SMART BRIDGE AI AND ML -- ASSIGNMENT-3

1.Import the 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("WA_Fn-UseC_-HR-Employee-Attrition.csv")
```

```
df
```

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

1470 rows × 35 columns

```
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount
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

```
df.tail()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCoun
1465	36	No	Travel_Frequently	884	Research & Development	23	2	Medical	
1466	39	No	Travel_Rarely	613	Research & Development	6	1	Medical	
1467	27	No	Travel_Rarely	155	Research & Development	4	3	Life Sciences	
1468	49	No	Travel_Frequently	1023	Sales	2	3	Medical	
1469	34	No	Travel_Rarely	628	Research & Development	8	3	Medical	

5 rows × 35 columns

```
df.shape
```

```
(1470, 35)
```

```
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
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000

8 rows × 26 columns

3.Checking for 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
```

```
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["EducationField"].value_counts()
```

```
Life Sciences       606
Medical             464
Marketing           159
Technical Degree    132
Other                82
Human Resources      27
Name: EducationField, dtype: int64
```

```
df["JobRole"].value_counts()
```

```
Sales Executive              326
Research Scientist           292
Laboratory Technician        259
Manufacturing Director       145
Healthcare Representative    131
Manager                      102
Sales Representative          83
Research Director             80
Human Resources               52
Name: JobRole, dtype: int64
```

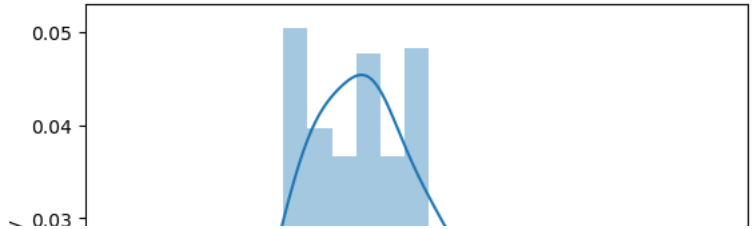
```
df["MaritalStatus"].value_counts()
```

```
Married     673
Single      470
Divorced    327
Name: MaritalStatus, dtype: int64
```

4.Data Visualization.

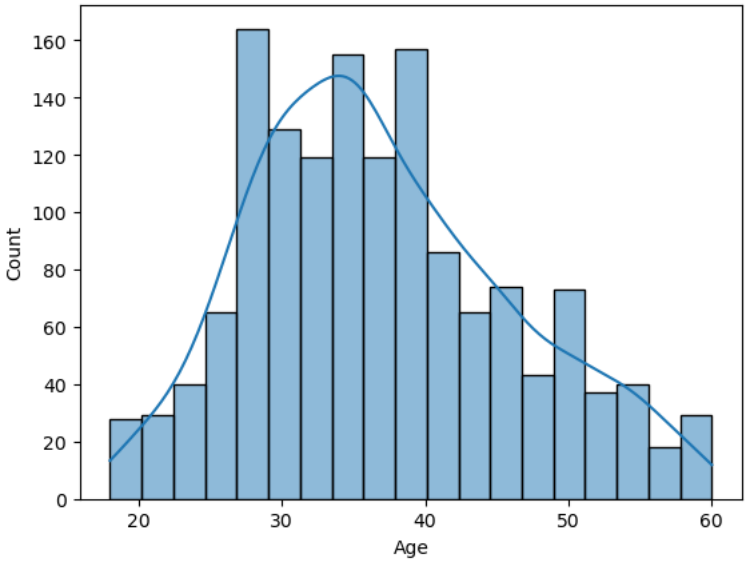
```
sns.distplot(df["Age"])
```

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_3668\2732350774.py:1: UserWarning:

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

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

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

  sns.distplot(df["Age"])
<Axes: xlabel='Age', ylabel='Density'>
```



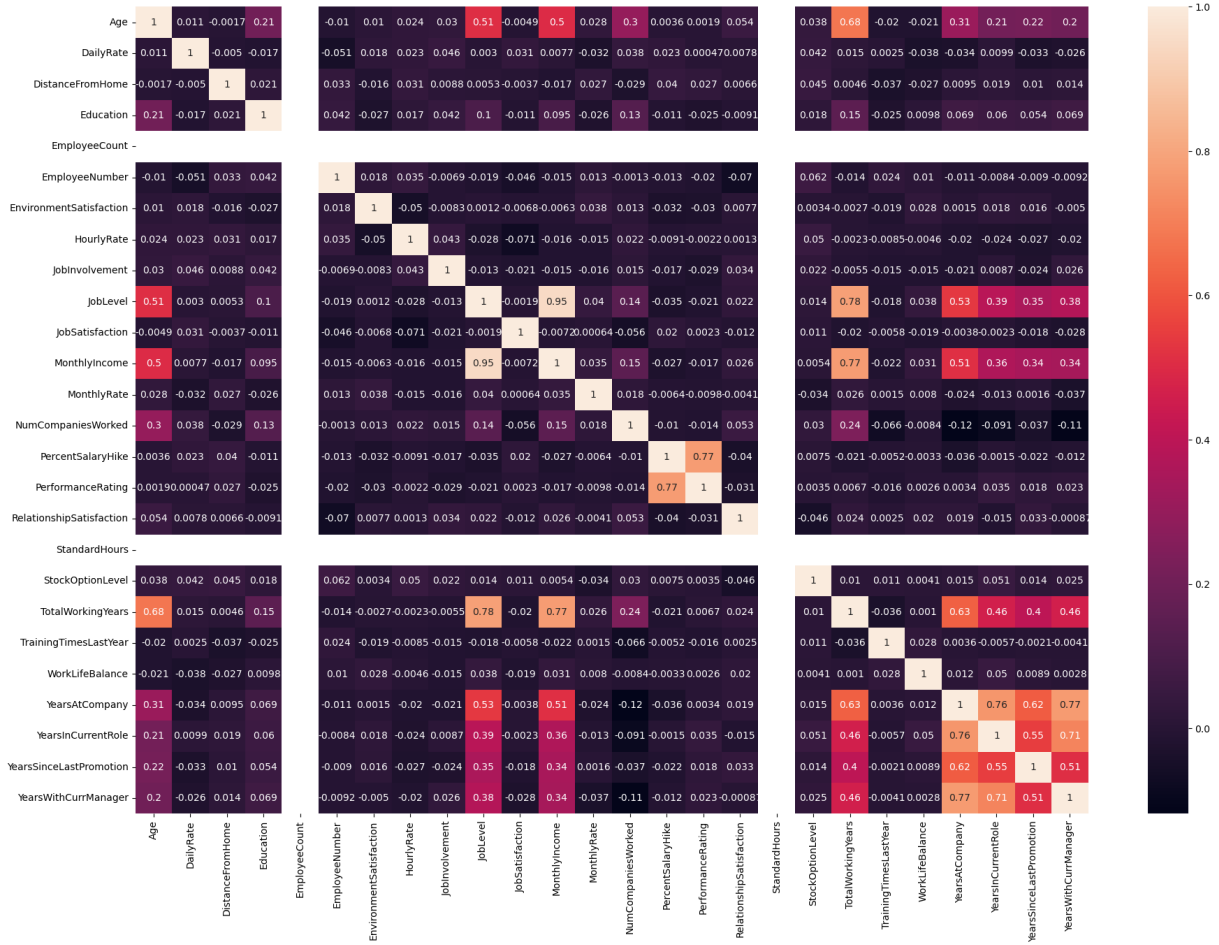

```
sns.histplot(x="Age",data=df,kde=True)
```

```
<Axes: xlabel='Age', ylabel='Count'>
```



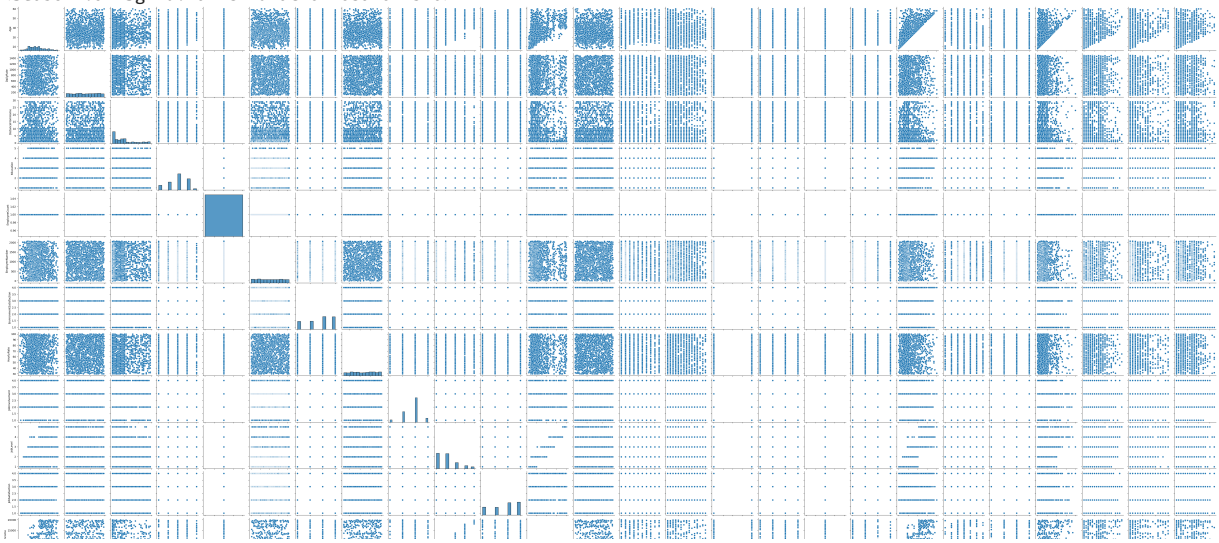

```
plt.subplots(figsize=(22,15))
sns.heatmap(df.corr(),annot=True)
```

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_3668\919060229.py:2: FutureWarning: The default value of numeric_only in Dat
  sns.heatmap(df.corr(),annot=True)
<Axes: >
```



```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x208af027b10>
```

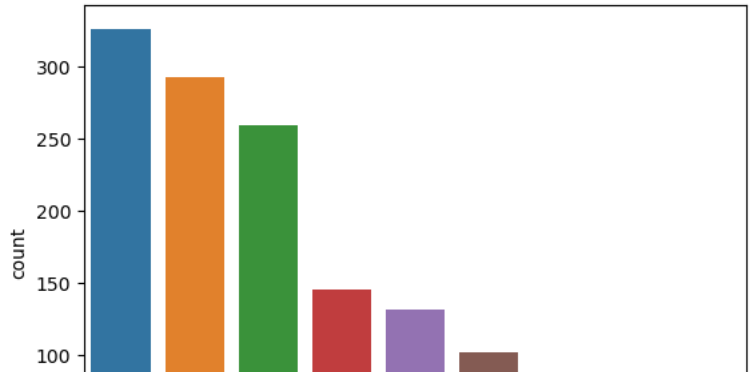


```
df.JobRole.value_counts().plot(kind="pie",autopct="%1.1f%%")
```

```
<Axes: ylabel='JobRole'>
```

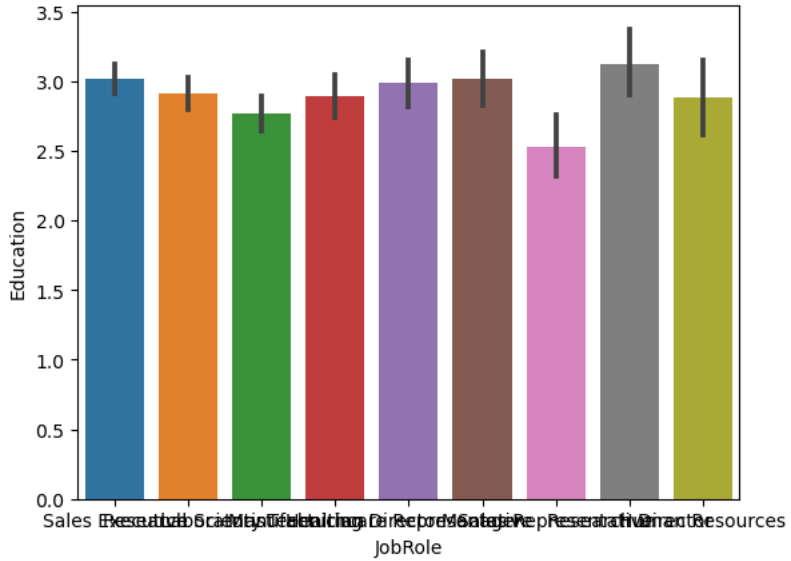

```
sns.countplot(x="JobRole",data=df)
```

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



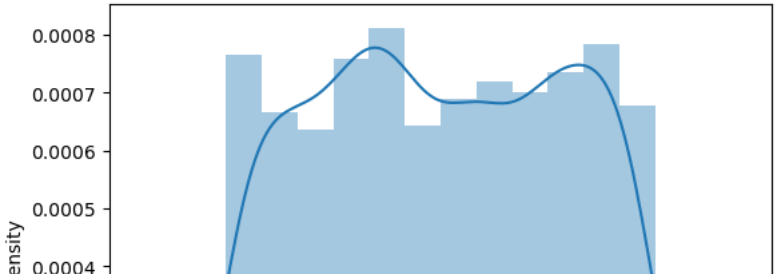

```
sns.barplot(x=df["JobRole"],y=df["Education"])
```

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



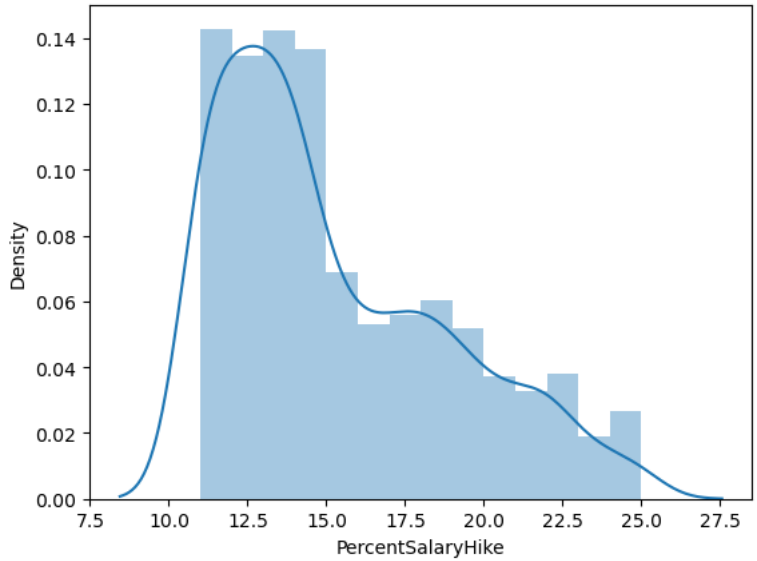

```
sns.distplot(df["DailyRate"])
```

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_3668\610149439.py:1: UserWarning:

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

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

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

  sns.distplot(df["DailyRate"])
<Axes: xlabel='DailyRate', ylabel='Density'>
```




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

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_3668\2990077479.py:1: UserWarning:

sns.distplot(df["PercentSalaryHike"])

C:\Users\Asus\AppData\Local\Temp\ipykernel_3668\3246365209.py:1: UserWarning:

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

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

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

  sns.distplot(df["PercentSalaryHike"])
<Axes: xlabel='PercentSalaryHike', ylabel='Density'>
```



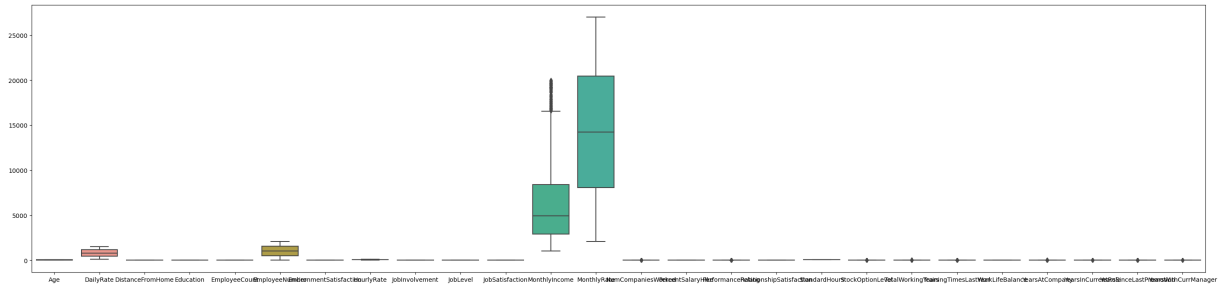
5.Outlier Detection

```
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1

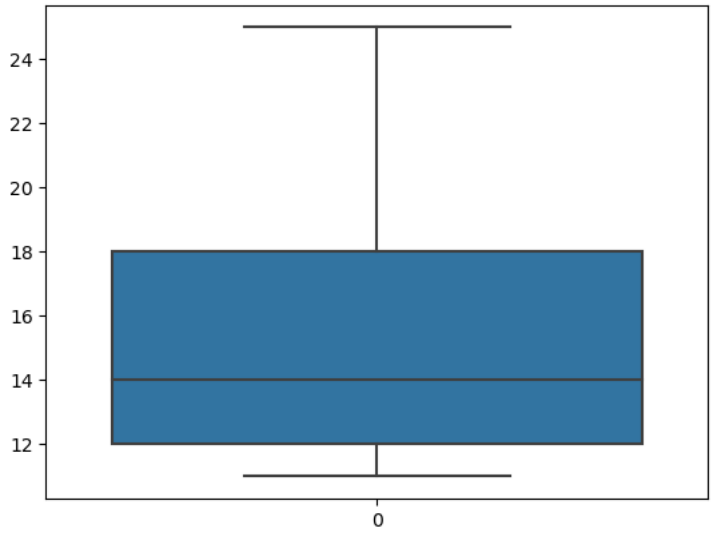
```
plt.figure(figsize=(35,8))
sns.boxplot(data=df)
```

```
<Axes: >
```



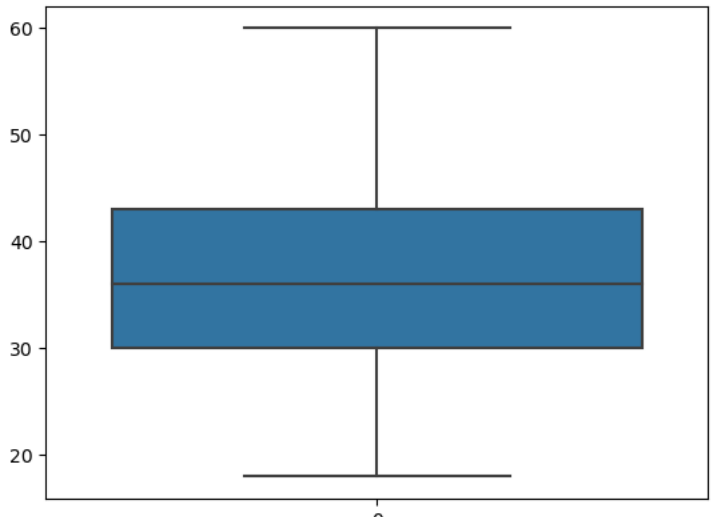
```
sns.boxplot(df.PercentSalaryHike)
```

```
<Axes: >
```



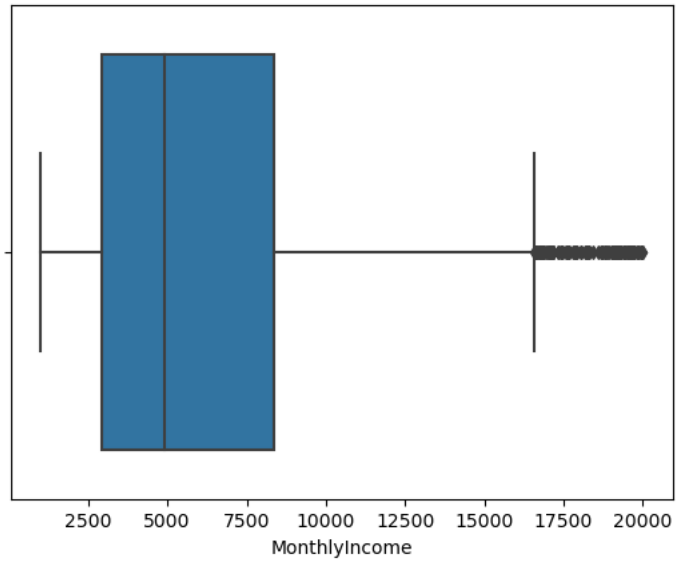

```
sns.boxplot(df.Age)
```

```
<Axes: >
```




```
from scipy import stats
z_scores = stats.zscore(df['MonthlyIncome'])
df_cleaned=df[(np.abs(z_scores)<=3)]


sns.boxplot(data=df_cleaned,x='MonthlyIncome')
```

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



6.Splitting Dependent and Independent variables

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

```
x.head()
```

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

5 rows × 34 columns

```
type(x)
```

```
pandas.core.frame.DataFrame
```

```
y=df["Attrition"]
```

```
y.head()
```

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

```
type(y)
```

```
pandas.core.series.Series
```

7.Encoding

```
#label Encoding
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

```
categorial_features=x.select_dtypes(include=['object']).columns.tolist()
x_encoded=pd.get_dummies(x,columns=categorial_features,drop_first=True)
```

```
x_encoded.head()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	Job
0	41	1102	1	2	1	1	2	94	
1	49	279	8	1	1	2	3	61	
2	37	1373	2	2	1	4	4	92	
3	33	1392	3	4	1	5	4	56	
4	27	591	2	1	1	7	1	40	

5 rows × 47 columns

8.Feature Scaling

```
from sklearn.preprocessing import StandardScaler
s=StandardScaler()
x_scaled=pd.DataFrame(s.fit_transform(x_encoded),columns=x_encoded.columns)
```

```
x_scaled.head()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate
0	0.446350	0.742527	-1.010909	-0.891688	0.0	-1.701283	-0.660531	1.383138
1	1.322365	-1.297775	-0.147150	-1.868426	0.0	-1.699621	0.254625	-0.240677
2	0.008343	1.414363	-0.887515	-0.891688	0.0	-1.696298	1.169781	1.284725
3	-0.429664	1.461466	-0.764121	1.061787	0.0	-1.694636	1.169781	-0.486709
4	-1.086676	-0.524295	-0.887515	-1.868426	0.0	-1.691313	-1.575686	-1.274014

5 rows × 47 columns

9.Splitting Data into Train and Test.

```
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)
```

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

```
((1176, 47), (294, 47), (1176,), (294,))
```

Model Building

Logistic Regression

1.Import the Libraries

```
from sklearn.linear_model import LogisticRegression
```

2.Initialize the model

```
lr=LogisticRegression()
```

3.Training and testing the model

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

```
LogisticRegression()
```

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

```
y_pred
```

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

```
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount
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

4.Evaluation of Model

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

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

```
0.8775510204081632
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[238,   7],
       [ 29,  20]], dtype=int64)
```

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

col_0	No	Yes
Attrition		

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

```
                  precision    recall  f1-score   support

              No       0.89      0.97      0.93       245
             Yes       0.74      0.41      0.53        49

        accuracy                           0.88       294
       macro avg       0.82      0.69      0.73       294
    weighted avg       0.87      0.88      0.86       294
```

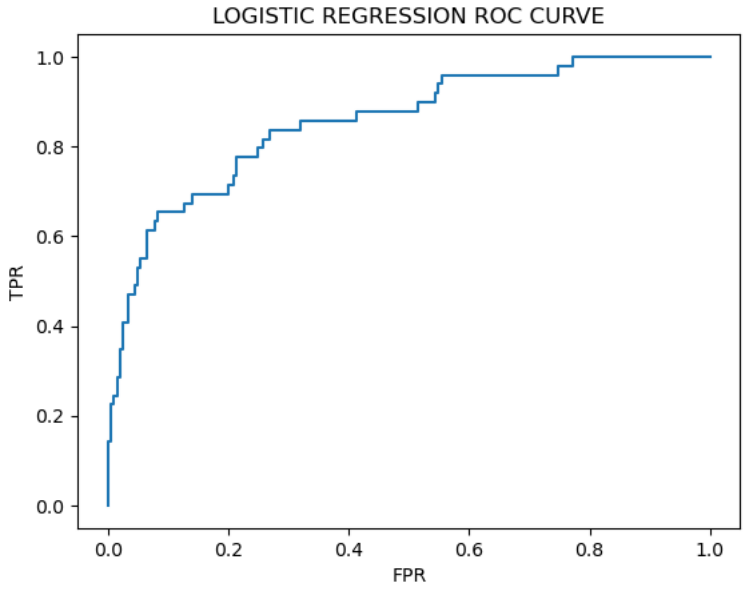
```
probability=lr.predict_proba(x_test)[:,1]
```

```
probability
```

```
array([6.61367680e-02, 8.06198087e-02, 5.95640954e-01, 1.42633628e-01,
       7.84184536e-01, 4.42409704e-02, 4.90141163e-01, 3.64319554e-02,
       8.44925839e-04, 4.30164151e-01, 3.70048030e-02, 1.96283567e-01,
       1.51964558e-02, 5.39471444e-01, 5.88816107e-02, 1.18804547e-02,
       1.13489514e-01, 3.82574810e-02, 2.32011787e-02, 2.66149260e-01,
       1.31568762e-01, 9.99913247e-03, 1.30286201e-02, 3.88975456e-02,
       7.51582081e-01, 4.58920880e-01, 6.43869830e-02, 2.97280577e-02,
       6.90209855e-01, 3.77607239e-02, 5.59306642e-03, 3.05902665e-01,
       3.65028773e-02, 3.90288488e-02, 2.27899092e-02, 5.90358760e-03,
       2.32731579e-01, 5.29386349e-02, 1.38241033e-02, 1.13353905e-01,
       3.42777217e-02, 8.33820691e-03, 1.07151981e-03, 5.52578825e-03,
       8.80282261e-03, 5.59254552e-01, 4.21978392e-01, 5.99712843e-04,
       4.00447399e-01, 3.68527002e-01, 3.55627577e-02, 8.16716675e-01,
       1.58976261e-02, 3.55101528e-01, 4.94410470e-01, 2.54453615e-01,
       1.03880530e-02, 4.22173742e-01, 2.22235299e-02, 2.95631863e-01,
       1.26998325e-02, 1.53660325e-01, 1.65718568e-01, 2.37981819e-02,
       1.13922041e-01, 2.33156457e-02, 3.48112915e-01, 1.78088100e-01,
       1.21220858e-01, 1.82213934e-01, 3.38118048e-02, 1.58551253e-01,
       9.58027931e-02, 3.08386596e-02, 7.44773642e-02, 4.13402445e-02,
       1.92788840e-02, 2.80415152e-02, 4.93838215e-01, 1.65502835e-02,
       3.08277540e-03, 1.74647710e-02, 2.61191408e-01, 1.97665118e-02,
       1.95892874e-02, 7.95183218e-02, 6.72183840e-04, 1.32195797e-02,
       1.24777389e-02, 6.32250258e-02, 5.79625808e-02, 7.28350500e-02,
       3.24892323e-01, 1.73612641e-01, 8.28152031e-04, 7.99639374e-02,
       4.97002372e-01, 6.41780395e-01, 5.52704989e-02, 7.62263342e-02,
       1.32740233e-01, 5.87050953e-01, 5.91342152e-01, 5.08443347e-03,
       1.06560968e-01, 6.69002781e-03, 6.09068154e-02, 2.11328359e-01,
       3.68701854e-02, 3.96643551e-01, 2.94309518e-02, 5.18292700e-02,
       3.44618460e-03, 2.96577876e-01, 2.24618906e-02, 2.45843670e-02,
       9.84944236e-03, 2.82977886e-02, 2.36629032e-03, 4.89083971e-03,
       1.06593614e-01, 3.22277227e-02, 8.43908260e-02, 8.89755562e-01,
       1.01655842e-02, 2.07510840e-03, 3.14944475e-03, 9.91294127e-02,
       1.39271996e-01, 1.05763678e-02, 4.62491869e-03, 3.29517891e-01,
       7.00654301e-01, 1.96667235e-01, 9.97786091e-03, 3.85631608e-01,
       6.96550561e-01, 2.28516613e-01, 1.82557187e-01, 4.88667496e-01,
       2.73199468e-02, 3.53431498e-02, 3.17068584e-02, 3.13677515e-01,
       4.71792593e-01, 3.75527841e-02, 2.67239299e-01, 3.83012181e-03,
       7.37038139e-02, 1.24740256e-01, 4.07667992e-03, 1.71768746e-01,
       7.20048951e-02, 1.75320014e-01, 7.13178095e-03, 2.39429405e-02,
       1.98658569e-01, 3.20547275e-01, 3.96927966e-03, 9.49330798e-03,
```

```
7.45555587e-01, 3.12330244e-03, 3.32408074e-02, 9.19392780e-01,
1.48852250e-02, 2.57190744e-01, 1.73014525e-01, 1.51785386e-01,
1.86931379e-02, 8.28851842e-04, 2.54342547e-01, 4.56483830e-02,
3.99204597e-02, 1.25490112e-01, 1.70919482e-02, 8.25299901e-02,
4.38992386e-02, 4.31506741e-02, 3.39152219e-02, 5.12964217e-02,
2.12219051e-02, 1.57474151e-01, 2.44952859e-03, 8.22895146e-01,
5.98438753e-02, 8.69783952e-02, 5.40701108e-01, 8.54091278e-03,
5.57078600e-01, 2.31232882e-01, 2.11246138e-01, 2.75461703e-01,
1.33154014e-01, 1.82199815e-02, 3.32865074e-02, 1.09429647e-01,
1.86460970e-02, 7.81476677e-03, 2.65574308e-01, 1.69272765e-02,
3.86686532e-01, 2.20988923e-01, 8.08612471e-01, 2.03966645e-02,
1.28650474e-01, 1.77680007e-02, 3.57289398e-01, 5.30474583e-04,
1.67105418e-01, 1.00764993e-02, 8.58290380e-02, 3.05739812e-01,
6.81826517e-02, 3.81711531e-01, 1.10859745e-01, 4.71842280e-03,
1.86894274e-02, 5.45048658e-02, 4.72421338e-02, 9.34489342e-02,
8.33653672e-02, 4.44029727e-01, 5.18663398e-01, 1.79271982e-01,
2.87090882e-01, 2.67710024e-03, 1.09509162e-01, 2.65241921e-01,
```

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
y_test= label_encoder.fit_transform(y_test)


fpr,tpr,thresholds = roc_curve(y_test,probability)


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



Decision Tree

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

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

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

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

```
y_pred
```

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

```
y_test
```

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

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

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

```
# Convert integer labels to string labels
y_test = ['No' if label == 0 else 'Yes' for label in y_test]

# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)
```

```
accuracy
```

```
0.7789115646258503
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[211,  34],
       [ 31,  18]], dtype=int64)
```

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

col_0	No	Yes
row_0		
No	211	34
Yes	31	18

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

```
                  precision    recall  f1-score   support

              No       0.87      0.86      0.87       245
             Yes       0.35      0.37      0.36        49

        accuracy                           0.78       294
       macro avg       0.61      0.61      0.61       294
    weighted avg       0.78      0.78      0.78       294
```

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

```
probability
```

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

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

Hyper parameter Tuning

```
from sklearn import tree
plt.figure(figsize=(25,15))
tree.plot_tree(dtc,filled=True)
```

```
[Text(0.331201688364524, 0.96875, 'x[19] <= -1.257\ngini = 0.269\nsamples = 1176\nvalue = [988, 188]'),
 Text(0.08462164361269324, 0.90625, 'x[45] <= 0.387\ngini = 0.5\nsamples = 78\nvalue = [39, 39]'),
 Text(0.04882017900732303, 0.84375, 'x[2] <= 0.902\ngini = 0.426\nsamples = 39\nvalue = [27, 12]'),
 Text(0.03254678600488202, 0.78125, 'x[26] <= 0.797\ngini = 0.312\nsamples = 31\nvalue = [25, 6]'),
 Text(0.01952807160292921, 0.71875, 'x[10] <= -1.114\ngini = 0.198\nsamples = 27\nvalue = [24, 3]'),
 Text(0.013018714401952807, 0.65625, 'x[46] <= 0.482\ngini = 0.5\nsamples = 6\nvalue = [3, 3]'),
 Text(0.006509357200976403, 0.59375, 'gini = 0.0\nsamples = 3\nvalue = [3, 0]'),
 Text(0.01952807160292921, 0.59375, 'gini = 0.0\nsamples = 3\nvalue = [0, 3]'),
 Text(0.026037428803905614, 0.65625, 'gini = 0.0\nsamples = 21\nvalue = [21, 0]'),
 Text(0.045565500406834825, 0.71875, 'x[8] <= -0.323\ngini = 0.375\nsamples = 4\nvalue = [1, 3]'),
 Text(0.03905614320585842, 0.65625, 'gini = 0.0\nsamples = 3\nvalue = [0, 3]'),
 Text(0.05207485760781123, 0.65625, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.06509357200976404, 0.78125, 'x[14] <= 1.446\ngini = 0.375\nsamples = 8\nvalue = [2, 6]'),
 Text(0.05858421480878763, 0.71875, 'gini = 0.0\nsamples = 6\nvalue = [0, 6]'),
 Text(0.07160292921074043, 0.71875, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(0.12042310821806347, 0.84375, 'x[41] <= 0.755\ngini = 0.426\nsamples = 39\nvalue = [12, 27]'),
 Text(0.09764035801464606, 0.78125, 'x[32] <= 0.397\ngini = 0.26\nsamples = 26\nvalue = [4, 22]'),
 Text(0.08462164361269324, 0.71875, 'x[7] <= 1.482\ngini = 0.095\nsamples = 20\nvalue = [1, 19]'),
 Text(0.07811228641171684, 0.65625, 'gini = 0.0\nsamples = 18\nvalue = [0, 18]'),
 Text(0.09113100081366965, 0.65625, 'x[16] <= -0.659\ngini = 0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(0.08462164361269324, 0.59375, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.09764035801464606, 0.59375, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.11065907241659886, 0.71875, 'x[5] <= -0.401\ngini = 0.5\nsamples = 6\nvalue = [3, 3]'),
 Text(0.10414971521562245, 0.65625, 'gini = 0.0\nsamples = 3\nvalue = [3, 0]'),
 Text(0.11716842961757526, 0.65625, 'gini = 0.0\nsamples = 3\nvalue = [0, 3]'),
 Text(0.14320585842148087, 0.78125, 'x[12] <= 1.103\ngini = 0.473\nsamples = 13\nvalue = [8, 5]'),
 Text(0.13669650122050447, 0.71875, 'x[1] <= 0.712\ngini = 0.32\nsamples = 10\nvalue = [8, 2]'),
 Text(0.13018714401952808, 0.65625, 'gini = 0.0\nsamples = 8\nvalue = [8, 0]'),
 Text(0.14320585842148087, 0.65625, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(0.1497152156224573, 0.71875, 'gini = 0.0\nsamples = 3\nvalue = [0, 3]'),
 Text(0.5777817331163547, 0.90625, 'x[46] <= 0.482\ngini = 0.235\nsamples = 1098\nvalue = [949, 149]'),
 Text(0.32521358828315705, 0.84375, 'x[21] <= -1.786\ngini = 0.162\nsamples = 798\nvalue = [727, 71]'),
 Text(0.1757526444263629, 0.78125, 'x[5] <= -0.173\ngini = 0.38\nsamples = 47\nvalue = [35, 12]'),
 Text(0.16273393002441008, 0.71875, 'x[45] <= 0.387\ngini = 0.1\nsamples = 19\nvalue = [18, 1]'),
 Text(0.1562245728234337, 0.65625, 'gini = 0.0\nsamples = 18\nvalue = [18, 0]'),
 Text(0.16924328722538648, 0.65625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.1887713588283157, 0.71875, 'x[11] <= -0.789\ngini = 0.477\nsamples = 28\nvalue = [17, 11]'),
 Text(0.1822620016273393, 0.65625, 'gini = 0.0\nsamples = 4\nvalue = [0, 4]'),
 Text(0.19528071602929212, 0.65625, 'x[5] <= 0.099\ngini = 0.413\nsamples = 24\nvalue = [17, 7]'),
 Text(0.1887713588283157, 0.59375, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(0.2017900732302685, 0.59375, 'x[25] <= 0.386\ngini = 0.351\nsamples = 22\nvalue = [17, 5]'),
 Text(0.1887713588283157, 0.53125, 'x[1] <= -1.649\ngini = 0.133\nsamples = 14\nvalue = [13, 1]'),
 Text(0.1822620016273393, 0.46875, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.19528071602929212, 0.46875, 'gini = 0.0\nsamples = 13\nvalue = [13, 0]'),
 Text(0.21480878763222133, 0.53125, 'x[1] <= -0.596\ngini = 0.5\nsamples = 8\nvalue = [4, 4]'),
 Text(0.2082994304312449, 0.46875, 'gini = 0.0\nsamples = 3\nvalue = [0, 3]'),
 Text(0.22131814483319773, 0.46875, 'x[24] <= 2.58\ngini = 0.32\nsamples = 5\nvalue = [4, 1]'),
 Text(0.21480878763222133, 0.40625, 'gini = 0.0\nsamples = 4\nvalue = [4, 0]'),
 Text(0.22782750203417412, 0.40625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.4746745321399512, 0.78125, 'x[19] <= 3.564\ngini = 0.145\nsamples = 751\nvalue = [692, 59]'),
 Text(0.46816517493897475, 0.71875, 'x[22] <= -0.41\ngini = 0.143\nsamples = 750\nvalue = [692, 58]'),
 Text(0.3279088689991863, 0.65625, 'x[6] <= -1.118\ngini = 0.218\nsamples = 257\nvalue = [225, 32]'),
 Text(0.2978030919446705, 0.59375, 'x[25] <= -0.455\ngini = 0.355\nsamples = 65\nvalue = [50, 15]'),
 Text(0.2766476810414972, 0.53125, 'x[25] <= -1.016\ngini = 0.303\nsamples = 59\nvalue = [48, 11]'),
 Text(0.25386493083807976, 0.46875, 'x[8] <= -0.323\ngini = 0.463\nsamples = 22\nvalue = [14, 8]'),
 Text(0.24084621643612694, 0.40625, 'x[7] <= -1.151\ngini = 0.198\nsamples = 9\nvalue = [8, 1]'),
 Text(0.23433685923515052, 0.34375, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
```

```
from sklearn.model_selection import GridSearchCV
parameter={
 'criterion':['gini','entropy'],
  'splitter':['best','random'],
  'max_depth':[1,2,3,4,5],
  'max_features':['auto', 'sqrt', 'log2']

}


grid_search=GridSearchCV(estimator=dtc,param_grid=parameter,cv=5,scoring="accuracy")


grid_search.fit(x_train,y_train)
```

```
C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.py:425: FitFailedWarning:
100 fits failed out of a total of 300.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
100 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\model_selection\_validation.py", line 732, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py", line 1144, in wrapper
    estimator._validate_params()
  File "C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\base.py", line 637, in _validate_params
    validate_parameter_constraints(
  File "C:\Users\Asus\anaconda3\Lib\site-packages\sklearn\utils\_param_validation.py", line 95, in validate_parameter_c
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter of DecisionTreeClassifier must be a
```

```
grid_search.best_params_
```

```
{'criterion': 'gini',
 'max_depth': 4,
 'max_features': 'sqrt',
 'splitter': 'random'}
```

```
       nan        nan 0.83588172 0.83758384 0.83928597 0.84013704
```

```
dtc_cv=DecisionTreeClassifier(criterion= 'entropy',
 max_depth=3,
 max_features='sqrt',
 splitter='best')
dtc_cv.fit(x_train,y_train)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=3, max_features='sqrt')
```

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

```
              precision    recall  f1-score   support

          No       0.87      0.86      0.87       245
         Yes       0.35      0.37      0.36        49

    accuracy                           0.78       294
   macro avg       0.61      0.61      0.61       294
weighted avg       0.78      0.78      0.78       294
```

Random Forest

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

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

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
y_pred2=dtc.predict(x_test)
```

```
y_pred2
```

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

```
y_test
```

```
['No',
 'No',
 'Yes',
 'No',
 'Yes',
 'No',
 'Yes',
 'No',
 'No',
 'Yes',
 'No',
 'Yes',
 'No',
 'Yes',
 'No',
 'No',
 'No',
 'Yes',
 'No',
 'No',
 'No',
```