

NAME: CHILUKURI NAGA VARDHAN

- REG NO: 21BCE7773
- CAMPUS: VIT-AP
- Assignment 4 on sept 22
- Morning Slot (10-12 am)
- Google colab Link: <https://colab.research.google.com/drive/1tt9UMfpygjGgjEJ-yCqAafaVca73oGps?usp=sharing>

Import necessary libraries

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

Import dataset

```
from google.colab import files
uploaded = files.upload()
```

Employee_Attrition.csv

- **Employee_Attrition.csv**(text/csv) - 227977 bytes, last modified: 9/28/2023 - 100% done

Saving Employee_Attrition.csv to Employee_Attrition.csv

```
df=pd.read_csv("/content/Employee_Attrition.csv")
```

```
df
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7
...
1465	36	No	Travel_Frequently	884	Research & Development	23	2	Medical	1	2061
1466	39	No	Travel_Rarely	613	Research & Development	6	1	Medical	1	2062
1467	27	No	Travel_Rarely	155	Research & Development	4	3	Life Sciences	1	2064
1468	49	No	Travel_Frequently	1023	Sales	2	3	Medical	1	2065
1469	34	No	Travel_Rarely	628	Research & Development	8	3	Medical	1	2068

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	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobIn
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000	14
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156	
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428	
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000	
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000	
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000	
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000	
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000	

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
```

**** Data visualization****

```
df.corr()
```

```
<ipython-input-9-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version,
df.corr()
```

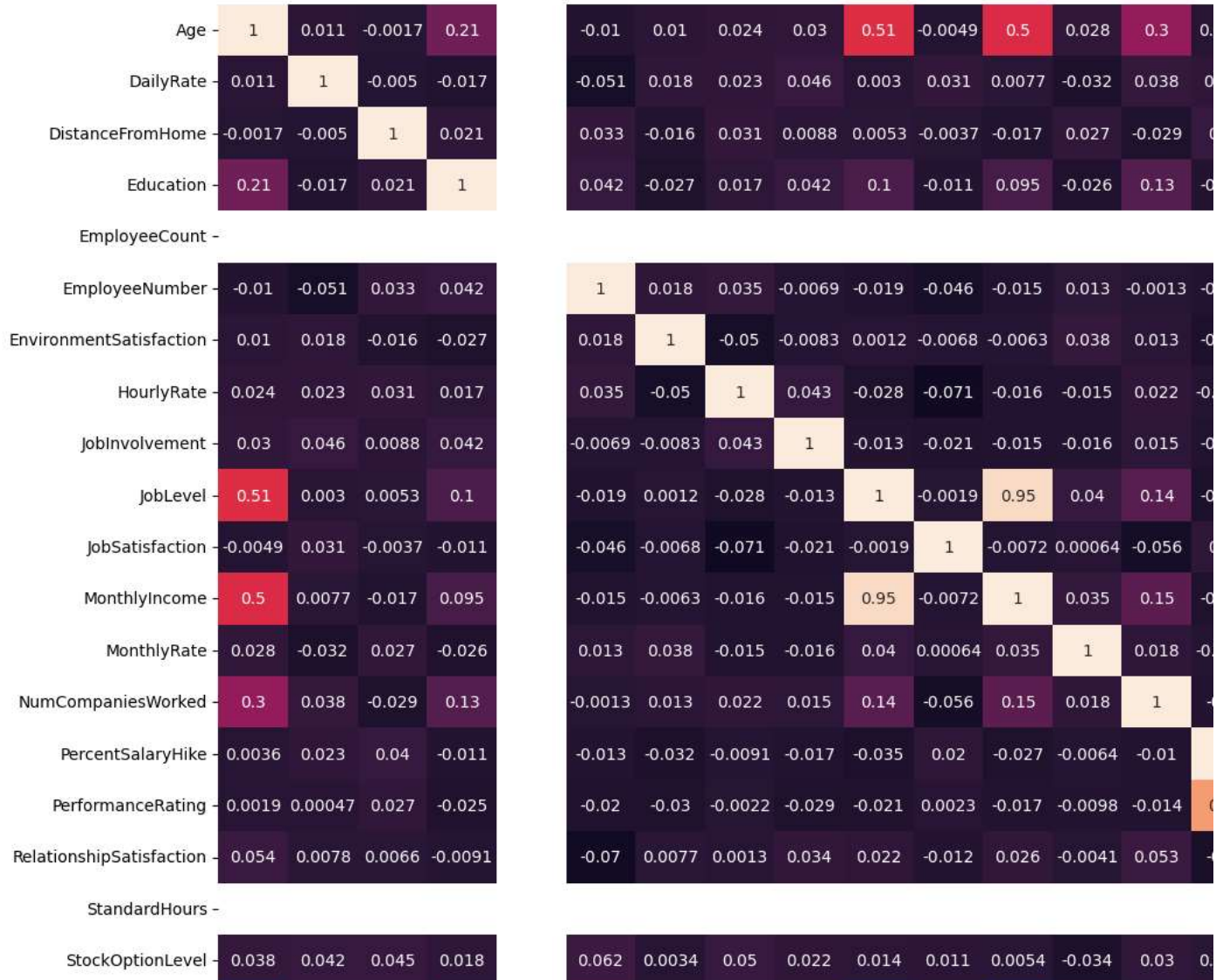
	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	Hourly
Age	1.000000	0.010661	-0.001686	0.208034	NaN	-0.010145	0.010146	0.02
DailyRate	0.010661	1.000000	-0.004985	-0.016806	NaN	-0.050990	0.018355	0.02
DistanceFromHome	-0.001686	-0.004985	1.000000	0.021042	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
RelationshipSatisfaction	0.053535	0.007846	0.006557	-0.009118	NaN	-0.069861	0.007665	0.00
StandardHours	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
StockOptionLevel	0.037510	0.042143	0.044872	0.018422	NaN	0.062227	0.003432	0.05
TotalWorkingYears	0.680381	0.014515	0.004628	0.148280	NaN	-0.014365	-0.002693	-0.00
TrainingTimesLastYear	-0.019621	0.002453	-0.036942	-0.025100	NaN	0.023603	-0.019359	-0.00
WorkLifeBalance	-0.021490	-0.037848	-0.026556	0.009819	NaN	0.010309	0.027627	-0.00
YearsAtCompany	0.311309	-0.034055	0.009508	0.069114	NaN	-0.011240	0.001458	-0.01
YearsInCurrentRole	0.212901	0.009932	0.018845	0.060236	NaN	-0.008416	0.018007	-0.02
YearsSinceLastPromotion	0.216513	-0.033229	0.010029	0.054254	NaN	-0.009019	0.016194	-0.02
YearsWithCurrManager	0.202089	-0.026363	0.014406	0.069065	NaN	-0.009197	-0.004999	-0.02

26 rows × 26 columns

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

<ipython-input-10-d92c41bf2ee0>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version
 sns.heatmap(df.corr(),annot=True)

<Axes: >

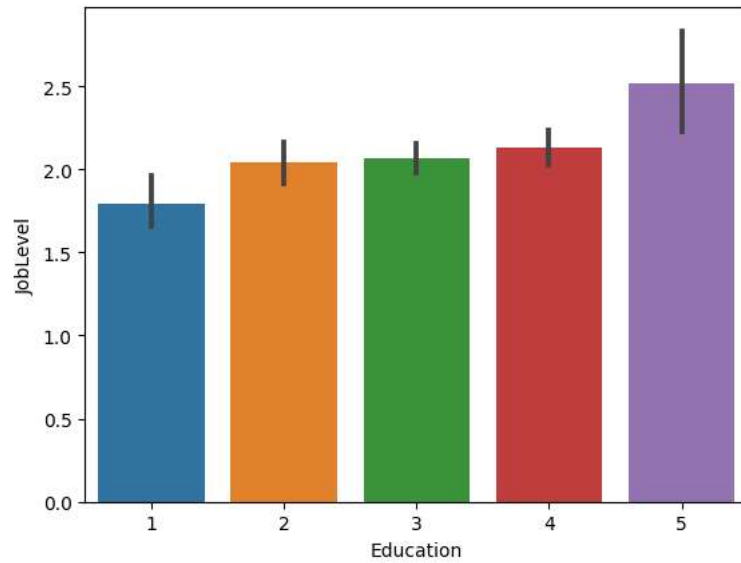


```
sns.countplot(x="Attrition",data=df)
```

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

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

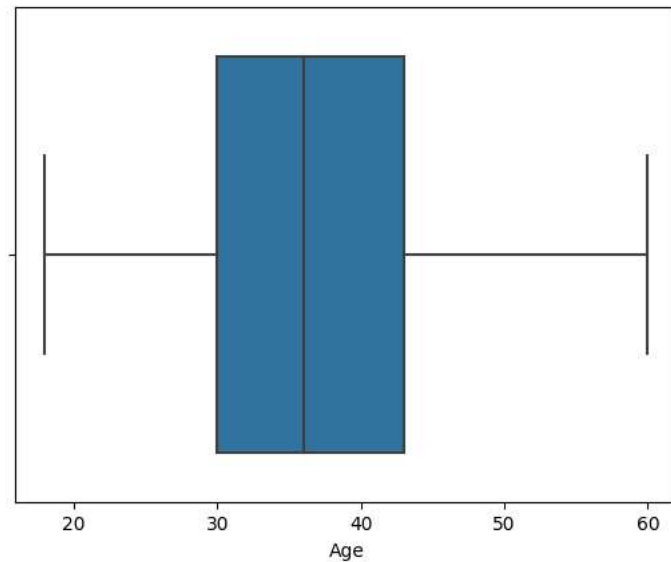
```
<Axes: xlabel='Education', ylabel='JobLevel'>
```



Outlier Detection

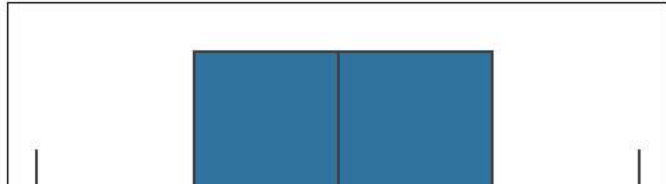
```
sns.boxplot(x="Age",data=df)
```

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



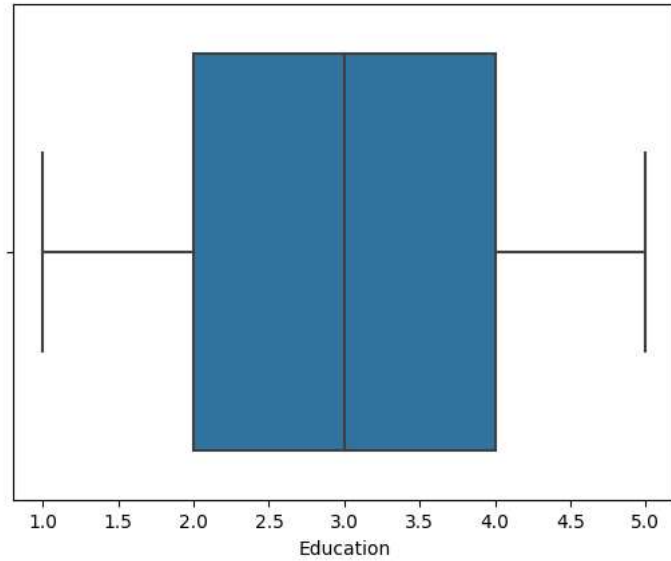
```
sns.boxplot(x="DailyRate",data=df)
```

<Axes: xlabel='DailyRate'>



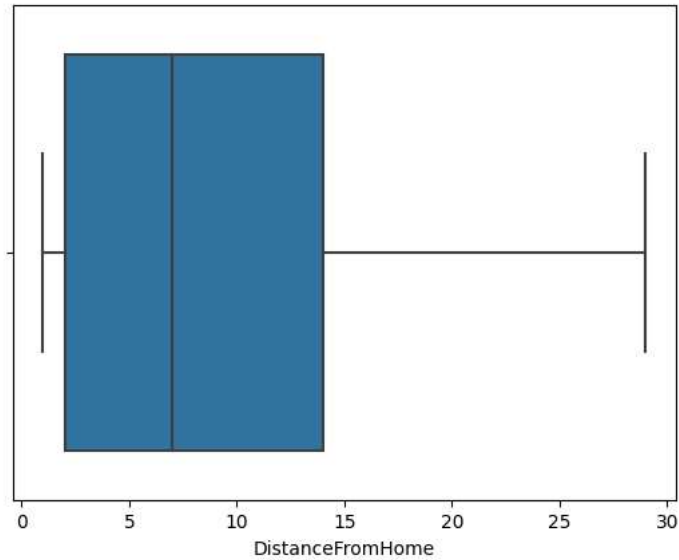
```
sns.boxplot(x="Education",data=df)
```

<Axes: xlabel='Education'>



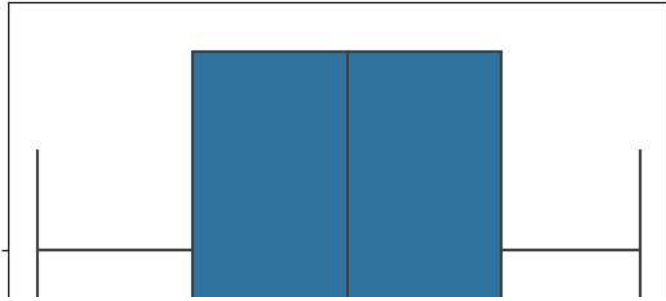
```
sns.boxplot(x="DistanceFromHome",data=df)
```

<Axes: xlabel='DistanceFromHome'>



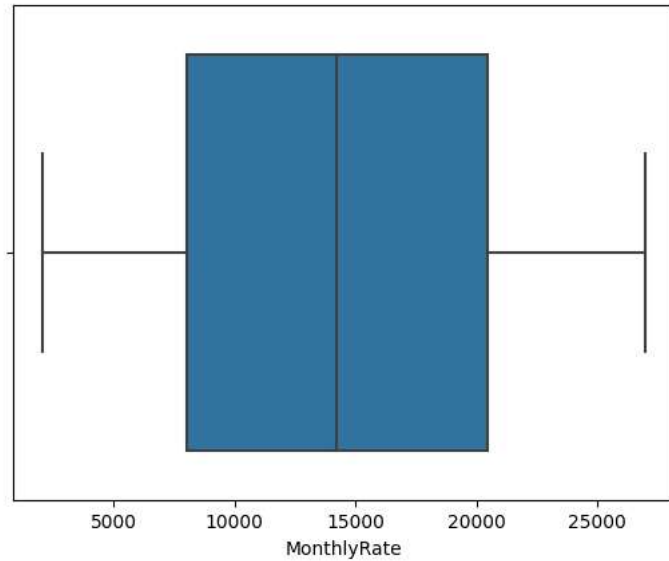
```
sns.boxplot(x="HourlyRate",data=df)
```

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



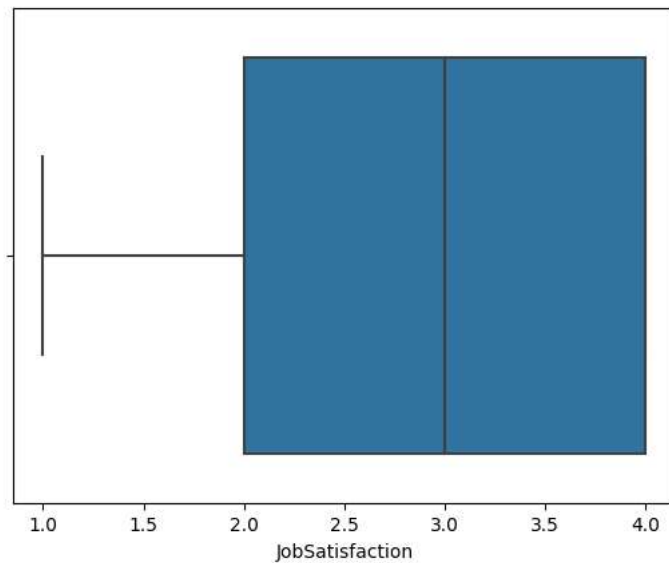
```
sns.boxplot(x="MonthlyRate",data=df)
```

```
<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']
```

```
y=df['Salary']
```

```
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-27-5f2403693ec8>: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-copy
x["Gender"] = le.fit_transform(x["Gender"])
```

```
x
```



	Age	Gender	Department	EnvironmentSatisfaction	HourlyRate	JobSatisfaction	Mo
0	41	0	Sales	2	94	4	
1	49	1	Research & Development	3	61	2	
2	37	1	Research & Development	4	92	3	
3	33	0	Research & Development	4	56	3	

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

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

```
le1=LabelEncoder()
```

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

<ipython-input-31-9d9230f200b4>: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>
x["Department"] = le1.fit_transform(x["Department"])

	Age	Gender	Department	EnvironmentSatisfaction	HourlyRate	JobSatisfaction	Month1
0	41	0	2	2	94	4	
1	49	1	1	3	61	2	
2	37	1	1	4	92	3	
3	33	0	1	4	56	3	

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

```
{'Human Resources': 0, 'Research & Development': 1, 'Sales': 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
```

10/16


```
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

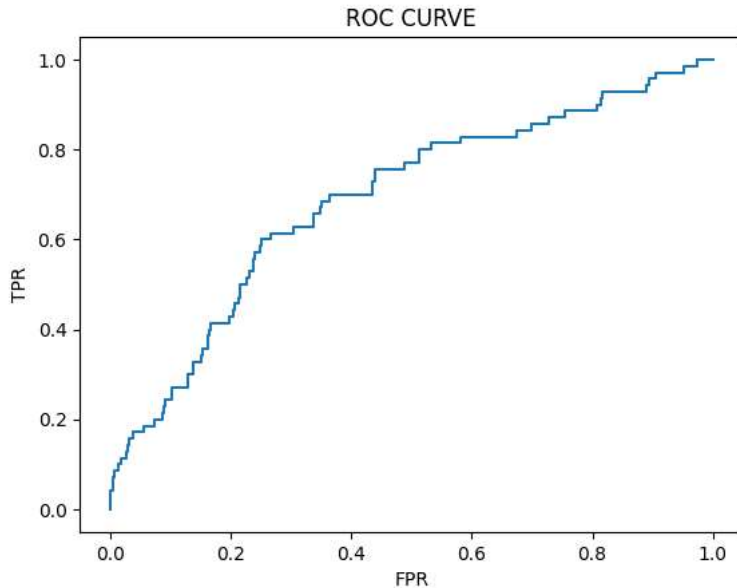
```
probability=model.predict_proba(X_test)[:,-1]
probability
```

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,

```
0.0/9/1346, 0.10846156, 0.208/622/, 0.18/91219, 0.062/5365,
0.04725651])
```

```
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', 'Yes', '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', 'Yes', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'Yes', 'Yes', 'No', 'No', 'No', 'No', 'No',
       'Yes', 'Yes', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No',
       'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No',
       'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No',
       'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No',
       'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No',
       'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No',
       'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'Yes',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'Yes', 'Yes', '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', 'Yes', 'Yes', 'No', 'No', 'Yes',
       'No', 'No', 'No', 'Yes', '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', 'Yes', 'No', 'No',
'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'Yes', 'Yes',
'No', 'No', 'No', 'Yes', 'No', 'No', 'Yes', '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', 'Yes',
'No', 'Yes', 'Yes', 'Yes', 'No', 'No', 'No', 'No', 'No', 'No',
'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes',
'No', 'No', 'Yes', 'No', 'Yes', 'No', 'No', 'Yes', 'No', 'Yes',
'No', 'No', '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', 'Yes', 'No', 'No', 'No', 'No',
'Yes', 'No', 'No', 'No', 'No', 'No', 'Yes', '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', '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 fitted
warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted
warnings.warn(
array(['Yes'], dtype=object)
```

```
accuracy_score(Y_test,pre)

0.746031746031746
```

```
classification_report(Y_test,pre)

              precision    recall  f1-score   support\n\n
0.85         0.85        0.37         0.58         441\n
accuracy          0.75         0.75         0.75         441\n
macro avg          0.85         0.21         0.53         441\n
weighted avg          0.85         0.37         0.58         441\n
```

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

col_0	No	Yes
Attrition		
No	314	57
Yes	55	15

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

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

```

ROC - CURVE for Decision tree

```

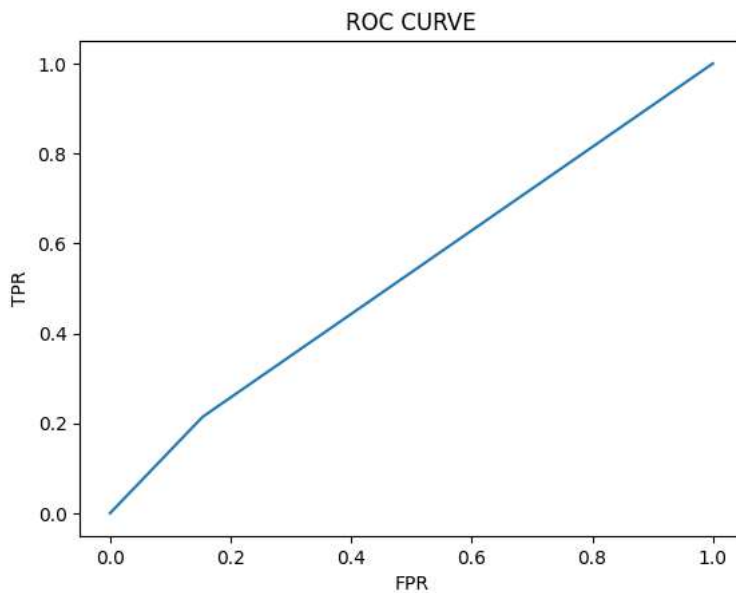
Y_test_bin2 = lb.fit_transform(Y_test)
fpr1,tpr1,thresholds = roc_curve(Y_test_bin2,prob)

```

```

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
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	No	Yes
0.98	0.91	0.371	0.53	0.13	0.21	0.86
accuracy			0.84	441	macro avg	0.69
0.56	0.441	weighted avg	0.80	0.84	0.80	0.55

```
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, thresholds = roc_curve(Y_test_bin3, prob)
```

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

