.06832962, 0.20850356, 0.09852004,
       0.03286342, 0.082464  , 0.05914568, 0.05256949, 0.05318322,
       0.05700568, 0.01903842, 0.01641415, 0.01302266, 0.02503352,
       0.50677165, 0.36259837, 0.00234831, 0.66839676, 0.44671953,
       0.13405863, 0.56997014, 0.07936646, 0.28134011, 0.69621889,
       0.24937791, 0.01621117, 0.38833096, 0.02564579, 0.17550708,
       0.02883122, 0.18284739, 0.14299095, 0.02734075, 0.34548398,
       0.04414777, 0.31497096, 0.14558263, 0.1235461 , 0.09541113,
       0.09102041, 0.2608112 , 0.07637309, 0.07676458, 0.10931979,
       0.05017179, 0.08388532, 0.10813308, 0.1900822 , 0.03545992,
       0.0091634 , 0.02462897, 0.16628409, 0.02543071, 0.03139766,
       0.07830403, 0.00499672, 0.07289363, 0.03522334, 0.12782832,
       0.1997292 , 0.14301624, 0.2646213 , 0.24404641, 0.01720617,
       0.20455338, 0.34599494, 0.25017011, 0.09201517, 0.05121543,
       0.2112655 , 0.72467912, 0.35414797, 0.02786452, 0.09955845,
       0.04508169, 0.06873754, 0.15215574, 0.10096503, 0.15594135,
       0.08245439, 0.04400721, 0.04334864, 0.14834368, 0.05975021,
       0.04272249, 0.04574552, 0.11551546, 0.00941756, 0.01223489,
       0.22613438, 0.04843507, 0.08376676, 0.80373244, 0.04366118,
       0.027391  , 0.01291323, 0.13356578, 0.17716949, 0.04168438,
       0.01438738, 0.30332401, 0.56809177, 0.26727437, 0.05807149,
       0.42124429, 0.56577335, 0.24697458, 0.06163264, 0.22610041,
       0.08386132, 0.07842809, 0.08930405, 0.17701088, 0.29890668,
       0.03919743, 0.13828096, 0.0033842 , 0.11208064, 0.13953154,
       0.05557145, 0.14898315, 0.05451647, 0.11730045, 0.0341553 ,
       0.04390226, 0.06912775, 0.07821587, 0.01381096, 0.01241026,
       0.38855565, 0.01307225, 0.11239813, 0.80343597, 0.1942669 ,
       0.33130457, 0.16264036, 0.13382165, 0.03038525, 0.00542577,
       0.03733729, 0.17353554, 0.17097854, 0.08239189, 0.0161542 ,
       0.11497677, 0.09675853, 0.09017036, 0.04375561, 0.09275858,
       0.02416675, 0.11140631, 0.00530973, 0.81022589, 0.06321252,
       0.04128112, 0.53764442, 0.04502352, 0.73399774, 0.0824389 ,
       0.34750978, 0.32974373, 0.31554156, 0.05230788, 0.07749644,
       0.21347688, 0.04652234, 0.01956346, 0.25828489, 0.05695346,
       0.15536   , 0.17308441, 0.63309302, 0.0554643 , 0.23351402,
       0.041376  , 0.4338105 , 0.00331211, 0.12265664, 0.02913167,
       0.11640111, 0.18765739, 0.09235799, 0.08987611, 0.24930837,
       0.0231433 , 0.01520841, 0.08704603, 0.0228926 , 0.12615554,
       0.09957933, 0.23980335, 0.67402732, 0.18515948, 0.35788124,
       0.02958548, 0.15886055, 0.16351833, 0.28564987, 0.02851941,
       0.03820667, 0.35638525, 0.05565143, 0.02953751, 0.16095279,
       0.28425   , 0.20774611, 0.01027417, 0.07141752, 0.01208132,
       0.19008834, 0.26995358, 0.01436142, 0.16016645, 0.05334459,
       0.03607947, 0.40769009, 0.4200565 , 0.0318672 , 0.10361636,
       0.4059028 , 0.35236182, 0.73182365, 0.04756796, 0.2262603 ,
       0.0848035 , 0.00518662, 0.62543897, 0.3014179 , 0.35547649,
       0.35354735, 0.0335931 , 0.19246384, 0.05074979, 0.05413504,
       0.15366914, 0.00781302, 0.212933  , 0.37675336, 0.06984907,
       0.10237479, 0.00992434, 0.1386785 , 0.05692345, 0.03202143,
       0.03304906, 0.06063555, 0.35044495, 0.35817183, 0.17578989,
       0.20989673, 0.01532338, 0.12868418, 0.08311898, 0.03062858,
```

```
       0.21635223, 0.00706043, 0.24320165, 0.00281585, 0.0253744 ,
       0.24037253, 0.67469459, 0.06700778, 0.26604659])
```

In [43]:
```
probability=dtc.predict_proba(x_test)[:,1]
probability
```

Out[43]:
```
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., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1.,
       1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 1., 0., 1., 0., 0., 0.,
       0., 0., 0., 1., 0., 0., 0., 1., 1., 0., 0., 0., 1., 0., 0., 1., 0.,
       0., 1., 0., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 1., 0., 0., 1.,
       1., 0., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 0.,
       0., 0., 1., 0., 0., 0., 1., 0., 0., 0., 0., 1., 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., 1., 0., 1., 1., 0., 0., 0., 0., 1., 0., 0., 1.,
       0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.,
       1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
       0., 1., 0., 0., 1., 0., 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., 0., 0., 0.,
       0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 1., 1., 0., 0., 0., 1., 0., 0., 1., 0., 0., 0.,
       0., 0., 0., 0., 0.])
```