In [7]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [8]:
```
df=pd.read_csv("Employee.csv")
```

In [9]:
```
df.head()
```

Out[9]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeN
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

In [10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1470 non-null   int64 
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64 
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64 
 6   Education                 1470 non-null   int64 
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64 
 9   EmployeeNumber            1470 non-null   int64 
 10  EnvironmentSatisfaction   1470 non-null   int64 
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64 
 13  JobInvolvement            1470 non-null   int64 
 14  JobLevel                  1470 non-null   int64 
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64 
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64 
 19  MonthlyRate               1470 non-null   int64 
 20  NumCompaniesWorked        1470 non-null   int64 
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64 
 24  PerformanceRating         1470 non-null   int64 
 25  RelationshipSatisfaction  1470 non-null   int64 
 26  StandardHours             1470 non-null   int64 
 27  StockOptionLevel          1470 non-null   int64 
 28  TotalWorkingYears         1470 non-null   int64 
 29  TrainingTimesLastYear     1470 non-null   int64 
 30  WorkLifeBalance           1470 non-null   int64 
 31  YearsAtCompany            1470 non-null   int64 
 32  YearsInCurrentRole        1470 non-null   int64 
 33  YearsSinceLastPromotion   1470 non-null   int64 
 34  YearsWithCurrManager      1470 non-null   int64 
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

In [11]:

```
df.isnull().any()
```

Out[11]:

```
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
```

In [12]:

```
df.corr()
```

```
C:\Users\Praveen\AppData\Local\Temp\ipykernel_25940\1134722465.py:1: FutureWarning: The default value of num
eric_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.
  df.corr()
```

Out[12]:

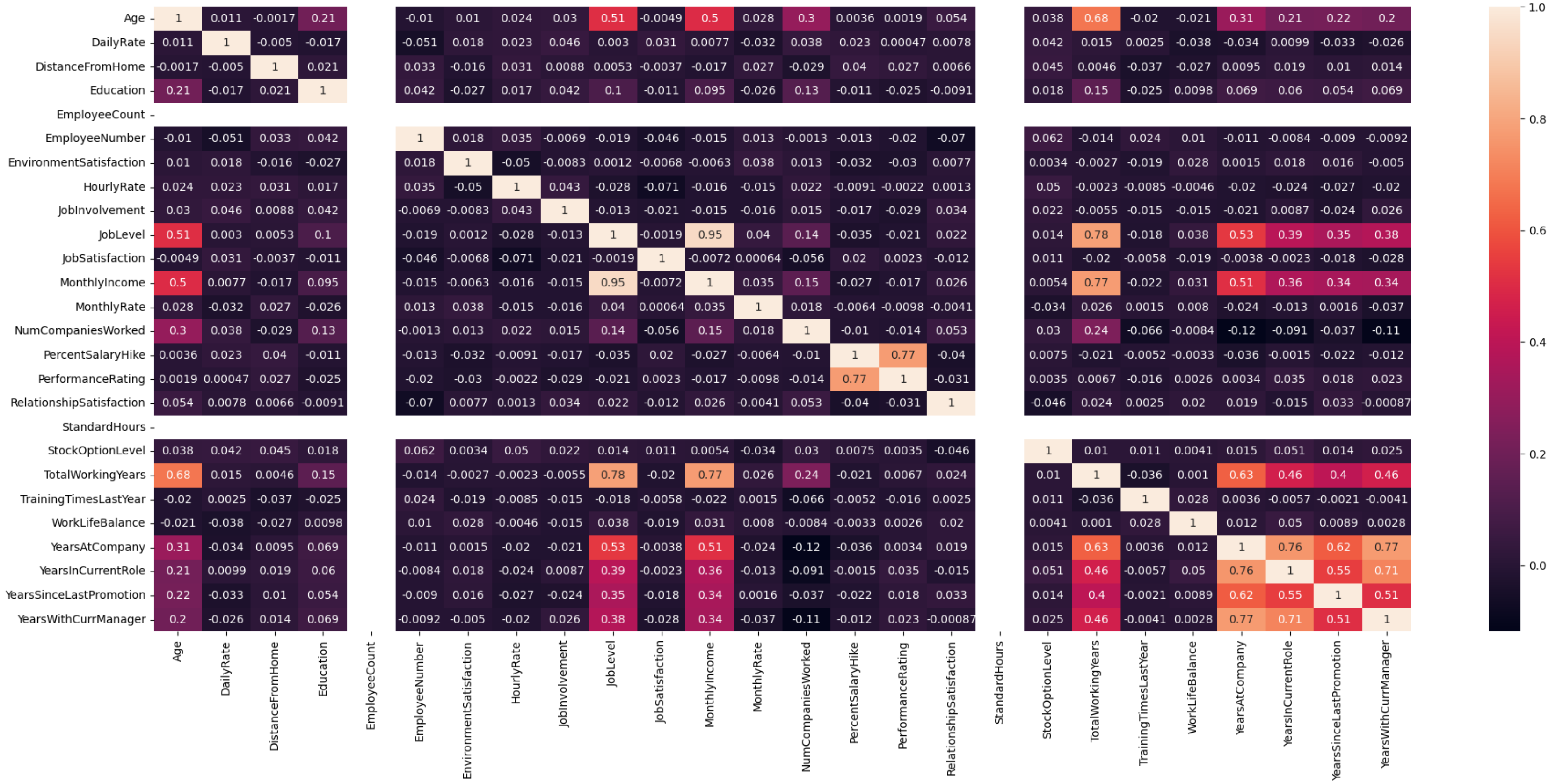
	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfacti
Age	1.000000	0.010661	-0.001686	0.208034	NaN	-0.010145	0.0101
DailyRate	0.010661	1.000000	-0.004985	-0.016806	NaN	-0.050990	0.0183
DistanceFromHome	-0.001686	-0.004985	1.000000	0.021042	NaN	0.032916	-0.0160
Education	0.208034	-0.016806	0.021042	1.000000	NaN	0.042070	-0.0271
EmployeeCount	NaN	NaN	NaN	NaN	NaN	NaN	N
EmployeeNumber	-0.010145	-0.050990	0.032916	0.042070	NaN	1.000000	0.0176
EnvironmentSatisfaction	0.010146	0.018355	-0.016075	-0.027128	NaN	0.017621	1.0000
HourlyRate	0.024287	0.023381	0.031131	0.016775	NaN	0.035179	-0.0498
JobInvolvement	0.029820	0.046135	0.008783	0.042438	NaN	-0.006888	-0.0082
JobLevel	0.509604	0.002966	0.005303	0.101589	NaN	-0.018519	0.0012
JobSatisfaction	-0.004892	0.030571	-0.003669	-0.011296	NaN	-0.046247	-0.0067
MonthlyIncome	0.497855	0.007707	-0.017014	0.094961	NaN	-0.014829	-0.0062
MonthlyRate	0.028051	-0.032182	0.027473	-0.026084	NaN	0.012648	0.0376
NumCompaniesWorked	0.299635	0.038153	-0.029251	0.126317	NaN	-0.001251	0.0125
PercentSalaryHike	0.003634	0.022704	0.040235	-0.011111	NaN	-0.012944	-0.0317
PerformanceRating	0.001904	0.000473	0.027110	-0.024539	NaN	-0.020359	-0.0295
RelationshipSatisfaction	0.053535	0.007846	0.006557	-0.009118	NaN	-0.069861	0.0076
StandardHours	NaN	NaN	NaN	NaN	NaN	NaN	N
StockOptionLevel	0.037510	0.042143	0.044872	0.018422	NaN	0.062227	0.0034
TotalWorkingYears	0.680381	0.014515	0.004628	0.148280	NaN	-0.014365	-0.0026
TrainingTimesLastYear	-0.019621	0.002453	-0.036942	-0.025100	NaN	0.023603	-0.0193
WorkLifeBalance	-0.021490	-0.037848	-0.026556	0.009819	NaN	0.010309	0.0276
YearsAtCompany	0.311309	-0.034055	0.009508	0.069114	NaN	-0.011240	0.0014
YearsInCurrentRole	0.212901	0.009932	0.018845	0.060236	NaN	-0.008416	0.0180
YearsSinceLastPromotion	0.216513	-0.033229	0.010029	0.054254	NaN	-0.009019	0.0161
YearsWithCurrManager	0.202089	-0.026363	0.014406	0.069065	NaN	-0.009197	-0.0049

26 rows × 26 columns

In [13]:
```
plt.figure(figsize=(25,10))
sns.heatmap(df.corr(),annot=True)
```

```
C:\Users\Praveen\AppData\Local\Temp\ipykernel_25940\1214538227.py:2: FutureWarning: The default value of num
eric_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(df.corr(),annot=True)
```

Out[13]:
```
<Axes: >
```



In [14]:
```
len(df.columns)
```

Out[14]:
```
35
```

In [15]:
```
df.head()
```

Out[15]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeN
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

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

Out[16]:

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

5 rows × 34 columns

In [17]:
```
y1=df.Attrition
y=y1.to_frame()
y.head()
```

Out[17]:

	Attrition
0	Yes
1	No
2	Yes
3	No
4	No

In [18]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
x.BusinessTravel=le.fit_transform(x.BusinessTravel)
print(le.classes_)
mapping=dict(zip(le.classes_,range(len(le.classes_))))
print(mapping)
```

```
['Non-Travel' 'Travel_Frequently' 'Travel_Rarely']
{'Non-Travel': 0, 'Travel_Frequently': 1, 'Travel_Rarely': 2}
```

In [19]:

```
x['Department']=le.fit_transform(x['Department'])
print(le.classes_)
mapping=dict(zip(le.classes_,range(len(le.classes_))))
print(mapping)
```

```
['Human Resources' 'Research & Development' 'Sales']
{'Human Resources': 0, 'Research & Development': 1, 'Sales': 2}
```

In [20]:

```
x.EducationField=le.fit_transform(x.EducationField)
print(le.classes_)
mapping=dict(zip(le.classes_,range(len(le.classes_))))
print(mapping)
```

```
['Human Resources' 'Life Sciences' 'Marketing' 'Medical' 'Other'
 'Technical Degree']
{'Human Resources': 0, 'Life Sciences': 1, 'Marketing': 2, 'Medical': 3, 'Other': 4, 'Technical Degree': 5}
```

In [21]:

```
x.Gender=le.fit_transform(x.Gender)
print(le.classes_)
mapping=dict(zip(le.classes_,range(len(le.classes_))))
print(mapping)
```

```
['Female' 'Male']
{'Female': 0, 'Male': 1}
```

In [22]:

```
x.JobRole=le.fit_transform(x.JobRole)
print(le.classes_)
mapping=dict(zip(le.classes_,range(len(le.classes_))))
print(mapping)
```

```
['Healthcare Representative' 'Human Resources' 'Laboratory Technician'
 'Manager' 'Manufacturing Director' 'Research Director'
 'Research Scientist' 'Sales Executive' 'Sales Representative']
{'Healthcare Representative': 0, 'Human Resources': 1, 'Laboratory Technician': 2, 'Manager': 3, 'Manufactur
ing Director': 4, 'Research Director': 5, 'Research Scientist': 6, 'Sales Executive': 7, 'Sales Representati
ve': 8}
```

In [23]:

```
x.MaritalStatus=le.fit_transform(x.MaritalStatus)
print(le.classes_)
mapping=dict(zip(le.classes_,range(len(le.classes_))))
print(mapping)
```

```
['Divorced' 'Married' 'Single']
{'Divorced': 0, 'Married': 1, 'Single': 2}
```

In [24]:

```
x.Over18 =le.fit_transform(x.Over18 )
print(le.classes_)
mapping=dict(zip(le.classes_,range(len(le.classes_))))
print(mapping)
```

```
['Y']
{'Y': 0}
```

In [25]:

```
x.OverTime =le.fit_transform(x.OverTime )
print(le.classes_)
mapping=dict(zip(le.classes_,range(len(le.classes_))))
print(mapping)
```

```
['No' 'Yes']
{'No': 0, 'Yes': 1}
```

In [26]:

```
y.Attrition =le.fit_transform(y.Attrition )
print(le.classes_)
mapping=dict(zip(le.classes_,range(len(le.classes_))))
print(mapping)
```

```
['No' 'Yes']
{'No': 0, 'Yes': 1}
```

In [27]:

```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 34 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   BusinessTravel            1470 non-null   int32
 2   DailyRate                 1470 non-null   int64
 3   Department                1470 non-null   int32
 4   DistanceFromHome          1470 non-null   int64
 5   Education                 1470 non-null   int64
 6   EducationField            1470 non-null   int32
 7   EmployeeCount             1470 non-null   int64
 8   EmployeeNumber            1470 non-null   int64
 9   EnvironmentSatisfaction   1470 non-null   int64
 10  Gender                    1470 non-null   int32
 11  HourlyRate                1470 non-null   int64
 12  JobInvolvement            1470 non-null   int64
 13  JobLevel                  1470 non-null   int64
 14  JobRole                   1470 non-null   int32
 15  JobSatisfaction           1470 non-null   int64
 16  MaritalStatus             1470 non-null   int32
 17  MonthlyIncome             1470 non-null   int64
 18  MonthlyRate               1470 non-null   int64
 19  NumCompaniesWorked        1470 non-null   int64
 20  Over18                    1470 non-null   int32
 21  OverTime                  1470 non-null   int32
 22  PercentSalaryHike         1470 non-null   int64
 23  PerformanceRating         1470 non-null   int64
 24  RelationshipSatisfaction  1470 non-null   int64
 25  StandardHours             1470 non-null   int64
 26  StockOptionLevel          1470 non-null   int64
 27  TotalWorkingYears         1470 non-null   int64
 28  TrainingTimesLastYear     1470 non-null   int64
 29  WorkLifeBalance           1470 non-null   int64
 30  YearsAtCompany            1470 non-null   int64
 31  YearsInCurrentRole        1470 non-null   int64
 32  YearsSinceLastPromotion   1470 non-null   int64
 33  YearsWithCurrManager      1470 non-null   int64
dtypes: int32(8), int64(26)
memory usage: 344.7 KB
```

In [28]:
```
x.head()
```

Out[28]:

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

5 rows × 34 columns

In [29]:
```
y.head()
```

Out[29]:

	Attrition
0	1
1	0
2	1
3	0
4	0

In [30]:
```
from sklearn.preprocessing import MinMaxScaler
ms=MinMaxScaler()
x_scaled=pd.DataFrame(ms.fit_transform(x),columns=x.columns)
```

In [31]:
```
x_scaled.head()
```

Out[31]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber
0	0.547619	1.0	0.715820	1.0	0.000000	0.25	0.2	0.0	0.000000
1	0.738095	0.5	0.126700	0.5	0.250000	0.00	0.2	0.0	0.000484
2	0.452381	1.0	0.909807	0.5	0.035714	0.25	0.8	0.0	0.001451
3	0.357143	0.5	0.923407	0.5	0.071429	0.75	0.2	0.0	0.001935
4	0.214286	1.0	0.350036	0.5	0.035714	0.00	0.6	0.0	0.002903

5 rows × 34 columns

In [32]:
```
x_scaled.shape
```

Out[32]:
```
(1470, 34)
```

In [33]:
```
y.shape
```

Out[33]:
```
(1470, 1)
```

In [34]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.2,random_state=1)
```

In [35]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1176, 34)
(294, 34)
(1176, 1)
(294, 1)
```

Logistic Regression

```
In [36]: from sklearn.linear_model import LogisticRegression
         lr=LogisticRegression()
         lr.fit(x_train,y_train)
```

```
C:\Users\Praveen\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWarning: A colu
mn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for exam
ple using ravel().
  y = column_or_1d(y, warn=True)
```

```
Out[36]: ▾ LogisticRegression

         LogisticRegression()
```

```
In [37]: y_pred=lr.predict(x_test)
```

```
In [38]: y_pred
```

```
Out[38]: array([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, 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, 0, 0, 0, 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, 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, 1, 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, 0, 0, 0, 1, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                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, 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, 1, 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])
```

In [39]:

```
y_test
```

Out[39]:

	Attrition
1291	1
1153	1
720	1
763	0
976	0
...	...
302	0
443	1
701	0
309	0
845	0

294 rows × 1 columns

In [40]:

```
print(pd.DataFrame({"Actual":y_test.Attrition,"Predicted":y_pred}))
```

```
      Actual  Predicted
1291       1          0
1153       1          1
720        1          1
763        0          0
976        0          0
...      ...        ...
302        0          0
443        1          0
701        0          0
309        0          0
845        0          0

[294 rows x 2 columns]
```

In [41]:

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

In [42]:

```
accuracy_score(y_test,y_pred)
```

Out[42]:

```
0.8333333333333334
```

In [43]:

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.98      0.90       236
           1       0.76      0.22      0.35        58

    accuracy                           0.83       294
   macro avg       0.80      0.60      0.63       294
weighted avg       0.82      0.83      0.79       294
```

In [44]:

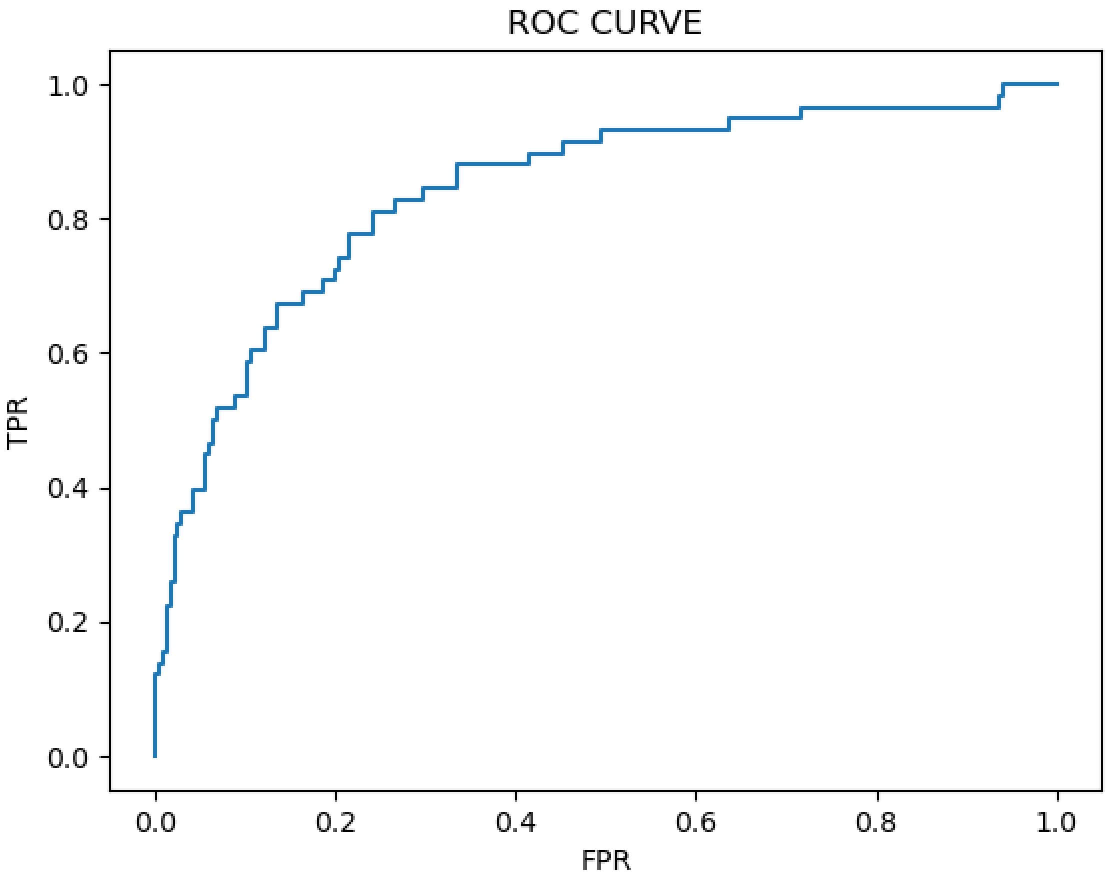
```
y_probability=lr.predict_proba(x_test)[:,1]
```

In [45]:

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

In [46]:

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



Hyperparameter Tuning (Logistic Regression)

In [47]:

```python
from sklearn.model_selection import GridSearchCV

# Define a range of hyperparameters to search
param_grid = {
    'penalty': ['l1', 'l2'],
    'C': [0.001, 0.01, 0.1, 1, 10, 100],  # Inverse of regularization strength
    'solver': ['liblinear', 'saga']
}

# Create a logistic regression classifier
lr = LogisticRegression()

# Create a GridSearchCV object with 5-fold cross-validation
grid_search = GridSearchCV(lr, param_grid, cv=5, scoring='accuracy')

# Fit the grid search to your training data
grid_search.fit(x_train, y_train)
```

```
olumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), fo
r example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\Praveen\anaconda3\Lib\site-packages\sklearn\linear_model\_sag.py:350: ConvergenceWarning: The ma
x_iter was reached which means the coef_ did not converge
  warnings.warn(
C:\Users\Praveen\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWarning: A c
olumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), fo
r example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\Praveen\anaconda3\Lib\site-packages\sklearn\linear_model\_sag.py:350: ConvergenceWarning: The ma
x_iter was reached which means the coef_ did not converge
  warnings.warn(
C:\Users\Praveen\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWarning: A c
olumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), fo
r example using ravel().
  y = column_or_1d(y, warn=True)

C:\Users\Praveen\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWarning: A c
olumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), fo
r example using ravel().
```

In [49]:

```
# Print the best hyperparameters found by grid search
print("Best Hyperparameters:")
print(grid_search.best_params_)
```

```
Best Hyperparameters:
{'C': 10, 'penalty': 'l1', 'solver': 'saga'}
```

In [50]:

```
# Get the best model
best_lr = grid_search.best_estimator_
best_lr
```

Out[50]:

```
▾                  LogisticRegression

LogisticRegression(C=10, penalty='l1', solver='saga')
```

In [51]:

```
# Evaluate the best model on the test set
y_pred_best = best_lr.predict(x_test)
print(pd.DataFrame({"Actual":y_test.Attrition,"Predicted":y_pred_best}))
```

```
      Actual  Predicted
1291       1          0
1153       1          1
720        1          1
763        0          0
976        0          0
...      ...        ...
302        0          0
443        1          0
701        0          0
309        0          0
845        0          0

[294 rows x 2 columns]
```

In [52]:

```
# Calculate accuracy and other metrics
accuracy_best = accuracy_score(y_test, y_pred_best)
classification_report_best = classification_report(y_test, y_pred_best)

print("Accuracy with Best Hyperparameters:", accuracy_best)
print("Classification Report with Best Hyperparameters:\n", classification_report_best)
```

```
Accuracy with Best Hyperparameters: 0.8469387755102041
Classification Report with Best Hyperparameters:
               precision    recall  f1-score   support

           0       0.85      0.98      0.91       236
           1       0.78      0.31      0.44        58

    accuracy                           0.85       294
   macro avg       0.82      0.64      0.68       294
weighted avg       0.84      0.85      0.82       294
```

Decision Tree

In [53]:

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

In [54]:

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

Out[54]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [55]:

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

In [56]:

```
print(pd.DataFrame({"Actual":y_test.Attrition,"Predicted":y_pred}))
```

```
      Actual  Predicted
1291       1          1
1153       1          1
720        1          0
763        0          0
976        0          1
...      ...        ...
302        0          1
443        1          0
701        0          0
309        0          0
845        0          0

[294 rows x 2 columns]
```

In [57]:

```
accuracy_score(y_test,y_pred)
```

Out[57]:

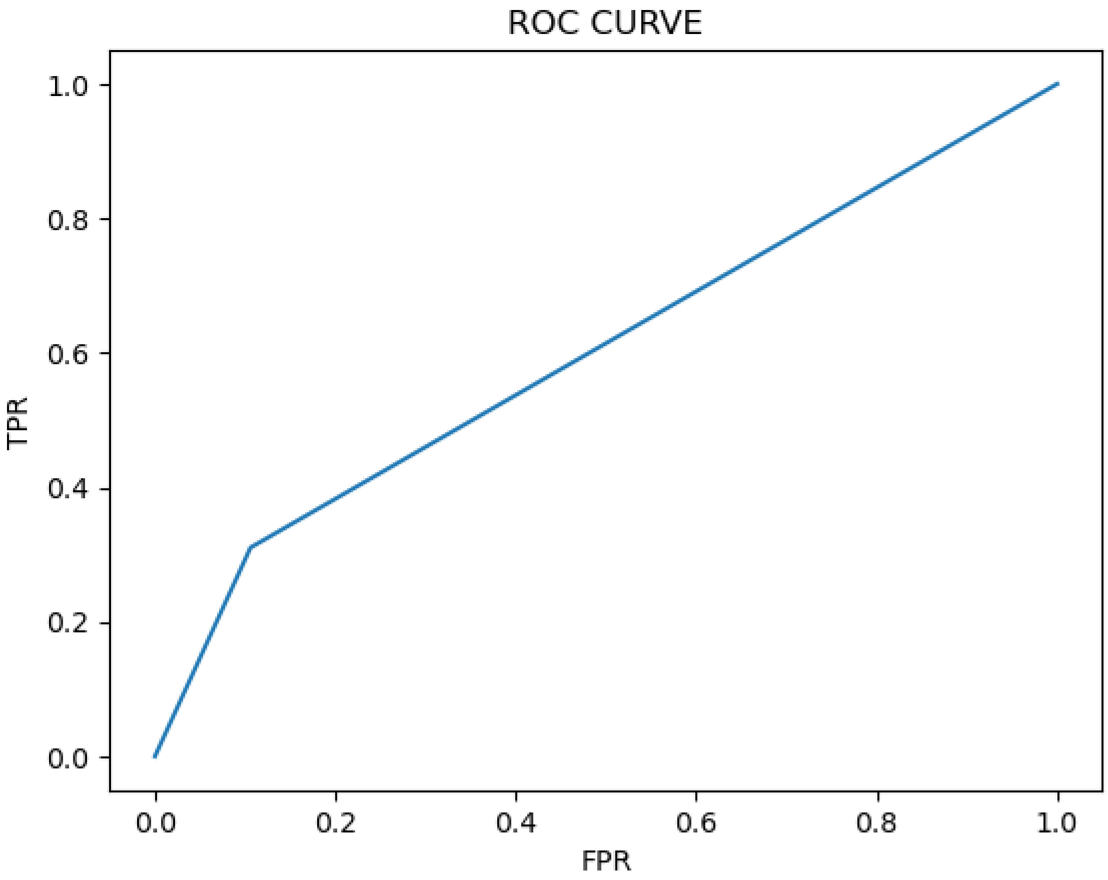
```
0.7789115646258503
```

In [58]:

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.89      0.87       236
           1       0.42      0.31      0.36        58

    accuracy                           0.78       294
   macro avg       0.63      0.60      0.61       294
weighted avg       0.76      0.78      0.77       294
```

In [59]:
```
y_prob=dtc.predict_proba(x_test)[:,1]
fpr,tpr,thresholds=roc_curve(y_test,y_prob)
plt.plot(fpr,tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC CURVE')
plt.show()
```



Hyperparameter tuning(Pre Pruning) for Decision Tree

In [60]:

```
para={
    'criterion':['entropy','gini'],
    'splitter':['best','random'],
    'max_features':['auto','sqrt','log2'],
    'max_depth':list(range(0,10))

}
```

In [61]:

```
grid_dtc=GridSearchCV(dtc,para,cv=10,scoring='accuracy')
```

In [62]:

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

```
C:\Users\Praveen\anaconda3\Lib\site-packages\sklearn\tree\_classes.py:269: FutureWarning: `max_features
='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly se
t `max_features='sqrt'`.
  warnings.warn(
C:\Users\Praveen\anaconda3\Lib\site-packages\sklearn\tree\_classes.py:269: FutureWarning: `max_features
='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly se
t `max_features='sqrt'`.
  warnings.warn(
C:\Users\Praveen\anaconda3\Lib\site-packages\sklearn\tree\_classes.py:269: FutureWarning: `max_features
='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly se
t `max_features='sqrt'`.
  warnings.warn(
C:\Users\Praveen\anaconda3\Lib\site-packages\sklearn\tree\_classes.py:269: FutureWarning: `max_features
='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly se
t `max_features='sqrt'`.
  warnings.warn(
C:\Users\Praveen\anaconda3\Lib\site-packages\sklearn\tree\_classes.py:269: FutureWarning: `max_features
='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly se
t `max_features='sqrt'`.
  warnings.warn(
```

In [63]:

```
grid_dtc.best_params_
```

Out[63]:

```
{'criterion': 'entropy',
 'max_depth': 7,
 'max_features': 'auto',
 'splitter': 'random'}
```

In [64]:
```
best_dtc=DecisionTreeClassifier(
  criterion=grid_dtc.best_params_['criterion'],
  max_depth=grid_dtc.best_params_['max_depth'],
  max_features=grid_dtc.best_params_["max_features"],
  splitter=grid_dtc.best_params_["splitter"]
)
best_dtc.fit(x_train,y_train)
```

```
C:\Users\Praveen\anaconda3\Lib\site-packages\sklearn\tree\_classes.py:269: FutureWarning: `max_features='aut
o'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_f
eatures='sqrt'`.
  warnings.warn(
```

Out[64]:
```
▾                          DecisionTreeClassifier

DecisionTreeClassifier(criterion='entropy', max_depth=7, max_features='auto',
                       splitter='random')
```

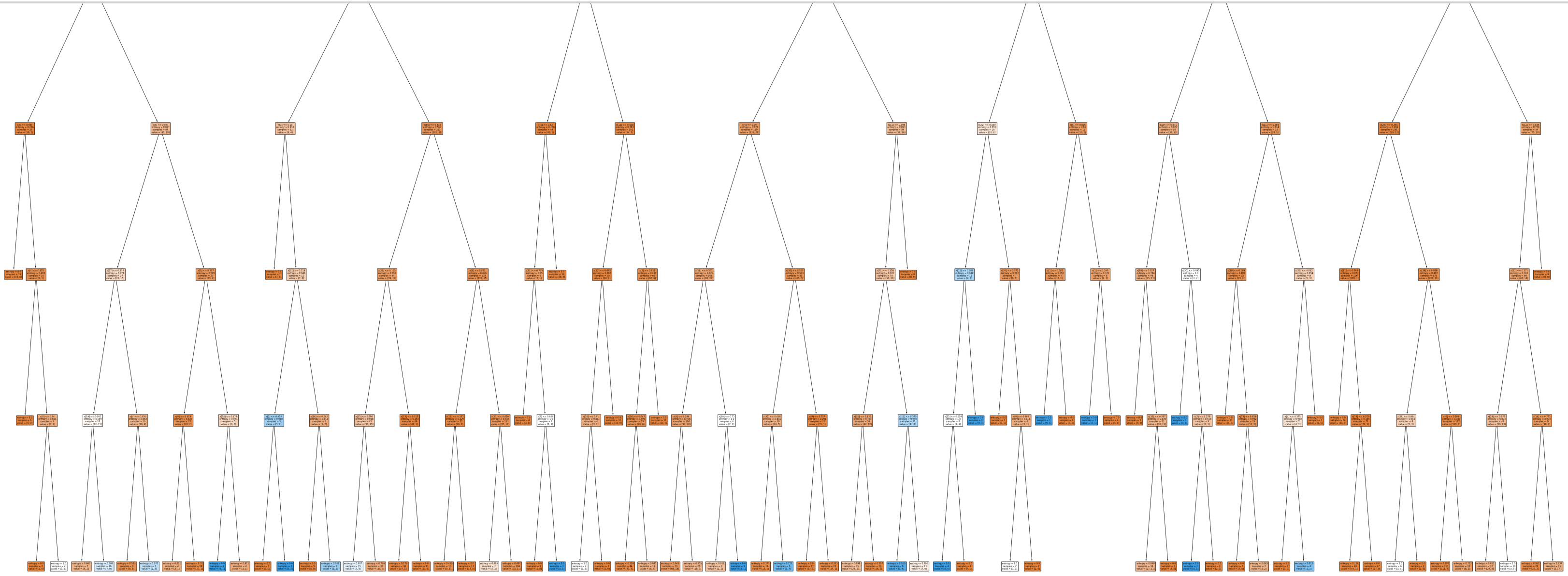
In [65]:
```
y_pred=best_dtc.predict(x_test)
print(pd.DataFrame({"Actual":y_test.Attrition,"Predicted":y_pred}))
```

```
      Actual  Predicted
1291       1          0
1153       1          0
720        1          0
763        0          0
976        0          0
...      ...        ...
302        0          0
443        1          0
701        0          0
309        0          0
845        0          0

[294 rows x 2 columns]
```

In [66]:
```
print(accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
0.7993197278911565
              precision    recall  f1-score   support

           0       0.82      0.95      0.88       236
           1       0.48      0.17      0.25        58

    accuracy                           0.80       294
   macro avg       0.65      0.56      0.57       294
weighted avg       0.76      0.80      0.76       294
```

In [88]:
```
from sklearn import tree
plt.figure(figsize=(80,60))
tree.plot_tree(best_dtc,filled=True)
```



Random Forest

```
In [67]: from sklearn.ensemble import RandomForestClassifier
         rfc=RandomForestClassifier()
```

```
In [68]: rfc.fit(x_train,y_train)
```

```
C:\Users\Praveen\AppData\Local\Temp\ipykernel_25940\4070307935.py:1: DataConversionWarning: A column-vector
y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using r
avel().
  rfc.fit(x_train,y_train)
```

```
Out[68]: ▾ RandomForestClassifier
         RandomForestClassifier()
```

```
In [69]: y_pred=rfc.predict(x_test)
```

```
In [70]: accuracy_score(y_test,y_pred)
```

```
Out[70]: 0.8231292517006803
```

```
In [71]: print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.99      0.90       236
           1       0.75      0.16      0.26        58

    accuracy                           0.82       294
   macro avg       0.79      0.57      0.58       294
weighted avg       0.81      0.82      0.77       294
```

Pre pruning Random Forest

In [92]:

```
para={
     'criterion':['gini','entropy'],
    'max_features':['best','sqrt','log2',None],
    'max_depth':[10, 20, 30, None],
}
```

In [93]:

```
rfc_cv=GridSearchCV(rfc,para,cv=5,scoring="accuracy")
```

In [94]:

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

```
les,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Praveen\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.py:686: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samp
les,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Praveen\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.py:686: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samp
les,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Praveen\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.py:686: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samp
les,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Praveen\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.py:686: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samp
les,), for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Praveen\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.py:686: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samp
```

In [95]:

```
rfc_cv.best_params_
```

Out[95]:

```
{'criterion': 'entropy', 'max_depth': 10, 'max_features': None}
```

In [96]:

```
best_rfc=RandomForestClassifier(
  criterion=rfc_cv.best_params_['criterion'],
  max_depth=rfc_cv.best_params_['max_depth'],
  max_features=rfc_cv.best_params_["max_features"],
)
```

In [97]:

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

```
C:\Users\Praveen\AppData\Local\Temp\ipykernel_25940\2820291153.py:1: DataConversionWarning: A column-vector
y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using r
avel().
  best_rfc.fit(x_train,y_train)
```

Out[97]:

```
▾                            RandomForestClassifier

RandomForestClassifier(criterion='entropy', max_depth=10, max_features=None)
```

In [98]:

```
y_pred=best_rfc.predict(x_test)
```

In [101]:

```
print("Accuracy:",accuracy_score(y_test,y_pred))
print("Classification Report:\n",classification_report(y_test,y_pred))
```

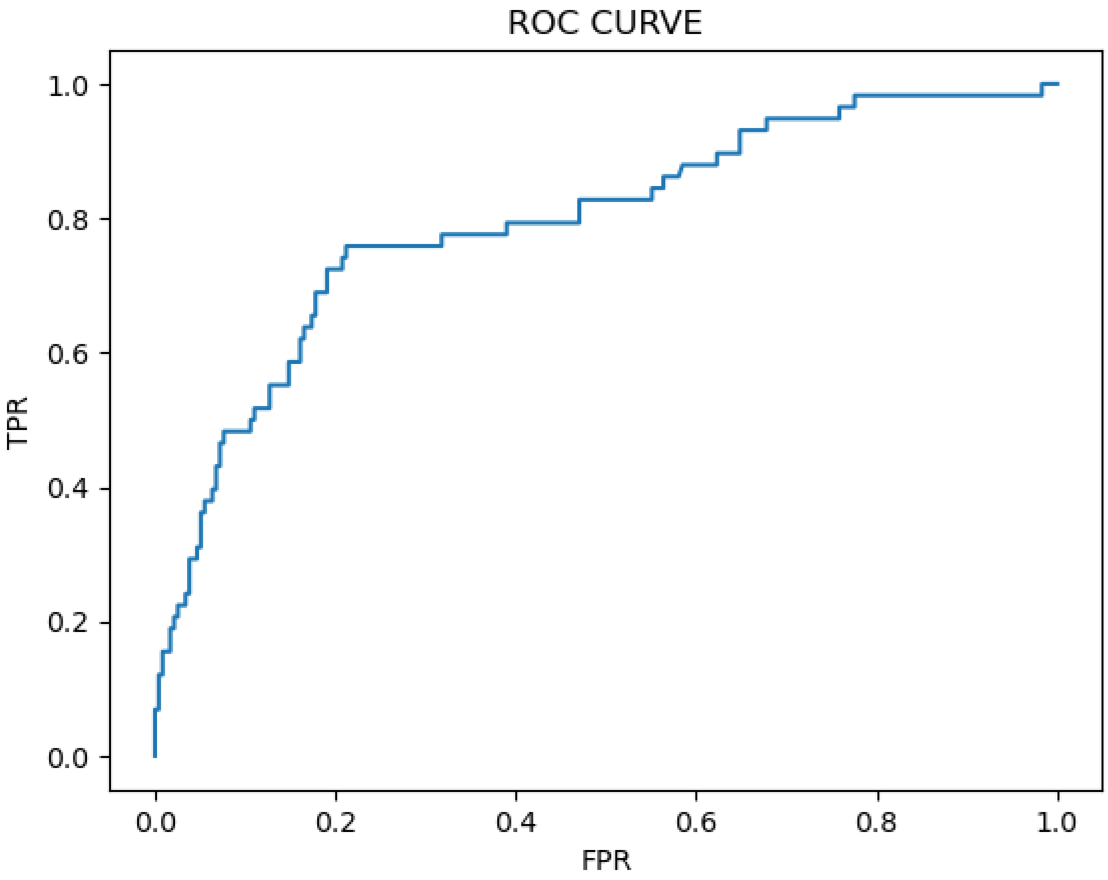
```
Accuracy: 0.8231292517006803
Classification Report:
               precision    recall  f1-score   support

           0       0.83      0.98      0.90       236
           1       0.69      0.19      0.30        58

    accuracy                           0.82       294
   macro avg       0.76      0.58      0.60       294
weighted avg       0.80      0.82      0.78       294
```

In [103]:

```
y_prob=best_rfc.predict_proba(x_test)[:,1]
```

In [106]:

```
fpr,tpr,thresholds=roc_curve(y_test,y_prob)
plt.plot(fpr,tpr)
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC CURVE")
plt.show()
```


ROC CURVE
TPR
FPR