NAME: CH.ABHIRAM

REG NO: 21BCE8707

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

```
#Import the Libraries.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
#Importing the dataset.
df=pd.read_csv("Employee-Attrition.csv")
```

```
df.head()
```

	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

```
df.shape
```

```
(1470, 35)
```

```
df.Age.value_counts()
```

```
35    78
34    77
36    69
31    69
29    68
32    61
30    60
33    58
38    58
40    57
37    50
27    48
28    48
42    46
39    42
45    41
41    40
26    39
44    33
46    33
43    32
50    30
25    26
24    26
49    24
47    24
```

```
55    22
51    19
53    19
48    19
54    18
52    18
22    16
56    14
23    14
58    14
21    13
20    11
59    10
19     9
18     8
60     5
57     4
Name: Age, dtype: int64
```

```
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
```

```
df.describe()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNu
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.00

```
#Checking for Null Values.
df.isnull().any()
```

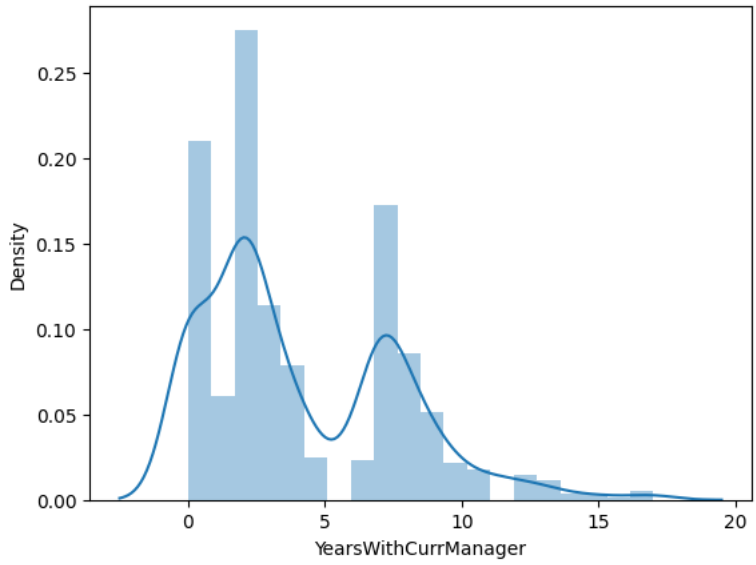
```
Age                         False
Attrition                   False
BusinessTravel              False
DailyRate                   False
Department                  False
DistanceFromHome            False
Education                   False
EducationField              False
EmployeeCount               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
```

```
df.isnull().sum()
```

```
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.
sns.distplot(df["YearsWithCurrManager"])
```

```
<ipython-input-12-71e8291be26b>:2: 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["YearsWithCurrManager"])
<Axes: xlabel='YearsWithCurrManager', ylabel='Density'>
```




```
df.corr()
```

```
<ipython-input-13-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in D
  df.corr()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount
Age	1.000000	0.010661	-0.001686	0.208034	NaN
DailyRate	0.010661	1.000000	-0.004985	-0.016806	NaN
DistanceFromHome	-0.001686	-0.004985	1.000000	0.021042	NaN
Education	0.208034	-0.016806	0.021042	1.000000	NaN
EmployeeCount	NaN	NaN	NaN	NaN	NaN
EmployeeNumber	-0.010145	-0.050990	0.032916	0.042070	NaN

```
df.head()
```

	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

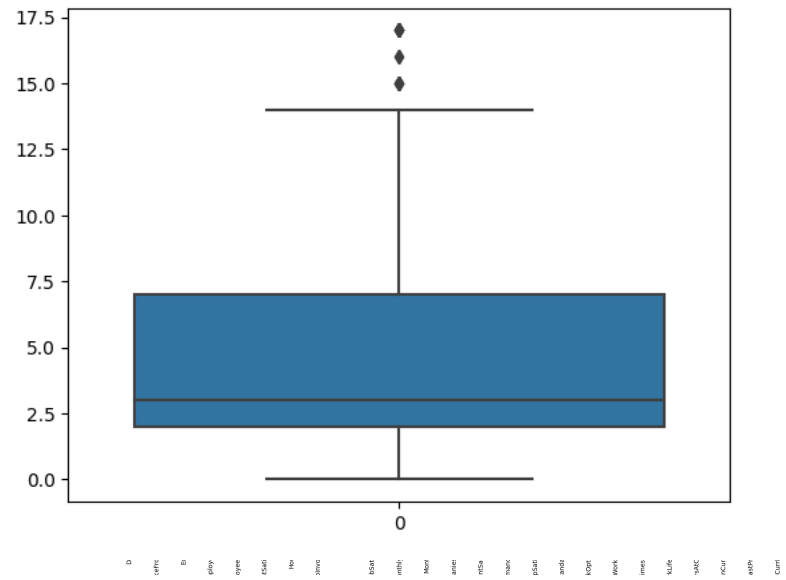
StandardHours	NaN	NaN		NaN	NaN	NaN

```
plt.subplots(figsize = (25,25))
sns.heatmap(df.corr(),annot=True)
```

```
<ipython-input-15-9329d5e70af4>:2: FutureWarning: The default value of numeric_only in D
  sns.heatmap(df.corr(),annot=True)
<Axes: >
```

```
sns.boxplot(df.YearsWithCurrManager)
```

```
<Axes: >
```



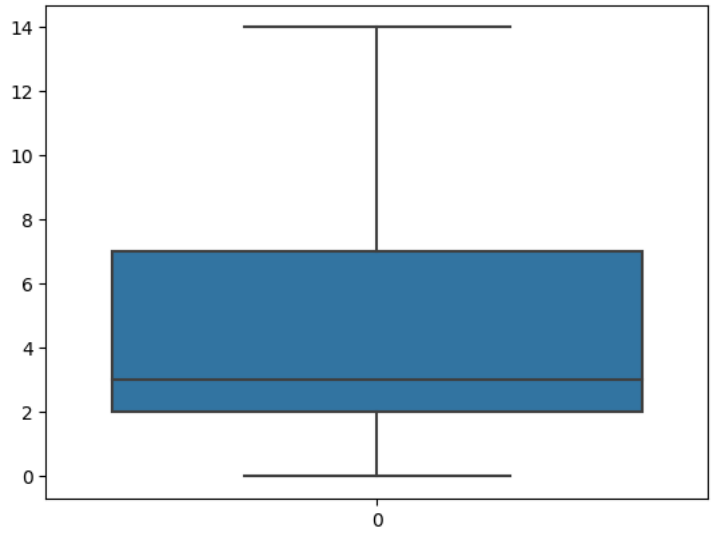

```
from scipy import stats
z_scores = np.abs(stats.zscore(df['YearsWithCurrManager']))
max_threshold=3
outliers = df['YearsWithCurrManager'][z_scores > max_threshold]

# Print and visualize the outliers
print("Outliers detected using Z-Score:")
print(outliers)
```

```
Outliers detected using Z-Score:
28      17
123     15
153     15
187     15
231     15
386     17
561     16
616     17
635     15
686     17
875     17
926     17
1078    17
1348    16
Name: YearsWithCurrManager, dtype: int64
```

```
q1 = df.YearsWithCurrManager.quantile(0.25)
q3 = df.YearsWithCurrManager.quantile(0.75)
print(q1)
print(q3)
upperlimit = q3+1.5*(q3-q1)
upperlimit
lowerlimit = q1-1.5*(q3-q1)
lowerlimit
df.median()
df["YearsWithCurrManager"]=np.where(df["YearsWithCurrManager"]>upperlimit,14,df['YearsWithCurrManager'])
```

```
sns.boxplot(df.YearsWithCurrManager)
```

```
2.0
7.0
<ipython-input-18-3a17581b0650>:9: FutureWarning: The default value of numeric_only in D
  df.median()
<Axes: >
```




```
from scipy import stats
z_scores = np.abs(stats.zscore(df['YearsWithCurrManager']))
max_threshold=3
outliers = df['YearsWithCurrManager'][z_scores > max_threshold]

# Print and visualize the outliers
print("Outliers detected using Z-Score:")
print(outliers)
```

```
Outliers detected using Z-Score:
Series([], Name: YearsWithCurrManager, dtype: int64)
```

```
df.head()
```

	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

```
x=df.drop('Attrition',axis=1)
x.head()
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationFi
0	41	Travel_Rarely	1102	Sales	1	2	Life Scie
1	49	Travel_Frequently	279	Research & Development	8	1	Life Scie
				Research &			

```
y=df.Attrition
y.head()
```

```
0    Yes
1     No
2    Yes
3     No
4     No
Name: Attrition, dtype: object
```

```
#label encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
x.BusinessTravel    =le.fit_transform(x.BusinessTravel  )
x.head()
x.Department    =le.fit_transform(x.Department  )
x.head()
x.EducationField    =le.fit_transform(x.EducationField  )
x.head()
x.Gender=le.fit_transform(x.Gender)
x.head()
x.JobRole   =le.fit_transform(x.JobRole )
x.head()
x.MaritalStatus =le.fit_transform(x.MaritalStatus   )
x.head()
x.Over18    =le.fit_transform(x.Over18  )
x.head()
x.OverTime  =le.fit_transform(x.OverTime    )
x.head()
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	EnvironmentSat
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

```
df.columns
```

```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

```
#feature scaling
from sklearn.preprocessing import MinMaxScaler
ms=MinMaxScaler()
x_scaled=pd.DataFrame(ms.fit_transform(x),columns=x.columns)
```

```
x_scaled
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	Enviro
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.
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)
```

```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

```
((1176, 34), (294, 34), (1176,), (294,))
```

```
x_train.head()
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	Enviro
1374	0.952381	1.0	0.360057	1.0	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	0.5	0.355762	1.0	0.821429	0.00	0.2	0.0	0.615385	

5 rows × 34 columns

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
```

```
model.fit(x_train,y_train)
pred=model.predict(x_test)
pred
```

```
array(['No', 'No', 'No', 'No', 'Yes', 'No', 'Yes', 'No', 'No', 'No', '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', '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', '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', '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',
       'No', '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)
```

```
#label encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
y=le.fit_transform(y)
```

```
y_test
```

```
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
```

```
df
```

	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

Evaluation of classification model

```
#Accuracy score
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_auc_score,roc_curve
```

```
accuracy_score(y_test,pred)
```

```
0.8843537414965986
```

```
confusion_matrix(y_test,pred)
```

```
array([[242,   3],
       [ 31,  18]])
```

```
pd.crosstab(y_test,pred)
```

col_0	No	Yes
Attrition		
No	242	3
Yes	31	18

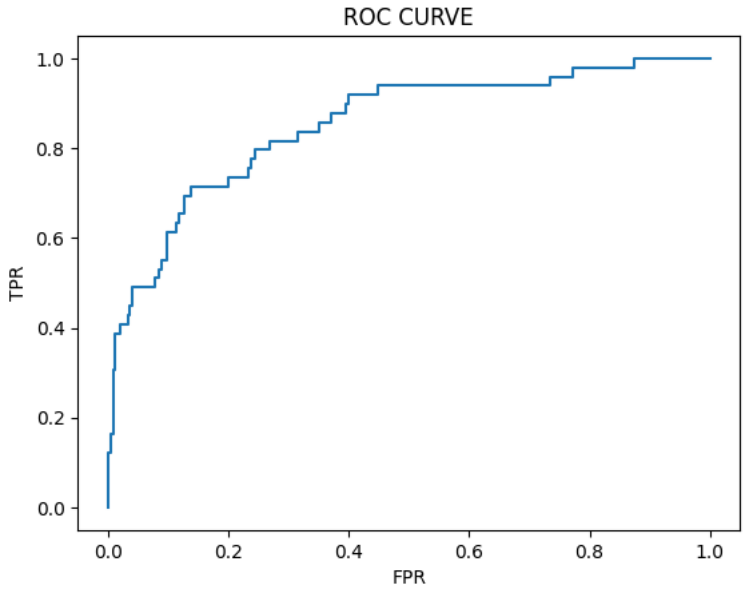
Roc-AUC curve

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

```
array([0.15843867, 0.20617997, 0.31691729, 0.09672152, 0.63876647,
       0.06205401, 0.61414184, 0.07466397, 0.00797252, 0.39157785,
       0.05281564, 0.33160211, 0.02022395, 0.6671328 , 0.19419683,
       0.0335299 , 0.10954936, 0.17130578, 0.043804  , 0.2241511 ,
       0.23531373, 0.01475346, 0.06562592, 0.05019163, 0.59115162,
       0.44667993, 0.07401303, 0.0449937 , 0.67637047, 0.05859033,
       0.01545736, 0.03386798, 0.07021403, 0.1707141 , 0.07767295,
       0.04154894, 0.08312937, 0.06997437, 0.03567429, 0.05269126,
       0.05742727, 0.02144976, 0.01779053, 0.01301572, 0.02825292,
       0.50162054, 0.41541766, 0.00299378, 0.74315718, 0.51799699,
       0.09708281, 0.48942319, 0.07941138, 0.25720931, 0.66861063,
       0.26482373, 0.01970983, 0.30281497, 0.02858501, 0.16213966,
       0.02040161, 0.2173984 , 0.13768821, 0.03568054, 0.37558052,
       0.03010741, 0.29718154, 0.15832399, 0.10264349, 0.08700774,
       0.0815183 , 0.30943969, 0.08708969, 0.07442596, 0.12300414,
       0.0618342 , 0.04633075, 0.07672219, 0.19834226, 0.03129952,
       0.00857215, 0.02394842, 0.13606932, 0.02587787, 0.03217004,
       0.0821409 , 0.00518749, 0.035308  , 0.03813342, 0.14270872,
       0.26418695, 0.16461435, 0.27401734, 0.24146954, 0.02119787,
       0.17774284, 0.34102562, 0.28338745, 0.06906981, 0.04948532,
       0.24465264, 0.74929682, 0.35691434, 0.01878265, 0.08772637,
       0.03239915, 0.05413857, 0.15215059, 0.07127406, 0.13828798,
       0.09342465, 0.04693869, 0.02494493, 0.15041914, 0.07133392,
       0.03025642, 0.05306455, 0.1165452 , 0.00872431, 0.01229042,
       0.17575238, 0.05005249, 0.09018395, 0.82857166, 0.03066995,
       0.0228189 , 0.00874605, 0.13496234, 0.16593413, 0.05060052,
       0.01520085, 0.29791945, 0.54919611, 0.33581407, 0.0469494 ,
       0.38773566, 0.61348127, 0.14171081, 0.07455884, 0.2409655 ,
       0.09528764, 0.06730943, 0.09797576, 0.20026612, 0.20053142,
       0.03046036, 0.14877431, 0.0036571 , 0.11146887, 0.15912883,
       0.06017571, 0.17964687, 0.06063618, 0.1199213 , 0.03284092,
       0.02688355, 0.06536903, 0.08335812, 0.01464284, 0.01536292,
       0.37701597, 0.01262506, 0.15004068, 0.80530948, 0.11655522,
       0.28461049, 0.17042029, 0.15392139, 0.02756879, 0.00599553,
       0.04142216, 0.09958411, 0.11567269, 0.10448555, 0.01830036,
       0.1444171 , 0.1048541 , 0.10079777, 0.05099176, 0.09183576,
       0.02893646, 0.09754427, 0.00516687, 0.75206394, 0.04227453,
       0.04018918, 0.37563319, 0.04457964, 0.72551665, 0.10583031,
       0.36656526, 0.38293703, 0.32923777, 0.05248015, 0.08216713,
       0.13748888, 0.04309097, 0.01429957, 0.2656631 , 0.06297408,
       0.16075744, 0.15388494, 0.67190498, 0.05834473, 0.28467369,
       0.04694404, 0.46237195, 0.00339026, 0.13927388, 0.02695884,
       0.12707414, 0.17395277, 0.0750947 , 0.10135673, 0.16496216,
       0.02583798, 0.01790826, 0.08850395, 0.02838351, 0.13795992,
       0.08655223, 0.22164621, 0.73379009, 0.17294814, 0.40907888,
       0.01503347, 0.11411826, 0.21412683, 0.32566668, 0.03366086,
       0.04472831, 0.32127248, 0.05442236, 0.0242917 , 0.16228044,
       0.32858438, 0.22879119, 0.00852736, 0.0798162 , 0.01140248,
       0.14102568, 0.29116266, 0.01282151, 0.17118076, 0.04051376,
       0.04165738, 0.42684273, 0.35009936, 0.0366853 , 0.11692325,
       0.37940034, 0.31562415, 0.79587005, 0.05488792, 0.21568794,
       0.06397987, 0.00569145, 0.66085682, 0.35796045, 0.37592133,
       0.3650533 , 0.03568965, 0.21192376, 0.05892118, 0.06428028,
       0.10143977, 0.00796354, 0.2678938 , 0.4288445 , 0.0652538 ,
       0.09309022, 0.01226927, 0.14314823, 0.04989664, 0.02304292,
       0.02508766, 0.06618985, 0.24272596, 0.26663754, 0.1979951 ,
       0.26504226, 0.01648205, 0.15826843, 0.08519882, 0.02669729,
       0.18757572, 0.00768502, 0.27928747, 0.0027473 , 0.02506718,
```

```
#label encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
y_test=le.fit_transform(y_test)
```

```
 # roc_curve
fpr,tpr,threshsholds = roc_curve(y_test,probability)


plt.plot(fpr,tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC CURVE')
plt.show()
```



DecisionTreeClassifier

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()


dtc.fit(x_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
pred=dtc.predict(x_test)


pred
```

```
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', 'Yes', 'No', 'No', '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', 'Yes', 'No',
       'Yes', 'Yes', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No',
       'No', 'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No',
       'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', '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', 'Yes', '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', 'Yes', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'Yes', 'No', 'Yes', 'Yes', 'No', 'No',
       'Yes', 'No', 'No', 'No', 'No', 'No', 'Yes', 'Yes', 'No', 'No', 'Yes',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'Yes', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', '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)
```

```
y_test
```

```
array([0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0])
```

```
df
```

	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

Evaluation of classification model

```
#Accuracy score
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_auc_score,roc_curve
```

```
#label encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
y=le.fit_transform(y)
#label encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
pred=le.fit_transform(pred)
```

```
y_test
```

```
array([0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
```

```
0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 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, 1,
1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 1, 0, 0])
```

```
accuracy_score(y_test,pred)
```

```
0.7482993197278912
```

```
confusion_matrix(y_test,pred)
```

```
array([[203,  42],
       [ 32,  17]])
```

```
pd.crosstab(y_test,pred)
```

col_0	0	1
row_0		
0	203	42
1	32	17

```
print(classification_report(y_test,pred))
```

```
                  precision    recall  f1-score   support

               0       0.86      0.83      0.85       245
               1       0.29      0.35      0.31        49

        accuracy                           0.75       294
       macro avg       0.58      0.59      0.58       294
    weighted avg       0.77      0.75      0.76       294
```

Roc-AUC curve

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

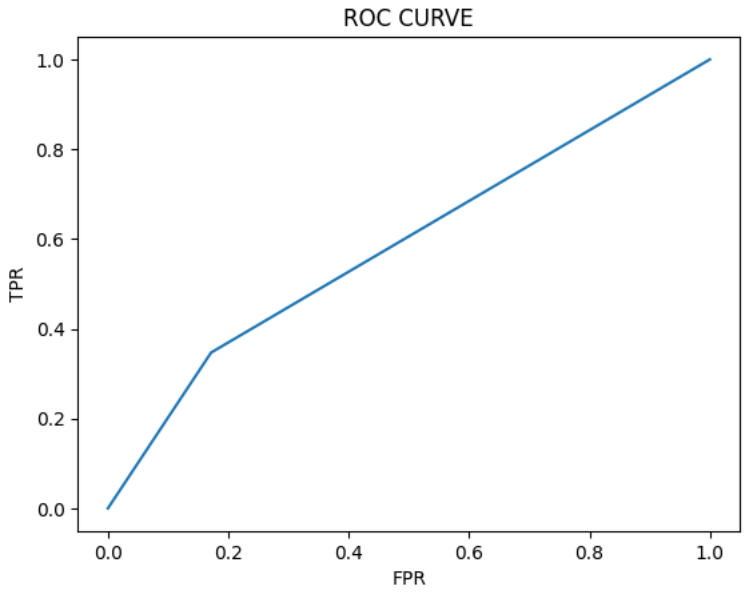
```
probability
```

```
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., 1., 0., 0., 0., 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., 0., 0., 0., 1., 0., 1., 0., 0., 0.,
       0., 0., 0., 1., 0., 1., 0., 1., 1., 0., 0., 0., 1., 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., 1., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 1., 0., 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., 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., 0., 1., 0., 0., 0., 0., 1., 0., 0., 1.,
       0., 0., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
       1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 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., 1., 0., 0.,
       0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 1., 0., 1., 0., 0., 1., 1., 0., 1., 0., 0., 1.,
       0., 0., 0., 0., 0.])
```

```
fpr,tpr,thresholds =  roc_curve(y_test,probability)
```

```
plt.plot(fpr,tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
```

```
plt.title('ROC CURVE')
plt.show()
```




```
from sklearn import tree
plt.figure(figsize=(25,15))
tree.plot_tree(dtc,filled=True)
```

```
[Text(0.3291164340101523, 0.9722222222222222, 'x[27] <= 0.038\ngini = 0.269\nsamples
= 1176\nvalue = [988, 188]'),
 Text(0.08121827411167512, 0.9166666666666666, 'x[16] <= 0.75\ngini = 0.5\nsamples =
78\nvalue = [39, 39]'),
 Text(0.050761421319796954, 0.8611111111111112, 'x[4] <= 0.554\ngini = 0.426\nsamples
= 39\nvalue = [27, 12]'),
 Text(0.0338409475465313, 0.8055555555555556, 'x[15] <= 0.167\ngini = 0.312\nsamples
= 31\nvalue = [25, 6]'),
```

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

}
```

```
 Text(0.04737732656514382, 0.75, 'x[19] <= 0.056\ngini = 0.153\nsamples = 24\nvalue
```

```
grid_search=GridSearchCV(estimator=dtc,param_grid=parameter,cv=5,scoring="accuracy")
```

```
     1]'),
```

```
grid_search.fit(x_train,y_train)
```

```
grid_search.best_params_
```

```
{'criterion': 'gini',
 'max_depth': 3,
 'max_features': 'auto',
 'splitter': 'random'}
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning:
```

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

```
DecisionTreeClassifier(criterion='entropy', max_depth=3, max_features='sqrt')
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning:
```

```
pred=dtc_cv.predict(x_test)
```

```
  warnings.warn(
```

```
#label encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
y=le.fit_transform(y)
#label encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
pred=le.fit_transform(pred)
```

```
  warnings.warn(
```

```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.85      0.97      0.90       245
           1       0.43      0.12      0.19        49

    accuracy                           0.83       294
   macro avg       0.64      0.54      0.55       294
weighted avg       0.78      0.83      0.78       294
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning:
```

RandomForestClassifier

```
  warnings.warn(
```

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
```

```
  warnings.warn(
```

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

```
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning:
```

```
rfc_cv= GridSearchCV(rfc,param_grid=forest_params,cv=10,scoring="accuracy")


rfc_cv.fit(x_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py:378: 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 "/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py", line 686, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py", line 340, in fit
    self._validate_params()
  File "/usr/local/lib/python3.10/dist-packages/sklearn/base.py", line 600, in _validate_params
    validate_parameter_constraints(
  File "/usr/local/lib/python3.10/dist-packages/sklearn/utils/_param_validation.py", line 97, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter of RandomForestClassifier must be an int in the rang
```

```
pred=rfc_cv.predict(x_test)
```

```
array([0.85882949, 0.85891501, 0.86209127, 0.85627088, 0.86127011, 0.8571245 ,
```

```
#label encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
y=le.fit_transform(y)
#label encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
pred=le.fit_transform(pred)
```

```
       0.86223381, 0.85712009, 0.86391424, 0.85882949])
```

```
print(classification_report(y_test,pred))
```

```
                  precision    recall  f1-score   support

               0       0.85      0.98      0.91       245
               1       0.67      0.16      0.26        49

        accuracy                           0.85       294
       macro avg       0.76      0.57      0.59       294
    weighted avg       0.82      0.85      0.81       294
```

```
rfc_cv.best_params_
```

```
{'max_depth': 12, 'max_features': 9}
```