# NAME: SHAIK MOHID BABU

# REG. NO: 21BCE9569

# Branch: Computer Science and Engineering with Specialization in Artificial Intelligence and Machine Learning

# Campus: VIT-AP

### Email: mohidbabu.21bce9569@vitapstudent.ac.in


## Index:

### 1. Data Preprocessing

```
I) Import the Libraries.

II) Importing the dataset.

III) Checking for Null Values.

IV) Data Visualization.

V) Outlier Detection

VI) Splitting Dependent and Independent variables

VII) Perform Encoding

VIII) Feature Scaling.

XI) Splitting Data into Train and Test
```

### 2. Logistic Regression

```
I) Evaluation Matrices
   i) Accuracy
   ii) classification Report
   iii) confusion matrix
   iv) ROC curve
```

### 3. A Logistic Regression model was built on the dataset without applying scaling

```
I) Evaluation Matrices
   i) Accuracy
   ii) classification Report
   iii) confusion matrix
   iv) ROC curve
```

### 4. HyperParameter Tuning on Logistic Regression

```
I) Evaluation Matrices
   i) Accuracy
   ii) classification Report
   iii) confusion matrix
   iv) ROC curve
```

### 5. Decision Tree

```
I) Evaluation Matrices
   i) Accuracy
   ii) classification Report
```

```
iii) confusion matrix
iv) ROC curve
```

**6. HyperParameter Tuning on Decision Tree**

```
I) Evaluation Matrices
   i) Accuracy
   ii) classification Report
   iii) confusion matrix
   iv) ROC curve
```

**7. Random Forest**

```
I) Evaluation Matrices
    i) Accuracy
    ii) classification Report
    iii) confusion matrix
    iv) ROC curve
```

**8. HyperParameter Tuning on Random Forest**

```
I) Evaluation Matrices
    i) Accuracy
    ii) classification Report
    iii) confusion matrix
    iv) ROC curve
```

**9. Report**

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

4.Calculate Performance metrics

# Data Preprocessing

Data Preprocessing

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

## 1. Import the Libraries.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
```

## 2. Importing the dataset.

```
df=pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv")
```

```
df.head()
```

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

5 rows × 35 columns

DataSet Discription

Education

1.'Below College' 2 'College' 3 'Bachelor' 4 'Master' 5 'Doctor'

EnvironmentSatisfaction

1 'Low' 2 'Medium' 3 'High' 4 'Very High'

JobInvolvement 1 'Low' 2 'Medium' 3 'High' 4 'Very High'

JobSatisfaction

1 'Low' 2 'Medium' 3 'High' 4 'Very High'

PerformanceRating

1 'Low' 2 'Good' 3 'Excellent' 4 'Outstanding'

RelationshipSatisfaction

1 'Low' 2 'Medium' 3 'High' 4 'Very High'

WorkLifeBalance

1 'Bad' 2 'Good' 3 'Better' 4 'Best'

```
df.EnvironmentSatisfaction
```

```
0       2
1       3
2       4
3       4
4       1
       ..
1465    3
1466    4
1467    2
1468    4
1469    2
Name: EnvironmentSatisfaction, Length: 1470, dtype: int64
```

```
df.JobRole.unique()
```

```
array(['Sales Executive', 'Research Scientist', 'Laboratory Technician',
       'Manufacturing Director', 'Healthcare Representative', 'Manager',
       'Sales Representative', 'Research Director', 'Human Resources'],
      dtype=object)
```

```
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
```

```python
df.BusinessTravel.unique()
```

```
array(['Travel_Rarely', 'Travel_Frequently', 'Non-Travel'], dtype=object)
```

```python
df.Department.unique()
```

```
array(['Sales', 'Research & Development', 'Human Resources'], dtype=object)
```

```python
df.EducationField.unique()
```

```
array(['Life Sciences', 'Other', 'Medical', 'Marketing',
       'Technical Degree', 'Human Resources'], dtype=object)
```

```python
df.JobRole.unique()
```

```
array(['Sales Executive', 'Research Scientist', 'Laboratory Technician',
       'Manufacturing Director', 'Healthcare Representative', 'Manager',
       'Sales Representative', 'Research Director', 'Human Resources'],
      dtype=object)
```

```python
df.MaritalStatus.unique()
```

```
array(['Single', 'Married', 'Divorced'], dtype=object)
```

```python
df.Over18.unique()
```

```
array(['Y'], dtype=object)
```

```python
df.OverTime.unique()
```

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

```python
for column in df.columns:
    print(f"{column}: Number of unique values ={df[column].nunique()}")
    print()
```

```
Education: Number of unique values =5

EducationField: Number of unique values =6

EmployeeCount: Number of unique values =1
```

```
JobRole: Number of unique values =9

JobSatisfaction: Number of unique values =4

MaritalStatus: Number of unique values =3

MonthlyIncome: Number of unique values =1349

MonthlyRate: Number of unique values =1427

NumCompaniesWorked: Number of unique values =10

Over18: Number of unique values =1

OverTime: Number of unique values =2

PercentSalaryHike: Number of unique values =15

PerformanceRating: Number of unique values =2

RelationshipSatisfaction: Number of unique values =4

StandardHours: Number of unique values =1

StockOptionLevel: Number of unique values =4

TotalWorkingYears: Number of unique values =40

TrainingTimesLastYear: Number of unique values =7

WorkLifeBalance: Number of unique values =4

YearsAtCompany: Number of unique values =37

YearsInCurrentRole: Number of unique values =19

YearsSinceLastPromotion: Number of unique values =16

YearsWithCurrManager: Number of unique values =18
```

## 3. 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
```

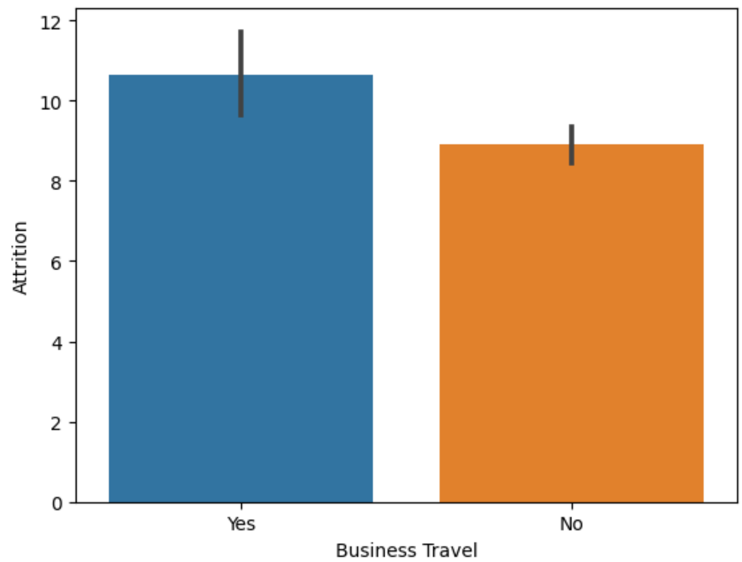
```
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      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
```

The dataset does not contain any null values.

## 4. Data Visualization.

```
sns.barplot(x="Attrition",y="DistanceFromHome",data=df)
plt.xlabel("Business Travel")
plt.ylabel("Attrition")
```

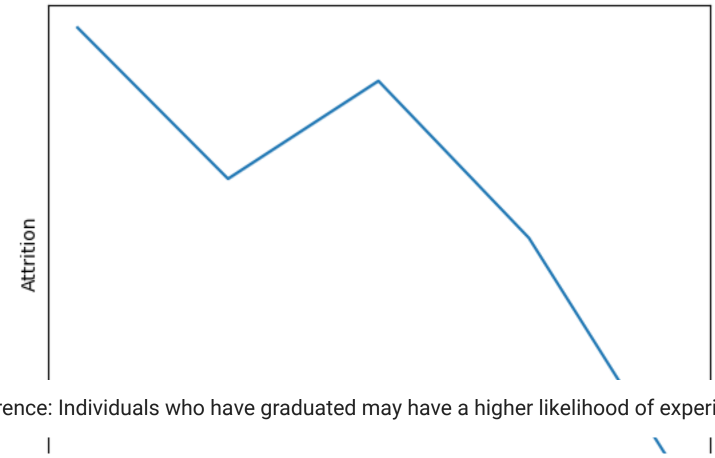
```
Text(0, 0.5, 'Attrition')
```




Inference: It is highly probable that employees who were distant from the office are more likely to experience attrition

```
sns.lineplot(x="Education",y="Attrition",data=df,ci=0)
```

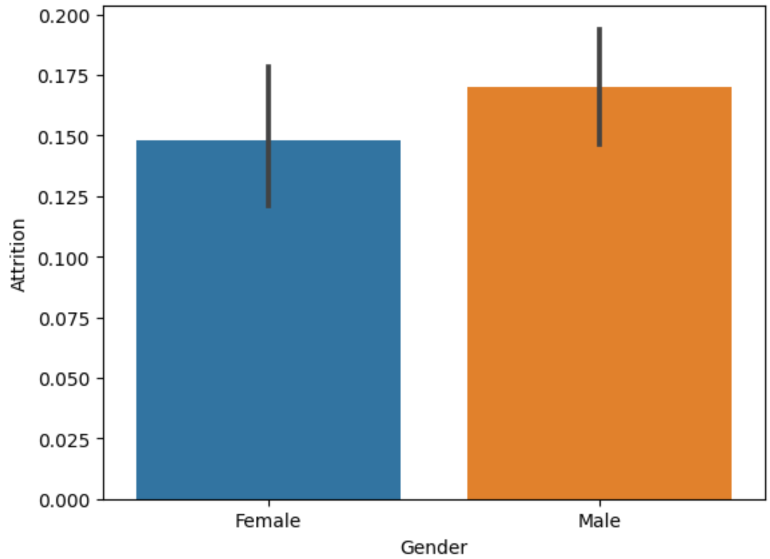
```
<ipython-input-207-93579bdd456f>:1: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=('ci', 0)` for the same effect.

  sns.lineplot(x="Education",y="Attrition",data=df,ci=0)
<Axes: xlabel='Education', ylabel='Attrition'>
```




Inference: Individuals who have graduated may have a higher likelihood of experiencing attrition.

```
df['Attrition'] = df['Attrition'].map({'Yes': 1, 'No': 0})
sns.barplot(x="Gender", y="Attrition", data=df)
```

```
<Axes: xlabel='Gender', ylabel='Attrition'>
```




Inference: Male employees are more prone to experiencing attrition.

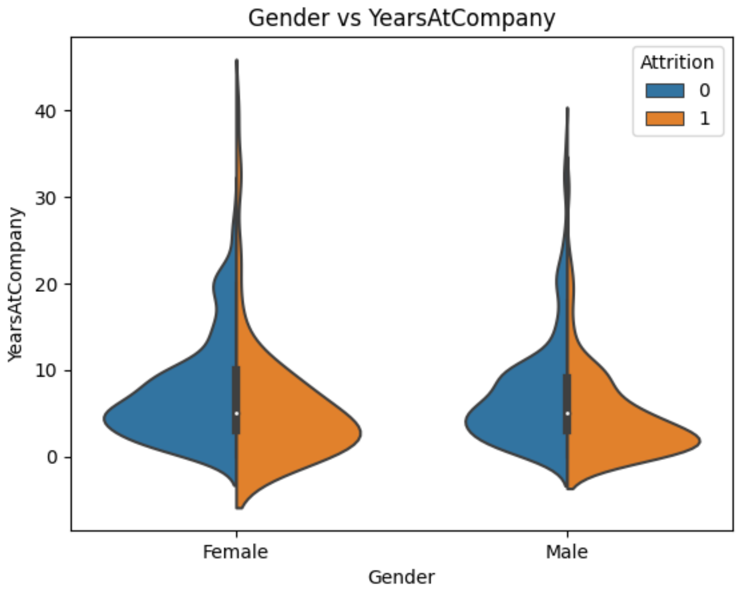
```
sns.distplot(df.PerformanceRating)
```

```
<ipython-input-209-f9adf02d0577>:1: UserWarning:

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

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

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

  sns.distplot(df.PerformanceRating)
```

Inference: Most employees who experience attrition have a performance rating of 3 (Excellent).

```
plt.title("Gender vs YearsAtCompany")
sns.violinplot(x="Gender",y="YearsAtCompany",hue="Attrition",split=True,data=df)
```

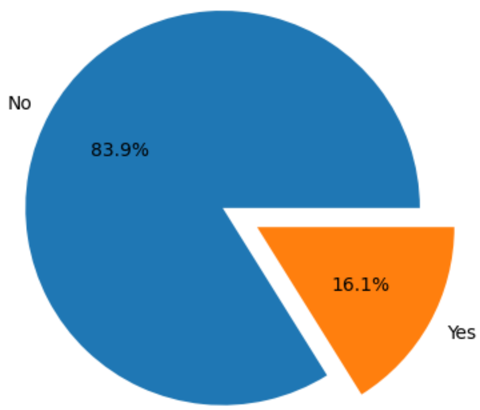
```
<Axes: title={'center': 'Gender vs YearsAtCompany'}, xlabel='Gender', ylabel='YearsAtCompany'>
```




```
df.Attrition.value_counts()
```

```
0    1233
1     237
Name: Attrition, dtype: int64
```

```
labels=["No","Yes"]
plt.pie(df.Attrition.value_counts(),labels=labels,autopct='%1.1f%%',explode=(0.2,0))
plt.show()
```



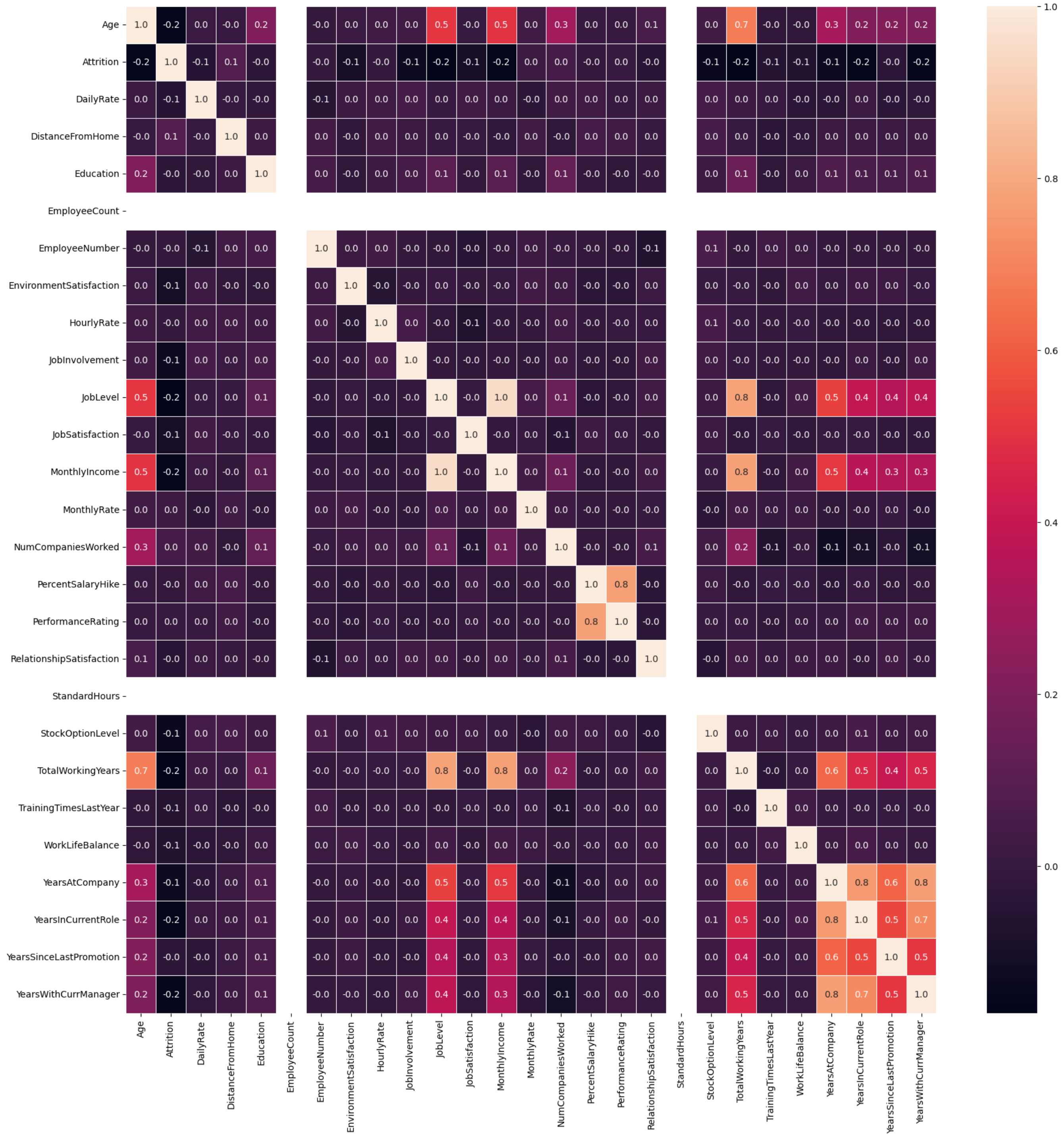

```
corr_matrix=df.select_dtypes(include=['number']).corr()
```

```
print(type(corr_matrix))
```

```
<class 'pandas.core.frame.DataFrame'>
```

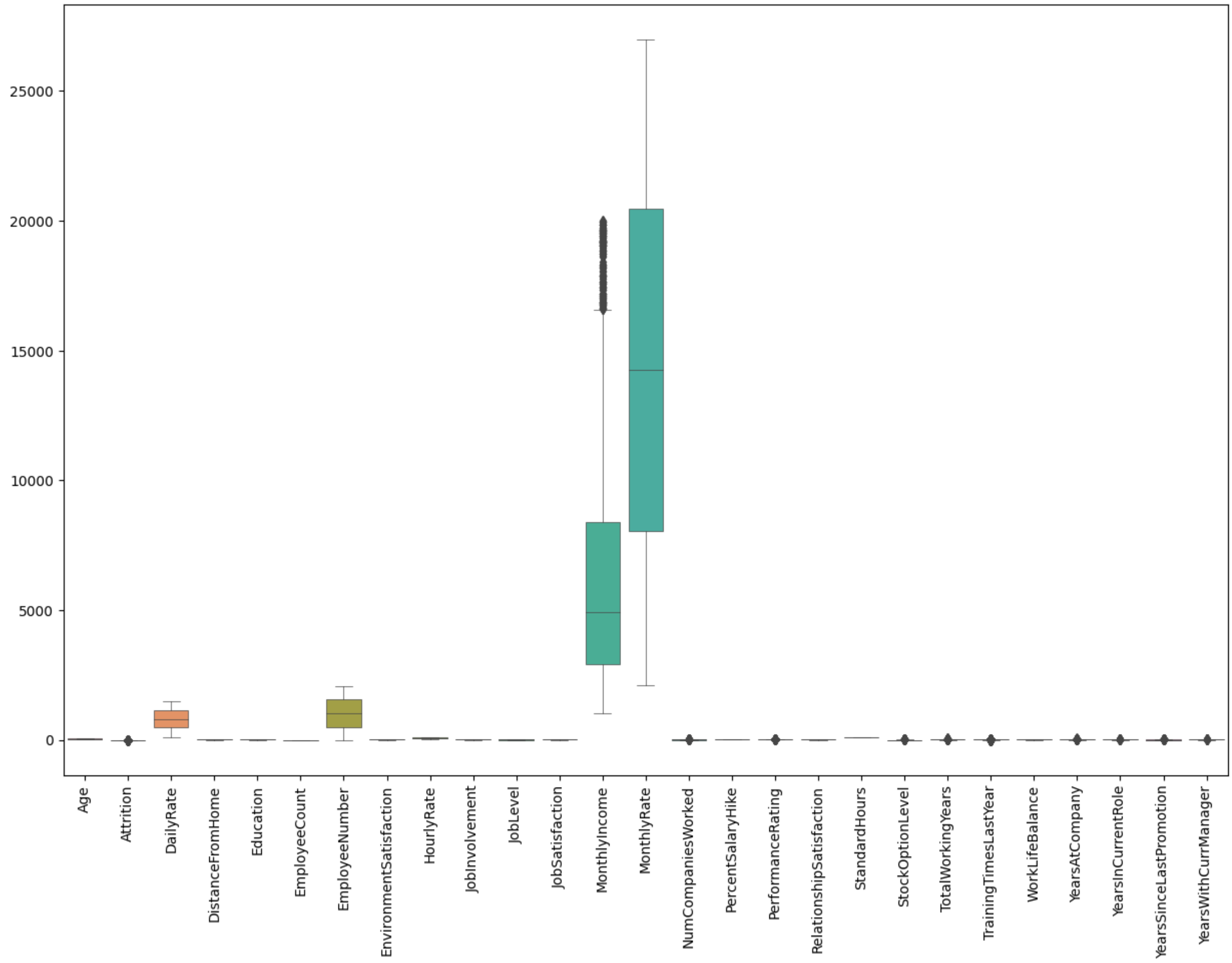
```
f,ax = plt.subplots(figsize=(20, 20))
sns.heatmap(corr_matrix, annot=True, linewidths=.5, fmt= '.1f',ax=ax)
```

```
<Axes: >
```



## 5. Outlier Detection

```
f, ax = plt.subplots(figsize=(15, 10))
sns.boxplot(data=df, linewidth=0.5, fliersize=5, ax=ax)
plt.xticks(rotation=90)
plt.show()
```

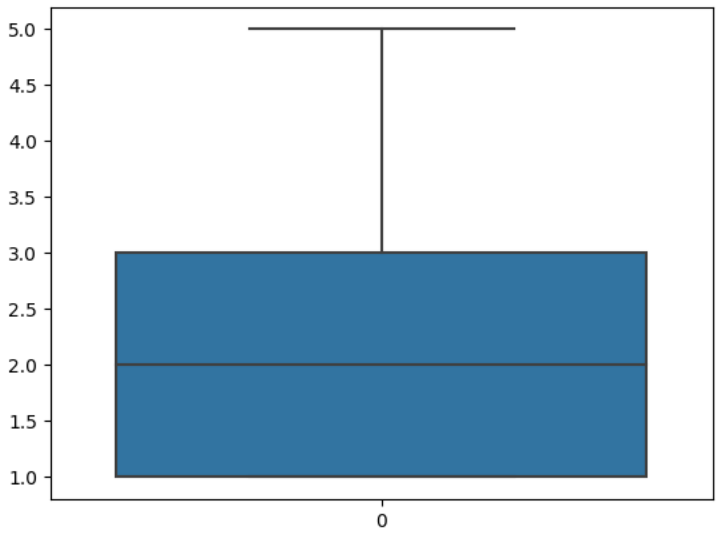



```
sns.boxplot(df.DistanceFromHome)
```

```
<Axes: >
```

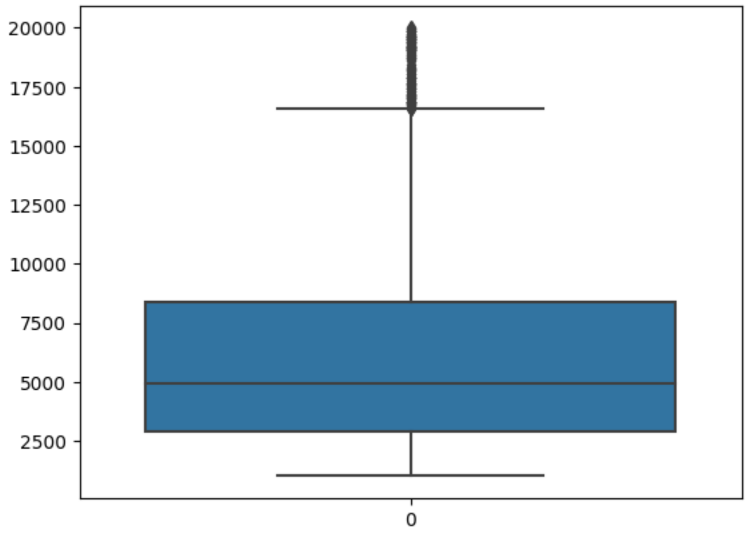
```
sns.boxplot(df.JobLevel)
```

```
<Axes: >
```



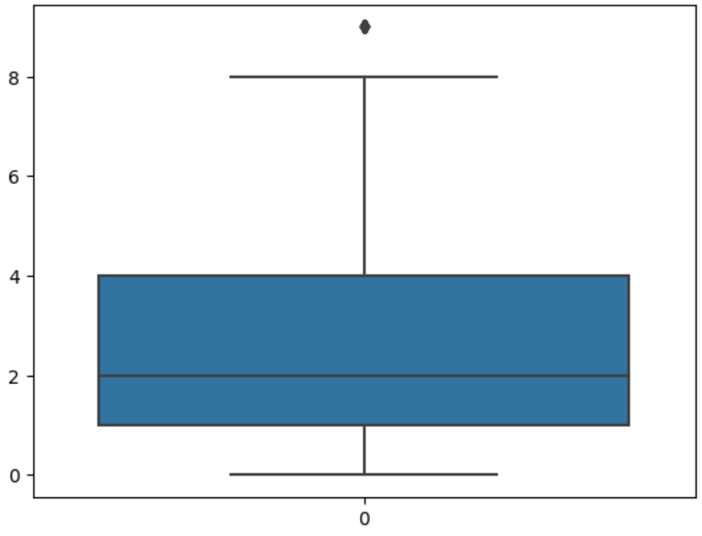
```
sns.boxplot(df.MonthlyIncome)
```

```
<Axes: >
```



```
sns.boxplot(df.NumCompaniesWorked)
```

```
<Axes: >
```



```
q1=df.NumCompaniesWorked.quantile(0.25)
q3=df.NumCompaniesWorked.quantile(0.75)
print(q1)
print(q3)
```

```
1.0
4.0
```

```
IQR=q3-q1
print(IQR)
```

```
3.0
```

```
upper_limit = q3+1.5*IQR
upper_limit
```

```
8.5
```

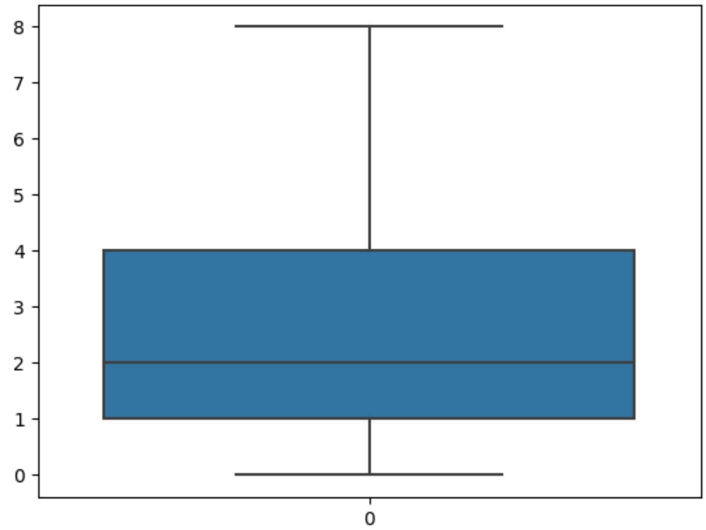
```
df.NumCompaniesWorked.median()
```

```
2.0
```

```
df.NumCompaniesWorked=np.where(df.NumCompaniesWorked>upper_limit,2,df.NumCompaniesWorked)
```

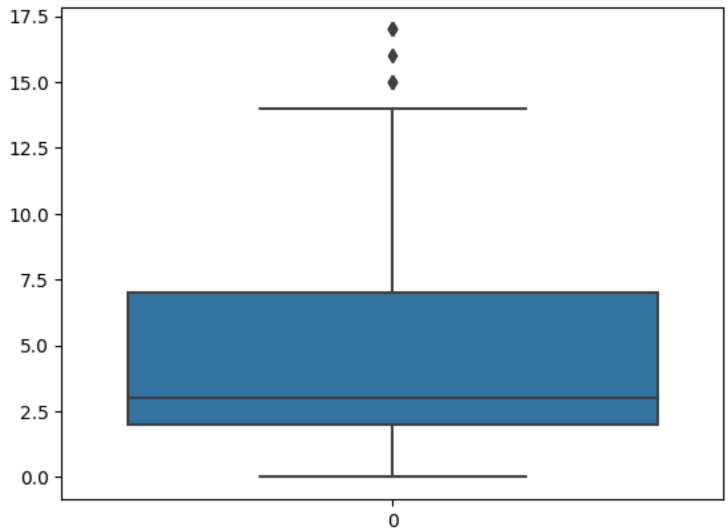
```
sns.boxplot(df.NumCompaniesWorked)
```

```
<Axes: >
```




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

```
<Axes: >
```




```
q1=df.YearsWithCurrManager.quantile(0.25)
q3=df.YearsWithCurrManager.quantile(0.75)
```

```
print(q1)
print(q3)
```

```
2.0
7.0
```

```
IQR=q3-q1
IQR
```

```
5.0
```

```
upper_limit=q3+1.5*IQR
upper_limit
```

```
14.5
```

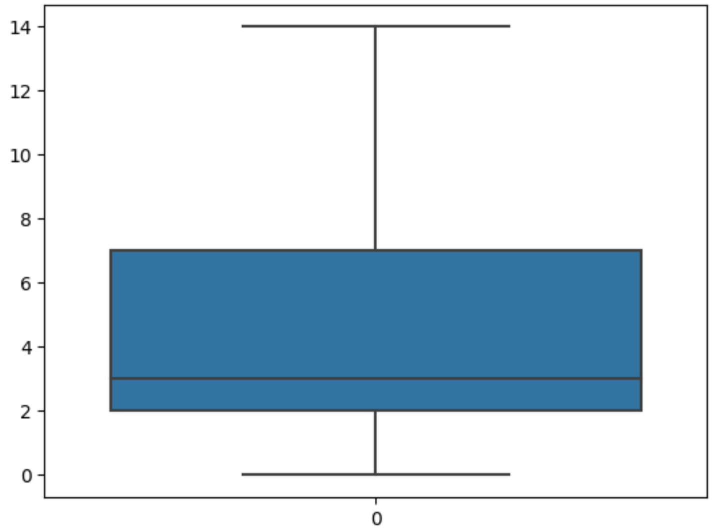
```
df.YearsWithCurrManager.median()
```

```
3.0
```

```
df.YearsWithCurrManager=np.where(df.YearsWithCurrManager>upper_limit,3,df.YearsWithCurrManager)
```

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

```
<Axes: >
```




```
df.PerformanceRating.unique()
```

```
array([3, 4])
```

The columns "YearsAtCompany," "PerformanceRating," and "MonthlyIncome" are critical in the dataset, and there is no requirement to address outliers in these columns.

## 6. Splitting Dependent and Independent variables

```
x=df.drop(["Attrition","Over18","EmployeeCount","StandardHours","EmployeeNumber"],axis=1)
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 30 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Age                      1470 non-null   int64
 1   BusinessTravel           1470 non-null   object
 2   DailyRate                1470 non-null   int64
 3   Department               1470 non-null   object
 4   DistanceFromHome         1470 non-null   int64
 5   Education                1470 non-null   int64
 6   EducationField           1470 non-null   object
 7   EnvironmentSatisfaction  1470 non-null   int64
 8   Gender                   1470 non-null   object
 9   HourlyRate               1470 non-null   int64
 10  JobInvolvement           1470 non-null   int64
```

```
 11  JobLevel                  1470 non-null   int64
 12  JobRole                   1470 non-null   object
 13  JobSatisfaction           1470 non-null   int64
 14  MaritalStatus             1470 non-null   object
 15  MonthlyIncome             1470 non-null   int64
 16  MonthlyRate               1470 non-null   int64
 17  NumCompaniesWorked        1470 non-null   int64
 18  OverTime                  1470 non-null   object
 19  PercentSalaryHike         1470 non-null   int64
 20  PerformanceRating         1470 non-null   int64
 21  RelationshipSatisfaction  1470 non-null   int64
 22  StockOptionLevel          1470 non-null   int64
 23  TotalWorkingYears         1470 non-null   int64
 24  TrainingTimesLastYear     1470 non-null   int64
 25  WorkLifeBalance           1470 non-null   int64
 26  YearsAtCompany            1470 non-null   int64
 27  YearsInCurrentRole        1470 non-null   int64
 28  YearsSinceLastPromotion   1470 non-null   int64
 29  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(23), object(7)
memory usage: 344.7+ KB
```

```
y=df.Attrition
print(y)
type(y)
```

```
0       1
1       0
2       1
3       0
4       0
       ..
1465    0
1466    0
1467    0
1468    0
1469    0
Name: Attrition, Length: 1470, dtype: int64
pandas.core.series.Series
```

## 7.Encoding

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OrdinalEncoder
```

```
le=LabelEncoder()
oe=OrdinalEncoder()
```

```
y=le.fit_transform(y)
y
```

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

```
x["Gender"]=le.fit_transform(x["Gender"])
x["Gender"]
```

```
0       0
1       1
2       1
3       0
4       1
       ..
1465    1
1466    1
1467    1
1468    1
1469    1
Name: Gender, Length: 1470, dtype: int64
```

```
x["OverTime"]=le.fit_transform(x["OverTime"])
x["OverTime"]
```

```
0       1
1       0
2       1
3       1
4       0
       ..
1465    0
1466    0
1467    1
```

```
1468    0
1469    0
Name: OverTime, Length: 1470, dtype: int64
```

```
custom_order = ["Non-Travel","Travel_Rarely","Travel_Frequently"]

# Create an instance of OrdinalEncoder with custom categories
encoder = OrdinalEncoder(categories=[custom_order])

# Fit and transform the "BusinessTravel" column
x["BusinessTravel"] = encoder.fit_transform(x[["BusinessTravel"]])
```

```
x.BusinessTravel
```

```
0       1.0
1       2.0
2       1.0
3       2.0
4       1.0
       ...
1465    2.0
1466    1.0
1467    1.0
1468    2.0
1469    1.0
Name: BusinessTravel, Length: 1470, dtype: float64
```

```
Department = pd.get_dummies(x["Department"], drop_first=True).astype(int)
x=pd.concat([x,Department],axis=1)
x.drop(["Department"],axis=1,inplace=True)
```

```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 31 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   BusinessTravel            1470 non-null   float64
 2   DailyRate                 1470 non-null   int64
 3   DistanceFromHome          1470 non-null   int64
 4   Education                 1470 non-null   int64
 5   EducationField            1470 non-null   object
 6   EnvironmentSatisfaction   1470 non-null   int64
 7   Gender                    1470 non-null   int64
 8   HourlyRate                1470 non-null   int64
 9   JobInvolvement            1470 non-null   int64
 10  JobLevel                  1470 non-null   int64
 11  JobRole                   1470 non-null   object
 12  JobSatisfaction           1470 non-null   int64
 13  MaritalStatus             1470 non-null   object
 14  MonthlyIncome             1470 non-null   int64
 15  MonthlyRate               1470 non-null   int64
 16  NumCompaniesWorked        1470 non-null   int64
 17  OverTime                  1470 non-null   int64
 18  PercentSalaryHike         1470 non-null   int64
 19  PerformanceRating         1470 non-null   int64
 20  RelationshipSatisfaction  1470 non-null   int64
 21  StockOptionLevel          1470 non-null   int64
 22  TotalWorkingYears         1470 non-null   int64
 23  TrainingTimesLastYear     1470 non-null   int64
 24  WorkLifeBalance           1470 non-null   int64
 25  YearsAtCompany            1470 non-null   int64
 26  YearsInCurrentRole        1470 non-null   int64
 27  YearsSinceLastPromotion   1470 non-null   int64
 28  YearsWithCurrManager      1470 non-null   int64
 29  Research & Development    1470 non-null   int64
 30  Sales                     1470 non-null   int64
dtypes: float64(1), int64(27), object(3)
memory usage: 356.1+ KB
```

```
MaritalStatus = pd.get_dummies(x["MaritalStatus"], drop_first=True).astype(int)
x=pd.concat([x,MaritalStatus],axis=1)
x.drop(["MaritalStatus"],axis=1,inplace=True)
x
```

| | Age | BusinessTravel | DailyRate | DistanceFromHome | Education | EducationField | EnvironmentSatisfaction | Gender | HourlyRate | JobI |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 41 | 1.0 | 1102 | 1 | 2 | Life Sciences | 2 | 0 | 94 | |
| **1** | 49 | 2.0 | 279 | 8 | 1 | Life Sciences | 3 | 1 | 61 | |
| **2** | 37 | 1.0 | 1373 | 2 | 2 | Other | 4 | 1 | 92 | |
| **3** | 33 | 2.0 | 1392 | 3 | 4 | Life Sciences | 4 | 0 | 56 | |
| **4** | 27 | 1.0 | 591 | 2 | 1 | Medical | 1 | 1 | 40 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **1465** | 36 | 2.0 | 884 | 23 | 2 | Medical | 3 | 1 | 41 | |
| **1466** | 39 | 1.0 | 613 | 6 | 1 | Medical | 4 | 1 | 42 | |
| **1467** | 27 | 1.0 | 155 | 4 | 3 | Life Sciences | 2 | 1 | 87 | |

```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 32 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Age                       1470 non-null   int64  
 1   BusinessTravel            1470 non-null   float64
 2   DailyRate                 1470 non-null   int64  
 3   DistanceFromHome          1470 non-null   int64  
 4   Education                 1470 non-null   int64  
 5   EducationField            1470 non-null   object 
 6   EnvironmentSatisfaction   1470 non-null   int64  
 7   Gender                    1470 non-null   int64  
 8   HourlyRate                1470 non-null   int64  
 9   JobInvolvement            1470 non-null   int64  
 10  JobLevel                  1470 non-null   int64  
 11  JobRole                   1470 non-null   object 
 12  JobSatisfaction           1470 non-null   int64  
 13  MonthlyIncome             1470 non-null   int64  
 14  MonthlyRate               1470 non-null   int64  
 15  NumCompaniesWorked        1470 non-null   int64  
 16  OverTime                  1470 non-null   int64  
 17  PercentSalaryHike         1470 non-null   int64  
 18  PerformanceRating         1470 non-null   int64  
 19  RelationshipSatisfaction  1470 non-null   int64  
 20  StockOptionLevel          1470 non-null   int64  
 21  TotalWorkingYears         1470 non-null   int64  
 22  TrainingTimesLastYear     1470 non-null   int64  
 23  WorkLifeBalance           1470 non-null   int64  
 24  YearsAtCompany            1470 non-null   int64  
 25  YearsInCurrentRole        1470 non-null   int64  
 26  YearsSinceLastPromotion   1470 non-null   int64  
 27  YearsWithCurrManager      1470 non-null   int64  
 28  Research & Development    1470 non-null   int64  
 29  Sales                     1470 non-null   int64  
 30  Married                   1470 non-null   int64  
 31  Single                    1470 non-null   int64  
dtypes: float64(1), int64(29), object(2)
memory usage: 367.6+ KB
```

```
JobRole = pd.get_dummies(x["JobRole"], drop_first=True).astype(int)
x=pd.concat([x,JobRole],axis=1)
x.drop(["JobRole"],axis=1,inplace=True)
x
```

| | Age | BusinessTravel | DailyRate | DistanceFromHome | Education | EducationField | EnvironmentSatisfaction | Gender | HourlyRate | JobI |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 41 | 1.0 | 1102 | 1 | 2 | Life Sciences | 2 | 0 | 94 | |

```
EducationField = pd.get_dummies(x["EducationField"], drop_first=True).astype(int)
x=pd.concat([x,EducationField],axis=1)
x.drop(["EducationField"],axis=1,inplace=True)
x
```

| | Age | BusinessTravel | DailyRate | DistanceFromHome | Education | EnvironmentSatisfaction | Gender | HourlyRate | JobInvolvement | JobI |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 41 | 1.0 | 1102 | 1 | 2 | 2 | 0 | 94 | 3 | |
| **1** | 49 | 2.0 | 279 | 8 | 1 | 3 | 1 | 61 | 2 | |
| **2** | 37 | 1.0 | 1373 | 2 | 2 | 4 | 1 | 92 | 2 | |
| **3** | 33 | 2.0 | 1392 | 3 | 4 | 4 | 0 | 56 | 3 | |
| **4** | 27 | 1.0 | 591 | 2 | 1 | 1 | 1 | 40 | 3 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **1465** | 36 | 2.0 | 884 | 23 | 2 | 3 | 1 | 41 | 4 | |
| **1466** | 39 | 1.0 | 613 | 6 | 1 | 4 | 1 | 42 | 2 | |
| **1467** | 27 | 1.0 | 155 | 4 | 3 | 2 | 1 | 87 | 4 | |
| **1468** | 49 | 2.0 | 1023 | 2 | 3 | 4 | 1 | 63 | 2 | |
| **1469** | 34 | 1.0 | 628 | 8 | 3 | 2 | 1 | 82 | 4 | |

1470 rows × 43 columns

```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 43 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Age                       1470 non-null   int64  
 1   BusinessTravel            1470 non-null   float64
 2   DailyRate                 1470 non-null   int64  
 3   DistanceFromHome          1470 non-null   int64  
 4   Education                 1470 non-null   int64  
 5   EnvironmentSatisfaction   1470 non-null   int64  
 6   Gender                    1470 non-null   int64  
 7   HourlyRate                1470 non-null   int64  
 8   JobInvolvement            1470 non-null   int64  
 9   JobLevel                  1470 non-null   int64  
 10  JobSatisfaction           1470 non-null   int64  
 11  MonthlyIncome             1470 non-null   int64  
 12  MonthlyRate               1470 non-null   int64  
 13  NumCompaniesWorked        1470 non-null   int64  
 14  OverTime                  1470 non-null   int64  
 15  PercentSalaryHike         1470 non-null   int64  
 16  PerformanceRating         1470 non-null   int64  
 17  RelationshipSatisfaction  1470 non-null   int64  
 18  StockOptionLevel          1470 non-null   int64  
 19  TotalWorkingYears         1470 non-null   int64  
 20  TrainingTimesLastYear     1470 non-null   int64  
 21  WorkLifeBalance           1470 non-null   int64  
 22  YearsAtCompany            1470 non-null   int64  
 23  YearsInCurrentRole        1470 non-null   int64  
 24  YearsSinceLastPromotion   1470 non-null   int64  
 25  YearsWithCurrManager      1470 non-null   int64  
 26  Research & Development    1470 non-null   int64  
 27  Sales                     1470 non-null   int64  
 28  Married                   1470 non-null   int64  
 29  Single                    1470 non-null   int64  
 30  Human Resources           1470 non-null   int64  
 31  Laboratory Technician     1470 non-null   int64  
 32  Manager                   1470 non-null   int64  
 33  Manufacturing Director    1470 non-null   int64  
 34  Research Director         1470 non-null   int64  
 35  Research Scientist        1470 non-null   int64  
 36  Sales Executive           1470 non-null   int64  
 37  Sales Representative      1470 non-null   int64  
 38  Life Sciences             1470 non-null   int64  
 39  Marketing                 1470 non-null   int64  
 40  Medical                   1470 non-null   int64  
 41  Other                     1470 non-null   int64  
 42  Technical Degree          1470 non-null   int64  
dtypes: float64(1), int64(42)
memory usage: 494.0 KB
```

## 8. Feature Scaling

```
ms=MinMaxScaler()
```

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

```
print(x_scaled)
```

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

      EnvironmentSatisfaction  Gender  HourlyRate  JobInvolvement  JobLevel  \
0                    0.333333     0.0    0.914286        0.666667      0.25   
1                    0.666667     1.0    0.442857        0.333333      0.25   
2                    1.000000     1.0    0.885714        0.333333      0.00   
3                    1.000000     0.0    0.371429        0.666667      0.00   
4                    0.000000     1.0    0.142857        0.666667      0.00   
...                       ...     ...         ...             ...       ...   
1465                 0.666667     1.0    0.157143        1.000000      0.25   
1466                 1.000000     1.0    0.171429        0.333333      0.50   
1467                 0.333333     1.0    0.814286        1.000000      0.25   
1468                 1.000000     1.0    0.471429        0.333333      0.25   
1469                 0.333333     1.0    0.742857        1.000000      0.25   

      ...  Manufacturing Director  Research Director  Research Scientist  \
0     ...                     0.0                0.0                 0.0   
1     ...                     0.0                0.0                 1.0   
2     ...                     0.0                0.0                 0.0   
3     ...                     0.0                0.0                 1.0   
4     ...                     0.0                0.0                 0.0   
...   ...                     ...                ...                 ...   
1465  ...                     0.0                0.0                 0.0   
1466  ...                     0.0                0.0                 0.0   
1467  ...                     1.0                0.0                 0.0   
1468  ...                     0.0                0.0                 0.0   
1469  ...                     0.0                0.0                 0.0   

      Sales Executive  Sales Representative  Life Sciences  Marketing  \
0                 1.0                   0.0            1.0        0.0   
1                 0.0                   0.0            1.0        0.0   
2                 0.0                   0.0            0.0        0.0   
3                 0.0                   0.0            1.0        0.0   
4                 0.0                   0.0            0.0        0.0   
...               ...                   ...            ...        ...   
1465              0.0                   0.0            0.0        0.0   
1466              0.0                   0.0            0.0        0.0   
1467              0.0                   0.0            1.0        0.0   
1468              1.0                   0.0            0.0        0.0   
1469              0.0                   0.0            0.0        0.0   

      Medical  Other  Technical Degree  
0         0.0    0.0               0.0  
1         0.0    0.0               0.0  
2         0.0    1.0               0.0  
3         0.0    0.0               0.0  
4         1.0    0.0               0.0  
```

```
x_scaled.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 43 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Age                       1470 non-null   float64
 1   BusinessTravel            1470 non-null   float64
 2   DailyRate                 1470 non-null   float64
 3   DistanceFromHome          1470 non-null   float64
 4   Education                 1470 non-null   float64
 5   EnvironmentSatisfaction   1470 non-null   float64
 6   Gender                    1470 non-null   float64
 7   HourlyRate                1470 non-null   float64
 8   JobInvolvement            1470 non-null   float64
```

```
 9   JobLevel                  1470 non-null   float64
 10  JobSatisfaction           1470 non-null   float64
 11  MonthlyIncome             1470 non-null   float64
 12  MonthlyRate               1470 non-null   float64
 13  NumCompaniesWorked        1470 non-null   float64
 14  OverTime                  1470 non-null   float64
 15  PercentSalaryHike         1470 non-null   float64
 16  PerformanceRating         1470 non-null   float64
 17  RelationshipSatisfaction  1470 non-null   float64
 18  StockOptionLevel          1470 non-null   float64
 19  TotalWorkingYears         1470 non-null   float64
 20  TrainingTimesLastYear     1470 non-null   float64
 21  WorkLifeBalance           1470 non-null   float64
 22  YearsAtCompany            1470 non-null   float64
 23  YearsInCurrentRole        1470 non-null   float64
 24  YearsSinceLastPromotion   1470 non-null   float64
 25  YearsWithCurrManager      1470 non-null   float64
 26  Research & Development    1470 non-null   float64
 27  Sales                     1470 non-null   float64
 28  Married                   1470 non-null   float64
 29  Single                    1470 non-null   float64
 30  Human Resources           1470 non-null   float64
 31  Laboratory Technician     1470 non-null   float64
 32  Manager                   1470 non-null   float64
 33  Manufacturing Director    1470 non-null   float64
 34  Research Director         1470 non-null   float64
 35  Research Scientist        1470 non-null   float64
 36  Sales Executive           1470 non-null   float64
 37  Sales Representative      1470 non-null   float64
 38  Life Sciences             1470 non-null   float64
 39  Marketing                 1470 non-null   float64
 40  Medical                   1470 non-null   float64
 41  Other                     1470 non-null   float64
 42  Technical Degree          1470 non-null   float64
dtypes: float64(43)
memory usage: 494.0 KB
```

## 9. Splitting Data into Train and Test

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

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

```
((1029, 43), (441, 43), (1029,), (441,))
```

```
x_train_scaled,x_test_scaled,y_train_scaled,y_test_scaled=train_test_split(x_scaled,y,test_size=0.3,random_state=0)
```

```
x_train_scaled.shape,x_test_scaled.shape,y_train_scaled.shape,y_test_scaled.shape
```

```
((1029, 43), (441, 43), (1029,), (441,))
```

## Logistic Regression

## Model Building

o Import the model building Libraries

o Initializing the model

o Training and testing the model

o Evaluation of Model

o Save the Model

## Import the model building Libraries

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

```
lr_model.fit(x_train_scaled,y_train_scaled)
```

```
▾ LogisticRegression
```

```
pred=lr_model.predict(x_test_scaled)
```

```
pred
```

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

```
y_test_scaled
```

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

# Evaluation of Matrices

## Logistic Regression was performed on the dataset using the scaled values for training

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

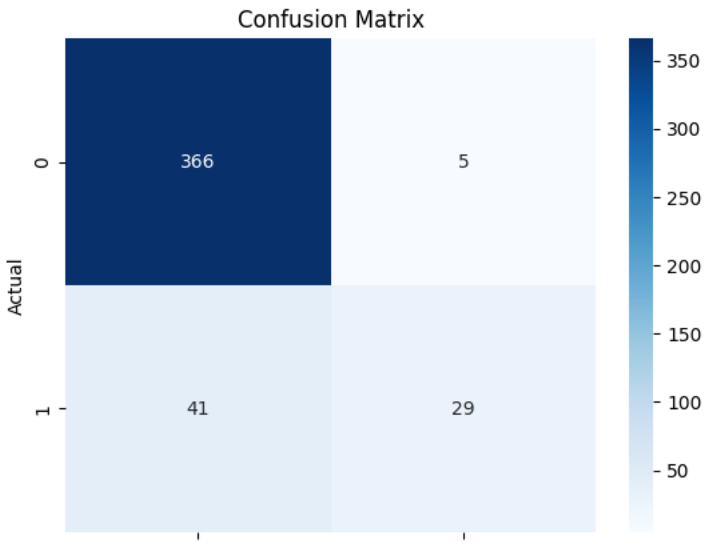
```
accuracy_score(y_test_scaled,pred)
```

```
0.8956916099773242
```

```
con_matrix=confusion_matrix(y_test_scaled,pred)
con_matrix
```

```
array([[366,   5],
       [ 41,  29]])
```

```
sns.heatmap(con_matrix, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```




```
print(classification_report(pred,y_test_scaled))
```

```
              precision    recall  f1-score   support

           0       0.99      0.90      0.94       407
           1       0.41      0.85      0.56        34

    accuracy                           0.90       441
   macro avg       0.70      0.88      0.75       441
weighted avg       0.94      0.90      0.91       441
```

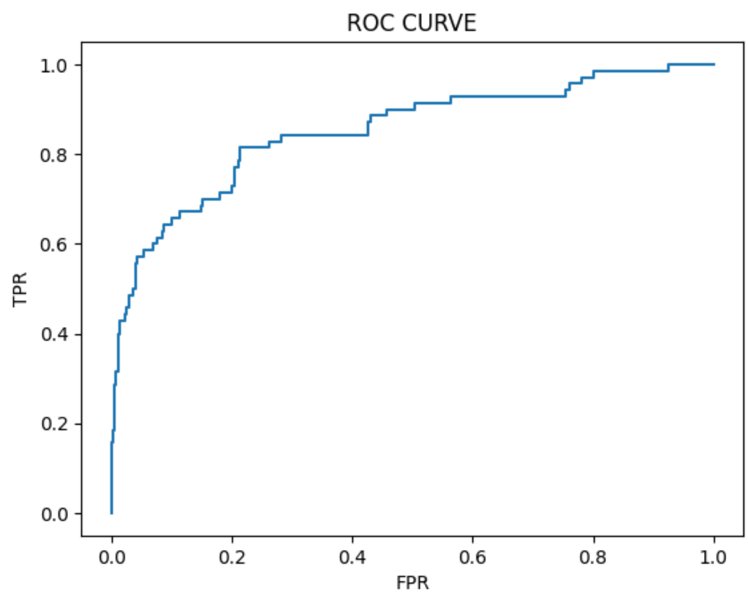
```
probability=lr_model.predict_proba(x_test_scaled)[:,1]
probability
```

```
array([0.06438748, 0.09475311, 0.5750844 , 0.11975144, 0.73738362,
       0.0548839 , 0.50798341, 0.06021682, 0.00322244, 0.18964254,
       0.03061441, 0.18362505, 0.02782101, 0.55931381, 0.14707078,
       0.02250489, 0.14499391, 0.08165404, 0.04298849, 0.17966132,
       0.15413954, 0.01017197, 0.01683991, 0.06607665, 0.6474417 ,
       0.46788466, 0.1001751 , 0.06404667, 0.6876234 , 0.06310492,
       0.01588842, 0.19814342, 0.06315573, 0.09291666, 0.03768926,
       0.01283036, 0.14999468, 0.07902595, 0.02325075, 0.11207308,
       0.12286207, 0.01953153, 0.00652322, 0.016441  , 0.02248894,
       0.62893543, 0.14189608, 0.00375846, 0.40026139, 0.31648035,
       0.14452742, 0.77944733, 0.02911634, 0.38681279, 0.58198361,
       0.3401339 , 0.02473516, 0.55340056, 0.02840217, 0.32713491,
       0.03341216, 0.16603527, 0.13336924, 0.04683476, 0.31114744,
       0.03464612, 0.4092714 , 0.18502382, 0.0992603 , 0.17627456,
       0.06393192, 0.16866074, 0.13032034, 0.04826457, 0.03286344,
       0.05181895, 0.04472508, 0.07531731, 0.40193988, 0.04310254,
       0.01059836, 0.04017779, 0.27474131, 0.03462497, 0.0410347 ,
       0.11854751, 0.00616709, 0.02350764, 0.0273443 , 0.07208579,
       0.09915454, 0.07392675, 0.31828217, 0.27826473, 0.00363969,
       0.13329399, 0.41668624, 0.40694796, 0.06499243, 0.11078655,
       0.17568886, 0.5158111 , 0.33934528, 0.01601024, 0.14251417,
       0.01198041, 0.08666495, 0.20192703, 0.10326314, 0.3005913 ,
       0.04233649, 0.06137641, 0.0077806 , 0.24938639, 0.03468254,
       0.0830945 , 0.01540979, 0.08894798, 0.0098337 , 0.01207957,
       0.10111047, 0.0500583 , 0.10135821, 0.82870502, 0.0255616 ,
       0.01125412, 0.01063293, 0.09723579, 0.11310603, 0.05551169,
       0.00957975, 0.38757445, 0.59501678, 0.20157986, 0.02887187,
       0.36316285, 0.60563574, 0.26617722, 0.19364849, 0.5745473 ,
       0.08608456, 0.08694995, 0.04861845, 0.23223325, 0.5106879 ,
       0.05955687, 0.23160877, 0.00996585, 0.08989206, 0.12760251,
       0.00710959, 0.1002393 , 0.08998554, 0.21076481, 0.03552147,
       0.03319055, 0.04884636, 0.2854546 , 0.01653081, 0.0153029 ,
       0.62131366, 0.01106981, 0.04543506, 0.8847407 , 0.03784182,
       0.25438087, 0.22522671, 0.13501784, 0.05293908, 0.00431995,
       0.10256115, 0.06880405, 0.1127886 , 0.11360801, 0.02128919,
       0.06999044, 0.07783347, 0.12296371, 0.05536815, 0.1375137 ,
       0.03602922, 0.21538814, 0.00567956, 0.80994564, 0.06002706,
       0.07897158, 0.46104975, 0.03037531, 0.48784617, 0.18900551,
       0.24556017, 0.32783392, 0.17782556, 0.04074752, 0.02780643,
       0.19273316, 0.03529899, 0.01617297, 0.2988878 , 0.03318387,
       0.31476841, 0.24682015, 0.70068974, 0.03702113, 0.1233906 ,
       0.02800771, 0.36636477, 0.00273029, 0.17617169, 0.01485424,
       0.09819079, 0.29363635, 0.06894514, 0.35679643, 0.09245615,
       0.01141046, 0.02307933, 0.07847448, 0.03038936, 0.1226029 ,
       0.11916471, 0.39430382, 0.48749883, 0.06112682, 0.30119439,
       0.00601101, 0.15499734, 0.34947204, 0.70891905, 0.09423065,
       0.04329395, 0.2143601 , 0.03928591, 0.05618273, 0.1289885 ,
       0.24970155, 0.2656108 , 0.00448779, 0.11191898, 0.00955181,
       0.18792737, 0.21909263, 0.00900874, 0.15377319, 0.07005297,
```

```
0.0186655 , 0.1166317 , 0.35331238, 0.07545233, 0.06797411,
0.22371492, 0.1553254 , 0.46234846, 0.03227259, 0.14524887,
0.10667592, 0.00551394, 0.54446252, 0.29509606, 0.32376476,
0.23062683, 0.05075229, 0.32717524, 0.07253238, 0.04896312,
0.12431835, 0.00409304, 0.45945493, 0.58505198, 0.04577969,
0.05256799, 0.02089617, 0.08328475, 0.05460047, 0.02662698,
0.01972966, 0.10506195, 0.44793447, 0.06194672, 0.21578698,
0.57439386, 0.00842774, 0.06177984, 0.12230851, 0.01852684,
```

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

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




## A Logistic Regression model was built on the dataset without applying scaling

```
lr_model_without_scaled=LogisticRegression()
```

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

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
▾ LogisticRegression
LogisticRegression()
```

```
pred_without_Scaled=lr_model.predict(x_test)
```

```
accuracy_score(y_test,pred_without_Scaled)
```

```
0.4580498866213152
```

```
print(classification_report(pred_without_Scaled,y_test))
```

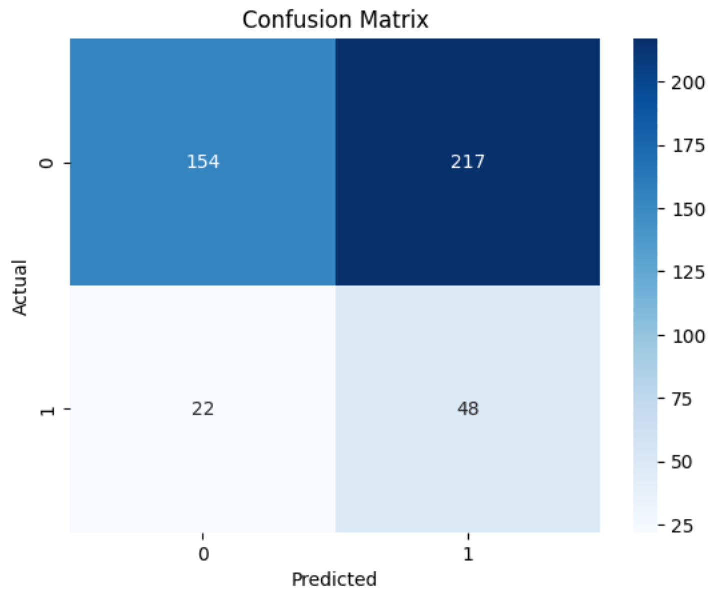
```
              precision    recall  f1-score   support

           0       0.42      0.88      0.56       176
           1       0.69      0.18      0.29       265

    accuracy                           0.46       441
   macro avg       0.55      0.53      0.42       441
```

```
weighted avg       0.58      0.46      0.40       441
```

```
con_matrix1=confusion_matrix(y_test_scaled,pred_without_Scaled)
con_matrix1
```

```
array([[154, 217],
       [ 22,  48]])
```

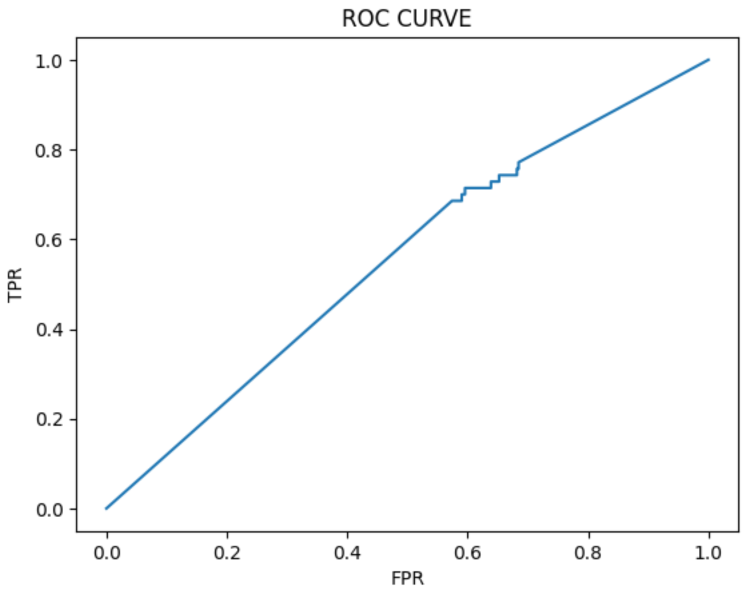
```
sns.heatmap(con_matrix1, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```




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

```
       1.00000000e+000, 2.82846343e-170, 1.00000000e+000, 0.00000000e+000,
       0.00000000e+000, 0.00000000e+000, 1.00000000e+000, 1.00000000e+000,
       0.00000000e+000, 1.00000000e+000, 0.00000000e+000, 0.00000000e+000,
       0.00000000e+000, 1.00000000e+000, 1.00000000e+000, 8.36394553e-051,
       1.00000000e+000, 1.00000000e+000, 1.00000000e+000, 1.00000000e+000,
       1.00000000e+000, 1.00000000e+000, 1.00000000e+000, 0.00000000e+000,
       1.00000000e+000, 1.00000000e+000, 0.00000000e+000, 1.00000000e+000,
       1.00000000e+000, 0.00000000e+000, 1.00000000e+000, 1.00000000e+000,
       2.92055918e-137, 0.00000000e+000, 3.49067311e-038, 0.00000000e+000,
       0.00000000e+000, 1.00000000e+000, 1.67027025e-092, 1.00000000e+000,
       0.00000000e+000, 1.00000000e+000, 6.88884358e-153, 1.00000000e+000,
       1.00000000e+000, 1.00000000e+000, 0.00000000e+000, 0.00000000e+000,
       1.00000000e+000, 0.00000000e+000, 0.00000000e+000, 1.00000000e+000,
       1.00000000e+000, 1.00000000e+000, 1.00000000e+000, 1.28349072e-279,
       1.00000000e+000, 0.00000000e+000, 1.00000000e+000, 1.00000000e+000,
       1.00000000e+000, 1.00000000e+000, 0.00000000e+000, 1.00000000e+000,
       1.00000000e+000])
```

```
# roc_curve
fpr,tpr,threshsholds = roc_curve(y_test_scaled,probability)
plt.plot(fpr,tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC CURVE')
plt.show()
```




**Scaling the dataset is crucial for improving the performance of the Logistic Regression model, as it significantly enhances its predictive accuracy and overall effectiveness.**

## Hyperparameter tuning on Logistic Regression

```
from sklearn.model_selection import GridSearchCV
```

```
param_grid = {
    'penalty': ['l2'],
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'solver': ['liblinear', 'newton-cg', 'lbfgs', 'sag'],
    'max_iter': [100, 200, 300]
}
```

```
grid_search=GridSearchCV(lr_model,param_grid=param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(x_train_scaled, y_train_scaled)
```

```
▸ GridSearchCV
▸ estimator: LogisticRegression
  ▸ LogisticRegression
```

```
g_pred=grid_search.predict(x_test_scaled)
g_pred
```

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

```
grid_search.best_params_
```

```
{'C': 1, 'max_iter': 100, 'penalty': 'l2', 'solver': 'liblinear'}
```

```
accuracy_score(y_test_scaled,g_pred)
```

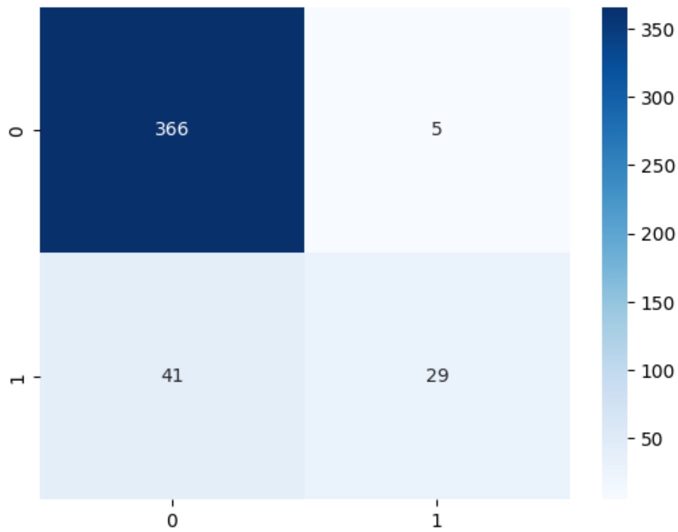
```
0.8956916099773242
```

```
print(classification_report(y_test_scaled,g_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.99      0.94       371
           1       0.85      0.41      0.56        70

    accuracy                           0.90       441
   macro avg       0.88      0.70      0.75       441
weighted avg       0.89      0.90      0.88       441
```

```
sns.heatmap(confusion_matrix(y_test_scaled,g_pred),annot=True,fmt='d',cmap="Blues")
```

```
&lt;Axes: &gt;
```




## Decision Tree

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

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

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
dtc_pred=dtc.predict(x_test)
```

```
dtc_pred
```

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

## Evalution Matrices

```
accuracy_score(y_test,dtc_pred)
```

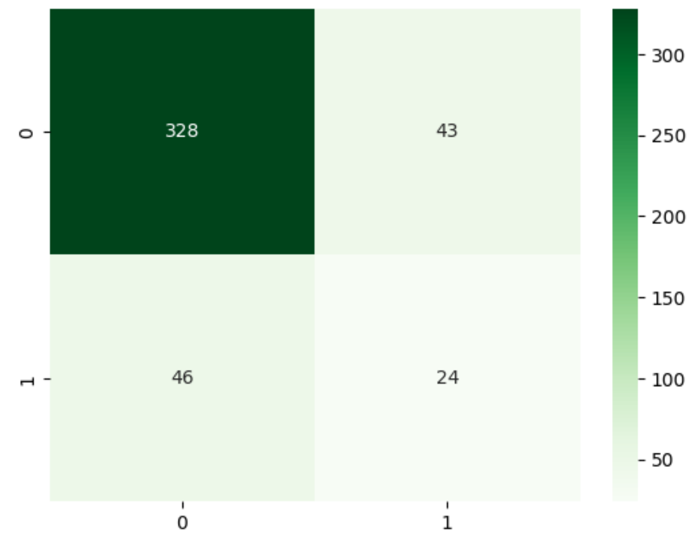
```
0.7981859410430839
```

```
print(classification_report(y_test,dtc_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.88      0.88       371
           1       0.36      0.34      0.35        70

    accuracy                           0.80       441
   macro avg       0.62      0.61      0.62       441
weighted avg       0.79      0.80      0.80       441
```

```
sns.heatmap(confusion_matrix(y_test,dtc_pred),annot=True,fmt='d',cmap='Greens')
```

<Axes: >

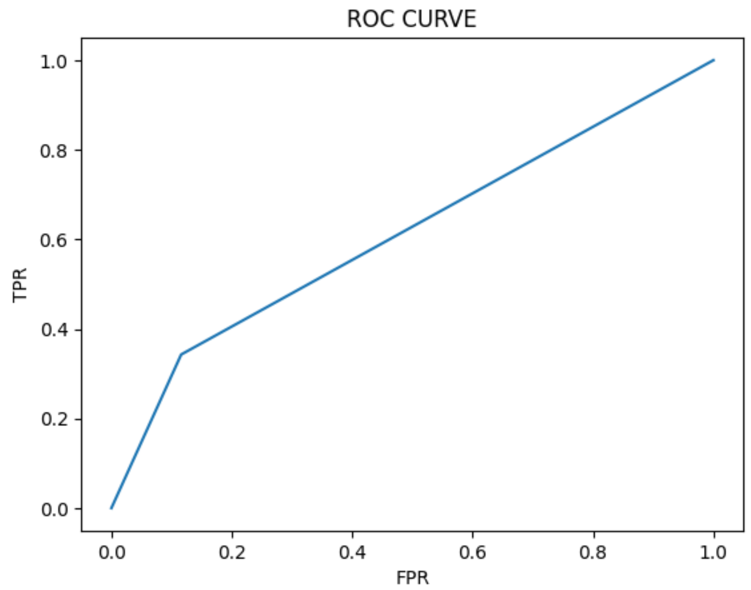



```
probability=dtc.predict_proba(x_test)[:,1]
probability
```

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

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

```
# 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()
```




## Hyperparameter tuning on Decision Tree

```
param_grid_dtc={
 'criterion':['gini','entropy'],
  'splitter':['best','random'],
  'max_depth':[1,2,3,4,5],
  'max_features':['auto', 'sqrt', 'log2']

}
```

```
grid_search_dtc=GridSearchCV(dtc,param_grid=param_grid_dtc,cv=5,scoring="accuracy", n_jobs=-1)
```

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

```
▸          GridSearchCV
▸ estimator: DecisionTreeClassifier
      ▸ DecisionTreeClassifier
```

```
grid_search_dtc.best_params_
```

```
{'criterion': 'entropy',
 'max_depth': 3,
```

```
 'max_features': 'sqrt',
 'splitter': 'best'}
```

```
dtc_grid_pred=grid_search_dtc.predict(x_test)
```

```
dtc_grid_pred
```

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

## Evalution Matrix for **Grid Search**

```
accuracy_score(y_test,dtc_grid_pred)
```

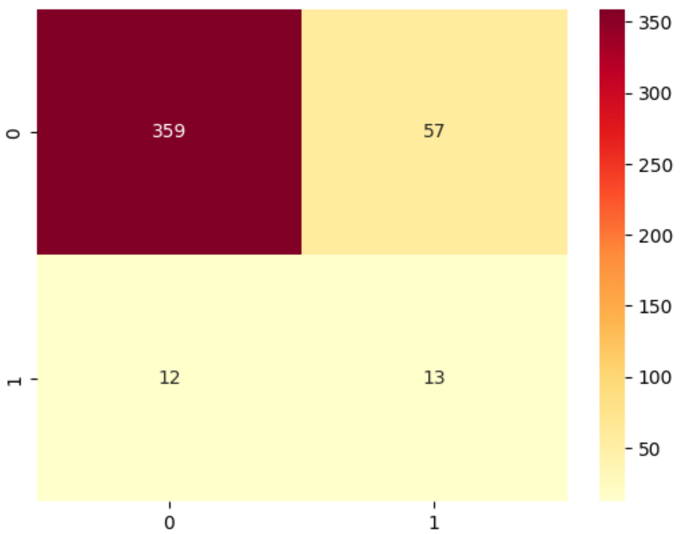
```
0.8435374149659864
```

```
print(classification_report(dtc_grid_pred,y_test))
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.97 | 0.86 | 0.91 | 416 |
| 1 | 0.19 | 0.52 | 0.27 | 25 |
| accuracy | | | 0.84 | 441 |
| macro avg | 0.58 | 0.69 | 0.59 | 441 |
| weighted avg | 0.92 | 0.84 | 0.88 | 441 |

```
sns.heatmap(confusion_matrix(dtc_grid_pred,y_test),annot=True, fmt='d', cmap='YlOrRd')
```

```
<Axes: >
```

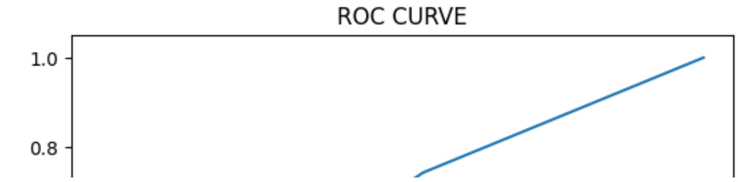



## ROC curve

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

```
array([0.15517241, 0.15517241, 0.05783133, 0.05783133, 0.53658537,
       0.38461538, 0.15517241, 0.15517241, 0.05783133, 0.19172932,
       0.05783133, 0.19172932, 0.05783133, 0.19172932, 0.05783133,
       0.19172932, 0.15517241, 0.05783133, 0.05783133, 0.19172932,
       0.38461538, 0.05783133, 0.05783133, 0.05783133, 0.05783133,
       0.05783133, 0.19172932, 0.05783133, 0.19172932, 0.05783133,
       0.15517241, 0.15517241, 0.05783133, 0.05783133, 0.05783133,
       0.05783133, 0.05783133, 0.19172932, 0.15517241, 0.17241379,
       0.15517241, 0.05783133, 0.05783133, 0.05783133, 0.19172932,
       0.19172932, 0.19172932, 0.05783133, 0.53658537, 0.19172932,
       0.05783133, 0.19172932, 0.05783133, 0.05783133, 0.53658537,
       0.15517241, 0.05783133, 0.05783133, 0.19172932, 0.17241379,
       0.19172932, 0.05783133, 0.15517241, 0.05783133, 1.        ,
       0.05783133, 0.19172932, 0.15517241, 0.19172932, 0.19172932,
       0.15517241, 0.19172932, 0.05783133, 0.05783133, 0.05783133,
       0.19172932, 0.05783133, 0.15517241, 0.17241379, 0.05783133,
       0.05783133, 0.05783133, 0.19172932, 0.19172932, 0.19172932,
       0.05783133, 0.15517241, 0.19172932, 0.05783133, 0.05783133,
       0.38461538, 0.05783133, 0.05783133, 0.19172932, 0.05783133,
       0.15517241, 0.15517241, 0.05783133, 0.19172932, 0.19172932,
       0.05783133, 0.53658537, 0.15517241, 0.05783133, 0.19172932,
       0.05783133, 0.15517241, 0.38461538, 0.19172932, 0.19172932,
       0.15517241, 0.15517241, 0.05783133, 0.05783133, 0.19172932,
       0.19172932, 0.15517241, 0.53658537, 0.05783133, 0.05783133,
       0.19172932, 0.19172932, 0.05783133, 0.19172932, 0.17241379,
       0.05783133, 0.05783133, 0.05783133, 0.17241379, 0.05783133,
       0.05783133, 0.5       , 0.53658537, 0.15517241, 0.15517241,
       0.15517241, 0.15517241, 0.19172932, 0.05783133, 0.05783133,
       0.53658537, 0.19172932, 0.05783133, 0.05783133, 0.05783133,
       0.05783133, 0.05783133, 0.05783133, 0.05783133, 0.05783133,
       0.05783133, 0.05783133, 0.19172932, 0.15517241, 0.15517241,
       0.05783133, 0.05783133, 0.19172932, 0.19172932, 0.05783133,
       0.15517241, 0.05783133, 0.5       , 0.53658537, 0.05783133,
       0.15517241, 0.05783133, 0.19172932, 0.05783133, 0.05783133,
       0.05783133, 0.15517241, 0.19172932, 0.38461538, 0.05783133,
       0.17241379, 0.05783133, 0.15517241, 0.19172932, 0.53658537,
       0.05783133, 0.05783133, 0.05783133, 0.15517241, 0.05783133,
       0.19172932, 0.15517241, 0.05783133, 0.53658537, 0.19172932,
       0.5       , 1.        , 0.05783133, 0.19172932, 0.19172932,
       0.05783133, 0.15517241, 0.05783133, 0.53658537, 0.19172932,
       0.5       , 0.19172932, 0.19172932, 0.05783133, 0.05783133,
       0.05783133, 0.53658537, 0.05783133, 0.05783133, 0.05783133,
       0.19172932, 0.19172932, 0.05783133, 0.19172932, 0.05783133,
       0.17241379, 0.05783133, 0.19172932, 0.17241379, 0.38461538,
       0.19172932, 0.38461538, 0.19172932, 0.05783133, 0.05783133,
       0.05783133, 0.17241379, 0.19172932, 0.38461538, 0.19172932,
       0.05783133, 0.19172932, 0.05783133, 0.05783133, 0.05783133,
       1.        , 0.19172932, 0.05783133, 0.19172932, 0.15517241,
       0.19172932, 0.05783133, 0.05783133, 0.05783133, 0.05783133,
       0.15517241, 0.05783133, 0.19172932, 0.05783133, 0.15517241,
       0.17241379, 0.15517241, 0.15517241, 0.05783133, 0.19172932,
       0.05783133, 0.05783133, 1.        , 0.19172932, 0.19172932,
       0.15517241, 0.05783133, 0.05783133, 0.05783133, 0.15517241,
       0.38461538, 0.05783133, 0.19172932, 0.05783133, 0.15517241,
       0.15517241, 0.05783133, 0.05783133, 0.05783133, 0.19172932,
       0.05783133, 0.05783133, 0.19172932, 0.05783133, 0.19172932,
       0.19172932, 0.19172932, 0.19172932, 0.15517241, 0.15517241,
       0.19172932, 0.05783133, 0.05783133, 0.05783133, 0.05783133,
```

```
# 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()
```




## RANDOM fOREST

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

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

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
rfc_pred=rfc.predict(x_test)
```

## Evalution Matrices

```
accuracy_score(y_test,rfc_pred)
```

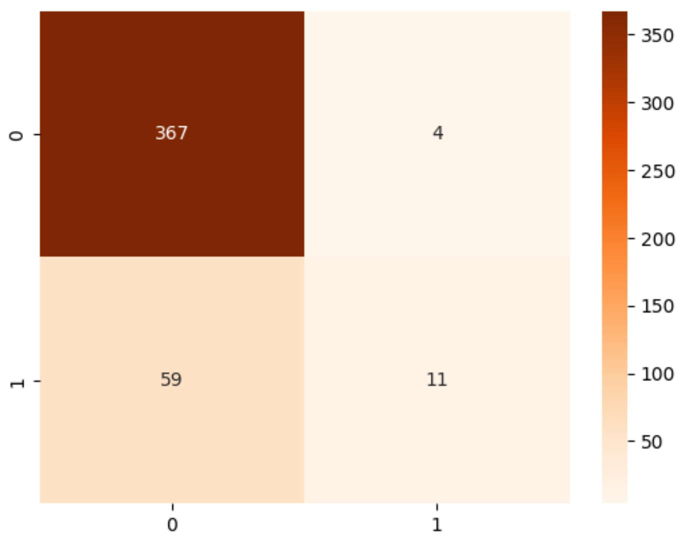
```
0.8571428571428571
```

```
print(classification_report(y_test,rfc_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.99      0.92       371
           1       0.73      0.16      0.26        70

    accuracy                           0.86       441
   macro avg       0.80      0.57      0.59       441
weighted avg       0.84      0.86      0.82       441
```

```
sns.heatmap(confusion_matrix(y_test,rfc_pred),annot=True,fmt='d',cmap='Oranges')
```

```
<Axes: >
```

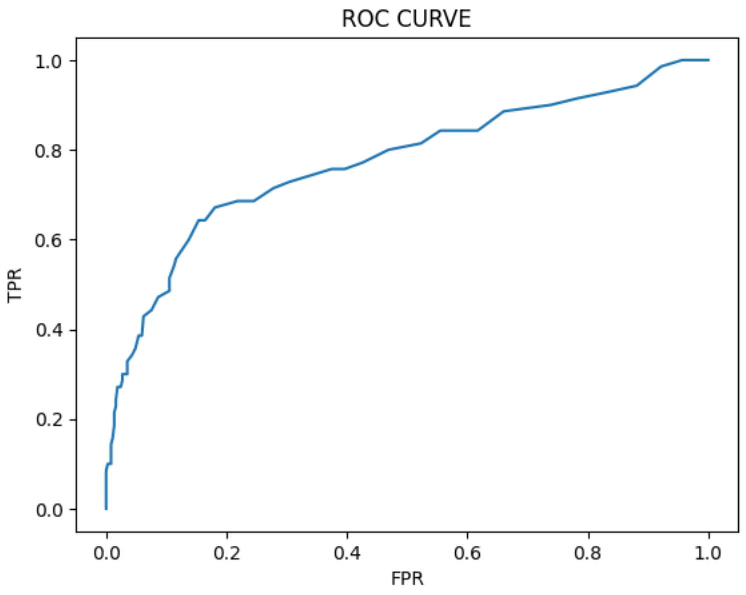



```
probability=rfc.predict_proba(x_test)[:,1]
probability
```

```
array([0.13, 0.08, 0.18, 0.19, 0.72, 0.38, 0.37, 0.14, 0.08, 0.13, 0.07,
       0.17, 0.12, 0.49, 0.07, 0.  , 0.13, 0.09, 0.07, 0.15, 0.55, 0.15,
       0.04, 0.03, 0.36, 0.21, 0.05, 0.06, 0.55, 0.1 , 0.09, 0.13, 0.2 ,
       0.08, 0.08, 0.04, 0.14, 0.13, 0.11, 0.18, 0.16, 0.03, 0.03, 0.17,
       0.13, 0.32, 0.08, 0.06, 0.54, 0.36, 0.25, 0.47, 0.21, 0.1 , 0.48,
       0.1 , 0.07, 0.05, 0.06, 0.33, 0.11, 0.18, 0.09, 0.2 , 0.46, 0.11,
       0.19, 0.13, 0.1 , 0.12, 0.07, 0.31, 0.09, 0.02, 0.04, 0.15, 0.12,
```

```
       0.04, 0.53, 0.01, 0.05, 0.05, 0.25, 0.19, 0.12, 0.06, 0.05, 0.26,
       0.05, 0.1 , 0.49, 0.04, 0.13, 0.2 , 0.06, 0.12, 0.1 , 0.42, 0.12,
       0.09, 0.28, 0.27, 0.33, 0.12, 0.14, 0.08, 0.09, 0.57, 0.19, 0.24,
       0.31, 0.17, 0.16, 0.02, 0.12, 0.16, 0.13, 0.24, 0.08, 0.02, 0.08,
       0.03, 0.04, 0.53, 0.24, 0.19, 0.04, 0.09, 0.12, 0.08, 0.04, 0.3 ,
       0.4 , 0.3 , 0.18, 0.21, 0.31, 0.18, 0.16, 0.22, 0.07, 0.16, 0.08,
       0.1 , 0.29, 0.09, 0.06, 0.08, 0.04, 0.08, 0.09, 0.2 , 0.12, 0.32,
       0.13, 0.05, 0.06, 0.32, 0.17, 0.12, 0.25, 0.04, 0.25, 0.49, 0.07,
       0.2 , 0.17, 0.05, 0.06, 0.02, 0.11, 0.14, 0.17, 0.29, 0.09, 0.21,
       0.07, 0.27, 0.19, 0.22, 0.01, 0.07, 0.05, 0.44, 0.06, 0.02, 0.29,
       0.08, 0.22, 0.2 , 0.24, 0.52, 0.09, 0.05, 0.17, 0.14, 0.08, 0.11,
       0.43, 0.12, 0.27, 0.25, 0.22, 0.06, 0.09, 0.15, 0.41, 0.05, 0.11,
       0.01, 0.08, 0.23, 0.07, 0.35, 0.04, 0.1 , 0.05, 0.18, 0.18, 0.39,
       0.15, 0.35, 0.43, 0.17, 0.21, 0.1 , 0.1 , 0.25, 0.64, 0.21, 0.01,
       0.16, 0.09, 0.11, 0.08, 0.29, 0.21, 0.05, 0.15, 0.09, 0.25, 0.13,
       0.05, 0.11, 0.1 , 0.14, 0.08, 0.31, 0.1 , 0.16, 0.2 , 0.17, 0.37,
       0.11, 0.22, 0.18, 0.06, 0.62, 0.13, 0.34, 0.16, 0.06, 0.17, 0.06,
       0.11, 0.29, 0.05, 0.24, 0.19, 0.04, 0.1 , 0.08, 0.12, 0.07, 0.03,
       0.12, 0.13, 0.38, 0.08, 0.29, 0.25, 0.21, 0.18, 0.17, 0.18, 0.19,
       0.02, 0.06, 0.1 , 0.08, 0.2 , 0.24, 0.01, 0.12, 0.13, 0.14, 0.1 ,
       0.18, 0.78, 0.11, 0.19, 0.19, 0.3 , 0.06, 0.13, 0.03, 0.27, 0.16,
       0.02, 0.09, 0.22, 0.11, 0.1 , 0.06, 0.22, 0.29, 0.21, 0.1 , 0.34,
       0.08, 0.59, 0.08, 0.25, 0.34, 0.03, 0.05, 0.31, 0.31, 0.01, 0.05,
       0.08, 0.24, 0.08, 0.41, 0.34, 0.36, 0.19, 0.07, 0.06, 0.24, 0.2 ,
       0.57, 0.17, 0.16, 0.16, 0.12, 0.14, 0.15, 0.12, 0.35, 0.06, 0.21,
       0.04, 0.08, 0.21, 0.09, 0.1 , 0.07, 0.1 , 0.23, 0.03, 0.07, 0.16,
       0.1 , 0.11, 0.25, 0.08, 0.26, 0.23, 0.52, 0.06, 0.19, 0.08, 0.02,
       0.24, 0.12, 0.02, 0.14, 0.3 , 0.13, 0.16, 0.04, 0.1 , 0.22, 0.07,
       0.58, 0.03, 0.21, 0.21, 0.07, 0.25, 0.29, 0.23, 0.04, 0.46, 0.03,
       0.07, 0.36, 0.05, 0.08, 0.27, 0.2 , 0.05, 0.09, 0.12, 0.03, 0.08,
       0.1 , 0.3 , 0.08, 0.21, 0.03, 0.19, 0.12, 0.3 , 0.14, 0.19, 0.28,
       0.11, 0.14, 0.43, 0.09, 0.04, 0.03, 0.03, 0.04, 0.06, 0.21, 0.1 ,
       0.09, 0.32, 0.42, 0.17, 0.13, 0.13, 0.17, 0.16, 0.12, 0.15, 0.04,
       0.14])
```

```
# 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()
```




## Hyperparameter Tuning on Random forest

```
parameter={
    'max_depth': list(range(10, 15)),
    'max_depth': [150,200,300],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', 'log2', 10, 0.5]
}
```

```
rfc_grid=GridSearchCV(rfc,param_grid=parameter,cv=5,scoring="accuracy",n_jobs=-1)
```

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

```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
  ▸ RandomForestClassifier
```

```
rfc_grid_pred=rfc_grid.predict(x_test)
```

```
rfc_grid_pred
```

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

```
rfc_grid.best_params_
```

```
{'max_depth': 300,
 'max_features': 10,
 'min_samples_leaf': 1,
 'min_samples_split': 10}
```

## Evalution Matrices of Gridsearchcv(Random Forest)

```
accuracy_score(y_test,rfc_grid_pred)
```

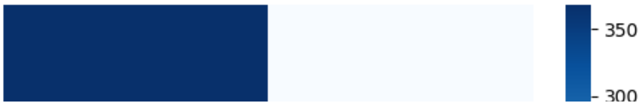
```
0.8684807256235828
```

```
print(classification_report(y_test,rfc_grid_pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.99      0.93       371
           1       0.83      0.21      0.34        70

    accuracy                           0.87       441
   macro avg       0.85      0.60      0.63       441
weighted avg       0.86      0.87      0.83       441
```

```
sns.heatmap(confusion_matrix(y_test,rfc_grid_pred),annot=True,fmt='d',cmap="Blues")
```

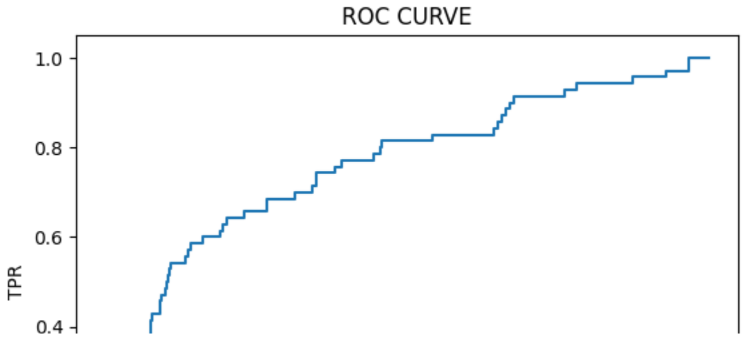
```
<Axes: >
```




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

```
array([0.06968065, 0.05635592, 0.18145954, 0.20476444, 0.71651423,
       0.49409068, 0.3365972 , 0.16821285, 0.11220681, 0.14758336,
       0.05046717, 0.08972515, 0.06004366, 0.57007986, 0.07570516,
       0.02197136, 0.06815058, 0.16289409, 0.03851741, 0.17449367,
       0.46577259, 0.14530556, 0.02093182, 0.06803247, 0.33023864,
       0.30025513, 0.03446864, 0.06512134, 0.62306093, 0.07024359,
       0.09943931, 0.16922883, 0.177579  , 0.09114494, 0.0730404 ,
       0.11482086, 0.12999089, 0.03685119, 0.06721154, 0.24086473,
       0.14218794, 0.0576829 , 0.0989508 , 0.11880851, 0.04928361,
       0.46581672, 0.09487902, 0.05750322, 0.63108769, 0.51628035,
       0.17731581, 0.45598414, 0.16653936, 0.20157733, 0.45693859,
       0.14485563, 0.05410212, 0.08463466, 0.11896527, 0.32847263,
       0.04382576, 0.13705231, 0.11250658, 0.20533626, 0.3236009 ,
       0.12589845, 0.2519329 , 0.15042585, 0.08464459, 0.1301577 ,
       0.10229183, 0.4013367 , 0.07320635, 0.08465754, 0.06044246,
       0.11233045, 0.14593372, 0.07114277, 0.36124512, 0.00253247,
       0.01982143, 0.06686869, 0.22936934, 0.21210541, 0.09063961,
       0.06669164, 0.07706876, 0.34434953, 0.0522212 , 0.06122106,
       0.53823825, 0.02415152, 0.18141947, 0.30701828, 0.07448296,
       0.03959402, 0.13408329, 0.42456501, 0.10987568, 0.07941886,
       0.18519708, 0.21504536, 0.26066557, 0.06134217, 0.0605926 ,
       0.06271284, 0.07942031, 0.38186673, 0.22959138, 0.22327519,
       0.19435578, 0.14376804, 0.07140625, 0.04536436, 0.10009106,
       0.17183866, 0.09834603, 0.13209774, 0.05135949, 0.        ,
       0.07374409, 0.05219848, 0.03657576, 0.68679609, 0.26797256,
       0.19941163, 0.02031818, 0.05080836, 0.11024875, 0.11924052,
       0.02777273, 0.33251743, 0.49579203, 0.17951957, 0.17401952,
       0.193125  , 0.38300095, 0.17437148, 0.15544885, 0.25253617,
       0.1023799 , 0.20166574, 0.05729224, 0.13599018, 0.33093001,
       0.04733858, 0.05748662, 0.11428441, 0.13220964, 0.10924396,
       0.08915508, 0.17446272, 0.13390107, 0.30292416, 0.14963303,
       0.0485776 , 0.10425234, 0.30464322, 0.19723693, 0.08213095,
       0.28351369, 0.05102579, 0.37753041, 0.537393  , 0.08459191,
       0.18105562, 0.22997278, 0.02640476, 0.12755069, 0.05999026,
       0.04576031, 0.1088683 , 0.13489284, 0.27703193, 0.08239623,
       0.26798078, 0.09700649, 0.32498214, 0.19332597, 0.15623964,
       0.04199377, 0.04203971, 0.0312619 , 0.41341055, 0.06250866,
       0.07668376, 0.26259992, 0.10008007, 0.17033028, 0.16599115,
       0.23964889, 0.56268486, 0.15073279, 0.02735462, 0.11837954,
       0.14148897, 0.09350321, 0.09201615, 0.41298532, 0.08046762,
       0.25836643, 0.1325245 , 0.22322721, 0.06941414, 0.12373747,
       0.1719427 , 0.39413649, 0.03099578, 0.17389184, 0.004     ,
       0.10564325, 0.15965812, 0.05970713, 0.24503056, 0.01758421,
       0.12439297, 0.06392806, 0.28354165, 0.1727817 , 0.45814859,
       0.15217503, 0.41628423, 0.47229244, 0.27155606, 0.15641321,
       0.11464827, 0.11382143, 0.3559227 , 0.66493497, 0.16012243,
       0.015     , 0.25088261, 0.06882251, 0.14763428, 0.01626858,
       0.37091857, 0.28645835, 0.09226472, 0.10495101, 0.09223107,
       0.28467328, 0.12034312, 0.04830682, 0.10447676, 0.10385212,
       0.10823458, 0.07957357, 0.31649269, 0.03643608, 0.20970754,
       0.17169904, 0.24800226, 0.3336872 , 0.10416563, 0.19233298,
       0.15694027, 0.05899341, 0.56731405, 0.09874986, 0.25531381,
       0.22515301, 0.0554206 , 0.23828632, 0.15536111, 0.10208531,
       0.27214468, 0.0160101 , 0.31124645, 0.2004338 , 0.04177194,
       0.09608621, 0.04319444, 0.16706795, 0.05887529, 0.04593927,
       0.05820051, 0.08443987, 0.38357844, 0.07768071, 0.36803295,
       0.31925373, 0.19991991, 0.13051689, 0.10863961, 0.19179842,
       0.18638066, 0.0377316 , 0.08476421, 0.06613541, 0.04243135,
```

```
# 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()
```




# REPORT

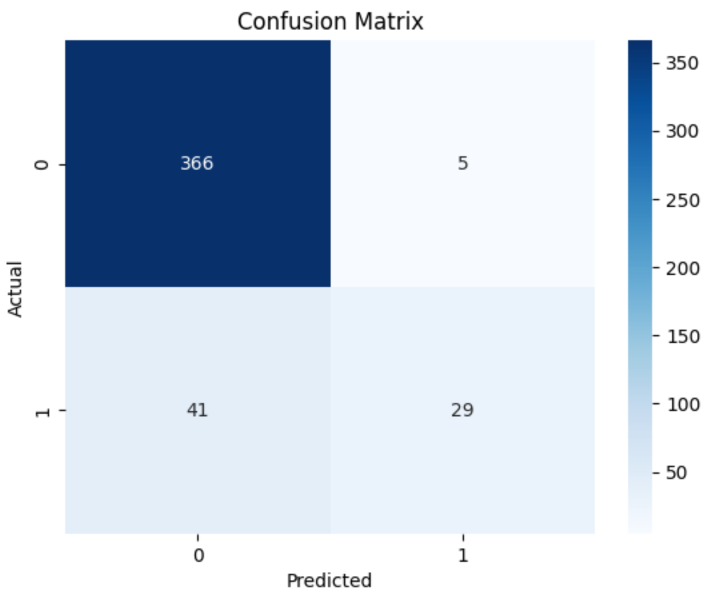
**Logistic Regression**

Accuracy_score: 90%

Classification Report:

```
              precision    recall  f1-score   support

           0       0.99      0.90      0.94       407
           1       0.41      0.85      0.56        34

    accuracy                           0.90       441
   macro avg       0.70      0.88      0.75       441
weighted avg       0.94      0.90      0.91       441
```

Confusion Matrix:



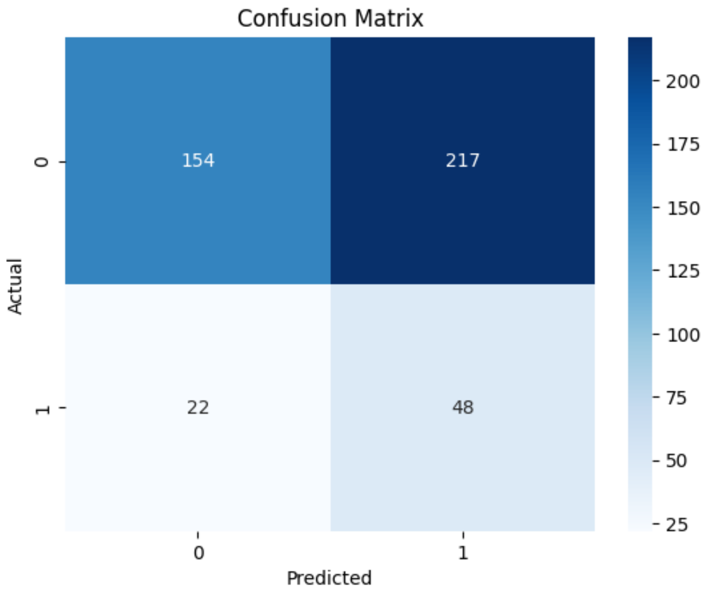

# Logistic Regression Without Scaling

**Accuracy: 46%**

**Classification Report:**

```
              precision    recall  f1-score   support

           0       0.42      0.88      0.56       176
           1       0.69      0.18      0.29       265

    accuracy                           0.46       441
   macro avg       0.55      0.53      0.42       441
weighted avg       0.58      0.46      0.40       441
```

**Cofusion Matrix**



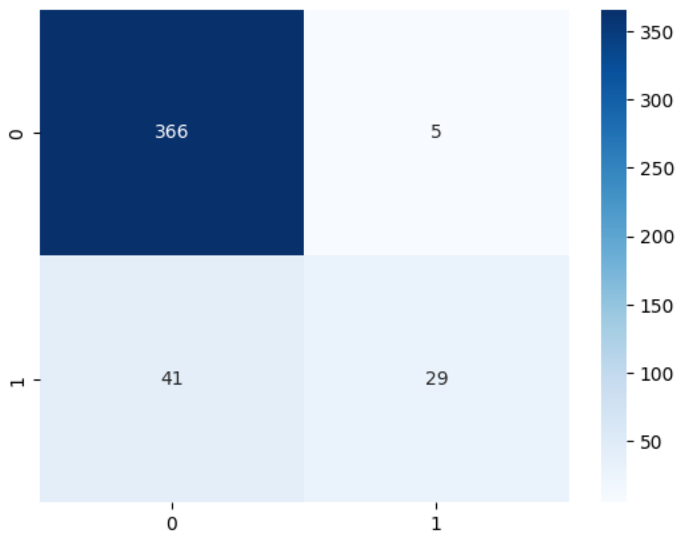

## HyperParameter Tuning on Logistic Regression

**Accuracy: 90%**

**Classification Report:**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.90 | 0.99 | 0.94 | 371 |
| 1 | 0.85 | 0.41 | 0.56 | 70 |
| accuracy | | | 0.90 | 441 |
| macro avg | 0.88 | 0.70 | 0.75 | 441 |
| weighted avg | 0.89 | 0.90 | 0.88 | 441 |

**Confusion Matrix:**

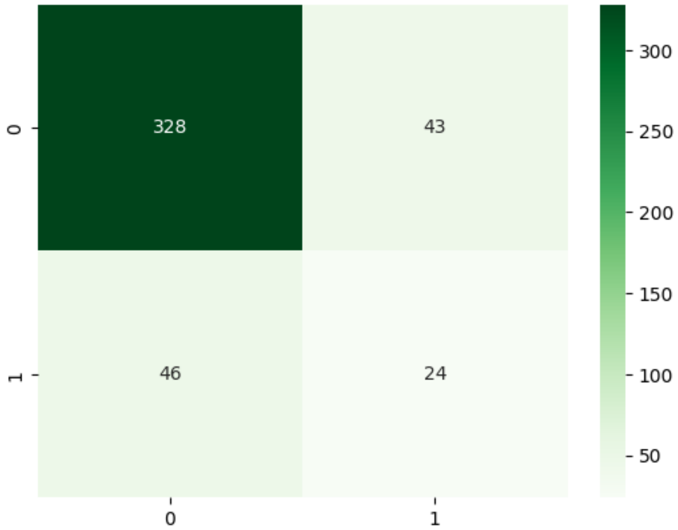



## Decision Tree

**Accuracy: 80%**

**Classification Report:**

```
              precision    recall  f1-score   support

           0       0.88      0.88      0.88       371
           1       0.36      0.34      0.35        70

    accuracy                           0.80       441
   macro avg       0.62      0.61      0.62       441
weighted avg       0.79      0.80      0.80       441
```




**Cofusion Matrix:**

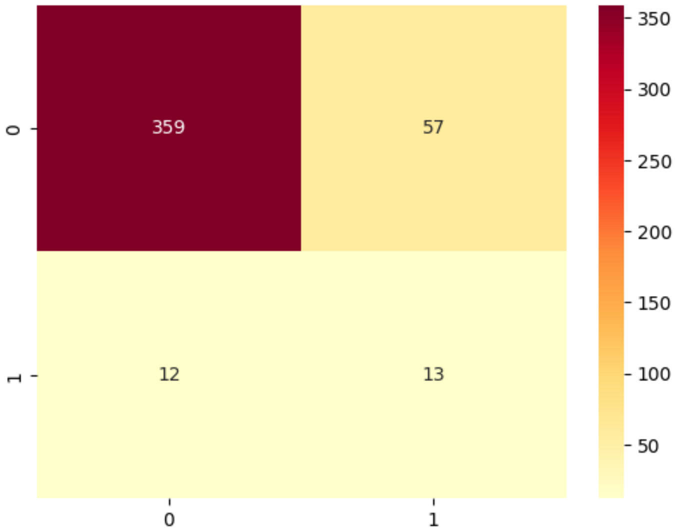
## Hyper Parameter Tuning on Decision Tree

**Accuracy: 84%**

**Classification Report:**

```
              precision    recall  f1-score   support

           0       0.97      0.86      0.91       416
           1       0.19      0.52      0.27        25

    accuracy                           0.84       441
   macro avg       0.58      0.69      0.59       441
weighted avg       0.92      0.84      0.88       441
```

**Cofusion Matrix:**



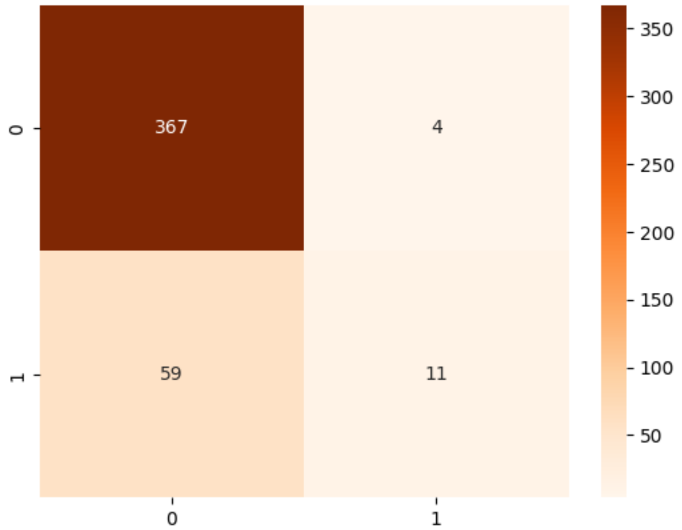

## Random Forest

**Accuracy: 86%**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.86 | 0.99 | 0.92 | 371 |
| 1 | 0.73 | 0.16 | 0.26 | 70 |
| accuracy | | | 0.86 | 441 |
| macro avg | 0.80 | 0.57 | 0.59 | 441 |
| weighted avg | 0.84 | 0.86 | 0.82 | 441 |

**Classification Report:**

**Cofusion Matrix:**




## HyperParameter Tuning on Random Forest

**Accuracy: 87%**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.87 | 0.99 | 0.93 | 371 |
| 1 | 0.83 | 0.21 | 0.34 | 70 |
| accuracy | | | 0.87 | 441 |
| macro avg | 0.85 | 0.60 | 0.63 | 441 |
| weighted avg | 0.86 | 0.87 | 0.83 | 441 |

**Classification Report:**

**Cofusion Matrix:**