___ASSIGNMENT--[4]--(SEP-22)____

Unsupported Cell Type. Double-Click to inspect/edit the content.

Unsupported Cell Type. Double-Click to inspect/edit the content.

DATA PREPROCESSING

1.IMPORT LIBRARIES

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

2.IMPORT THE DATASET

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

```
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educatic
0	41	Yes	Travel_Rarely	1102	Sales	1	
1	49	No	Travel_Frequently	279	Research & Development	8	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	
4	27	No	Travel_Rarely	591	Research & Development	2	

5 rows × 35 columns

```
df.info()
```

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

```
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

```
df.describe()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	Employe
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	147(
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024
std	9.135373	403.509100	8.106864	1.024165	0.0	602
min	18.000000	102.000000	1.000000	1.000000	1.0	1
25%	30.000000	465.000000	2.000000	2.000000	1.0	491
50%	36.000000	802.000000	7.000000	3.000000	1.0	102(
75%	43.000000	1157.000000	14.000000	4.000000	1.0	155
max	60.000000	1499.000000	29.000000	5.000000	1.0	206

8 rows × 26 columns

```
df.shape
```

```
(1470, 35)
```

3.checking null values

```
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
```

4.Data Visualization.

```
sns.scatterplot(x="YearsWithCurrManager",y="YearsSinceLastPromotion",data=df)
```

```
plt.subplots(figsize=(20,15))
sns.heatmap(df.corr(),annot=True)
```

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

```
sns.boxplot(df["MonthlyIncome"])
```

```
df.head()
```

5.Splitting Dependent and Independent variables

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

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

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

6.Encoding

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

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

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

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

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

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

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

```
x["Over18"]=le.fit_transform(x["Over18"])
```

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

```
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
```

7.Feature scaling

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

```
x_scaled.head()
```

8.SPLITTING OF TRAINING AND TESTING DATA

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.2,random_state=0)
```

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

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

MODEL BUILDING using decision tree

Unsupported Cell Type. Double-Click to inspect/edit the content.

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

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

```
DecisionTreeClassifier()
```

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

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

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

```
y_test
```

```
442      No
1091     No
981     Yes
785      No
1332    Yes
       ...
1439     No
481      No
124     Yes
198      No
1229     No
Name: Attrition, Length: 294, dtype: object
```

```
x.iloc[:,0:16]
```

```
dtc.predict(ms.transform([[41,2,1102,2,1,2,1,1,1,2,0,94,3,2,7,4,2,5993,19479,8,0,1,11,3,1,80,0,8,0,1,6,4,0,5 ]]))
```

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

```
y.head()
```

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

Evaluation of classification model

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

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

```
0.7619047619047619
```

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

```
array([[207,  38],
       [ 32,  17]], dtype=int64)
```

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

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

```
                  precision    recall  f1-score   support

              No       0.87      0.84      0.86       245
             Yes       0.31      0.35      0.33        49

        accuracy                           0.76       294
       macro avg       0.59      0.60      0.59       294
    weighted avg       0.77      0.76      0.77       294
```

Roc-AUC curve

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

```
probability
```

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

```
from sklearn.metrics import roc_curve
```

```
fpr, tpr, thresholds = roc_curve(y_test, probability, pos_label='Yes')
```

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

Model building using logistic regression

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

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

```
LogisticRegression()
```

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

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

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

```
y_test
```

```
442      No
1091     No
981     Yes
785      No
1332    Yes
       ...
1439     No
481      No
124     Yes
198      No
1229     No
Name: Attrition, Length: 294, dtype: object
```

```
model.predict(ms.transform([[41,2,1102,2,1,2,1,1,1,2,0,94,3,2,7,4,2,5993,19479,8,0,1,11,3,1,80,0,8,0,1,6,4,0,5]]))
```

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

```
x_train
```

Evaluation of model

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

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

```
0.8843537414965986
```

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

```
array([[242,   3],
       [ 31,  18]], dtype=int64)
```

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

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

```
                  precision    recall  f1-score   support

              No       0.89      0.99      0.93       245
             Yes       0.86      0.37      0.51        49

        accuracy                           0.88       294
       macro avg       0.87      0.68      0.72       294
    weighted avg       0.88      0.88      0.86       294
```

roc auc curve

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

```
array([0.16000127, 0.20600667, 0.31532384, 0.09242886, 0.63667551,
       0.06153061, 0.61819432, 0.0757087 , 0.00841372, 0.3912069 ,
       0.05398439, 0.33293123, 0.02020698, 0.67215483, 0.19786547,
       0.03454902, 0.11043981, 0.17101703, 0.04477777, 0.22783614,
       0.2335018 , 0.01553905, 0.06464492, 0.05029956, 0.58792413,
```

```
       0.44849464, 0.07412714, 0.04460935, 0.67666632, 0.0584383 ,
       0.01599026, 0.03521098, 0.06963085, 0.17397462, 0.07830857,
       0.04288032, 0.08150424, 0.07106342, 0.03622137, 0.05223965,
       0.04862098, 0.02091497, 0.01819361, 0.01362467, 0.02873997,
       0.50236969, 0.41553218, 0.00306874, 0.73976412, 0.51382382,
       0.09637213, 0.48845516, 0.08036228, 0.25757243, 0.66516772,
       0.26308027, 0.01964858, 0.30198497, 0.02919946, 0.16038964,
       0.02102747, 0.21670232, 0.13981568, 0.0358316 , 0.37208403,
       0.03002317, 0.29091186, 0.16041142, 0.10437497, 0.08695177,
       0.08217589, 0.30984518, 0.08531362, 0.07420689, 0.12268651,
       0.06192552, 0.04640904, 0.07624712, 0.19738483, 0.03236316,
       0.00884439, 0.0244108 , 0.13635803, 0.0260104 , 0.03341008,
       0.08186888, 0.00499397, 0.03474852, 0.03858027, 0.14602694,
       0.26167665, 0.16667357, 0.27400109, 0.24159565, 0.02160421,
       0.17748606, 0.34076078, 0.28022482, 0.06914126, 0.05003806,
       0.24437761, 0.74698271, 0.35438567, 0.01920627, 0.08778845,
       0.03255847, 0.05461351, 0.15123251, 0.06843702, 0.13752637,
       0.09584388, 0.04669882, 0.02493091, 0.15383171, 0.07081259,
       0.03089296, 0.0537667 , 0.11554316, 0.00881616, 0.01263271,
       0.17552253, 0.05045234, 0.08823238, 0.82995757, 0.03017756,
       0.0236819 , 0.0087012 , 0.1349589 , 0.16474801, 0.05202613,
       0.01524549, 0.29278083, 0.54767448, 0.34275448, 0.04629541,
       0.38966344, 0.61333366, 0.14552367, 0.07402366, 0.24143471,
       0.09418418, 0.0689069 , 0.10061956, 0.19346327, 0.20026293,
       0.03004939, 0.14900424, 0.00348846, 0.11225149, 0.15843155,
       0.06047573, 0.18601882, 0.06085869, 0.12221317, 0.03280184,
       0.02738799, 0.06356425, 0.08302382, 0.01541716, 0.014665  ,
       0.38517822, 0.01264231, 0.14961974, 0.80508787, 0.11598661,
       0.2842811 , 0.17020143, 0.1530583 , 0.02764153, 0.00613226,
       0.04191632, 0.09782393, 0.11551417, 0.10377982, 0.01779313,
       0.14371315, 0.10615435, 0.10298963, 0.05132621, 0.09061081,
       0.02897383, 0.09924087, 0.00512032, 0.75108423, 0.04296968,
       0.04062134, 0.37518972, 0.04563128, 0.7251816 , 0.10671665,
       0.36949086, 0.38146941, 0.32095493, 0.05266802, 0.08172004,
       0.13947833, 0.04334317, 0.01469593, 0.26413988, 0.06330966,
       0.1614747 , 0.15380517, 0.67152357, 0.05840793, 0.27891823,
       0.04512564, 0.46033865, 0.00348431, 0.14068967, 0.02747401,
       0.12714133, 0.17284246, 0.07341066, 0.10099827, 0.16870885,
       0.02560842, 0.01824031, 0.08670796, 0.02834237, 0.13710215,
       0.08778935, 0.2200061 , 0.73401148, 0.15938978, 0.4095449 ,
       0.01513845, 0.11306309, 0.21497506, 0.32337575, 0.03409266,
       0.04256318, 0.32157531, 0.05454465, 0.02348479, 0.16423352,
       0.32696147, 0.22892063, 0.00877159, 0.08198819, 0.01156361,
       0.1408691 , 0.29235147, 0.01270305, 0.17329916, 0.04081391,
       0.04094165, 0.42771425, 0.34958286, 0.03766772, 0.12025286,
       0.37698923, 0.3192629 , 0.79559338, 0.05385659, 0.21597037,
       0.06383728, 0.00570991, 0.66018187, 0.35855286, 0.37783606,
       0.36781398, 0.03554512, 0.21718203, 0.05943622, 0.06554485,
       0.10081475, 0.00818713, 0.26591316, 0.42809675, 0.06542835,
       0.09296803, 0.01259826, 0.14226651, 0.05072662, 0.02372258,
       0.02586923, 0.06760427, 0.24315648, 0.26961432, 0.19831733,
       0.2652296 , 0.0165923 , 0.15784236, 0.08398982, 0.02711775,
       0.18750547, 0.00783535, 0.2844239 , 0.00270742, 0.02484969
```

```
from sklearn.metrics import roc_curve

fpr, tpr, thresholds = roc_curve(y_test, probability, pos_label='Yes')
```

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