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

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

```
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	Emplo
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.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.shape
```

```
(1470, 35)
```

```
df.Attrition.value_counts()
```

```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

```
df.corr()
```

```
<ipython-input-7-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a
  df.corr()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfac
Age	1.000000	0.010661	-0.001686	0.208034	NaN	-0.010145	0.01
DailyRate	0.010661	1.000000	-0.004985	-0.016806	NaN	-0.050990	0.01
DistanceFromHome	-0.001686	-0.004985	1.000000	0.021042	NaN	0.032916	-0.01
Education	0.208034	-0.016806	0.021042	1.000000	NaN	0.042070	-0.02
EmployeeCount	NaN	NaN	NaN	NaN	NaN	NaN	
EmployeeNumber	-0.010145	-0.050990	0.032916	0.042070	NaN	1.000000	0.01
EnvironmentSatisfaction	0.010146	0.018355	-0.016075	-0.027128	NaN	0.017621	1.00
HourlyRate	0.024287	0.023381	0.031131	0.016775	NaN	0.035179	-0.04
JobInvolvement	0.029820	0.046135	0.008783	0.042438	NaN	-0.006888	-0.00
JobLevel	0.509604	0.002966	0.005303	0.101589	NaN	-0.018519	0.00
JobSatisfaction	-0.004892	0.030571	-0.003669	-0.011296	NaN	-0.046247	-0.00
MonthlyIncome	0.497855	0.007707	-0.017014	0.094961	NaN	-0.014829	-0.00
MonthlyRate	0.028051	-0.032182	0.027473	-0.026084	NaN	0.012648	0.03
NumCompaniesWorked	0.299635	0.038153	-0.029251	0.126317	NaN	-0.001251	0.01
PercentSalaryHike	0.003634	0.022704	0.040235	-0.011111	NaN	-0.012944	-0.03
PerformanceRating	0.001904	0.000473	0.027110	-0.024539	NaN	-0.020359	-0.02
RelationshipSatisfaction	0.053535	0.007846	0.006557	-0.009118	NaN	-0.069861	0.00
StandardHours	NaN	NaN	NaN	NaN	NaN	NaN	
StockOptionLevel	0.037510	0.042143	0.044872	0.018422	NaN	0.062227	0.00
TotalWorkingYears	0.680381	0.014515	0.004628	0.148280	NaN	-0.014365	-0.00
TrainingTimesLastYear	-0.019621	0.002453	-0.036942	-0.025100	NaN	0.023603	-0.01
WorkLifeBalance	-0.021490	-0.037848	-0.026556	0.009819	NaN	0.010309	0.02
YearsAtCompany	0.311309	-0.034055	0.009508	0.069114	NaN	-0.011240	0.00
YearsInCurrentRole	0.212901	0.009932	0.018845	0.060236	NaN	-0.008416	0.01
YearsSinceLastPromotion	0.216513	-0.033229	0.010029	0.054254	NaN	-0.009019	0.01
YearsWithCurrManager	0.202089	-0.026363	0.014406	0.069065	NaN	-0.009197	-0.00

26 rows × 26 columns

```
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
```

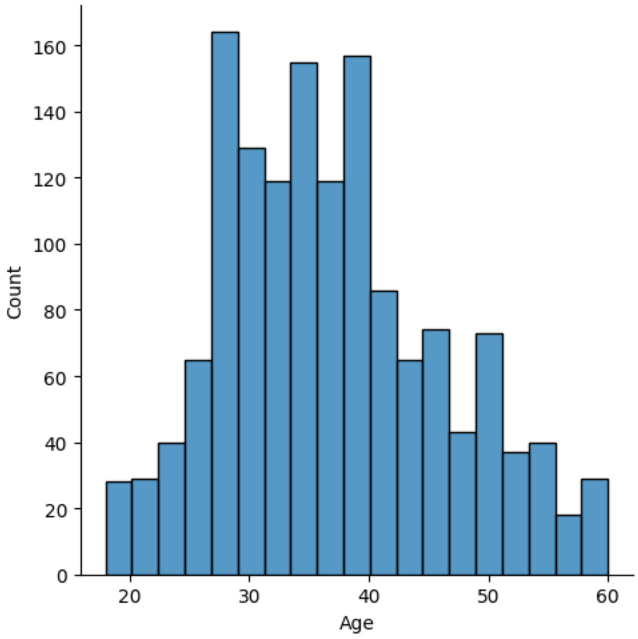
```
    WorkLifeBalance             False
    YearsAtCompany              False
    YearsInCurrentRole          False
    YearsSinceLastPromotion     False
    YearsWithCurrManager        False
    dtype: bool
```

```
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
```

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

```
    <seaborn.axisgrid.FacetGrid at 0x7f005861f5b0>
```



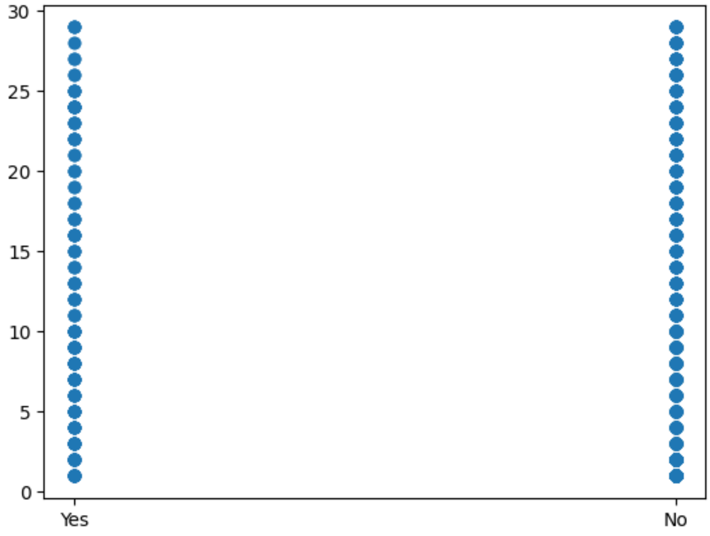
```
plt.scatter(df['Attrition'],df['BusinessTravel'])
```

```
<matplotlib.collections.PathCollection at 0x7f0018a06320>
```

Non-Travel
Travel_Frequently

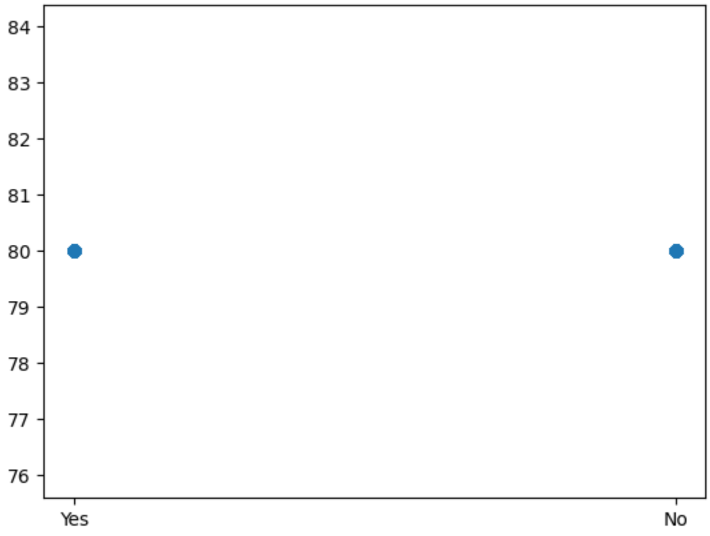
```
plt.scatter(df['Attrition'],df['DistanceFromHome'])
```

```
<matplotlib.collections.PathCollection at 0x7f001887cdf0>
```



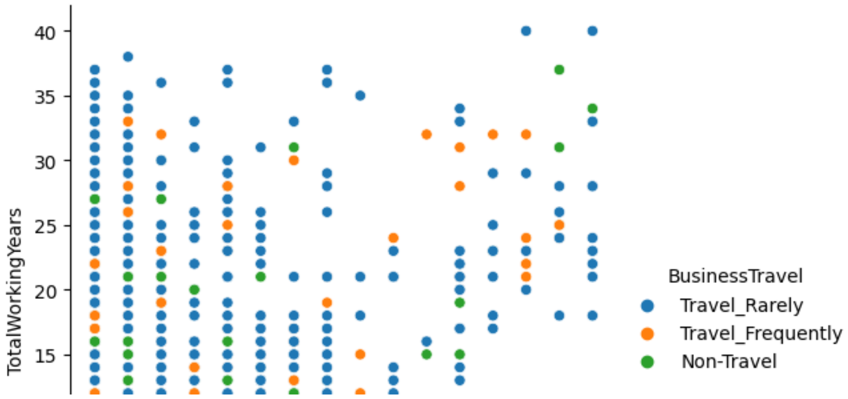
```
plt.scatter(df['Attrition'],df['StandardHours'])
```

```
<matplotlib.collections.PathCollection at 0x7f00188d7310>
```



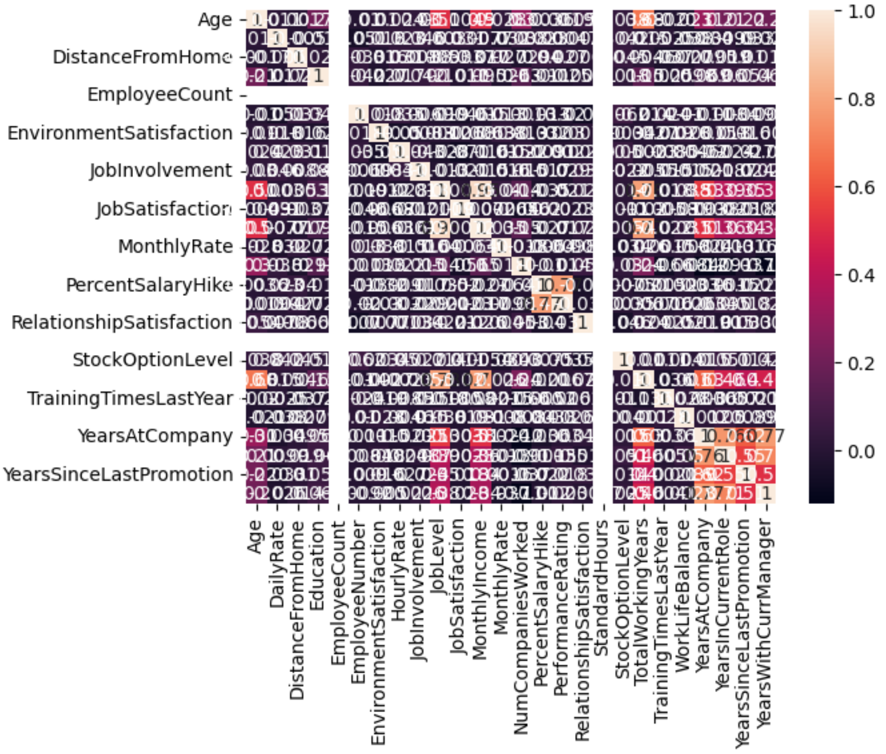
```
sns.relplot(x="YearsSinceLastPromotion",y="TotalWorkingYears",data=df,hue="BusinessTravel")
```

```
<seaborn.axisgrid.FacetGrid at 0x7f0018a47700>
```



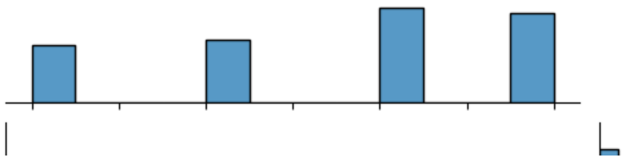
```
sns.heatmap(df.corr(),annot=True)
```

```
<ipython-input-15-8df7bcac526d>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a
  sns.heatmap(df.corr(),annot=True)
<Axes: >
```



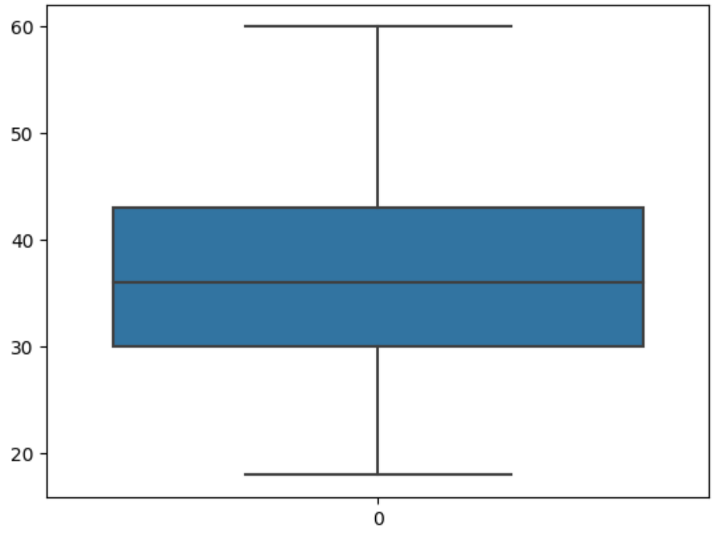
```
sns.jointplot(x="RelationshipSatisfaction",y="Attrition",data=df)
```

```
<seaborn.axisgrid.JointGrid at 0x7f00187e6d40>
```



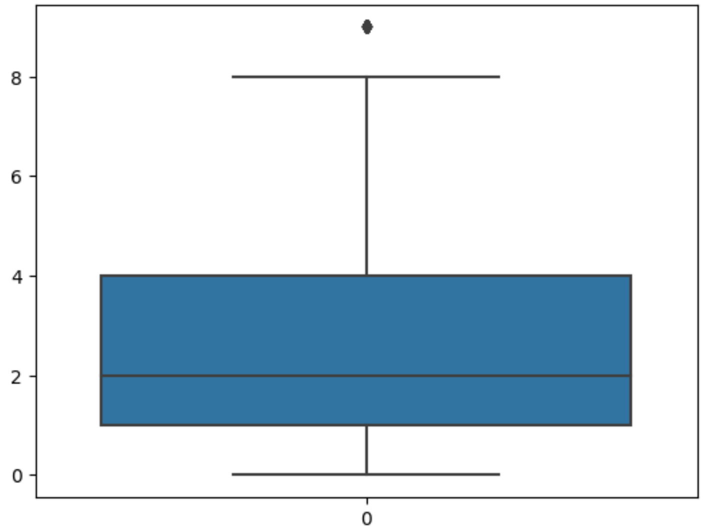
```
sns.boxplot(df.Age)
```

```
<Axes: >
```



```
sns.boxplot(df.NumCompaniesWorked)
```

```
<Axes: >
```



```
q1=df.NumCompaniesWorked.quantile(0.25)
q3=df.NumCompaniesWorked.quantile(0.75)
```

```
print(q1)
print(q3)
```

```
1.0
4.0
```

```
IQR=q3-q1
```

```
IQR
```

```
3.0
```

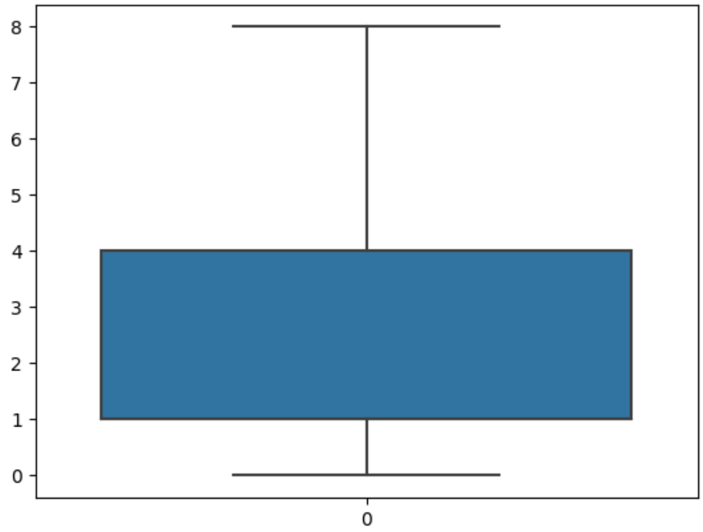
```
upper_limit=q3+1.5*IQR
upper_limit
```

```
8.5
```

```
df=df[df.NumCompaniesWorked<upper_limit]
```

```
sns.boxplot(df.NumCompaniesWorked)
```

```
<Axes: >
```



```
#dependent variable
y=df.Attrition
```

```
y.head()
```

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

```
#independent varible
x=df.drop(["Attrition"],axis=1)
```

```
x.head()
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber
0	41	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1
1	49	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2
2	37	Travel_Rarely	1373	Research & Development	2	2	Other	1	4
3	33	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5
5	32	Travel_Frequently	1005	Research & Development	2	2	Life Sciences	1	8

5 rows × 34 columns

```
x.shape
```

```
(1418, 34)
```

```
y.shape
```

```
(1418,)
```

```
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	Emplo
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	

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
x["BusinessTravel"]=le.fit_transform(x["BusinessTravel"])
```

```
x["BusinessTravel"]
```

```
0       2
1       1
2       2
3       1
5       1
       ..
1465    1
1466    2
1467    2
1468    1
1469    2
Name: BusinessTravel, Length: 1418, dtype: int64
```

```
x.head()
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber
0	41	2	1102	Sales	1	2	Life Sciences	1	1
1	49	1	279	Research & Development	8	1	Life Sciences	1	2
2	37	2	1373	Research & Development	2	2	Other	1	4
3	33	1	1392	Research & Development	3	4	Life Sciences	1	5
5	32	1	1005	Research & Development	2	2	Life Sciences	1	8

5 rows × 34 columns

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

```
x["Department"]
```

```
0       2
1       1
2       1
3       1
5       1
       ..
1465    1
1466    1
1467    1
1468    2
1469    1
Name: Department, Length: 1418, dtype: int64
```

```
x.head()
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber

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

1	49	1	279	1	8	1	Life Sciences	1	2

```
x["EducationField"]
```

```
0       1
1       1
2       4
3       1
5       1
       ..
1465    3
1466    3
1467    1
1468    3
1469    3
Name: EducationField, Length: 1418, dtype: int64
```

```
x.head()
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber
0	41	2	1102	2	1	2	1	1	1
1	49	1	279	1	8	1	1	1	2
2	37	2	1373	1	2	2	4	1	4
3	33	1	1392	1	3	4	1	1	5
5	32	1	1005	1	2	2	1	1	8

5 rows × 34 columns

```
non_numeric_columns = x.select_dtypes(exclude=['number']).columns
```

```
print(non_numeric_columns)
```

```
Index(['Gender', 'JobRole', 'MaritalStatus', 'Over18', 'OverTime'], dtype='object')
```

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

```
x["Gender"]
```

```
0       0
1       1
2       1
3       0
5       1
       ..
1465    1
1466    1
1467    1
1468    1
1469    1
Name: Gender, Length: 1418, dtype: int64
```

```
x.head()
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber
0	41	2	1102	2	1	2	1	1	1
1	49	1	279	1	8	1	1	1	2
2	37	2	1373	1	2	2	4	1	4
3	33	1	1392	1	3	4	1	1	5
5	32	1	1005	1	2	2	1	1	8

5 rows × 34 columns

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

```
x["JobRole"]
```

```
0       7
1       6
2       2
3       6
5       2
       ..
1465    2
1466    0
1467    4
1468    7
1469    2
Name: JobRole, Length: 1418, dtype: int64
```

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

```
x["MaritalStatus"]
```

```
0       2
1       1
2       2
3       1
5       2
       ..
1465    1
1466    1
1467    1
1468    1
1469    1
Name: MaritalStatus, Length: 1418, dtype: int64
```

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

```
x["Over18"]
```

```
0       0
1       0
2       0
3       0
5       0
       ..
1465    0
1466    0
1467    0
1468    0
1469    0
Name: Over18, Length: 1418, dtype: int64
```

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

```
x["OverTime"]
```

```
0       1
1       0
2       1
3       1
5       0
       ..
1465    0
1466    0
1467    1
1468    0
1469    0
Name: OverTime, Length: 1418, dtype: int64
```

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

```
x_scaled
```

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeN
0	0.547619	1.0	0.715820	1.0	0.000000	0.25	0.2	0.0	0.0
1	0.738095	0.5	0.126700	0.5	0.250000	0.00	0.2	0.0	0.0
2	0.452381	1.0	0.909807	0.5	0.035714	0.25	0.8	0.0	0.0
3	0.357143	0.5	0.923407	0.5	0.071429	0.75	0.2	0.0	0.0
4	0.333333	0.5	0.646385	0.5	0.035714	0.25	0.2	0.0	0.0
...	...	...	...	...	...	...	...	...	

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

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

```
(1134, 34) (284, 34) (1134,) (284,)
```

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

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

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

```
y_test
```

```
451     No
639     No
832     No
1287    No
1277    No
        ..
521     No
550     No
1113    No
335     No
917     No
Name: Attrition, Length: 284, dtype: object
```

```
df
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	Em
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	
5	32	No	Travel_Frequently	1005	Research & Development	2	2	Life Sciences	1	
...	...	...	...	...	...	...	...	...	...	
1465	36	No	Travel_Frequently	884	Research & Development	23	2	Medical	1	
1466	39	No	Travel_Rarely	613	Research & Development	6	1	Medical	1	
1467	27	No	Travel_Rarely	155	Research & Development	4	3	Life Sciences	1	
1468	49	No	Travel_Frequently	1023	Sales	2	3	Medical	1	
1469	34	No	Travel_Rarely	628	Research &	8	3	Medical	1	

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

1418 rows × 35 columns

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

```
0.8697183098591549
```

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

```
array([[237,   3],
       [ 34,  10]])
```

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

col_0	No	Yes
Attrition		
No	237	3
Yes	34	10

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

```
              precision    recall  f1-score   support

          No       0.87      0.99      0.93       240
         Yes       0.77      0.23      0.35        44

    accuracy                           0.87       284
   macro avg       0.82      0.61      0.64       284
weighted avg       0.86      0.87      0.84       284
```

```
#ROC_AUC Curve

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

```
array([0.13657895, 0.03742004, 0.08053736, 0.08659374, 0.023358  ,
       0.10563069, 0.13815154, 0.00229225, 0.06771379, 0.12744425,
       0.08172802, 0.05965762, 0.0638561 , 0.12855128, 0.2275486 ,
       0.08936636, 0.06240484, 0.09603478, 0.21199145, 0.05717384,
       0.01180209, 0.00367791, 0.07898725, 0.02473968, 0.11962886,
       0.12904799, 0.0184306 , 0.0365714 , 0.02049336, 0.10008116,
       0.16143025, 0.03099261, 0.05571065, 0.04469354, 0.21600549,
       0.42230677, 0.2197372 , 0.5227653 , 0.18101958, 0.10182865,
       0.03088844, 0.18054679, 0.08248226, 0.01733578, 0.19733818,
       0.06725397, 0.01197982, 0.01366601, 0.02702768, 0.18659878,
       0.04323244, 0.00445696, 0.05192806, 0.1866853 , 0.1632088 ,
       0.27853238, 0.07437663, 0.09816652, 0.00573849, 0.00449716,
       0.0059488 , 0.03111943, 0.00839901, 0.00669404, 0.04253402,
       0.18695255, 0.19941885, 0.03278527, 0.00238087, 0.01663221,
       0.58136087, 0.1578733 , 0.21711936, 0.03898385, 0.04521495,
       0.03220001, 0.06616953, 0.19809653, 0.10991992, 0.22934288,
       0.05904098, 0.02037218, 0.66970453, 0.26829173, 0.08216447,
       0.04010601, 0.11590138, 0.27057603, 0.22694055, 0.20450222,
       0.56793147, 0.22053355, 0.36393157, 0.01755166, 0.01233427,
       0.01492107, 0.2081514 , 0.12205625, 0.40315397, 0.04856193,
       0.07330096, 0.25379683, 0.14516211, 0.28647266, 0.02781388,
       0.18391223, 0.26396952, 0.01946723, 0.28598072, 0.04347479,
       0.15563751, 0.13357455, 0.00963796, 0.02116195, 0.07528362,
       0.05922541, 0.11977388, 0.00903596, 0.36455439, 0.05168354,
       0.20310448, 0.01231492, 0.05158269, 0.57453501, 0.07656055,
       0.03508536, 0.30385493, 0.0309728 , 0.42983322, 0.02371366,
       0.05130702, 0.02103465, 0.04602763, 0.01905589, 0.32734204,
       0.19614051, 0.06294798, 0.0186783 , 0.00440507, 0.12521514,
       0.35937712, 0.01824423, 0.03851794, 0.36623505, 0.0761209 ,
       0.26592758, 0.03553327, 0.02772604, 0.0193432 , 0.28332535,
       0.31642215, 0.02571374, 0.12136821, 0.32580669, 0.13472202,
       0.06624905, 0.08617629, 0.03661786, 0.01839348, 0.15357873,
       0.39926896, 0.71257736, 0.89315923, 0.00546009, 0.00246771,
       0.02778452, 0.05857899, 0.36399558, 0.01646451, 0.14794275,
       0.47711028, 0.03384135, 0.01739   , 0.04238425, 0.20976761,
       0.54481958, 0.02510394, 0.01863455, 0.24136931, 0.06312414,
       0.03643677, 0.00616726, 0.1100783 , 0.15064248, 0.07821613,
       0.10409581, 0.20971698, 0.13795456, 0.28657845, 0.02226441,
       0.23272876, 0.23596972, 0.16844684, 0.00414635, 0.03126561,
       0.44815074, 0.01643598, 0.10900941, 0.01603778, 0.0333788 ,
       0.27797218, 0.14158042, 0.05577601, 0.09399929, 0.24091949,
       0.09998247, 0.01242131, 0.02205424, 0.1890573 , 0.06235382,
       0.09115454, 0.00728886, 0.19906759, 0.1575069 , 0.20840636,
       0.13738917, 0.05410298, 0.18636277, 0.08545779, 0.2373784 ,
       0.04893286, 0.28718093, 0.07707427, 0.25024676, 0.11690009,
       0.05663235, 0.06336832, 0.1402614 , 0.09635028, 0.5603858 ,
       0.07966128, 0.18409077, 0.00949154, 0.04702311, 0.16756119,
       0.03001824, 0.51420487, 0.00555785, 0.09370631, 0.01171392,
       0.12695966, 0.03659918, 0.3821563 , 0.13188418, 0.17530265,
       0.19609367, 0.10202889, 0.74603311, 0.05622724, 0.15448205,
       0.17941515, 0.07061508, 0.07724554, 0.11220407, 0.19871038,
       0.08215216, 0.00188234, 0.15323164, 0.06851284, 0.02069078,
       0.71737346, 0.17804198, 0.15215912, 0.00469619, 0.23093543,
       0.03742954, 0.06874542, 0.45373149, 0.6448183 , 0.09910567,
       0.3574995 , 0.02215789, 0.00967421, 0.07067802, 0.35407627,
       0.31550123, 0.01930184, 0.08248221, 0.07689043, 0.01921869,
       0.13324521, 0.08754501, 0.22298726, 0.42007529])
```

```
y_test_encoded = le.fit_transform(y_test)
```