1. Importing necessary modules

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Importing the dataset

```
dataset = pd.read_csv("Employee_Attrition.csv")

dataset.head()
```

```
   Age Attrition     BusinessTravel  DailyRate                Department
\
0   41       Yes      Travel_Rarely       1102                     Sales

1   49        No  Travel_Frequently        279  Research & Development

2   37       Yes      Travel_Rarely       1373  Research & Development

3   33        No  Travel_Frequently       1392  Research & Development

4   27        No      Travel_Rarely        591  Research & Development


   DistanceFromHome  Education EducationField  EmployeeCount
EmployeeNumber  \
0                 1          2  Life Sciences              1
1
1                 8          1  Life Sciences              1
2
2                 2          2          Other              1
4
3                 3          4  Life Sciences              1
5
4                 2          1        Medical              1
7

   ...  RelationshipSatisfaction StandardHours  StockOptionLevel  \
0  ...                         1            80                 0
1  ...                         4            80                 1
2  ...                         2            80                 0
3  ...                         3            80                 0
4  ...                         4            80                 1

   TotalWorkingYears  TrainingTimesLastYear WorkLifeBalance
YearsAtCompany  \
0                  8                      0               1
6
```

```
1                  10                         3                3
10
2                   7                         3                3
0
3                   8                         3                3
8
4                   6                         3                3
2

  YearsInCurrentRole  YearsSinceLastPromotion  YearsWithCurrManager
0                  4                        0                    5
1                  7                        1                    7
2                  0                        0                    0
3                  7                        3                    0
4                  2                        2                    2

[5 rows x 35 columns]
```

```
dataset.shape
```

```
(1470, 35)
```

```
dataset.describe()
```

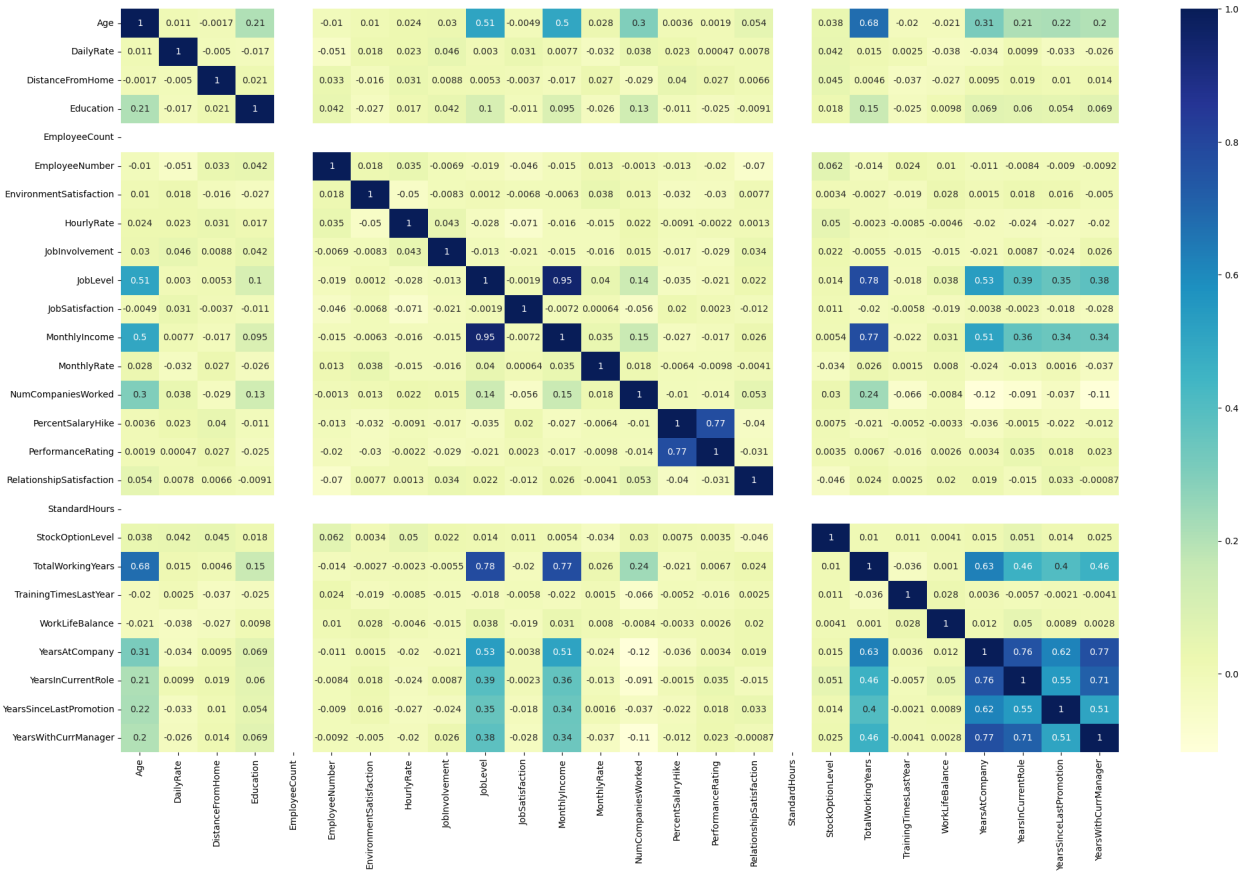
```
                Age    DailyRate  DistanceFromHome    Education
EmployeeCount  \
count  1470.000000  1470.000000       1470.000000  1470.000000
1470.0
mean     36.923810   802.485714          9.192517     2.912925
1.0
std       9.135373   403.509100          8.106864     1.024165
0.0
min      18.000000   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
75%      43.000000  1157.000000         14.000000     4.000000
1.0
max      60.000000  1499.000000         29.000000     5.000000
1.0

       EmployeeNumber  EnvironmentSatisfaction   HourlyRate
JobInvolvement  \
count     1470.000000              1470.000000  1470.000000
1470.000000
mean      1024.865306                 2.721769    65.891156
2.729932
std        602.024335                 1.093082    20.329428
```

```
0.711561
min          1.000000                  1.000000    30.000000
1.000000
25%        491.250000                  2.000000    48.000000
2.000000
50%       1020.500000                  3.000000    66.000000
3.000000
75%       1555.750000                  4.000000    83.750000
3.000000
max       2068.000000                  4.000000   100.000000
4.000000

          JobLevel  ...  RelationshipSatisfaction  StandardHours  \
count  1470.000000  ...               1470.000000         1470.0
mean      2.063946  ...                  2.712245           80.0
std       1.106940  ...                  1.081209            0.0
min       1.000000  ...                  1.000000           80.0
25%       1.000000  ...                  2.000000           80.0
50%       2.000000  ...                  3.000000           80.0
75%       3.000000  ...                  4.000000           80.0
max       5.000000  ...                  4.000000           80.0

       StockOptionLevel  TotalWorkingYears  TrainingTimesLastYear  \
count       1470.000000        1470.000000            1470.000000
mean           0.793878          11.279592               2.799320
std            0.852077           7.780782               1.289271
min            0.000000           0.000000               0.000000
25%            0.000000           6.000000               2.000000
50%            1.000000          10.000000               3.000000
75%            1.000000          15.000000               3.000000
max            3.000000          40.000000               6.000000

       WorkLifeBalance  YearsAtCompany  YearsInCurrentRole  \
count      1470.000000     1470.000000         1470.000000
mean          2.761224        7.008163            4.229252
std           0.706476        6.126525            3.623137
min           1.000000        0.000000            0.000000
25%           2.000000        3.000000            2.000000
50%           3.000000        5.000000            3.000000
75%           3.000000        9.000000            7.000000
max           4.000000       40.000000           18.000000

       YearsSinceLastPromotion  YearsWithCurrManager
count              1470.000000           1470.000000
mean                  2.187755              4.123129
std                   3.222430              3.568136
min                   0.000000              0.000000
25%                   0.000000              2.000000
50%                   1.000000              3.000000
75%                   3.000000              7.000000
```

```
max                      15.000000              17.000000

[8 rows x 26 columns]
```

```
plt.subplots(figsize = [25,15])
sns.heatmap(dataset.corr(), annot = True, cmap = "YlGnBu")
```

```
C:\Users\raman\AppData\Local\Temp\ipykernel_6424\3250113890.py:2:
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(dataset.corr(), annot = True, cmap = "YlGnBu")

<Axes: >
```




```
dataset.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
```

3. Checking null values

```
dataset.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
```

```
dataset.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
```

There are no null values in the dataset

4. Outlier detection

```
dataset.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 
```

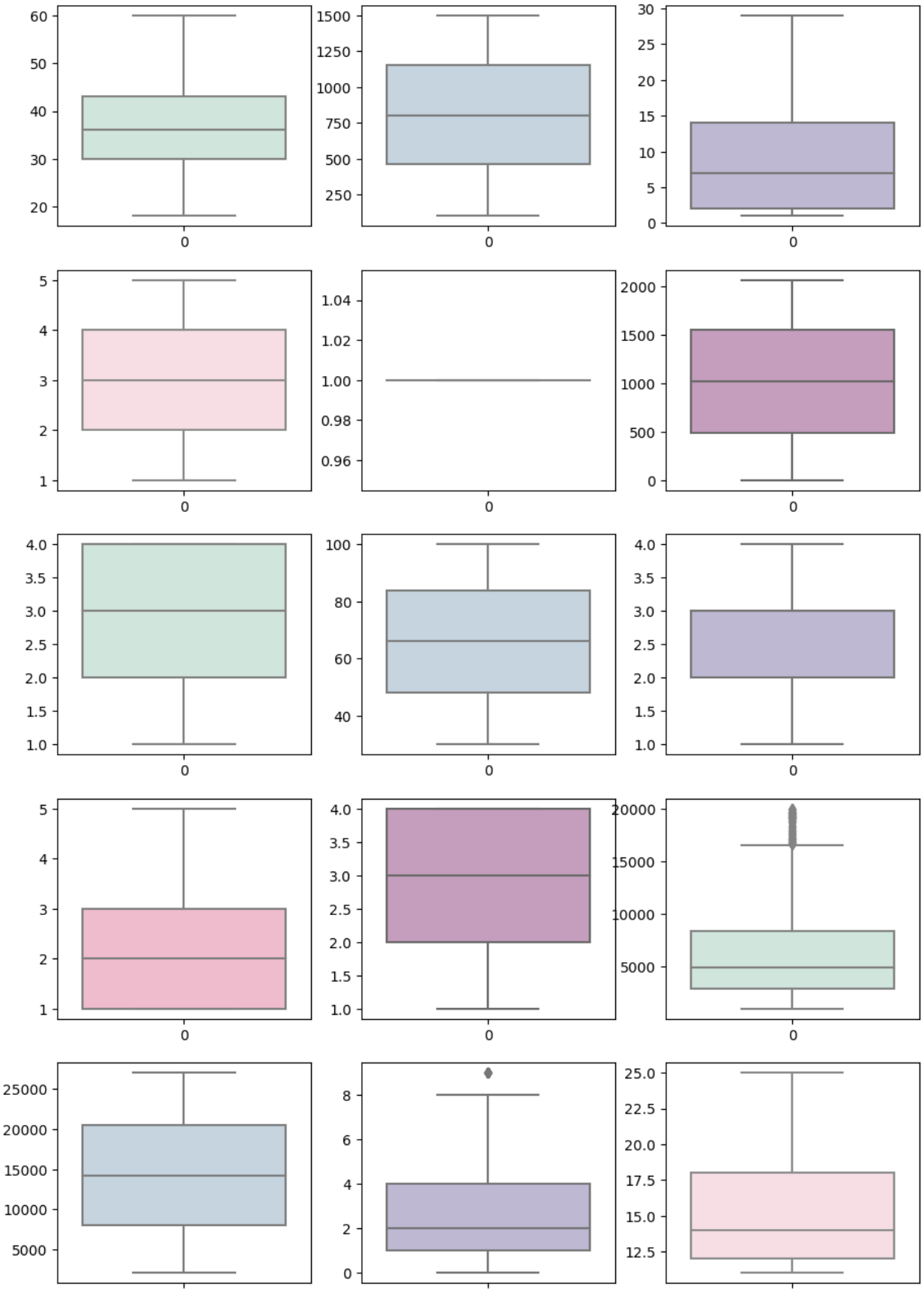
```
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
 24  PerformanceRating         1470 non-null   int64
 25  RelationshipSatisfaction  1470 non-null   int64
 26  StandardHours             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
```

```
# Checking outliers for all numerical columns via boxplot
plt.subplots(figsize = [11,30])
plt.subplot(9,3,1)
sns.boxplot(dataset["Age"], color = "#CCE8DB")
plt.subplot(9,3,2)
sns.boxplot(dataset["DailyRate"], color = "#C1D4E3")
plt.subplot(9,3,3)
sns.boxplot(dataset["DistanceFromHome"], color = "#BBB4D6")
plt.subplot(9,3,4)
sns.boxplot(dataset["Education"], color = "#FADAE2")
plt.subplot(9,3,5)
sns.boxplot(dataset["EmployeeCount"], color = "#F8B3CA")
plt.subplot(9,3,6)
sns.boxplot(dataset["EmployeeNumber"], color = "#CC97C1")
plt.subplot(9,3,7)
sns.boxplot(dataset["EnvironmentSatisfaction"], color = "#CCE8DB")
plt.subplot(9,3,8)
sns.boxplot(dataset["HourlyRate"], color = "#C1D4E3")
plt.subplot(9,3,9)
sns.boxplot(dataset["JobInvolvement"], color = "#BBB4D6")
plt.subplot(9,3,10)
sns.boxplot(dataset["JobLevel"], color = "#F8B3CA")
plt.subplot(9,3,11)
sns.boxplot(dataset["JobSatisfaction"], color = "#CC97C1")
plt.subplot(9,3,12)
sns.boxplot(dataset["MonthlyIncome"], color = "#CCE8DB")
plt.subplot(9,3,13)
sns.boxplot(dataset["MonthlyRate"], color = "#C1D4E3")
plt.subplot(9,3,14)
sns.boxplot(dataset["NumCompaniesWorked"], color = "#BBB4D6")
plt.subplot(9,3,15)
sns.boxplot(dataset["PercentSalaryHike"], color = "#FADAE2")
plt.subplot(9,3,16)
```

```
sns.boxplot(dataset["PerformanceRating"], color = "#F8B3CA")
plt.subplot(9,3,17)
sns.boxplot(dataset["RelationshipSatisfaction"], color = "#CC97C1")
plt.subplot(9,3,18)
sns.boxplot(dataset["StandardHours"], color = "#FADAE2")
plt.subplot(9,3,19)
sns.boxplot(dataset["StockOptionLevel"], color = "#CCE8DB")
plt.subplot(9,3,20)
sns.boxplot(dataset["TotalWorkingYears"], color = "#C1D4E3")
plt.subplot(9,3,21)
sns.boxplot(dataset["TrainingTimesLastYear"], color = "#BBB4D6")
plt.subplot(9,3,22)
sns.boxplot(dataset["WorkLifeBalance"], color = "#FADAE2")
plt.subplot(9,3,23)
sns.boxplot(dataset["YearsAtCompany"], color = "#F8B3CA")
plt.subplot(9,3,24)
sns.boxplot(dataset["YearsInCurrentRole"], color = "#CC97C1")
plt.subplot(9,3,25)
sns.boxplot(dataset["YearsSinceLastPromotion"], color = "#CCE8DB")
plt.subplot(9,3,26)
sns.boxplot(dataset["YearsWithCurrManager"], color = "#C1D4E3")
```

```
C:\Users\raman\AppData\Local\Temp\ipykernel_6424\2558338779.py:3:
MatplotlibDeprecationWarning: Auto-removal of overlapping axes is
deprecated since 3.6 and will be removed two minor releases later;
explicitly call ax.remove() as needed.
  plt.subplot(9,3,1)
```

```
<Axes: >
```


60
50
40
30
20
0
1500
1250
1000
750
500
250
0
30
25
20
15
10
5
0
0
5
4
3
2
1
0
1.04
1.02
1.00
0.98
0.96
0
2000
1500
1000
500
0
0
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
100
80
60
40
0
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
5
4
3
2
1
0
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
20000
15000
10000
5000
0
25000
20000
15000
10000
5000
8
6
4
2
0
25.0
22.5
20.0
17.5
15.0
12.5

Below columns have outliers

1. MonthlyIncome
2. NumCompaniesWorked
3. PerformanceRating
4. StockOptionLevel
5. TotalWorkingYears
6. TrainingTimesLastYear
7. YearsAtCompany
8. YearsInCurrentRole
9. YearsSinceLastPromotion
10. YearsWithCurrManager

```
from scipy import stats
def remove_outliers(column_name, method = "z-score", threshold= 3): #
Default method is z-score
    global dataset

    if method == "iqr":
        q1 = dataset[column_name].quantile(0.25)
        q3 = dataset[column_name].quantile(0.75)
        iqr = q3 - q1
        upper_limit = q3 + (1.5 * iqr)
        lower_limit = q1 - (1.5 * iqr)

        dataset = dataset[dataset[column_name] < upper_limit]
        dataset = dataset[dataset[column_name] > lower_limit]

    elif method == "z-score":
        zscore = stats.zscore(dataset[column_name])

        dataset = dataset[np.abs(zscore) <= 3]

    elif method == "percentile":
        p99 = p99 = dataset[column_name].quantile(0.99)

        dataset = dataset[dataset[column_name] <= p99]
```

Removing outliers via IQR method

```
remove_outliers("MonthlyIncome","iqr")
remove_outliers("NumCompaniesWorked","iqr")
remove_outliers("PerformanceRating")
remove_outliers("StockOptionLevel","iqr")
```

Removing outliers via Z-score method

```
remove_outliers("TotalWorkingYears")
remove_outliers("TrainingTimesLastYear")
remove_outliers("YearsAtCompany")
```

Removing outliers via Percentile method

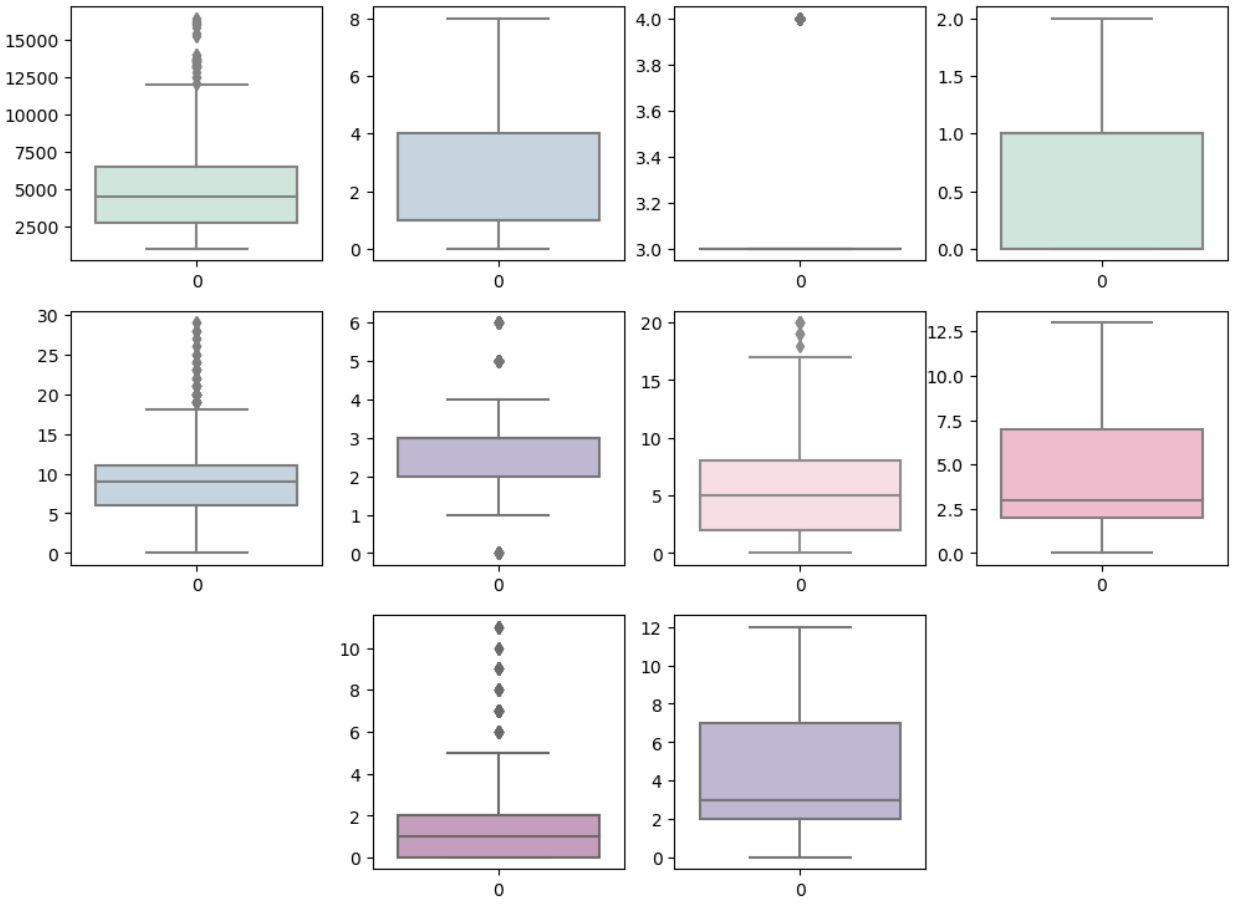
```
remove_outliers("YearsInCurrentRole","percentile")
remove_outliers("YearsSinceLastPromotion","percentile")
remove_outliers("YearsWithCurrManager","percentile")
```

Verifying Outlier removal

```
plt.subplots(figsize = [12,9])
plt.subplot(3,4,1)
sns.boxplot(dataset["MonthlyIncome"], color = "#CCE8DB")
plt.subplot(3,4,2)
sns.boxplot(dataset["NumCompaniesWorked"], color = "#C1D4E3")
plt.subplot(3,4,3)
sns.boxplot(dataset["PerformanceRating"], color = "#BBB4D6")
plt.subplot(3,4,4)
sns.boxplot(dataset["StockOptionLevel"], color = "#CCE8DB")
plt.subplot(3,4,5)
sns.boxplot(dataset["TotalWorkingYears"], color = "#C1D4E3")
plt.subplot(3,4,6)
sns.boxplot(dataset["TrainingTimesLastYear"], color = "#BBB4D6")
plt.subplot(3,4,7)
sns.boxplot(dataset["YearsAtCompany"], color = "#FADAE2")
plt.subplot(3,4,8)
sns.boxplot(dataset["YearsInCurrentRole"], color = "#F8B3CA")
plt.subplot(3,4,10)
sns.boxplot(dataset["YearsSinceLastPromotion"], color = "#CC97C1")
plt.subplot(3,4,11)
sns.boxplot(dataset["YearsWithCurrManager"], color = "#BBB4D6")
```

```
C:\Users\raman\AppData\Local\Temp\ipykernel_6424\874727411.py:2:
MatplotlibDeprecationWarning: Auto-removal of overlapping axes is
deprecated since 3.6 and will be removed two minor releases later;
explicitly call ax.remove() as needed.
  plt.subplot(3,4,1)
```

```
<Axes: >
```



```
dataset.describe()
```

```
                Age    DailyRate  DistanceFromHome    Education
EmployeeCount  \
count  1162.000000  1162.000000       1162.000000  1162.000000
1162.0
mean     35.327022   797.548193          9.371773     2.883821
1.0
std       8.509392   404.025135          8.101498     1.028163
0.0
min      18.000000   103.000000          1.000000     1.000000
1.0
25%      29.000000   462.500000          2.000000     2.000000
1.0
50%      34.000000   798.000000          7.000000     3.000000
1.0
75%      40.000000  1149.250000         14.750000     4.000000
1.0
max      60.000000  1498.000000         29.000000     5.000000
1.0

       EmployeeNumber  EnvironmentSatisfaction   HourlyRate
```

```
JobInvolvement  \
count     1162.000000              1162.000000  1162.000000
1162.000000
mean      1027.776248                 2.717728    65.935456
2.735800
std        608.524702                 1.090802    20.290274
0.710382
min          1.000000                 1.000000    30.000000
1.000000
25%        485.250000                 2.000000    48.000000
2.000000
50%       1009.500000                 3.000000    66.000000
3.000000
75%       1567.000000                 4.000000    84.000000
3.000000
max       2068.000000                 4.000000   100.000000
4.000000

          JobLevel  ...  RelationshipSatisfaction  StandardHours  \
count  1162.000000  ...               1162.000000         1162.0
mean      1.768503  ...                  2.686747           80.0
std       0.774667  ...                  1.090234            0.0
min       1.000000  ...                  1.000000           80.0
25%       1.000000  ...                  2.000000           80.0
50%       2.000000  ...                  3.000000           80.0
75%       2.000000  ...                  4.000000           80.0
max       4.000000  ...                  4.000000           80.0

       StockOptionLevel  TotalWorkingYears  TrainingTimesLastYear  \
count       1162.000000        1162.000000            1162.000000
mean           0.659208           9.086919               2.808090
std            0.686430           5.347222               1.317151
min            0.000000           0.000000               0.000000
25%            0.000000           6.000000               2.000000
50%            1.000000           9.000000               3.000000
75%            1.000000          11.000000               3.000000
max            2.000000          29.000000               6.000000

       WorkLifeBalance  YearsAtCompany  YearsInCurrentRole  \
count      1162.000000     1162.000000         1162.000000
mean          2.758176        5.670396            3.704819
std           0.717114        3.979573            3.098130
min           1.000000        0.000000            0.000000
25%           2.000000        2.000000            2.000000
50%           3.000000        5.000000            3.000000
75%           3.000000        8.000000            7.000000
max           4.000000       20.000000           13.000000

       YearsSinceLastPromotion  YearsWithCurrManager
count              1162.000000           1162.000000
```

```
mean                    1.684165              3.635112
std                     2.418542              3.065083
min                     0.000000              0.000000
25%                     0.000000              2.000000
50%                     1.000000              3.000000
75%                     2.000000              7.000000
max                    11.000000             12.000000

[8 rows x 26 columns]
```

Change in the mean and standard deviation indicates that the outliers are removed

5. Separating dependant and independant variables

```
for i in dataset.columns:
    print(i)
```

```
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

# Dropping Unnecessary columns
dataset.drop(['Over18',
'EmployeeNumber','EmployeeCount','StandardHours'],axis=1, inplace =
True)

# Attrition is the dependant variable in this dataset
x = dataset.drop(columns = ["Attrition"])
y = dataset["Attrition"]

x.head()
```

```
   Age     BusinessTravel  DailyRate              Department  \
0   41      Travel_Rarely       1102                   Sales
1   49  Travel_Frequently        279  Research & Development
2   37      Travel_Rarely       1373  Research & Development
3   33  Travel_Frequently       1392  Research & Development
5   32  Travel_Frequently       1005  Research & Development

   DistanceFromHome  Education EducationField  EnvironmentSatisfaction
\
0                 1          2  Life Sciences                        2

1                 8          1  Life Sciences                        3

2                 2          2         Other                        4

3                 3          4  Life Sciences                        4

5                 2          2  Life Sciences                        4


   Gender  HourlyRate  ...  PerformanceRating
RelationshipSatisfaction  \
0  Female          94  ...                  3
1
1    Male          61  ...                  4
4
2    Male          92  ...                  3
2
3  Female          56  ...                  3
3
5    Male          79  ...                  3
3

  StockOptionLevel  TotalWorkingYears TrainingTimesLastYear
WorkLifeBalance  \
0                0                  8                     0
1
```

```
1                  1                   10                        3
3
2                  0                    7                        3
3
3                  0                    8                        3
3
5                  0                    8                        2
2

   YearsAtCompany  YearsInCurrentRole YearsSinceLastPromotion  \
0               6                   4                        0
1              10                   7                        1
2               0                   0                        0
3               8                   7                        3
5               7                   7                        3

   YearsWithCurrManager
0                     5
1                     7
2                     0
3                     0
5                     6

[5 rows x 30 columns]
```

```
x.shape
```

```
(1162, 30)
```

```
y.head()
```

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

```
y.shape
```

```
(1162,)
```

6. Encoding

```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1162 entries, 0 to 1469
Data columns (total 30 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
```

```
 0   Age                       1162 non-null   int64
 1   BusinessTravel            1162 non-null   object
 2   DailyRate                 1162 non-null   int64
 3   Department                1162 non-null   object
 4   DistanceFromHome          1162 non-null   int64
 5   Education                 1162 non-null   int64
 6   EducationField            1162 non-null   object
 7   EnvironmentSatisfaction   1162 non-null   int64
 8   Gender                    1162 non-null   object
 9   HourlyRate                1162 non-null   int64
 10  JobInvolvement            1162 non-null   int64
 11  JobLevel                  1162 non-null   int64
 12  JobRole                   1162 non-null   object
 13  JobSatisfaction           1162 non-null   int64
 14  MaritalStatus             1162 non-null   object
 15  MonthlyIncome             1162 non-null   int64
 16  MonthlyRate               1162 non-null   int64
 17  NumCompaniesWorked        1162 non-null   int64
 18  OverTime                  1162 non-null   object
 19  PercentSalaryHike         1162 non-null   int64
 20  PerformanceRating         1162 non-null   int64
 21  RelationshipSatisfaction  1162 non-null   int64
 22  StockOptionLevel          1162 non-null   int64
 23  TotalWorkingYears         1162 non-null   int64
 24  TrainingTimesLastYear     1162 non-null   int64
 25  WorkLifeBalance           1162 non-null   int64
 26  YearsAtCompany            1162 non-null   int64
 27  YearsInCurrentRole        1162 non-null   int64
 28  YearsSinceLastPromotion   1162 non-null   int64
 29  YearsWithCurrManager      1162 non-null   int64
dtypes: int64(23), object(7)
memory usage: 313.7+ KB
```

Below columns have non integer values

1. Business Travel
2. Department
3. EducationField
4. Gender
5. JobRole
6. MaritalStatus
7. OverTime

```
print(x.BusinessTravel.nunique(), x.Department.nunique(),
x.EducationField.nunique(), x.Gender.nunique(), x.JobRole.nunique(),
x.MaritalStatus.nunique(), x.OverTime.nunique())

3 3 6 2 9 3 2
```

Using One hot encoding for Overtime as it has only two columns

```
x.shape
```

```
(1162, 30)
```

```
overtime = pd.get_dummies(x["OverTime"],drop_first = True)
overtime.head()
```

```
   Yes
0    1
1    0
2    1
3    1
5    0
```

```
x = pd.concat([x, overtime], axis = 1)
```

```
x.head()
```

```
   Age     BusinessTravel  DailyRate              Department  \
0   41      Travel_Rarely       1102                   Sales
1   49  Travel_Frequently        279  Research & Development
2   37      Travel_Rarely       1373  Research & Development
3   33  Travel_Frequently       1392  Research & Development
5   32  Travel_Frequently       1005  Research & Development

   DistanceFromHome  Education EducationField  EnvironmentSatisfaction
\
0                 1          2  Life Sciences                        2

1                 8          1  Life Sciences                        3

2                 2          2          Other                        4

3                 3          4  Life Sciences                        4

5                 2          2  Life Sciences                        4


   Gender  HourlyRate  ...  RelationshipSatisfaction  StockOptionLevel
\
0  Female          94  ...                         1                 0

1    Male          61  ...                         4                 1

2    Male          92  ...                         2                 0

3  Female          56  ...                         3                 0

5    Male          79  ...                         3                 0
```

```
  TotalWorkingYears  TrainingTimesLastYear WorkLifeBalance
YearsAtCompany  \
0                  8                      0               1
6
1                 10                      3               3
10
2                  7                      3               3
0
3                  8                      3               3
8
5                  8                      2               2
7

   YearsInCurrentRole  YearsSinceLastPromotion YearsWithCurrManager
Yes
0                   4                        0                    5
1
1                   7                        1                    7
0
2                   0                        0                    0
1
3                   7                        3                    0
1
5                   7                        3                    6
0

[5 rows x 31 columns]
```

```
x.drop(columns = ["OverTime"], axis = 1, inplace = True)
```

```
x.head()
```

```
   Age     BusinessTravel  DailyRate              Department  \
0   41      Travel_Rarely       1102                   Sales
1   49  Travel_Frequently        279  Research & Development
2   37      Travel_Rarely       1373  Research & Development
3   33  Travel_Frequently       1392  Research & Development
5   32  Travel_Frequently       1005  Research & Development

   DistanceFromHome  Education EducationField  EnvironmentSatisfaction
\
0                 1          2  Life Sciences                        2

1                 8          1  Life Sciences                        3

2                 2          2         Other                         4

3                 3          4  Life Sciences                        4

5                 2          2  Life Sciences                        4
```

```
   Gender  HourlyRate  ...  RelationshipSatisfaction  StockOptionLevel
\
0  Female          94  ...                         1                 0

1    Male          61  ...                         4                 1

2    Male          92  ...                         2                 0

3  Female          56  ...                         3                 0

5    Male          79  ...                         3                 0


  TotalWorkingYears  TrainingTimesLastYear WorkLifeBalance
YearsAtCompany  \
0                 8                      0               1
6
1                10                      3               3
10
2                 7                      3               3
0
3                 8                      3               3
8
5                 8                      2               2
7

   YearsInCurrentRole  YearsSinceLastPromotion  YearsWithCurrManager
Yes
0                   4                        0                     5
1
1                   7                        1                     7
0
2                   0                        0                     0
1
3                   7                        3                     0
1
5                   7                        3                     6
0

[5 rows x 30 columns]
```

```
x.shape
```

```
(1162, 30)
```

Using Label encoding for the other columns

```
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.head()
```

```
   Age  BusinessTravel  DailyRate  Department  DistanceFromHome
Education  \
0   41               2       1102           2                 1
2
1   49               1        279           1                 8
1
2   37               2       1373           1                 2
2
3   33               1       1392           1                 3
4
5   32               1       1005           1                 2
2

   EducationField  EnvironmentSatisfaction  Gender  HourlyRate  ...  \
0               1                        2       0          94  ...
1               1                        3       1          61  ...
2               4                        4       1          92  ...
3               1                        4       0          56  ...
5               1                        4       1          79  ...

   RelationshipSatisfaction  StockOptionLevel  TotalWorkingYears  \
0                         1                 0                  8
1                         4                 1                 10
2                         2                 0                  7
3                         3                 0                  8
5                         3                 0                  8

   TrainingTimesLastYear  WorkLifeBalance  YearsAtCompany
YearsInCurrentRole  \
0                      0                1               6
4
1                      3                3              10
7
2                      3                3               0
0
3                      3                3               8
7
5                      2                2               7
7

   YearsSinceLastPromotion  YearsWithCurrManager  Yes
```

```
0                          0                          5    1
1                          1                          7    0
2                          0                          0    1
3                          3                          0    1
5                          3                          6    0

[5 rows x 30 columns]
```

7. Feature Scaling

We will use min-max scaling to scale this data

```
from sklearn.preprocessing import MinMaxScaler
ms=MinMaxScaler()

x = pd.DataFrame(ms.fit_transform(x), columns = x.columns)

x.head()
```

```
        Age  BusinessTravel  DailyRate  Department
DistanceFromHome  \
0  0.547619             1.0   0.716129         1.0          0.000000

1  0.738095             0.5   0.126165         0.5          0.250000

2  0.452381             1.0   0.910394         0.5          0.035714

3  0.357143             0.5   0.924014         0.5          0.071429

4  0.333333             0.5   0.646595         0.5          0.035714


   Education  EducationField  EnvironmentSatisfaction  Gender
HourlyRate  \
0       0.25             0.2                 0.333333     0.0
0.914286
1       0.00             0.2                 0.666667     1.0
0.442857
2       0.25             0.8                 1.000000     1.0
0.885714
3       0.75             0.2                 1.000000     0.0
0.371429
4       0.25             0.2                 1.000000     1.0
0.700000

   ...  RelationshipSatisfaction  StockOptionLevel  TotalWorkingYears
\
0  ...                  0.000000               0.0           0.275862

1  ...                  1.000000               0.5           0.344828
```

```
2  ...                    0.333333               0.0           0.241379

3  ...                    0.666667               0.0           0.275862

4  ...                    0.666667               0.0           0.275862


   TrainingTimesLastYear  WorkLifeBalance  YearsAtCompany
YearsInCurrentRole  \
0               0.000000         0.000000            0.30
0.307692
1               0.500000         0.666667            0.50
0.538462
2               0.500000         0.666667            0.00
0.000000
3               0.500000         0.666667            0.40
0.538462
4               0.333333         0.333333            0.35
0.538462

   YearsSinceLastPromotion  YearsWithCurrManager  Yes
0                 0.000000              0.416667  1.0
1                 0.090909              0.583333  0.0
2                 0.000000              0.000000  1.0
3                 0.272727              0.000000  1.0
4                 0.272727              0.500000  0.0

[5 rows x 30 columns]
```

8. Train Test Split

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x, y, test_size =0.2,
random_state =0)

print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
```

```
(929, 30) (233, 30) (929,) (233,)
```

Model Building and Performance Metrics for Different Models

1. Logistic Regression

- Model Building

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()

model.fit(x_train, y_train)

LogisticRegression()

lor_pred = model.predict(x_test)

lor_pred
```

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

```
y_test
```

```
1169     No
735      No
138      No
1062     No
1363     No
       ...
1254     No
54       No
828     Yes
1293     No
1314     No
Name: Attrition, Length: 233, dtype: object
```

- Performance Metrics

```
from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report,roc_auc_score,ro
c_curve

accuracy_score(y_test,lor_pred)
```

```
0.8583690987124464
```

```
confusion_matrix(y_test,lor_pred)
```

```
array([[190,   3],
       [ 30,  10]], dtype=int64)
```

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

```
col_0       No  Yes
Attrition
No         190    3
Yes         30   10
```

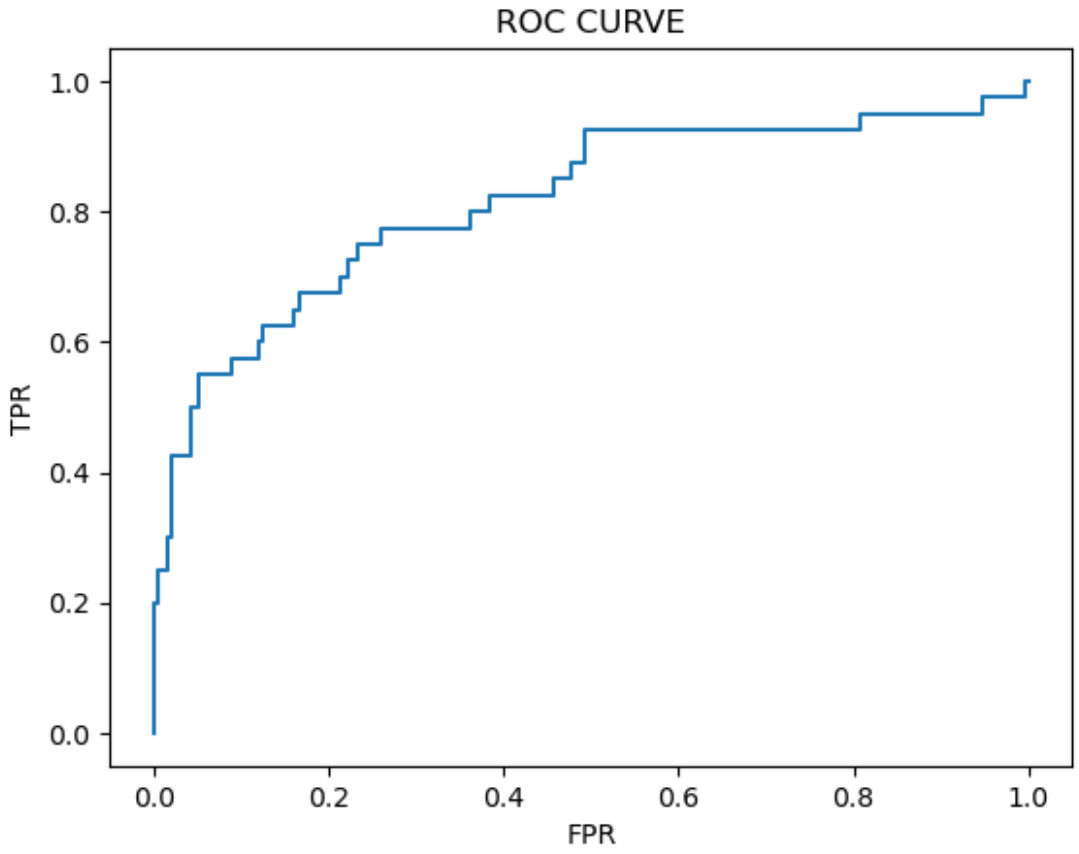
```
print(classification_report(y_test,lor_pred))
```

```
              precision    recall  f1-score   support

          No       0.86      0.98      0.92       193
         Yes       0.77      0.25      0.38        40

    accuracy                           0.86       233
   macro avg       0.82      0.62      0.65       233
weighted avg       0.85      0.86      0.83       233


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

```
array([0.21884798, 0.1842936 , 0.48672887, 0.29242406, 0.15494834,
       0.4509023 , 0.15880704, 0.02010031, 0.06299702, 0.42140598,
       0.02880934, 0.24797127, 0.08680376, 0.38809258, 0.03070421,
       0.02670382, 0.06256929, 0.07348411, 0.09201031, 0.15369572,
       0.09104493, 0.0124355 , 0.23542447, 0.16599539, 0.03887133,
       0.38852922, 0.74293717, 0.01293179, 0.00746001, 0.01046156,
       0.013157  , 0.12400704, 0.02274732, 0.0310812 , 0.11309698,
       0.16168035, 0.03026736, 0.31490537, 0.23926839, 0.03865847,
       0.11998491, 0.50818449, 0.00775669, 0.02791368, 0.45499291,
       0.45696161, 0.15823854, 0.00699001, 0.06858861, 0.34636604,
       0.00420666, 0.01968675, 0.06845107, 0.04005472, 0.43537184,
       0.13753695, 0.10502666, 0.08329553, 0.14211074, 0.20668035,
       0.02086264, 0.10023007, 0.22116233, 0.12555039, 0.25667201,
       0.18405196, 0.60544091, 0.06666404, 0.0651559 , 0.51218045,
       0.01444444, 0.02917515, 0.09702769, 0.02182263, 0.03809518,
       0.21803465, 0.0249948 , 0.01243764, 0.21068717, 0.01603947,
       0.30259644, 0.1784549 , 0.05253635, 0.11896636, 0.03459003,
       0.049032  , 0.05483025, 0.06591024, 0.67855319, 0.01238514,
       0.05124055, 0.17350621, 0.39369213, 0.13253071, 0.18044809,
       0.07465871, 0.17158792, 0.2106257 , 0.26269848, 0.11069508,
       0.01817009, 0.61971188, 0.55199484, 0.04495393, 0.515992  ,
       0.06977832, 0.15391458, 0.39689805, 0.039876  , 0.37799568,
       0.08363938, 0.2135062 , 0.10944132, 0.12969897, 0.01001751,
       0.01224257, 0.02260569, 0.776841  , 0.44799534, 0.15799751,
       0.06824998, 0.0306223 , 0.00920981, 0.00730615, 0.2635132 ,
       0.11410432, 0.01379653, 0.01597104, 0.05613282, 0.09051378,
       0.0350764 , 0.2075274 , 0.01012133, 0.03199128, 0.2014498 ,
       0.05359915, 0.12207974, 0.40239535, 0.28690228, 0.05950859,
       0.08895944, 0.63539672, 0.01966978, 0.04455405, 0.01754886,
       0.09129724, 0.0170675 , 0.06040123, 0.05300055, 0.0161252 ,
       0.02178709, 0.03717715, 0.03097989, 0.02459999, 0.49921926,
       0.18743479, 0.01588419, 0.24219907, 0.21871844, 0.13921929,
       0.44731576, 0.16014089, 0.0690518 , 0.06451598, 0.43260022,
       0.07013957, 0.07681354, 0.01867066, 0.17470481, 0.07358382,
       0.04302043, 0.04332387, 0.04218175, 0.05197101, 0.00908731,
       0.02514873, 0.41410446, 0.09820258, 0.18573655, 0.16493664,
       0.03322204, 0.35464561, 0.00487803, 0.03016803, 0.4268862 ,
```

```
       0.01815954, 0.01725307, 0.02310657, 0.4589503 , 0.06269727,
       0.82548228, 0.10505924, 0.09761444, 0.13819751, 0.00274523,
       0.48685137, 0.22158393, 0.08931115, 0.14132182, 0.05345247,
       0.07680376, 0.28020637, 0.09475478, 0.05961934, 0.36132201,
       0.01144816, 0.04520835, 0.05464981, 0.06818839, 0.21752456,
       0.71509775, 0.19939384, 0.34459675, 0.17873873, 0.16256608,
       0.1229724 , 0.44308411, 0.16181926, 0.00372869, 0.00189897,
       0.20264724, 0.02781871, 0.11902035, 0.1241076 , 0.01844732,
       0.06745283, 0.45183238, 0.08211287, 0.02628573, 0.58663873,
       0.42735461, 0.39076145, 0.01172702])
```

```
# Building ROC curve
# encoding y_test to fit the table
y_true = y_test.map({'Yes': 1, 'No': 0})
fpr,tpr,threshsholds = roc_curve(y_true,probability)

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



2. Decision Tree

- Model Building

```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()

dtc.fit(x_train,y_train)

DecisionTreeClassifier()

dt_pred = dtc.predict(x_test)

dt_pred
```

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

```
       'No', 'No', 'Yes', 'No', 'No', 'Yes', 'No', 'No', 'No'],
      dtype=object)
```

```
y_test
```

```
1169     No
735      No
138      No
1062     No
1363     No
       ...
1254     No
54       No
828     Yes
1293     No
1314     No
Name: Attrition, Length: 233, dtype: object
```

- Performance Metrics

```
accuracy_score(y_test,dt_pred)
```

```
0.7811158798283262
```

```
confusion_matrix(y_test,dt_pred)
```

```
array([[166,  27],
       [ 24,  16]], dtype=int64)
```

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

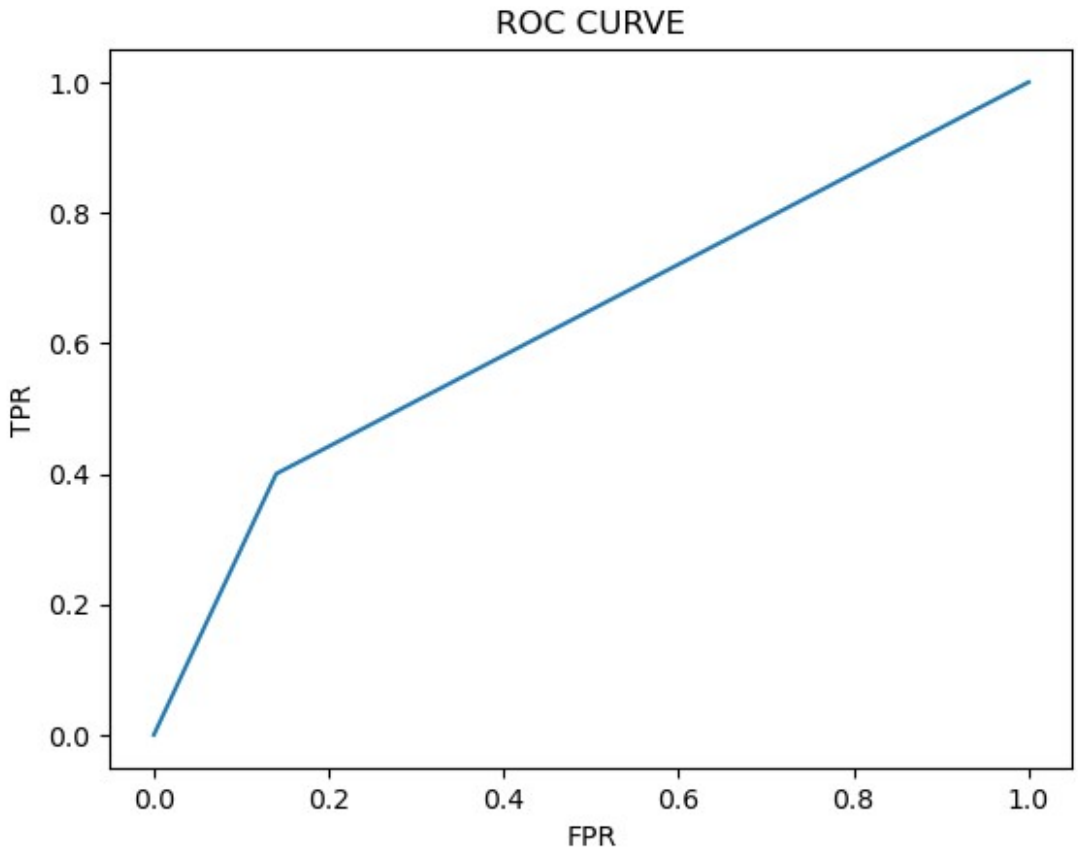
```
col_0       No  Yes
Attrition
No         166   27
Yes         24   16
```

```
print(classification_report(y_test,dt_pred))
```

```
              precision    recall  f1-score   support

          No       0.87      0.86      0.87       193
         Yes       0.37      0.40      0.39        40

    accuracy                           0.78       233
   macro avg       0.62      0.63      0.63       233
weighted avg       0.79      0.78      0.78       233


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

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

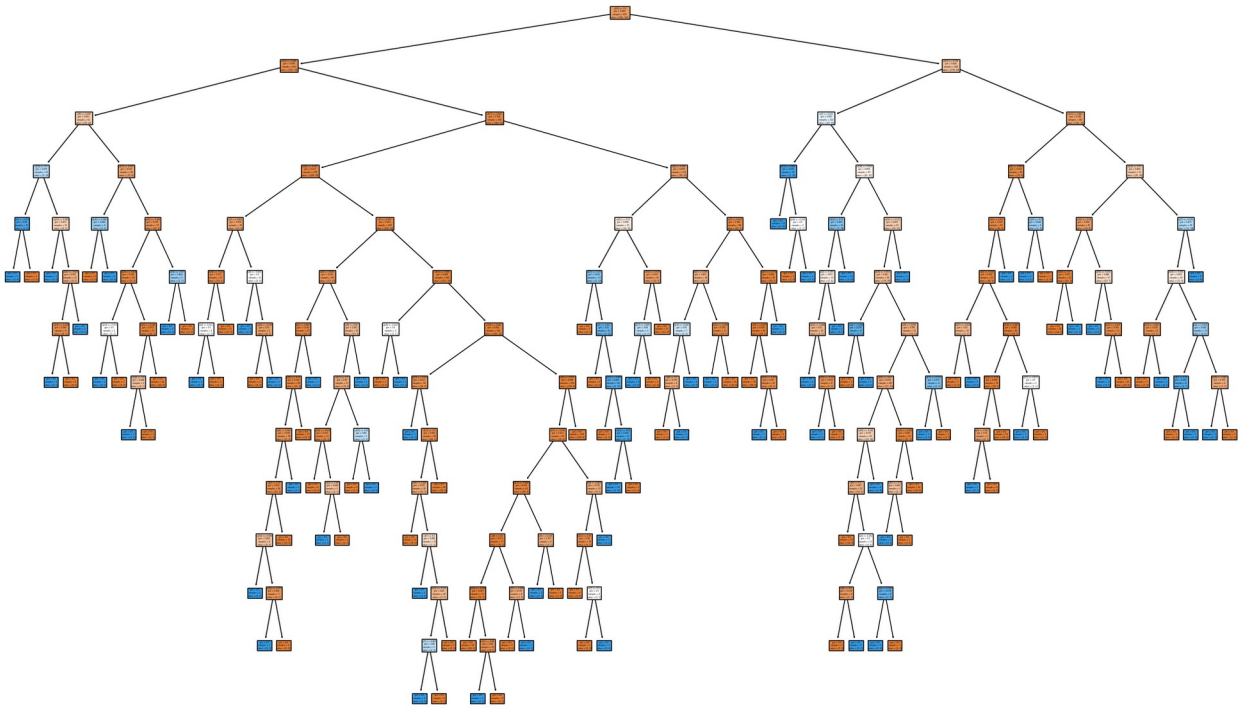
```
# Building ROC curve
# encoding y_test to fit the table
y_true = y_test.map({'Yes': 1, 'No': 0})
fpr,tpr,threshsholds = roc_curve(y_true,probability)

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



The accuracy of this model is not sufficient

- Handling imbalanced data using hyper parameters (for better accuracy)

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



```
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']
}

grid_search=GridSearchCV(estimator=dtc,param_grid=parameter,cv=5,scori
ng="accuracy")

grid_search.fit(x_train,y_train)
```

```
C:\Users\raman\anaconda3\Lib\site-packages\sklearn\model_selection\
_validation.py:425: FitFailedWarning:
100 fits failed out of a total of 300.
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:
--------------------------------------------------------------------
----------
100 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\raman\anaconda3\Lib\site-packages\sklearn\
```

```
model_selection\_validation.py", line 732, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\raman\anaconda3\Lib\site-packages\sklearn\base.py",
line 1144, in wrapper
    estimator._validate_params()
  File "C:\Users\raman\anaconda3\Lib\site-packages\sklearn\base.py",
line 637, in _validate_params
    validate_parameter_constraints(
  File "C:\Users\raman\anaconda3\Lib\site-packages\sklearn\utils\
_param_validation.py", line 95, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The
'max_features' parameter of DecisionTreeClassifier must be an int in
the range [1, inf), a float in the range (0.0, 1.0], a str among
{'sqrt', 'log2'} or None. Got 'auto' instead.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\raman\anaconda3\Lib\site-packages\sklearn\model_selection\
_search.py:976: UserWarning: One or more of the test scores are non-
finite: [       nan        nan 0.82454519 0.82454519 0.82454519
0.82454519
        nan        nan 0.81915141 0.82454519 0.82238303 0.827771
        nan        nan 0.82346411 0.82024993 0.83207207 0.82024412
        nan        nan 0.824551   0.81915722 0.8202383  0.82128451
        nan        nan 0.80519035 0.82560302 0.81593142 0.81702412
        nan        nan 0.82454519 0.82562046 0.82454519 0.82454519
        nan        nan 0.82778262 0.82239465 0.82884627 0.82454519
        nan        nan 0.80520198 0.82454519 0.8159256  0.82666667
        nan        nan 0.82992153 0.82990991 0.81057251 0.80842197
        nan        nan 0.81916303 0.81807614 0.80946237 0.83636152]
  warnings.warn(
```

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

```
grid_search.best_params_
```

```
{'criterion': 'entropy',
 'max_depth': 5,
 'max_features': 'log2',
 'splitter': 'random'}
```

```
dtc_cv=DecisionTreeClassifier(criterion = 'entropy', max_depth = 5,
max_features = 'log2', splitter = 'random')
```

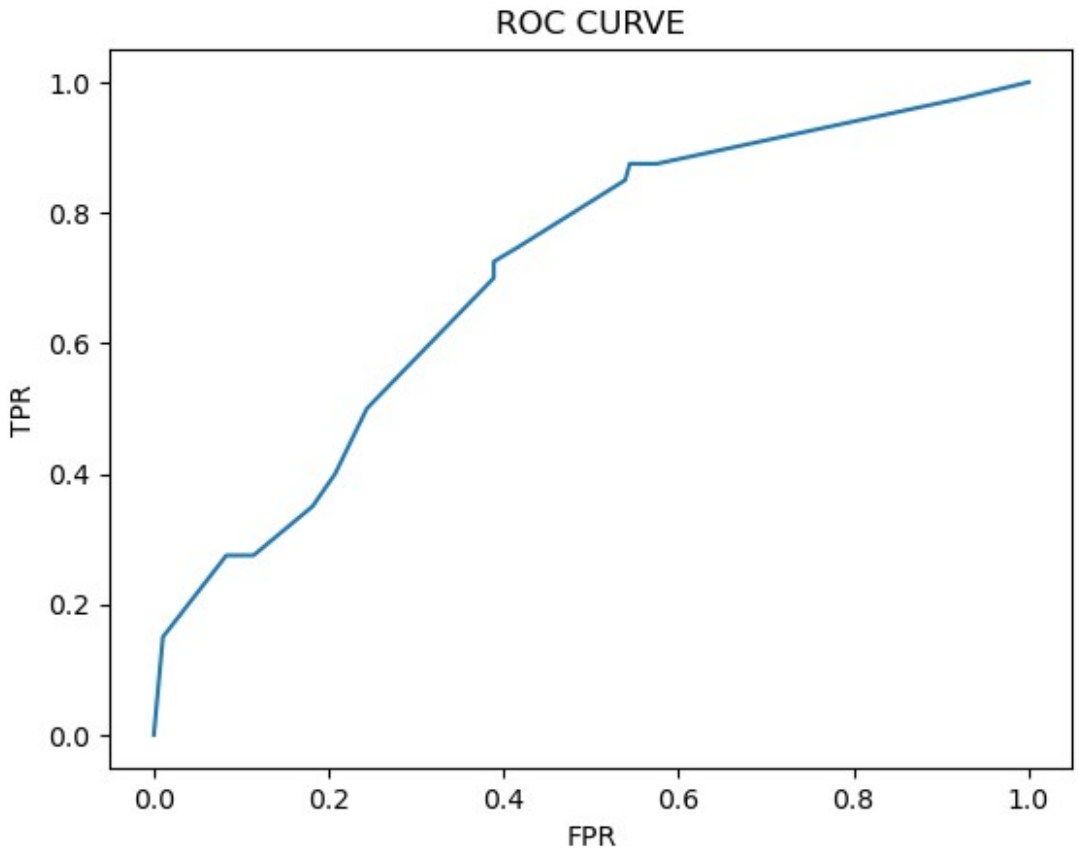
```
dtc_cv.fit(x_train,y_train)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=5,
max_features='log2',
                       splitter='random')

dt_cv_pred=dtc_cv.predict(x_test)

print(classification_report(y_test,dt_cv_pred))
```

```
              precision    recall  f1-score   support

          No       0.85      0.99      0.91       193
         Yes       0.75      0.15      0.25        40

    accuracy                           0.85       233
   macro avg       0.80      0.57      0.58       233
weighted avg       0.83      0.85      0.80       233


accuracy_score(y_test,dt_cv_pred)

0.8454935622317596

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

array([0.04605263, 0.08333333, 0.        , 0.44186047, 0.14457831,
       0.18382353, 0.14457831, 0.04605263, 0.        , 0.08333333,
       0.04605263, 0.18382353, 0.18382353, 0.44186047, 0.04605263,
       0.04605263, 0.20930233, 0.34042553, 0.34042553, 0.14457831,
       0.04605263, 0.04605263, 0.14457831, 0.625     , 0.18382353,
       0.18382353, 0.18181818, 0.04605263, 0.04605263, 0.04605263,
       0.        , 0.        , 0.        , 0.34042553, 0.18382353,
       0.04605263, 0.18382353, 0.44444444, 0.20930233, 0.04605263,
       0.20930233, 0.18382353, 0.04605263, 0.18382353, 0.625     ,
       0.11111111, 0.44444444, 0.04605263, 0.14457831, 0.14457831,
       0.04605263, 0.04605263, 0.04605263, 0.04605263, 0.14457831,
       0.        , 0.04605263, 0.04605263, 0.34042553, 0.11111111,
       0.44444444, 0.18382353, 0.44444444, 0.44444444, 0.44444444,
       0.        , 0.20930233, 0.04605263, 0.        , 0.44444444,
       0.14457831, 0.04605263, 0.04605263, 0.20930233, 0.04605263,
       0.18382353, 0.04605263, 0.04605263, 0.04605263, 0.04605263,
       0.18382353, 0.18382353, 0.18382353, 0.08333333, 0.04605263,
       0.08333333, 0.04605263, 0.15      , 0.14457831, 0.        ,
       0.14457831, 0.04605263, 0.04605263, 0.3       , 0.18382353,
       0.        , 0.14457831, 0.44186047, 0.34042553, 0.20930233,
       0.04605263, 0.3       , 0.625     , 0.3       , 0.18382353,
       0.04605263, 0.14457831, 0.625     , 0.04605263, 0.3       ,
       0.14457831, 0.18382353, 0.14457831, 0.34042553, 0.15      ,
       0.04605263, 0.34042553, 0.625     , 0.44186047, 0.14457831,
       0.        , 0.04605263, 0.18382353, 0.04605263, 0.18382353,
       0.625     , 0.04605263, 0.04605263, 0.04605263, 0.20930233,
```

```
       0.04605263, 0.15      , 0.04605263, 0.14457831, 0.3       ,
       0.18382353, 0.04605263, 0.20930233, 0.08333333, 0.15      ,
       0.14457831, 0.44444444, 0.04605263, 0.04605263, 0.04605263,
       0.20930233, 0.18382353, 0.04605263, 0.15      , 0.04605263,
       0.14457831, 0.14457831, 0.04605263, 0.04605263, 0.34042553,
       0.44444444, 0.34042553, 0.18382353, 0.18382353, 0.44444444,
       0.44444444, 0.44444444, 0.34042553, 0.14457831, 0.625     ,
       0.14457831, 0.04605263, 0.34042553, 0.        , 0.14457831,
       0.04605263, 0.04605263, 0.        , 0.18382353, 0.34042553,
       0.14457831, 0.44186047, 0.04605263, 0.04605263, 0.18382353,
       0.        , 0.20930233, 0.04605263, 0.34042553, 0.625     ,
       0.04605263, 0.14457831, 0.08333333, 0.34042553, 0.34042553,
       0.44444444, 0.18382353, 0.44186047, 0.        , 0.04605263,
       0.18382353, 0.18382353, 0.44444444, 0.44444444, 0.14457831,
       0.44444444, 0.14457831, 0.04605263, 0.15      , 0.3       ,
       0.04605263, 0.04605263, 0.18382353, 0.04605263, 0.18382353,
       0.3       , 0.18382353, 0.18382353, 0.44444444, 0.18382353,
       0.        , 0.18382353, 0.04605263, 0.04605263, 0.04605263,
       0.20930233, 0.04605263, 0.18382353, 0.15      , 0.04605263,
       0.04605263, 0.34042553, 0.04605263, 0.14457831, 0.18382353,
       0.44444444, 0.44444444, 0.04605263])
```

```
# Building ROC curve
# encoding y_test to fit the table
y_true = y_test.map({'Yes': 1, 'No': 0})
fpr,tpr,threshsholds = roc_curve(y_true,probability)

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



There is a satisfactory increase in the accuracy

3. Random Forest

- Model Building

```
from sklearn.ensemble import RandomForestClassifier

rfc=RandomForestClassifier()

rfc.fit(x_train,y_train)

RandomForestClassifier()

rf_pred = rfc.predict(x_test)

rf_pred

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

```
y_test
```

```
1169     No
735      No
138      No
1062     No
1363     No
       ...
1254     No
54       No
828     Yes
1293     No
```

```
1314     No
Name: Attrition, Length: 233, dtype: object
```

- Performance Metrics

```
accuracy_score(y_test,rf_pred)
```

```
0.8669527896995708
```

```
confusion_matrix(y_test,rf_pred)
```

```
array([[193,   0],
       [ 31,   9]], dtype=int64)
```

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

```
col_0       No  Yes
Attrition
No         193    0
Yes         31    9
```

```
print(classification_report(y_test,rf_pred))
```

```
              precision    recall  f1-score   support

          No       0.86      1.00      0.93       193
         Yes       1.00      0.23      0.37        40

    accuracy                           0.87       233
   macro avg       0.93      0.61      0.65       233
weighted avg       0.89      0.87      0.83       233
```