Kilaru Soma Sekhar

21BCE7019

ASSIGNMENT-4

AI AND ML MORNING SLOT

somasekhar.21bce7019@vitapstudent.ac.in

Import necessary libraries

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

Import dataset

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

```
df
```

	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	
...	...	...	...	...	...	...	
1465	36	No	Travel_Frequently	884	Research & Development	23	
1466	39	No	Travel_Rarely	613	Research & Development	6	
1467	27	No	Travel_Rarely	155	Research & Development	4	
1468	49	No	Travel_Frequently	1023	Sales	2	
1469	34	No	Travel_Rarely	628	Research & Development	8	

1470 rows × 35 columns

Information and statistics about dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1470 non-null   int64 
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64 
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64 
 6   Education                 1470 non-null   int64 
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64 
```

```
 9   EmployeeNumber            1470 non-null   int64
 10  EnvironmentSatisfaction   1470 non-null   int64
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64
 13  JobInvolvement            1470 non-null   int64
 14  JobLevel                  1470 non-null   int64
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64
 19  MonthlyRate               1470 non-null   int64
 20  NumCompaniesWorked        1470 non-null   int64
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
 24  PerformanceRating         1470 non-null   int64
 25  RelationshipSatisfaction  1470 non-null   int64
 26  StandardHours             1470 non-null   int64
 27  StockOptionLevel          1470 non-null   int64
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

```
df.describe()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNu
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.00
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.86
std	9.135373	403.509100	8.106864	1.024165	0.0	602.02
min	18.000000	102.000000	1.000000	1.000000	1.0	1.00
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.25
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.50
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.75
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.00

8 rows × 26 columns

```
df.shape
```

```
(1470, 35)
```

Check if any null values present in the dataset

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

```
Age                         False
Attrition                   False
BusinessTravel              False
DailyRate                   False
Department                  False
DistanceFromHome            False
Education                   False
EducationField              False
EmployeeCount               False
EmployeeNumber              False
EnvironmentSatisfaction     False
Gender                      False
HourlyRate                  False
JobInvolvement              False
JobLevel                    False
JobRole                     False
JobSatisfaction             False
MaritalStatus               False
MonthlyIncome               False
MonthlyRate                 False
NumCompaniesWorked          False
```

```
Over18                    False
OverTime                  False
PercentSalaryHike         False
PerformanceRating         False
RelationshipSatisfaction  False
StandardHours             False
StockOptionLevel          False
TotalWorkingYears         False
TrainingTimesLastYear     False
WorkLifeBalance           False
YearsAtCompany            False
YearsInCurrentRole        False
YearsSinceLastPromotion   False
YearsWithCurrManager      False
dtype: bool
```

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

```
Age                       0
Attrition                 0
BusinessTravel            0
DailyRate                 0
Department                0
DistanceFromHome          0
Education                 0
EducationField            0
EmployeeCount             0
EmployeeNumber            0
EnvironmentSatisfaction   0
Gender                    0
HourlyRate                0
JobInvolvement            0
JobLevel                  0
JobRole                   0
JobSatisfaction           0
MaritalStatus             0
MonthlyIncome             0
MonthlyRate               0
NumCompaniesWorked        0
Over18                    0
OverTime                  0
PercentSalaryHike         0
PerformanceRating         0
RelationshipSatisfaction  0
StandardHours             0
StockOptionLevel          0
TotalWorkingYears         0
TrainingTimesLastYear     0
WorkLifeBalance           0
YearsAtCompany            0
YearsInCurrentRole        0
YearsSinceLastPromotion   0
YearsWithCurrManager      0
dtype: int64
```

** Data visualzation**

```
df.corr()
```

```
<ipython-input-11-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version
  df.corr()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	Hourly
Age	1.000000	0.010661	-0.001686	0.208034	NaN	-0.010145	0.010146	0.02
DailyRate	0.010661	1.000000	-0.004985	-0.016806	NaN	-0.050990	0.018355	0.02
DistanceFromHome	-0.001686	-0.004985	1.000000	0.021042	NaN	0.032916	-0.016075	0.03
Education	0.208034	-0.016806	0.021042	1.000000	NaN	0.042070	-0.027128	0.01
EmployeeCount	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
EmployeeNumber	-0.010145	-0.050990	0.032916	0.042070	NaN	1.000000	0.017621	0.03
EnvironmentSatisfaction	0.010146	0.018355	-0.016075	-0.027128	NaN	0.017621	1.000000	-0.04
HourlyRate	0.024287	0.023381	0.031131	0.016775	NaN	0.035179	-0.049857	1.00
JobInvolvement	0.029820	0.046135	0.008783	0.042438	NaN	-0.006888	-0.008278	0.04
JobLevel	0.509604	0.002966	0.005303	0.101589	NaN	-0.018519	0.001212	-0.02
JobSatisfaction	-0.004892	0.030571	-0.003669	-0.011296	NaN	-0.046247	-0.006784	-0.07
MonthlyIncome	0.497855	0.007707	-0.017014	0.094961	NaN	-0.014829	-0.006259	-0.01
MonthlyRate	0.028051	-0.032182	0.027473	-0.026084	NaN	0.012648	0.037600	-0.01
NumCompaniesWorked	0.299635	0.038153	-0.029251	0.126317	NaN	-0.001251	0.012594	0.02
PercentSalaryHike	0.003634	0.022704	0.040235	-0.011111	NaN	-0.012944	-0.031701	-0.00
PerformanceRating	0.001904	0.000473	0.027110	-0.024539	NaN	-0.020359	-0.029548	-0.00

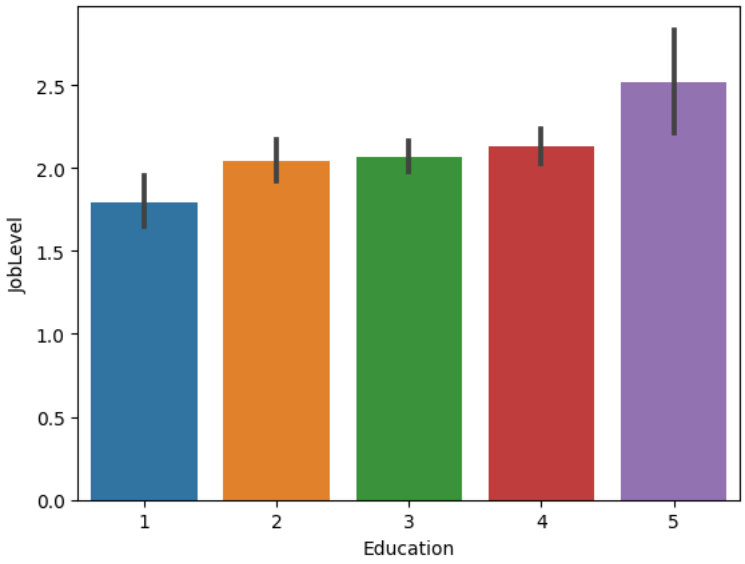
```
plt.figure(figsize=(25,15))
sns.heatmap(df.corr(),annot=True)
```

```
sns.countplot(x="Attrition",data=df)
```

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


```
sns.barplot(x="Education",y="JobLevel",data=df)
```

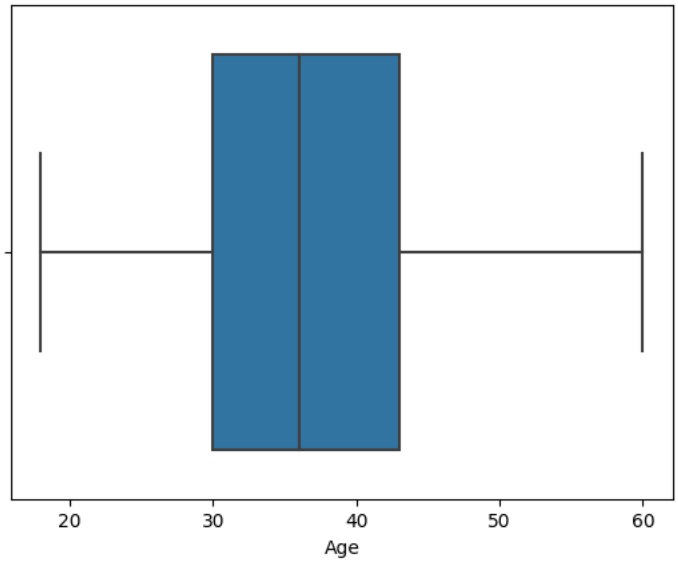
```
<Axes: xlabel='Education', ylabel='JobLevel'>
```



Outlier Detection

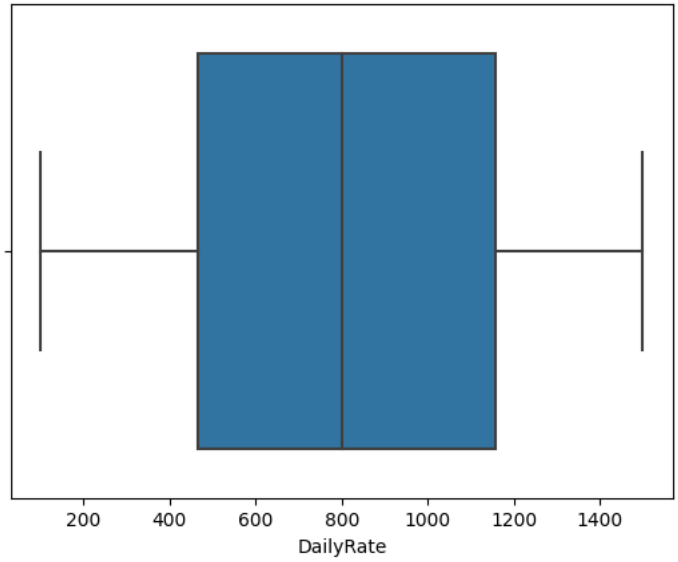
```
sns.boxplot(x="Age",data=df)
```

```
<Axes: xlabel='Age'>
```



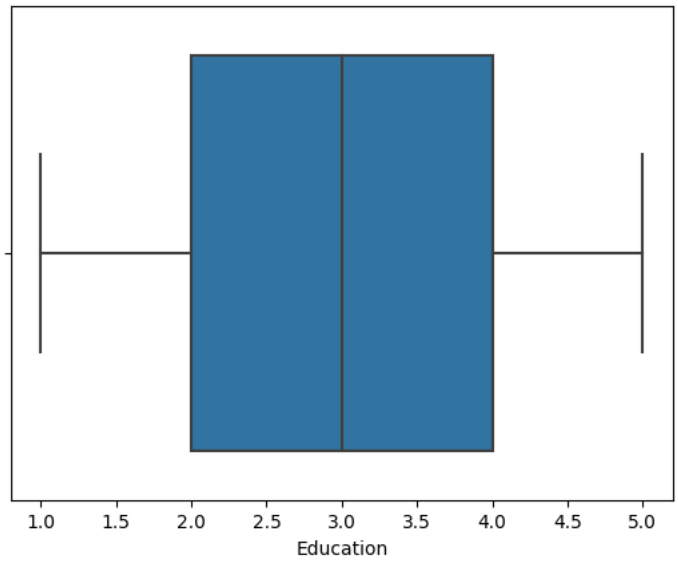

```
sns.boxplot(x="DailyRate",data=df)
```

```
<Axes: xlabel='DailyRate'>
```



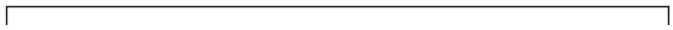

```
sns.boxplot(x="Education",data=df)
```

```
<Axes: xlabel='Education'>
```



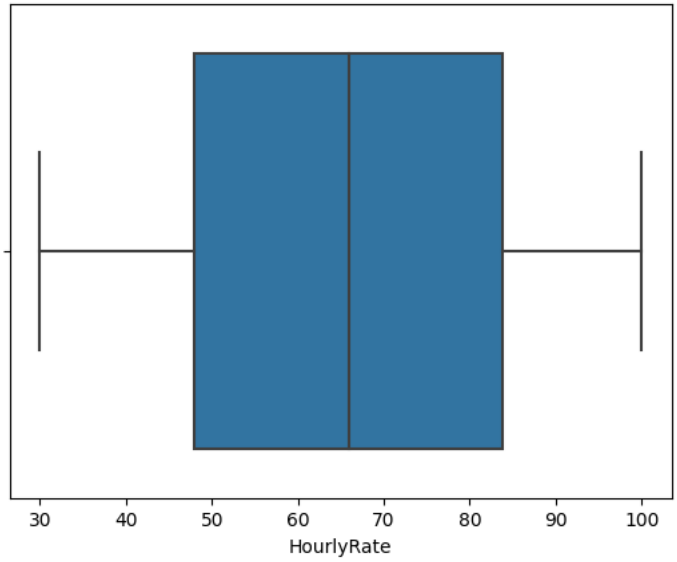

```
sns.boxplot(x="DistanceFromHome",data=df)
```

```
<Axes: xlabel='DistanceFromHome'>
```



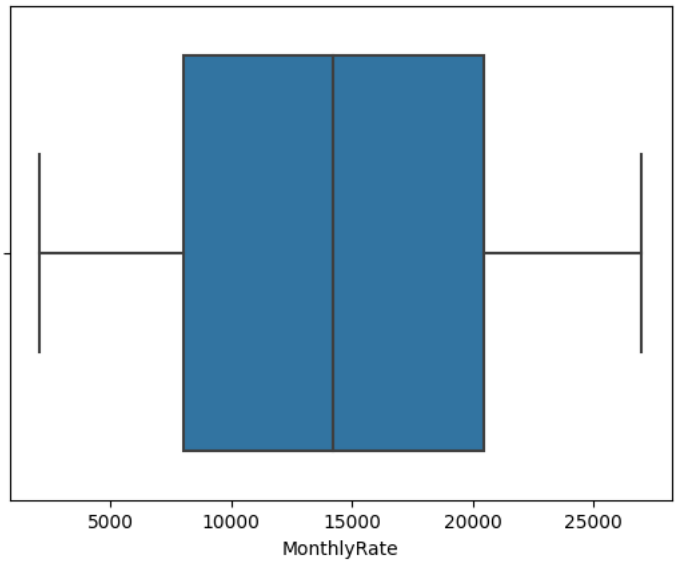
```
sns.boxplot(x="HourlyRate",data=df)
```

```
<Axes: xlabel='HourlyRate'>
```




```
sns.boxplot(x="MonthlyRate",data=df)
```

```
<Axes: xlabel='MonthlyRate'>
```




```
sns.boxplot(x="JobSatisfaction",data=df)
```

```
<Axes: xlabel='JobSatisfaction'>
```

There are no outliers in the data.

Splitting Dependent and Independent Variables

```
x=df[['Age','Gender','Department','EnvironmentSatisfaction','HourlyRate','JobSatisfaction','MonthlyIncome','WorkLifeBalance','YearsAtCompany'
```

```
x.head(10)
```

	Age	Gender	Department	EnvironmentSatisfaction	HourlyRate	JobSatisfaction	MonthlyIncome	WorkLifeBalance	YearsAtCompa
0	41	Female	Sales	2	94	4	5993	1	
1	49	Male	Research & Development	3	61	2	5130	3	
2	37	Male	Research & Development	4	92	3	2090	3	
3	33	Female	Research & Development	4	56	3	2909	3	
4	27	Male	Research & Development	1	40	2	3468	3	
5	32	Male	Research & Development	4	79	4	3068	2	
6	59	Female	Research & Development	3	81	1	2670	2	
7	30	Male	Research & Development	4	67	3	2693	3	
8	38	Male	Research & Development	4	44	3	9526	3	
9	36	Male	Research & Development	3	94	3	5237	2	

```
x.shape
```

```
(1470, 9)
```

```
y=df.Attrition
```

```
y.head(10)
```

```
0    Yes
1     No
2    Yes
3     No
4     No
5     No
6     No
7     No
8     No
9     No
Name: Attrition, dtype: object
```

```
y.shape
```

```
(1470,)
```

Encoding

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

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

```
<ipython-input-110-5f2403693ec8>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  x["Gender"] = le.fit_transform(x["Gender"])
```

```
x
```

	Age	Gender	Department	EnvironmentSatisfaction	HourlyRate	JobSatisfaction	MonthlyIncome	WorkLifeBalance	YearsAtCompany
0	41	0	2	2	94	4	5993	1	6
1	49	1	1	3	61	2	5130	3	10
2	37	1	1	4	92	3	2090	3	0
3	33	0	1	4	56	3	2909	3	8
4	27	1	1	1	40	2	3468	3	2
...	...	...	...	...	...	...	...	...	...
1465	36	1	1	3	41	4	2571	3	5
1466	39	1	1	4	42	1	9991	3	7
1467	27	1	1	2	87	2	6142	3	6
1468	49	1	2	4	63	2	5390	2	9
1469	34	1	1	2	82	3	4404	4	4

1470 rows × 9 columns

```
print(le.classes_)
```

```
['Female' 'Male']
```

```
le1=LabelEncoder()
```

```
x["Department"] = le1.fit_transform(x["Department"])
x.head()
```

```
<ipython-input-114-9d9230f200b4>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  x["Department"] = le1.fit_transform(x["Department"])
```

	Age	Gender	Department	EnvironmentSatisfaction	HourlyRate	JobSatisfaction	MonthlyIncome	WorkLifeBalance	YearsAtCompany
0	41	0	2	2	94	4	5993	1	6
1	49	1	1	3	61	2	5130	3	10
2	37	1	1	4	92	3	2090	3	0
3	33	0	1	4	56	3	2909	3	8
4	27	1	1	1	40	2	3468	3	2

```
dict(zip(le1.classes_,range(len(le1.classes_))))
```

```
{0: 0, 1: 1, 2: 2}
```

```
x.shape
```

```
(1470, 9)
```

Feature Scaling

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

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

```
x_scaled
```

	Age	Gender	Department	EnvironmentSatisfaction	HourlyRate	JobSatisfaction	MonthlyIncome	WorkLifeBalance	YearsAtCompany
0	0.547619	0.0	1.0	0.333333	0.914286	1.000000	0.262454	0.000000	0.150
1	0.738095	1.0	0.5	0.666667	0.442857	0.333333	0.217009	0.666667	0.250
2	0.452381	1.0	0.5	1.000000	0.885714	0.666667	0.056925	0.666667	0.000
3	0.357143	0.0	0.5	1.000000	0.371429	0.666667	0.100053	0.666667	0.200
4	0.214286	1.0	0.5	0.000000	0.142857	0.333333	0.129489	0.666667	0.050
...	...	...	...	...	...	...	...	...	...
1465	0.428571	1.0	0.5	0.666667	0.157143	1.000000	0.082254	0.666667	0.125
1466	0.500000	1.0	0.5	1.000000	0.171429	0.000000	0.472986	0.666667	0.175
1467	0.214286	1.0	0.5	0.333333	0.814286	0.333333	0.270300	0.666667	0.150
1468	0.738095	1.0	1.0	1.000000	0.471429	0.333333	0.230700	0.333333	0.225
1469	0.380952	1.0	0.5	0.333333	0.742857	0.666667	0.178778	1.000000	0.100

1470 rows × 9 columns

Splitting data into test and train

```
from sklearn.model_selection import train_test_split
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, 9), (441, 9), (1029,), (441,))
```

```
X_train.head()
```

	Age	Gender	Department	EnvironmentSatisfaction	HourlyRate	JobSatisfaction	MonthlyIncome	WorkLifeBalance	YearsAtCompany
338	30	0	2	4	30	3	6118	3	10
363	33	0	2	4	34	3	2851	3	1
759	45	1	0	2	36	2	2177	3	6
793	28	1	1	1	50	3	2207	2	4
581	30	1	1	4	38	3	3833	3	2

Model Building

LOGISTIC REGRESSION

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

```
model.fit(X_train,Y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
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
```

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

```
pred
```

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

```
Y_test
```

```
442      No
1091     No
981     Yes
785      No
1332    Yes
       ...
817      No
399      No
458      No
406      No
590      No
Name: Attrition, Length: 441, dtype: object
```

```
x
```

	Age	Gender	Department	EnvironmentSatisfaction	HourlyRate	JobSatisfaction	MonthlyIncome	WorkLifeBalance	YearsAtCompany
0	41	0	2	2	94	4	5993	1	6
1	49	1	1	3	61	2	5130	3	10
2	37	1	1	4	92	3	2090	3	0
3	33	0	1	4	56	3	2909	3	8
4	27	1	1	1	40	2	3468	3	2
...	...	...	...	...	...	...	...	...	...
1465	36	1	1	3	41	4	2571	3	5
1466	39	1	1	4	42	1	9991	3	7

```
model.predict(ms.transform([[49,2,1,3,50,2,6130,3,10]]))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but MinMaxScaler was fitt
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LogisticRegression wa
  warnings.warn(
array(['Yes'], dtype=object)
```

Evalution Of Classification Model

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

```
accuracy_score(Y_test,pred)
```

```
0.8412698412698413
```

```
confusion_matrix(Y_test,pred)
```

```
array([[371,   0],
       [ 70,   0]])
```

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

col_0	No
Attrition	
No	371
Yes	70

Roc-AUC curve for Logistic Regression

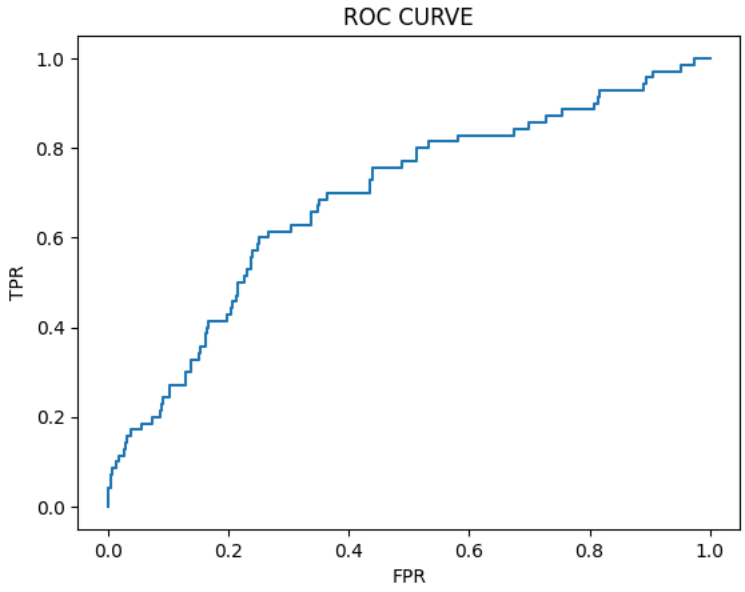
```
probability=model.predict_proba(X_test)[:,1]
probability
```

```
array([0.13999482, 0.13510158, 0.1921227 , 0.07945617, 0.15875883,
       0.17834534, 0.22894027, 0.17975852, 0.0816263 , 0.12894142,
       0.02625   , 0.19346052, 0.1099566 , 0.25687042, 0.11080215,
       0.12536022, 0.1425527 , 0.16104372, 0.06978485, 0.17773204,
       0.28786116, 0.02713025, 0.11553851, 0.12937922, 0.23766872,
       0.18435573, 0.16092767, 0.1004844 , 0.42077291, 0.22029387,
       0.05635244, 0.08639324, 0.11341627, 0.07143224, 0.06133954,
       0.07935575, 0.0727648 , 0.11656844, 0.09683589, 0.14170519,
       0.05094741, 0.13565384, 0.0416719 , 0.16134199, 0.14019893,
       0.22327038, 0.2931985 , 0.11283951, 0.30708606, 0.23919713,
       0.05897769, 0.33105969, 0.23612045, 0.31001386, 0.24284322,
       0.0982353 , 0.11300746, 0.19606216, 0.08591789, 0.31824832,
       0.07390978, 0.36673268, 0.05828606, 0.16660581, 0.11448615,
       0.07052734, 0.14451697, 0.14454887, 0.21802454, 0.1345294 ,
       0.08622783, 0.18907035, 0.15998653, 0.14163856, 0.18257291,
       0.13997776, 0.19145035, 0.02935521, 0.21334342, 0.210938  ,
       0.03467244, 0.20601926, 0.37109332, 0.09151443, 0.07788364,
       0.24828662, 0.01650076, 0.22441584, 0.12488893, 0.15502913,
       0.31510813, 0.13901478, 0.29153704, 0.29253313, 0.04715788,
       0.26669387, 0.19630646, 0.15881301, 0.19187441, 0.1221024 ,
```

```
0.45577361, 0.20405161, 0.01512562, 0.15070495, 0.17753412,
0.10330058, 0.09697001, 0.31830702, 0.2552207 , 0.08297351,
0.2057899 , 0.01040383, 0.0737379 , 0.11431993, 0.17018445,
0.09901762, 0.08552111, 0.06292763, 0.05003386, 0.08871715,
0.12747644, 0.17116197, 0.24116762, 0.18919257, 0.15702442,
0.06884623, 0.05802877, 0.24963821, 0.11314453, 0.04959177,
0.04662589, 0.23285363, 0.16131207, 0.14469419, 0.06005167,
0.14184321, 0.22191068, 0.14757936, 0.27954017, 0.21916053,
0.05079891, 0.10507781, 0.10114028, 0.09215138, 0.30986573,
0.19320262, 0.03939071, 0.01378995, 0.08197973, 0.35044325,
0.14932747, 0.09864343, 0.18785562, 0.04784092, 0.07881059,
0.110635  , 0.02049974, 0.25525038, 0.19591951, 0.1387285 ,
0.08836309, 0.16656477, 0.24516129, 0.29224717, 0.07253421,
0.11021663, 0.13287916, 0.21831026, 0.21483336, 0.0212246 ,
0.05312628, 0.1462309 , 0.14348759, 0.25549958, 0.0173791 ,
0.16125078, 0.15407691, 0.04999541, 0.19967622, 0.05353541,
0.14687622, 0.0829353 , 0.03162126, 0.26786194, 0.09041993,
0.11196688, 0.33312713, 0.39186162, 0.16356838, 0.12671923,
0.19812856, 0.2559796 , 0.11243345, 0.14635096, 0.13627759,
0.13885117, 0.13633964, 0.04296887, 0.1707587 , 0.12149756,
0.29587996, 0.34010533, 0.46374588, 0.15077557, 0.16605278,
0.13987893, 0.23790641, 0.02869572, 0.21614303, 0.07006171,
0.18153057, 0.29257269, 0.17949356, 0.37165878, 0.13341095,
0.07871589, 0.05089413, 0.10433459, 0.02175516, 0.37188397,
0.2332626 , 0.35519468, 0.23128027, 0.0163058 , 0.34708024,
0.11137203, 0.1118143 , 0.16288471, 0.40980612, 0.14680036,
0.01474042, 0.10039386, 0.18780139, 0.07931762, 0.15092164,
0.11181745, 0.23668318, 0.10110865, 0.16747711, 0.01906994,
0.20296809, 0.3113231 , 0.04053303, 0.15083689, 0.1418575 ,
0.05488641, 0.12040902, 0.16760565, 0.15486252, 0.07657531,
0.24187002, 0.12348867, 0.39873316, 0.06743135, 0.14829617,
0.27033343, 0.00860165, 0.20511307, 0.19126526, 0.22357328,
0.15265071, 0.25304601, 0.07880407, 0.1320606 , 0.16195514,
0.19712524, 0.13476191, 0.2239766 , 0.34098474, 0.12207818,
0.28414977, 0.09183016, 0.22245087, 0.13823236, 0.1285389 ,
0.11913034, 0.25505142, 0.1653026 , 0.09986998, 0.14338829,
0.1890425 , 0.14277232, 0.32542082, 0.10054703, 0.01041131,
```

```
from sklearn.preprocessing import LabelBinarizer
lb = LabelBinarizer()
Y_test_bin = lb.fit_transform(Y_test)
fpr, tpr, thresholds = roc_curve(Y_test_bin, probability)
```

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



Decision Tree

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

```
dt.fit(X_train,Y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
pre=dt.predict(X_test)
pre
```

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

```
Y_test
```

```
442      No
1091     No
981     Yes
785      No
1332    Yes
       ...
817      No
399      No
458      No
406      No
590      No
Name: Attrition, Length: 441, dtype: object
```

```
dt.predict(ms.transform([[49,2,1,3,50,2,6130,3,10]]))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but MinMaxScaler was fitt
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionTreeClassifie
  warnings.warn(
array(['Yes'], dtype=object)
```

```
accuracy_score(Y_test,pre)
```

```
0.764172335600907
```

```
classification_report(Y_test,pre)
```

```
'              precision    recall  f1-score   support\n\n          No       0.85      0.87      0.86       371\n         Yes       0.23
acro avg       0.54      0.54      0.54       441\nweighted avg       0.75      0.76      0.76       441\n'
```

```
pd.crosstab(Y_test,pre)
```

col_0	No	Yes
Attrition		
No	323	48
Yes	56	14

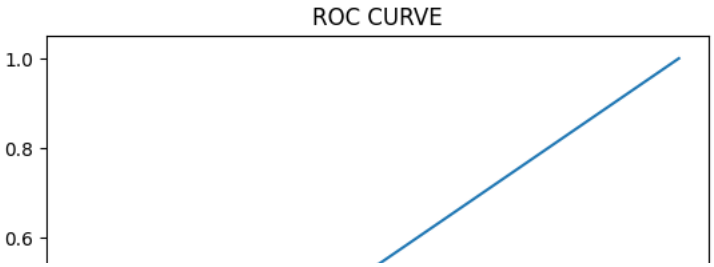
```
prob=dt.predict_proba(X_test)[:,1]
prob
```

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

ROC - CURVE for Decision tree

```
Y_test_bin2 = lb.fit_transform(Y_test)
fpr1,tpr1,threshsholds = roc_curve(Y_test_bin2,prob)
```

```
plt.plot(fpr1,tpr1)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC CURVE')
plt.show()
```



Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier(random_state=50)
```

```
rfc.fit(X_train, Y_train)
```

```
RandomForestClassifier(random_state=50)
```

```
rfc_predictions=rfc.predict(X_test)
```

```
accuracy_score(Y_test, rfc_predictions)
```

```
0.8435374149659864
```

```
classification_report(Y_test, rfc_predictions)
```

```
'              precision    recall  f1-score   support\n\n          No       0.86      0.98      0.91       371\n         Yes       0.53
acro avg       0.69      0.55      0.56       441\nweighted avg       0.80      0.84      0.80       441\n'
```

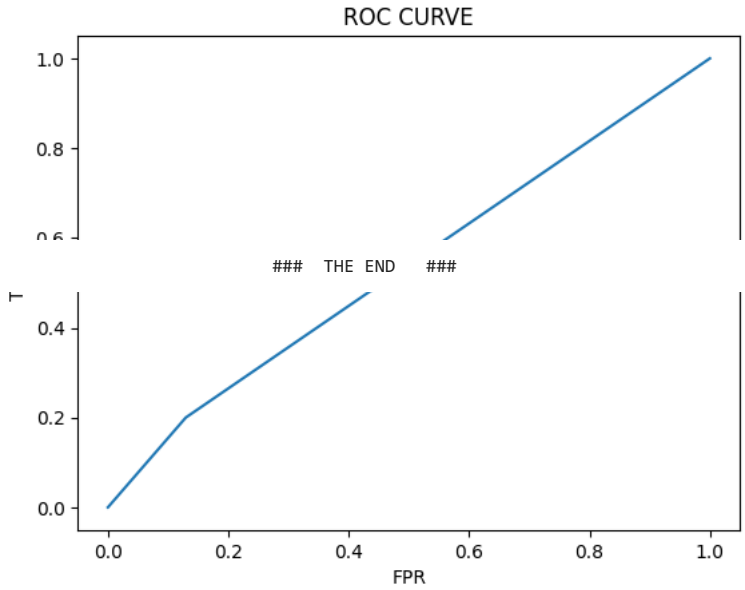
```
pd.crosstab(Y_test,rfc_predictions)
```

col_0	No	Yes
Attrition		
No	363	8
Yes	61	9

Roc - Auc curve for Random Forest Classifier

```
Y_test_bin3 = lb.fit_transform(Y_test)
fpr2,tpr2,threshsholds = roc_curve(Y_test_bin3,prob)
```

```
plt.plot(fpr2,tpr2)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC CURVE')
plt.show()
```


ROC CURVE
1.0
0.8
0.6
THE END
T
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
FPR