

- **Data Preprocessing.**

- Import the Libraries.
- Importing the dataset.
- Checking for Null Values.
- Data Visualization.
- Outlier Detection
- Splitting Dependent and Independent variables
- Encoding
- Feature Scaling.
- Splitting Data into Train and Test.

1.Import the Libraries.

```
In [4]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

2.Importing the dataset.

```
In [5]: df=pd.read_csv("Employee-Attrition.csv")
```

```
In [6]: df.head()
```

Out[6]:

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

5 rows × 35 columns



In [7]: df.tail()

Out[7]:

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

5 rows × 35 columns



In [8]: 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
```

```
In [9]: df.describe()
```

Out[9]:

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

8 rows × 26 columns



In [10]: `df.shape`

Out[10]: (1470, 35)

In [11]: `df.corr()`

Out[11]:

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

26 rows × 26 columns

In [12]: `df.isnull().any()`

3. Checking for Null Values.

```
Out[12]: 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
```

```
In [13]: df.isnull().sum()
```

```
Out[13]: 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
```

```
In [14]: df["Department"].value_counts()
```

```
Out[14]: Research & Development    961  
Sales                  446  
Human Resources          63  
Name: Department, dtype: int64
```

```
In [15]: df["BusinessTravel"].value_counts()
```

```
Out[15]: Travel_Rarely      1043  
Travel_Frequently      277  
Non-Travel            150  
Name: BusinessTravel, dtype: int64
```

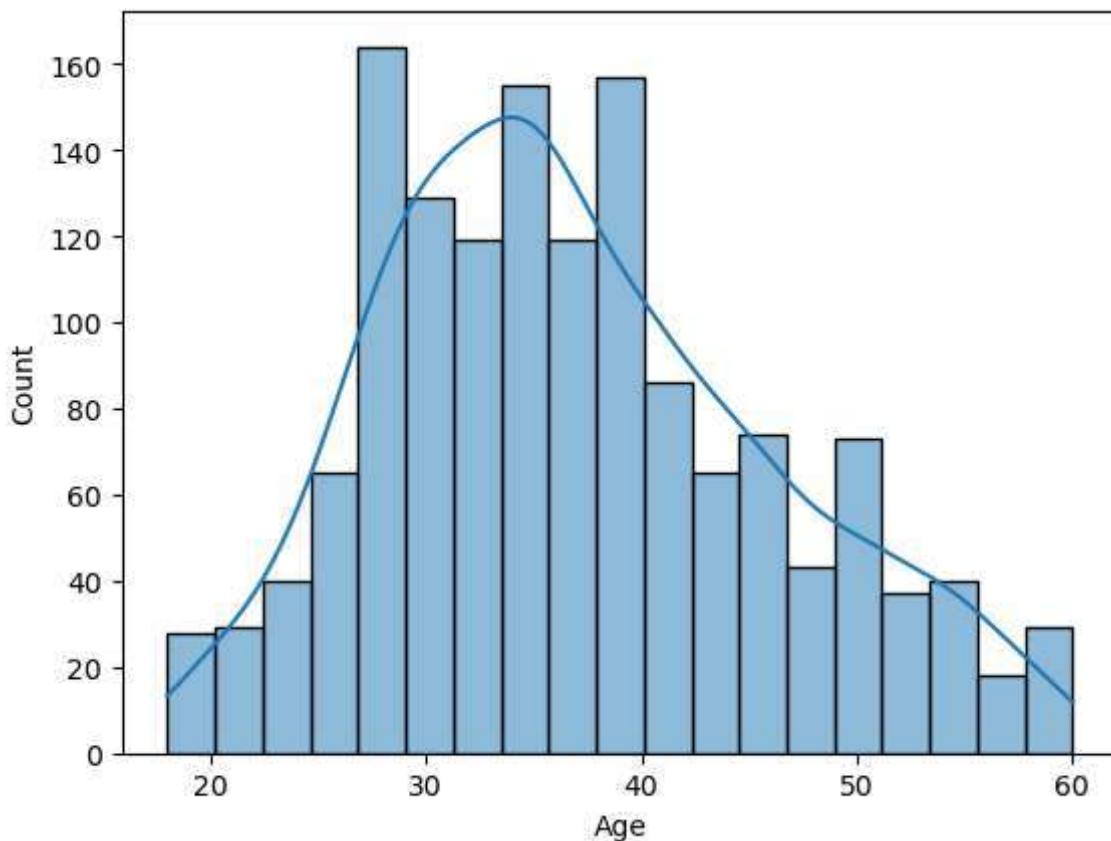
```
In [16]: df["EducationField"].value_counts()
```

```
Out[16]: Life Sciences      606  
Medical                464  
Marketing               159  
Technical Degree        132  
Other                  82  
Human Resources          27  
Name: EducationField, dtype: int64
```

4.Data Visualization.

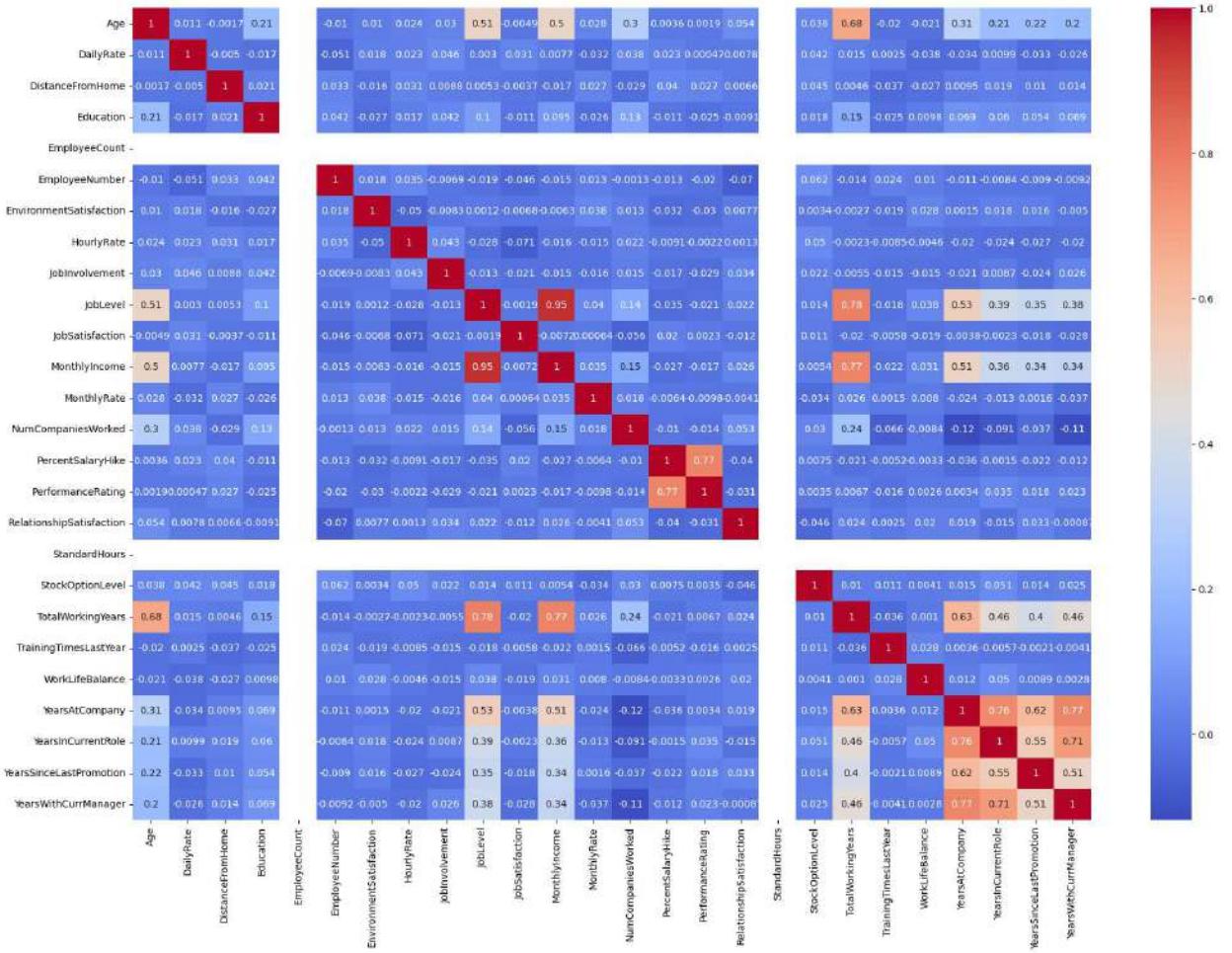
```
In [17]: sns.histplot(x="Age", data=df, kde=True)
```

```
Out[17]: <AxesSubplot:xlabel='Age', ylabel='Count'>
```



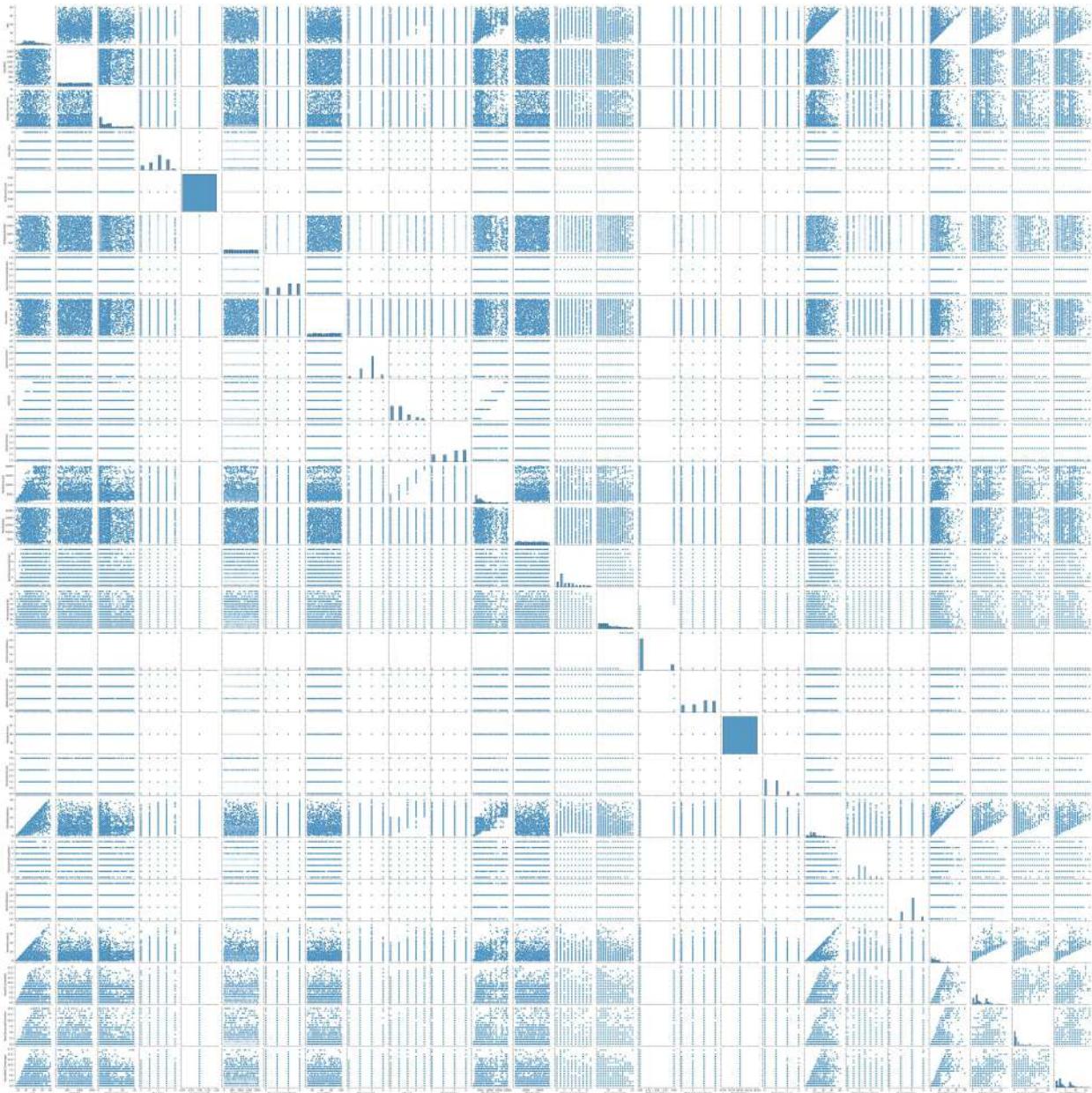
```
In [15]: plt.subplots(figsize=(22,15))  
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
```

```
Out[15]: <AxesSubplot:>
```



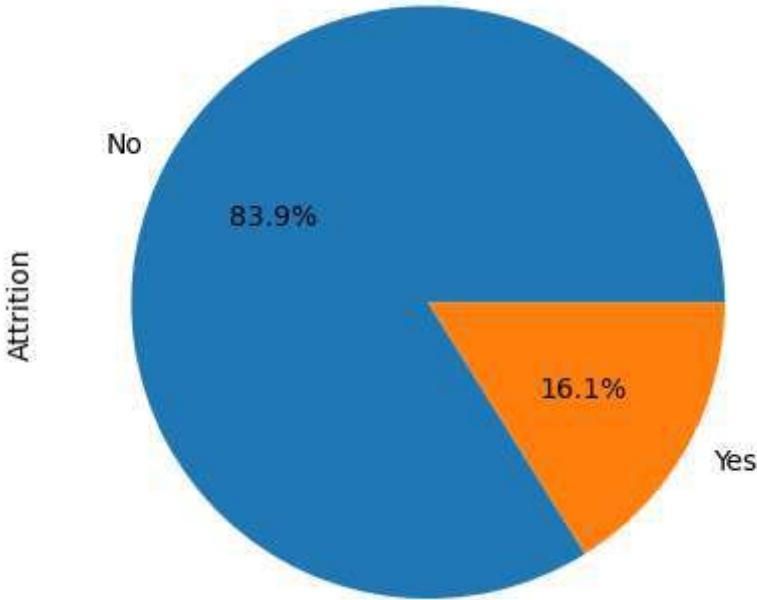
```
In [16]: sns.pairplot(df)
```

```
Out[16]: <seaborn.axisgrid.PairGrid at 0x1e9dd3441f0>
```



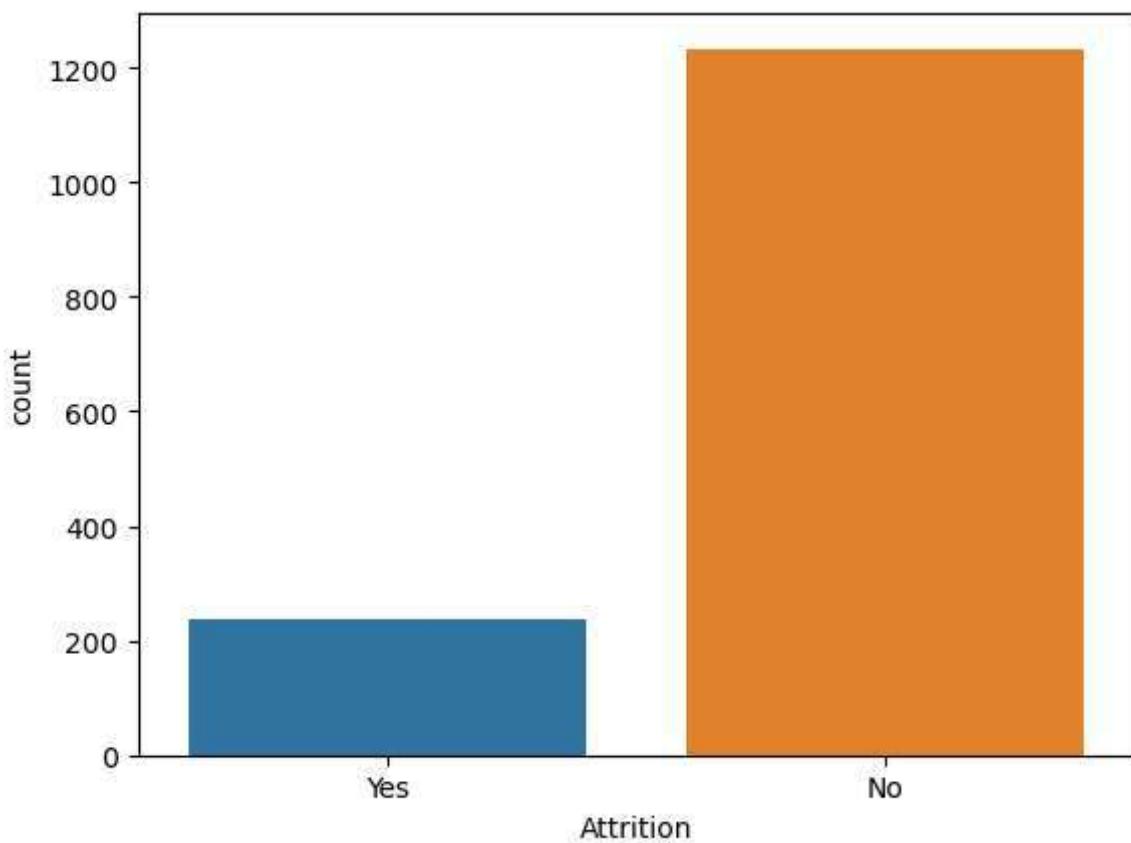
```
In [18]: df.Attrition.value_counts().plot(kind="pie", autopct="%1.1f%%")
```

```
Out[18]: <AxesSubplot: ylabel='Attrition'>
```



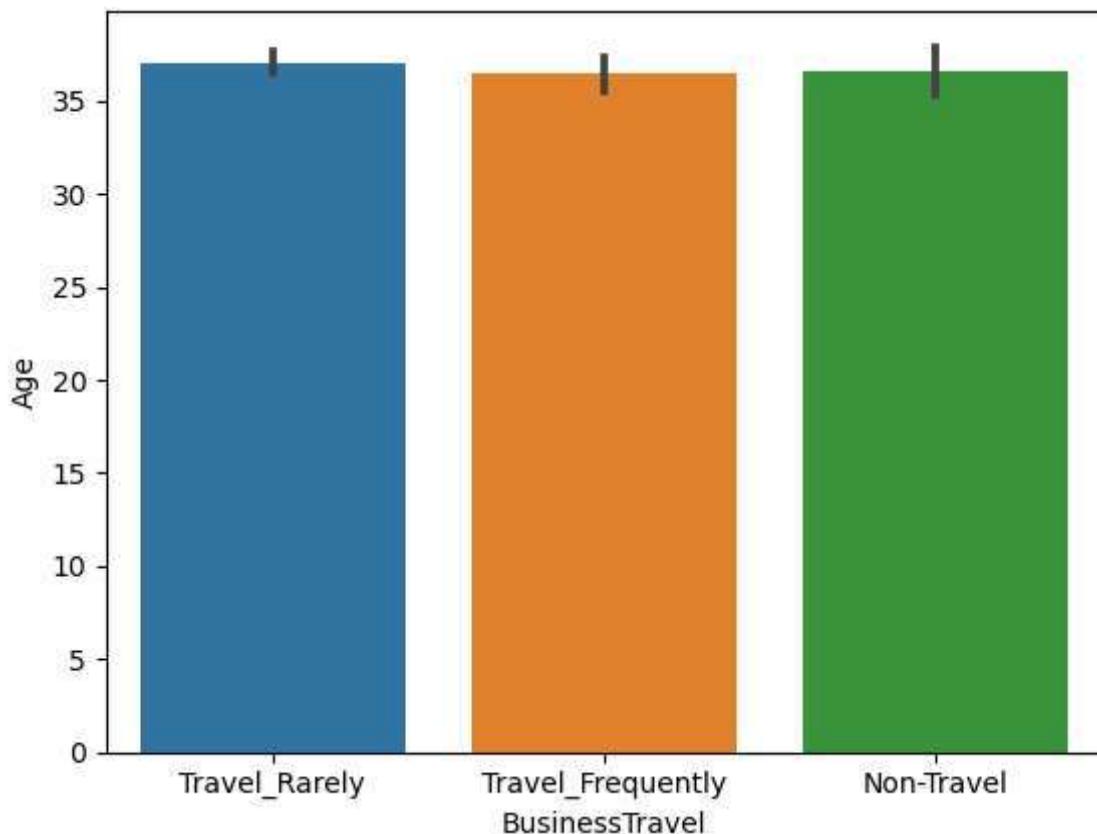
```
In [19]: sns.countplot(x="Attrition", data=df)
```

```
Out[19]: <AxesSubplot:xlabel='Attrition', ylabel='count'>
```



```
In [20]: sns.barplot(x=df["BusinessTravel"], y=df["Age"])
```

```
Out[20]: <AxesSubplot:xlabel='BusinessTravel', ylabel='Age'>
```



5. Outlier Detection

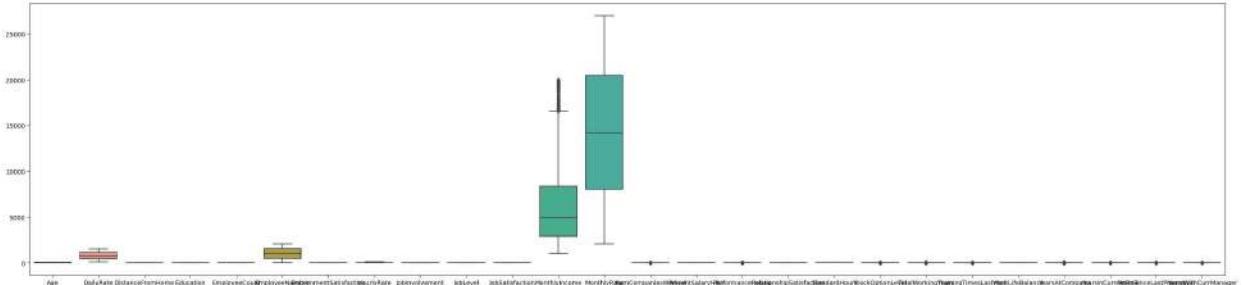
```
In [21]: df.head()
```

```
Out[21]:   Age Attrition BusinessTravel DailyRate Department DistanceFromHome Education Education
0    41      Yes  Travel_Rarely     1102        Sales              1         2  Life Sci
1    49       No  Travel_Frequently    279  Research & Development      8         1  Life Sci
2    37      Yes  Travel_Rarely     1373  Research & Development      2         2  Life Sci
3    33       No  Travel_Frequently    1392  Research & Development      3         4  Life Sci
4    27       No  Travel_Rarely      591  Research & Development      2         1  Mktg
```

5 rows × 35 columns

```
In [22]: plt.figure(figsize=(35,8))
sns.boxplot(data=df)
```

```
Out[22]: <AxesSubplot:>
```

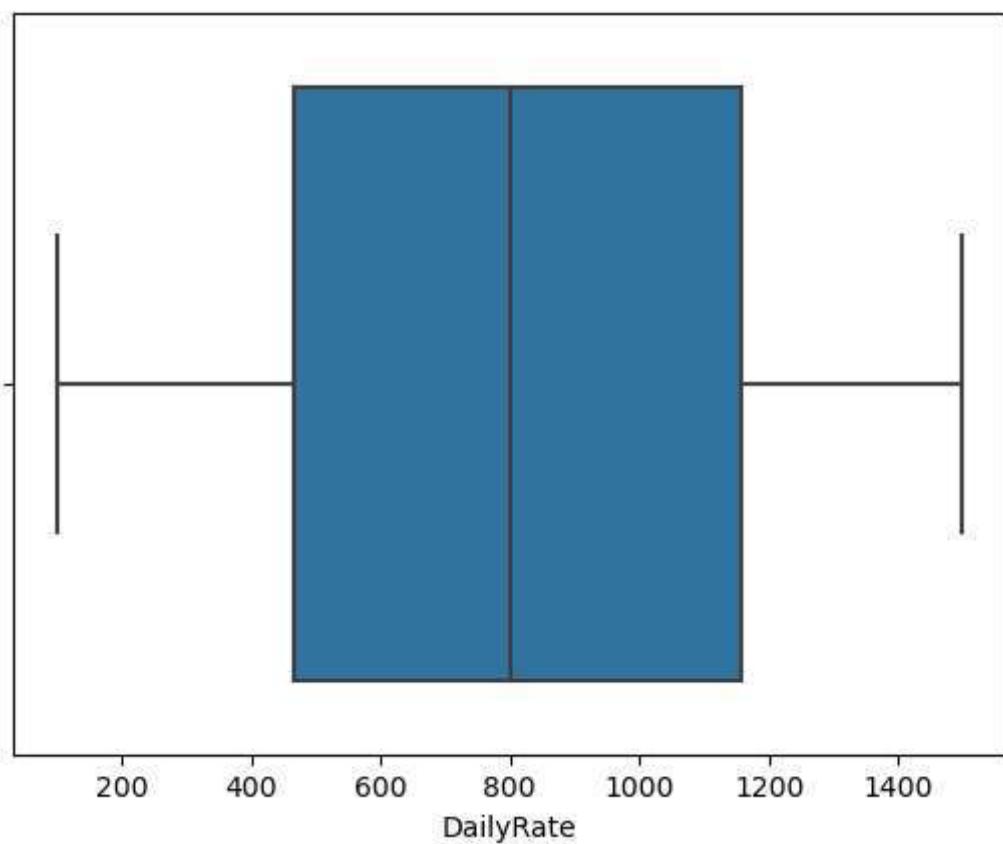


```
In [23]: sns.boxplot(df[ "DailyRate"])
```

C:\Users\admin\anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po
sitional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.

```
    warnings.warn(
```

```
Out[23]: <AxesSubplot:xlabel='DailyRate'>
```

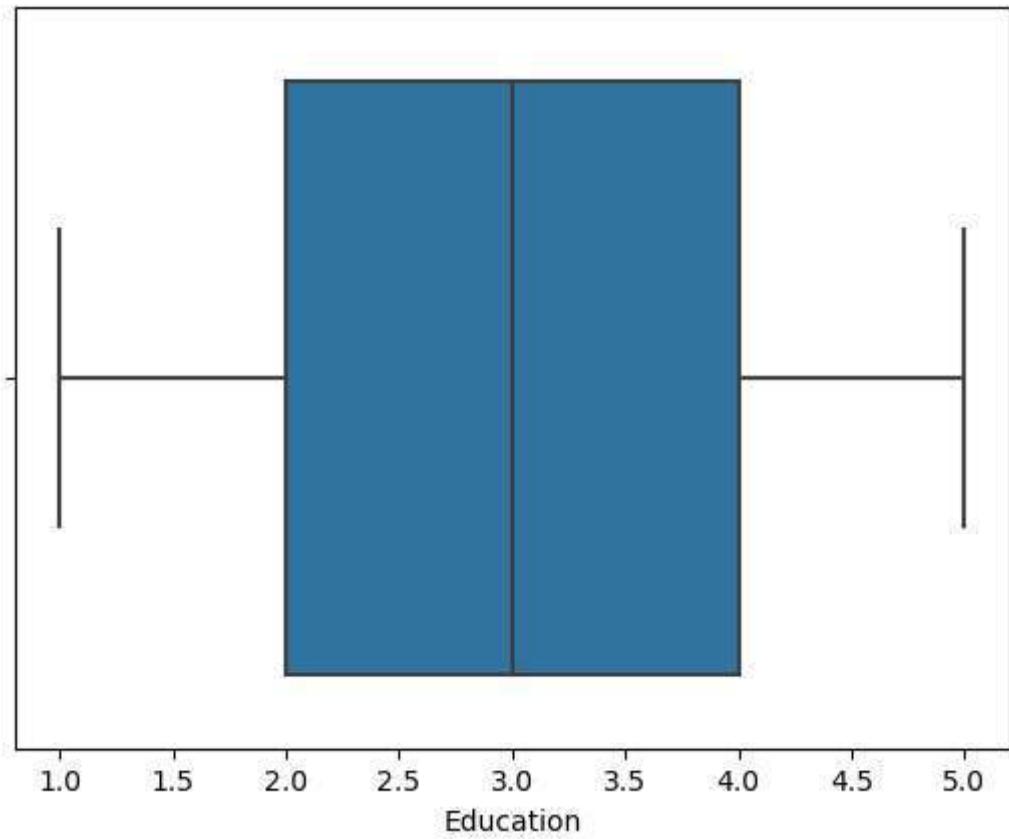


```
In [24]: sns.boxplot(df[ "Education"])
```

C:\Users\admin\anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only valid po
sitional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.

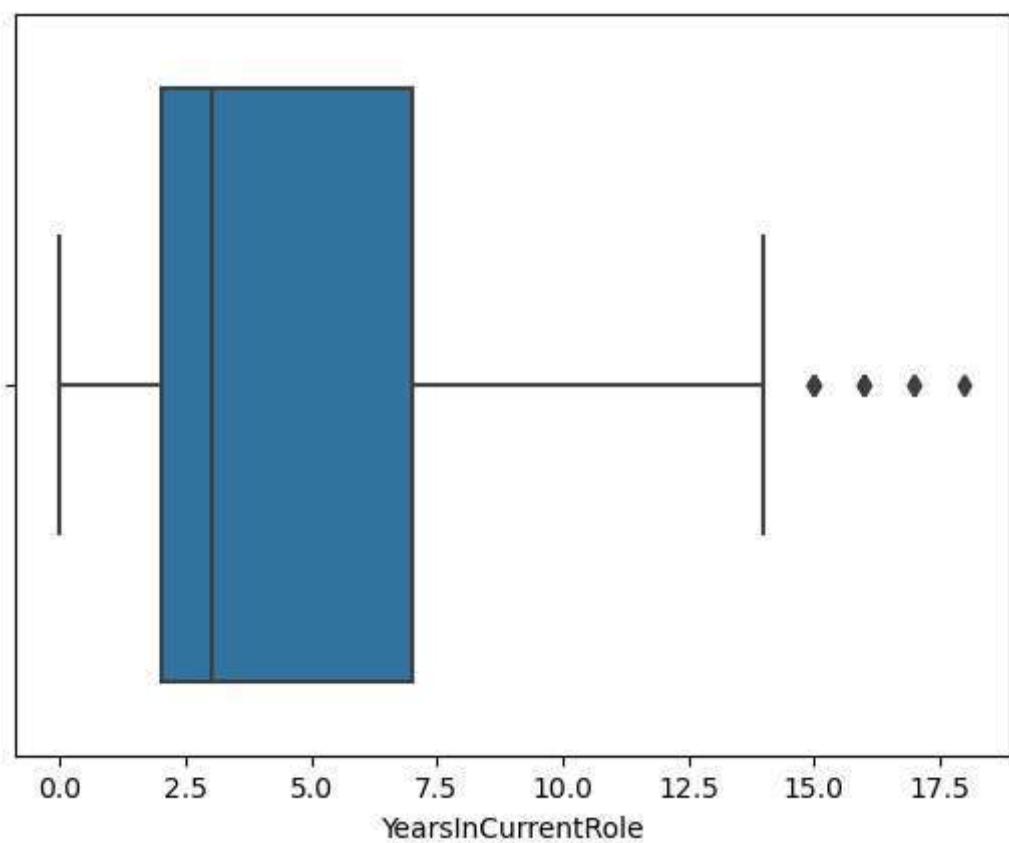
```
    warnings.warn(
```

```
Out[24]: <AxesSubplot:xlabel='Education'>
```



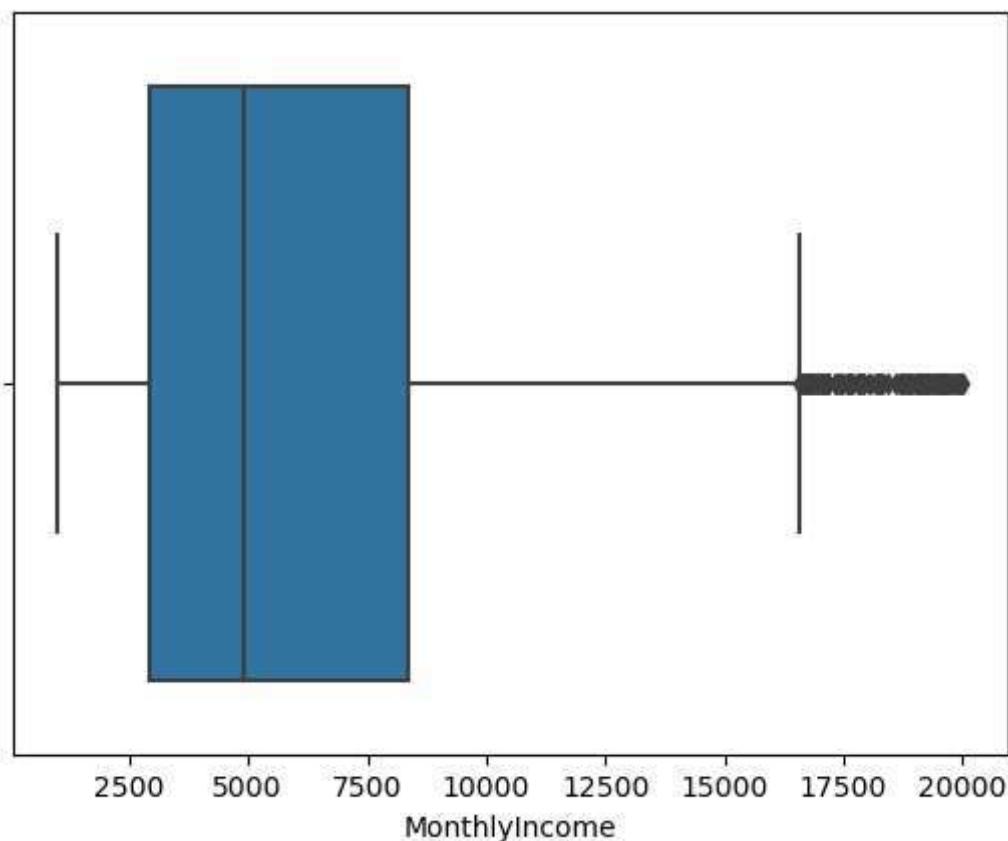
```
In [25]: sns.boxplot(data=df, x='YearsInCurrentRole')
```

```
Out[25]: <AxesSubplot:xlabel='YearsInCurrentRole'>
```



```
In [26]: sns.boxplot(data=df, x='MonthlyIncome')
```

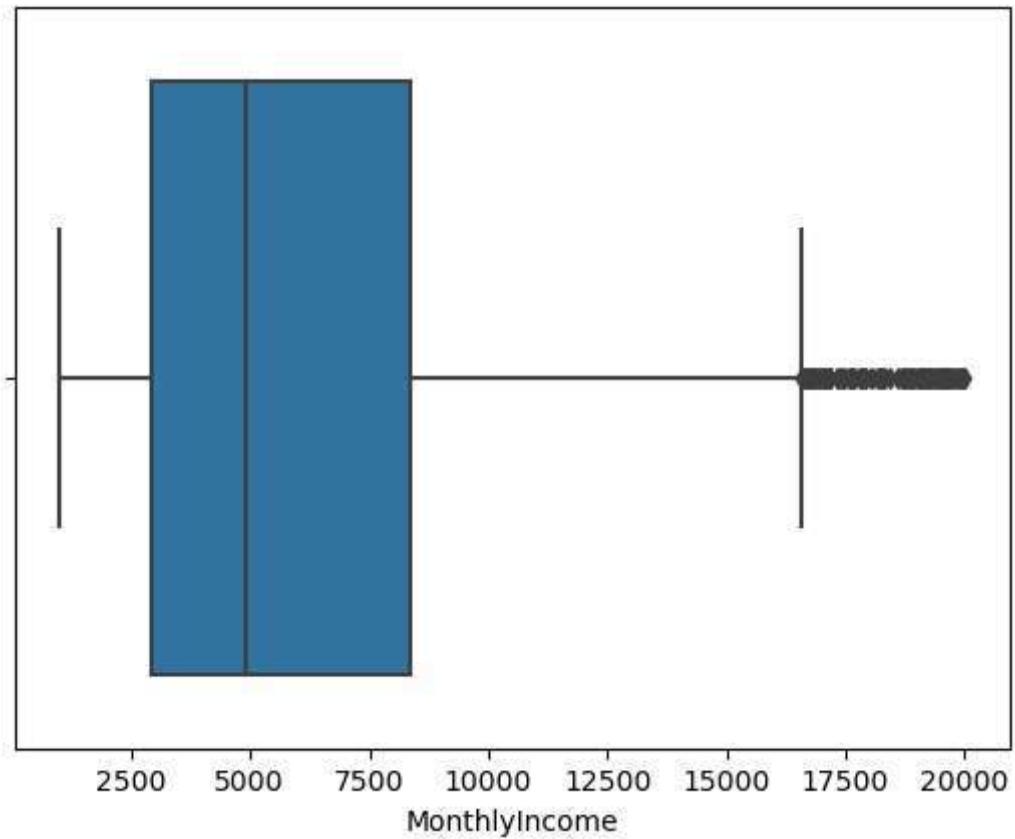
```
Out[26]: <AxesSubplot:xlabel='MonthlyIncome'>
```



```
In [27]: from scipy import stats  
z_scores = stats.zscore(df['MonthlyIncome'])  
df_cleaned = df[(np.abs(z_scores) <= 3)]
```

```
In [28]: sns.boxplot(data=df_cleaned, x='MonthlyIncome')
```

```
Out[28]: <AxesSubplot:xlabel='MonthlyIncome'>
```



So the outliers are in large quantity and they are inside the threshold, so let us not remove the outliers

6. Splitting Dependent and Independent variables

```
In [29]: x=df.drop(columns=["Attrition"],axis=1)
```

```
In [30]: x.head()
```

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

5 rows × 34 columns

```
In [31]: type(x)
```

```
Out[31]: pandas.core.frame.DataFrame
```

```
In [32]: y=df["Attrition"]
```

```
In [33]: y.head()
```

```
Out[33]: 0    Yes  
1    No  
2    Yes  
3    No  
4    No  
Name: Attrition, dtype: object
```

```
In [34]: type(y)
```

```
Out[34]: pandas.core.series.Series
```

7.Encoding

```
In [35]: from sklearn.preprocessing import LabelEncoder  
le=LabelEncoder()
```

```
In [36]: categorical_features = x.select_dtypes(include=['object']).columns.tolist()  
x_encoded = pd.get_dummies(x, columns=categorical_features, drop_first=True)
```

```
In [37]: x_encoded.head()
```

```
Out[37]:
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSat
0	41	1102		1	2	1	1
1	49	279		8	1	1	2
2	37	1373		2	2	1	4
3	33	1392		3	4	1	5
4	27	591		2	1	1	7

5 rows × 47 columns

8.Feature Scaling

```
In [38]: from sklearn.preprocessing import StandardScaler  
s=StandardScaler()  
x_scaled=pd.DataFrame(s.fit_transform(x_encoded),columns=x_encoded.columns)
```

```
In [39]: x_scaled.head()
```

Out[39]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	JobRole	JobSatisfaction	MaritalStatus	OverTime	RelationshipSatisfaction	StandardHours	TotalWorkingYears	WorkLifeBalance	YearsAtCompany	YearsInCurrentRole	YearsOnSite
0	0.446350	0.742527	-1.010909	-0.891688	0.0	-1.701283	0.0	-0.891688	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	1.322365	-1.297775	-0.147150	-1.868426	0.0	-1.699621	0.0	-0.891688	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.008343	1.414363	-0.887515	-0.891688	0.0	-1.696298	0.0	-0.891688	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	-0.429664	1.461466	-0.764121	1.061787	0.0	-1.694636	0.0	-0.891688	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	-1.086676	-0.524295	-0.887515	-1.868426	0.0	-1.691313	0.0	-0.891688	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 47 columns

9. Splitting Data into Train and Test.

```
In [40]: 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=42)  
  
In [41]: x_train.shape,x_test.shape,y_train.shape,y_test.shape  
Out[41]: ((1176, 47), (294, 47), (1176,), (294,))
```

• Model Building

- o Import the model building Libraries
- o Initializing the model
- o Training and testing the model
- o Evaluation of Model
- o Save the Model

Logistic Regression

1.Import the model building Libraries

```
In [42]: from sklearn.linear_model import LogisticRegression
```

2.Initializing the model

```
In [43]: lr=LogisticRegression()
```

3.Training and testing the model

```
In [44]: lr.fit(x_train,y_train)
```

```
Out[44]: LogisticRegression()
```

```
In [45]: y_pred=lr.predict(x_test)
```

```
In [96]: y_pred
```

```
Out[96]: array(['No', 'No', 'No', 'No', 'Yes', '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', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'Yes', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', '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)
```

```
In [97]: y_test
```

```
Out[97]: 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
```

```
In [48]: df.head()
```

Out[48]:

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

5 rows × 35 columns

◀ ▶

4.Evaluation of Model

In [49]: `from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_`

In [50]: `accuracy_score(y_test,y_pred)`

Out[50]: `0.8775510204081632`

In [51]: `confusion_matrix(y_test,y_pred)`

Out[51]: `array([[238, 7],
 [29, 20]], dtype=int64)`

In [52]: `pd.crosstab(y_test,y_pred)`

Out[52]:

	col_0	No	Yes
Attrition			
No	238	7	
Yes	29	20	

Attrition

No 238 7

Yes 29 20

In [53]: `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

In [54]: `probability=lr.predict_proba(x_test)[:,1]`

In [55]: probability

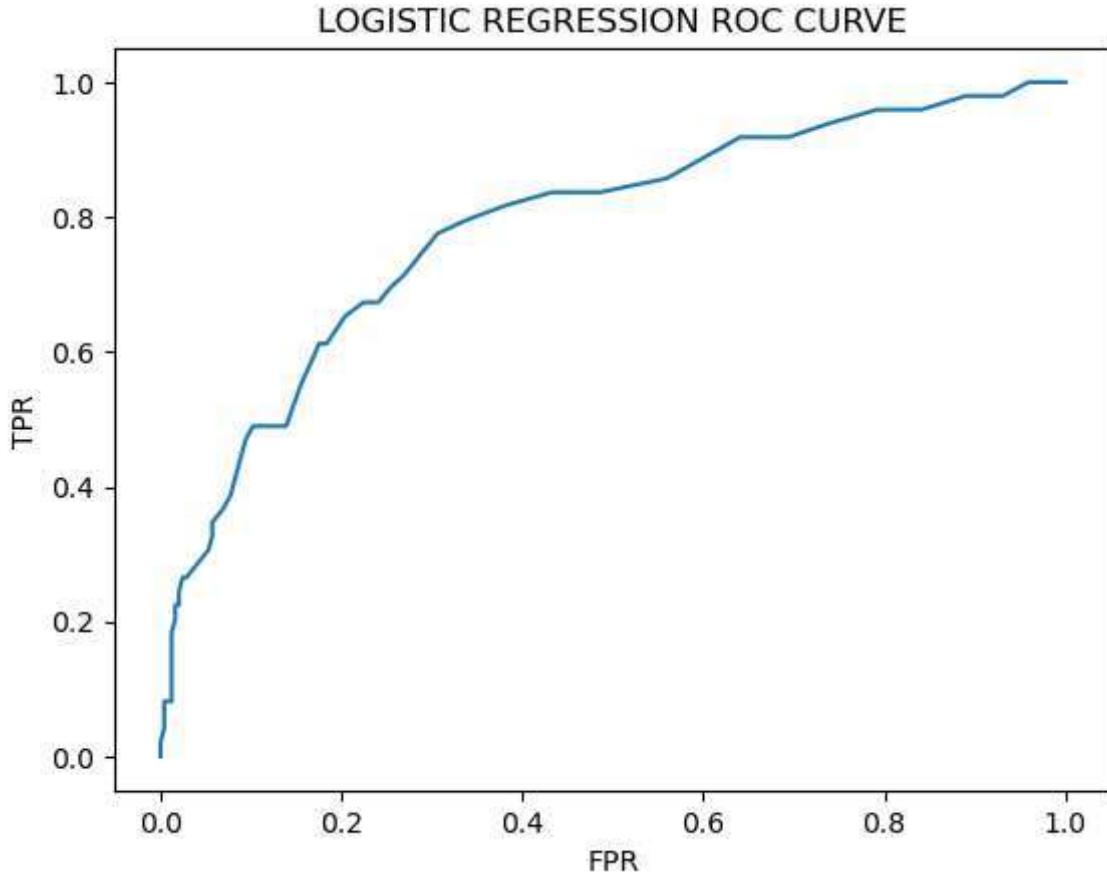
```
Out[55]: array([6.61367680e-02, 8.06198087e-02, 5.95640954e-01, 1.42633628e-01,
 7.84184536e-01, 4.42409704e-02, 4.90141163e-01, 3.64319554e-02,
 8.44925839e-04, 4.30164151e-01, 3.70048030e-02, 1.96283567e-01,
 1.51964558e-02, 5.39471444e-01, 5.88816107e-02, 1.18804547e-02,
 1.13489514e-01, 3.82574810e-02, 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,
 7.65032998e-01, 4.64180665e-02, 1.97677279e-01, 2.06495230e-01,
 3.00475823e-02, 4.28680929e-02, 9.01257179e-02, 1.16204269e-01,
 2.66591825e-01, 9.07332777e-04, 7.47052094e-02, 1.49451887e-03,
```

```
1.41520152e-01, 2.92363790e-01, 5.07029822e-03, 1.26501783e-01,
3.79539275e-02, 1.77150306e-02, 1.54456455e-01, 3.43384106e-01,
4.27481928e-02, 3.44423944e-02, 2.93165700e-01, 7.16660273e-02,
5.38151329e-01, 1.63749013e-02, 1.23951727e-01, 6.52278702e-02,
2.45783680e-03, 6.25002778e-01, 6.25608255e-01, 3.74616723e-01,
1.41256456e-01, 2.84604841e-02, 2.91843045e-01, 5.95069780e-02,
3.21045151e-02, 6.99486994e-02, 1.15155181e-03, 4.47317458e-01,
5.39084197e-01, 2.55697184e-02, 2.85616158e-02, 8.43094503e-03,
6.55595901e-02, 2.48427612e-02, 9.60265542e-03, 5.93422188e-03,
8.58521685e-02, 3.64680251e-01, 1.20449475e-01, 1.84226608e-01,
6.91761349e-01, 3.24455148e-03, 4.85788438e-02, 7.34675868e-02,
5.35355633e-02, 7.58287245e-02, 2.30602991e-04, 1.53563842e-01,
1.43056931e-03, 9.15060156e-03, 1.08956580e-01, 6.50047900e-01,
2.36548949e-02, 7.52606168e-02])
```

```
In [98]: from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
y_test = label_encoder.fit_transform(y_test)
```

```
In [99]: fpr,tpr,thresholds = roc_curve(y_test,probability)
```

```
In [100... plt.plot(fpr,tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('LOGISTIC REGRESSION ROC CURVE')
plt.show()
```



Decision Tree

```
In [57]: from sklearn.tree import DecisionTreeClassifier  
dtc=DecisionTreeClassifier()
```

```
In [58]: dtc.fit(x_train,y_train)
```

```
Out[58]: DecisionTreeClassifier()
```

```
In [59]: y_pred1=dtc.predict(x_test)
```

In [101... y_pred1

```
Out[101]: array(['No', 'No', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'No', 'No',
   'No', 'No', 'No', 'Yes', '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', 'No', 'No', 'No', 'No', 'No', 'No',
   'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'Yes', 'Yes', 'No',
   'No', 'No', 'No', 'No', 'No', 'No', 'Yes', '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', 'No', 'Yes', 'No', 'No',
   'No', 'Yes', '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', 'No', 'No', 'No', 'No',
   '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', 'Yes', 'No', 'Yes', 'No', 'No', 'No', 'Yes',
   'No', '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', 'No', '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', 'No',
   'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No',
   'Yes', 'No', 'Yes',
   'No', 'Yes', 'No', 'No', 'Yes', 'No', 'No', 'Yes', 'No', 'Yes', 'No', 'No'],
  dtype=object)
```

```
In [64]: y_test
```

```
Out[64]:
```

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

```
In [65]: #Accuracy score
```

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc
```

```
In [66]: accuracy_score(y_test,y_pred1)
```

```
Out[66]: 0.7687074829931972
```

```
In [67]: confusion_matrix(y_test,y_pred1)
```

```
Out[67]: array([[210,  35],  
                 [ 33,  16]], dtype=int64)
```

```
In [68]: pd.crosstab(y_test,y_pred1)
```

Out[68]:

Attrition		
No	210	35
Yes	33	16

```
In [69]: print(classification_report(y_test,y_pred1))
```

	precision	recall	f1-score	support
No	0.86	0.86	0.86	245
Yes	0.31	0.33	0.32	49
accuracy			0.77	294
macro avg	0.59	0.59	0.59	294
weighted avg	0.77	0.77	0.77	294

```
In [70]: probability=dtc.predict_proba(x_test)[:,1]
```

```
In [71]: probability
```

Hyper parameter Tuning

```
In [67]: from sklearn import tree  
plt.figure(figsize=(25,15))  
tree.plot_tree(dtc,filled=True)
```

```
Out[67]: [Text(0.32960321576763485, 0.96875, 'X[19] <= -1.257\ngini = 0.269\nsamples = 1176\nvalue = [988, 188']),  
Text(0.08298755186721991, 0.90625, 'X[45] <= 0.387\ngini = 0.5\nsamples = 78\nvalue = [39, 39']),  
Text(0.04979253112033195, 0.84375, 'X[2] <= 0.902\ngini = 0.426\nsamples = 39\nvalue = [27, 12']),  
Text(0.03319502074688797, 0.78125, 'X[26] <= 0.797\ngini = 0.312\nsamples = 31\nvalue = [25, 6']),  
Text(0.01991701244813278, 0.71875, 'X[10] <= -1.114\ngini = 0.198\nsamples = 27\nvalue = [24, 3']),  
Text(0.013278008298755186, 0.65625, 'X[46] <= 0.482\ngini = 0.5\nsamples = 6\nvalue = [3, 3']),  
Text(0.006639004149377593, 0.59375, 'gini = 0.0\nsamples = 3\nvalue = [3, 0']),  
Text(0.01991701244813278, 0.59375, 'gini = 0.0\nsamples = 3\nvalue = [0, 3']),  
Text(0.026556016597510373, 0.65625, 'gini = 0.0\nsamples = 21\nvalue = [21, 0']),  
Text(0.046473029045643155, 0.71875, 'X[6] <= 0.712\ngini = 0.375\nsamples = 4\nvalue = [1, 3']),  
Text(0.03983402489626556, 0.65625, 'gini = 0.0\nsamples = 3\nvalue = [0, 3']),  
Text(0.053112033195020746, 0.65625, 'gini = 0.0\nsamples = 1\nvalue = [1, 0']),  
Text(0.06639004149377593, 0.78125, 'X[5] <= -0.378\ngini = 0.375\nsamples = 8\nvalue = [2, 6']),  
Text(0.05975103734439834, 0.71875, 'gini = 0.0\nsamples = 2\nvalue = [2, 0']),  
Text(0.07302904564315353, 0.71875, 'gini = 0.0\nsamples = 6\nvalue = [0, 6']),  
Text(0.11618257261410789, 0.84375, 'X[41] <= 0.755\ngini = 0.426\nsamples = 39\nvalue = [12, 27']),  
Text(0.0995850622406639, 0.78125, 'X[32] <= 0.397\ngini = 0.26\nsamples = 26\nvalue = [4, 22']),  
Text(0.08630705394190871, 0.71875, 'X[7] <= 1.482\ngini = 0.095\nsamples = 20\nvalue = [1, 19']),  
Text(0.07966804979253111, 0.65625, 'gini = 0.0\nsamples = 18\nvalue = [0, 18']),  
Text(0.09294605809128631, 0.65625, 'X[14] <= 0.763\ngini = 0.5\nsamples = 2\nvalue = [1, 1']),  
Text(0.08630705394190871, 0.59375, 'gini = 0.0\nsamples = 1\nvalue = [1, 0']),  
Text(0.0995850622406639, 0.59375, 'gini = 0.0\nsamples = 1\nvalue = [0, 1']),  
Text(0.11286307053941909, 0.71875, 'X[5] <= -0.401\ngini = 0.5\nsamples = 6\nvalue = [3, 3']),  
Text(0.10622406639004149, 0.65625, 'gini = 0.0\nsamples = 3\nvalue = [3, 0']),  
Text(0.11950207468879669, 0.65625, 'gini = 0.0\nsamples = 3\nvalue = [0, 3']),  
Text(0.13278008298755187, 0.78125, 'X[16] <= -1.122\ngini = 0.473\nsamples = 13\nvalue = [8, 5']),  
Text(0.12614107883817427, 0.71875, 'gini = 0.0\nsamples = 3\nvalue = [0, 3']),  
Text(0.13941908713692946, 0.71875, 'X[7] <= -1.545\ngini = 0.32\nsamples = 10\nvalue = [8, 2']),  
Text(0.13278008298755187, 0.65625, 'gini = 0.0\nsamples = 2\nvalue = [0, 2']),  
Text(0.14605809128630706, 0.65625, 'gini = 0.0\nsamples = 8\nvalue = [8, 0']),  
Text(0.5762188796680497, 0.90625, 'X[46] <= 0.482\ngini = 0.235\nsamples = 1098\nvalue = [949, 149']),  
Text(0.3279564315352697, 0.84375, 'X[21] <= -1.786\ngini = 0.162\nsamples = 798\nvalue = [727, 71']),  
Text(0.17925311203319502, 0.78125, 'X[5] <= -0.173\ngini = 0.38\nsamples = 47\nvalue = [35, 12']),  
Text(0.16597510373443983, 0.71875, 'X[45] <= 0.387\ngini = 0.1\nsamples = 19\nvalue = [18, 1']),  
Text(0.15933609958506223, 0.65625, 'gini = 0.0\nsamples = 18\nvalue = [18, 0']),  
Text(0.17261410788381742, 0.65625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1']),  
Text(0.19253112033195022, 0.71875, 'X[11] <= -0.789\ngini = 0.477\nsamples = 28\nvalue = [17, 11']),  
Text(0.18589211618257262, 0.65625, 'gini = 0.0\nsamples = 4\nvalue = [0, 4']),  
Text(0.1991701244813278, 0.65625, 'X[5] <= 0.099\ngini = 0.413\nsamples = 24\nvalue = [17, 7']),
```

```
Text(0.19253112033195022, 0.59375, 'gini = 0.0\nsamples = 2\nvalue = [0, 2']),
Text(0.20580912863070538, 0.59375, 'X[25] <= 0.386\nngini = 0.351\nsamples = 22\nvalue = [17, 5'],
Text(0.19253112033195022, 0.53125, 'X[1] <= -1.649\nngini = 0.133\nsamples = 14\nvalue = [13, 1'],
Text(0.18589211618257262, 0.46875, 'gini = 0.0\nsamples = 1\nvalue = [0, 1'],
Text(0.1991701244813278, 0.46875, 'gini = 0.0\nsamples = 13\nvalue = [13, 0'],
Text(0.21908713692946058, 0.53125, 'X[1] <= -0.596\nngini = 0.5\nsamples = 8\nvalue = [4, 4'],
Text(0.21244813278008298, 0.46875, 'gini = 0.0\nsamples = 3\nvalue = [0, 3'],
Text(0.22572614107883818, 0.46875, 'X[13] <= 0.523\nngini = 0.32\nsamples = 5\nvalue = [4, 1'],
Text(0.21908713692946058, 0.40625, 'gini = 0.0\nsamples = 4\nvalue = [4, 0'],
Text(0.23236514522821577, 0.40625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1'],
Text(0.4766597510373444, 0.78125, 'X[19] <= 3.564\nngini = 0.145\nsamples = 751\nvalue = [692, 59'],
Text(0.4700207468879668, 0.71875, 'X[22] <= -0.41\nngini = 0.143\nsamples = 750\nvalue = [692, 58'],
Text(0.33443983402489624, 0.65625, 'X[6] <= -1.118\nngini = 0.218\nsamples = 257\nvalue = [225, 32'],
Text(0.30373443983402487, 0.59375, 'X[25] <= -0.455\nngini = 0.355\nsamples = 65\nvalue = [50, 15'],
Text(0.2821576763485477, 0.53125, 'X[25] <= -1.016\nngini = 0.303\nsamples = 59\nvalue = [48, 11'],
Text(0.25892116182572616, 0.46875, 'X[8] <= -0.323\nngini = 0.463\nsamples = 22\nvalue = [14, 8'],
Text(0.24564315352697094, 0.40625, 'X[7] <= -1.151\nngini = 0.198\nsamples = 9\nvalue = [8, 1'],
Text(0.23900414937759337, 0.34375, 'gini = 0.0\nsamples = 1\nvalue = [0, 1'],
Text(0.25228215767634854, 0.34375, 'gini = 0.0\nsamples = 8\nvalue = [8, 0'],
Text(0.2721991701244813, 0.40625, 'X[7] <= -0.388\nngini = 0.497\nsamples = 13\nvalue = [6, 7'],
Text(0.26556016597510373, 0.34375, 'gini = 0.0\nsamples = 4\nvalue = [4, 0'],
Text(0.27883817427385893, 0.34375, 'X[2] <= -0.024\nngini = 0.346\nsamples = 9\nvalue = [2, 7'],
Text(0.2721991701244813, 0.28125, 'X[13] <= 2.325\nngini = 0.444\nsamples = 3\nvalue = [2, 1'],
Text(0.26556016597510373, 0.21875, 'gini = 0.0\nsamples = 2\nvalue = [2, 0'],
Text(0.27883817427385893, 0.21875, 'gini = 0.0\nsamples = 1\nvalue = [0, 1'],
Text(0.2854771784232365, 0.28125, 'gini = 0.0\nsamples = 6\nvalue = [0, 6'],
Text(0.3053941908713693, 0.46875, 'X[10] <= -1.114\nngini = 0.149\nsamples = 37\nvalue = [34, 3'],
Text(0.2987551867219917, 0.40625, 'X[21] <= -0.37\nngini = 0.5\nsamples = 6\nvalue = [3, 3'],
Text(0.2921161825726141, 0.34375, 'gini = 0.0\nsamples = 3\nvalue = [3, 0'],
Text(0.3053941908713693, 0.34375, 'gini = 0.0\nsamples = 3\nvalue = [0, 3'],
Text(0.3120331950207469, 0.40625, 'gini = 0.0\nsamples = 31\nvalue = [31, 0'],
Text(0.3253112033195021, 0.53125, 'X[5] <= -1.479\nngini = 0.444\nsamples = 6\nvalue = [2, 4'],
Text(0.31867219917012446, 0.46875, 'gini = 0.0\nsamples = 2\nvalue = [2, 0'],
Text(0.33195020746887965, 0.46875, 'gini = 0.0\nsamples = 4\nvalue = [0, 4'],
Text(0.3651452282157676, 0.59375, 'X[30] <= 0.178\nngini = 0.161\nsamples = 192\nvalue = [175, 17'],
Text(0.35850622406639004, 0.53125, 'X[21] <= -0.37\nngini = 0.24\nsamples = 122\nvalue = [105, 17'],
Text(0.34522821576763485, 0.46875, 'X[7] <= 0.399\nngini = 0.463\nsamples = 22\nvalue = [14, 8'],
Text(0.3385892116182573, 0.40625, 'X[0] <= -0.156\nngini = 0.444\nsamples = 12\nvalue = [4, 8'],
Text(0.33195020746887965, 0.34375, 'X[14] <= 0.626\nngini = 0.198\nsamples = 9\nvalue
```

```
= [1, 8']),
Text(0.3253112033195021, 0.28125, 'gini = 0.0\nsamples = 8\nvalue = [0, 8']),
Text(0.3385892116182573, 0.28125, 'gini = 0.0\nsamples = 1\nvalue = [1, 0']),
Text(0.34522821576763485, 0.34375, 'gini = 0.0\nsamples = 3\nvalue = [3, 0']),
Text(0.3518672199170125, 0.40625, 'gini = 0.0\nsamples = 10\nvalue = [10, 0']),
Text(0.37178423236514524, 0.46875, 'X[1] <= -1.711\ngini = 0.164\nsamples = 100\nvalue = [91, 9']),
Text(0.3651452282157676, 0.40625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]),
Text(0.3784232365145228, 0.40625, 'X[5] <= -1.627\ngini = 0.149\nsamples = 99\nvalue = [91, 8']),
Text(0.35933609958506224, 0.34375, 'X[19] <= -0.486\ngini = 0.5\nsamples = 4\nvalue = [2, 2]),
Text(0.35269709543568467, 0.28125, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]),
Text(0.3659751037344398, 0.28125, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]),
Text(0.3975103734439834, 0.34375, 'X[1] <= 1.528\ngini = 0.118\nsamples = 95\nvalue = [89, 6]),
Text(0.379253112033195, 0.28125, 'X[3] <= 1.55\ngini = 0.086\nsamples = 89\nvalue = [85, 4]),
Text(0.362655601659751, 0.21875, 'X[0] <= 2.144\ngini = 0.047\nsamples = 83\nvalue = [81, 2]),
Text(0.34937759336099583, 0.15625, 'X[8] <= -1.729\ngini = 0.024\nsamples = 81\nvalue = [80, 1]),
Text(0.34273858921161826, 0.09375, 'X[19] <= -1.129\ngini = 0.32\nsamples = 5\nvalue = [4, 1]),
Text(0.3360995850622407, 0.03125, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]),
Text(0.34937759336099583, 0.03125, 'gini = 0.0\nsamples = 4\nvalue = [4, 0]),
Text(0.35601659751037346, 0.09375, 'gini = 0.0\nsamples = 76\nvalue = [76, 0]),
Text(0.3759336099585062, 0.15625, 'X[14] <= 1.856\ngini = 0.5\nsamples = 2\nvalue = [1, 1]),
Text(0.36929460580912865, 0.09375, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]),
Text(0.3825726141078838, 0.09375, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]),
Text(0.395850622406639, 0.21875, 'X[12] <= 0.348\ngini = 0.444\nsamples = 6\nvalue = [4, 2]),
Text(0.3892116182572614, 0.15625, 'gini = 0.0\nsamples = 4\nvalue = [4, 0]),
Text(0.4024896265560166, 0.15625, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]),
Text(0.4157676348547718, 0.28125, 'X[22] <= -0.573\ngini = 0.444\nsamples = 6\nvalue = [4, 2]),
Text(0.4091286307053942, 0.21875, 'gini = 0.0\nsamples = 4\nvalue = [4, 0]),
Text(0.4224066390041494, 0.21875, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]),
Text(0.37178423236514524, 0.53125, 'gini = 0.0\nsamples = 70\nvalue = [70, 0]),
Text(0.6056016597510373, 0.65625, 'X[22] <= 3.999\ngini = 0.1\nsamples = 493\nvalue = [467, 26]),
Text(0.5697095435684647, 0.59375, 'X[10] <= -0.207\ngini = 0.094\nsamples = 486\nvalue = [462, 24]),
Text(0.5178423236514523, 0.53125, 'X[43] <= 1.922\ngini = 0.154\nsamples = 191\nvalue = [175, 16]),
Text(0.5112033195020746, 0.46875, 'X[12] <= -0.035\ngini = 0.145\nsamples = 190\nvalue = [175, 15]),
Text(0.4871369294605809, 0.40625, 'X[12] <= -0.073\ngini = 0.221\nsamples = 95\nvalue = [83, 12]),
Text(0.48049792531120333, 0.34375, 'X[25] <= 2.629\ngini = 0.207\nsamples = 94\nvalue = [83, 11]),
Text(0.4738589211618257, 0.28125, 'X[26] <= 0.797\ngini = 0.192\nsamples = 93\nvalue = [83, 10]),
Text(0.43900414937759336, 0.21875, 'X[11] <= 0.439\ngini = 0.124\nsamples = 75\nvalue = [70, 5]),
Text(0.4157676348547718, 0.15625, 'X[39] <= 1.346\ngini = 0.037\nsamples = 53\nvalue = [52, 1]),
Text(0.4091286307053942, 0.09375, 'gini = 0.0\nsamples = 45\nvalue = [45, 0]),
Text(0.4224066390041494, 0.09375, 'X[16] <= -1.122\ngini = 0.219\nsamples = 8\nvalue = [8, 0])
```

```

= [7, 1']),
Text(0.4157676348547718, 0.03125, 'gini = 0.0\nsamples = 1\nvalue = [0, 1']),
Text(0.42904564315352695, 0.03125, 'gini = 0.0\nsamples = 7\nvalue = [7, 0']),
Text(0.46224066390041496, 0.15625, 'X[3] <= -0.403\ngini = 0.298\nsamples = 22\nvalue = [18, 4']),
Text(0.44896265560165977, 0.09375, 'X[23] <= 1.593\ngini = 0.375\nsamples = 4\nvalue = [1, 3']),
Text(0.44232365145228214, 0.03125, 'gini = 0.0\nsamples = 3\nvalue = [0, 3']),
Text(0.45560165975103734, 0.03125, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]),
Text(0.4755186721991701, 0.09375, 'X[2] <= -0.949\ngini = 0.105\nsamples = 18\nvalue = [17, 1']),
Text(0.46887966804979253, 0.03125, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]),
Text(0.4821576763485477, 0.03125, 'gini = 0.0\nsamples = 17\nvalue = [17, 0]),
Text(0.5087136929460581, 0.21875, 'X[45] <= 0.387\ngini = 0.401\nsamples = 18\nvalue = [13, 5]),
Text(0.4954356846473029, 0.15625, 'X[23] <= -0.615\ngini = 0.142\nsamples = 13\nvalue = [12, 1]),
Text(0.4887966804979253, 0.09375, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]),
Text(0.5020746887966805, 0.09375, 'gini = 0.0\nsamples = 12\nvalue = [12, 0]),
Text(0.5219917012448133, 0.15625, 'X[7] <= -0.585\ngini = 0.32\nsamples = 5\nvalue = [1, 4]),
Text(0.5153526970954356, 0.09375, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]),
Text(0.5286307053941909, 0.09375, 'gini = 0.0\nsamples = 4\nvalue = [0, 4]),
Text(0.4871369294605809, 0.28125, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]),
Text(0.49377593360995853, 0.34375, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]),
Text(0.5352697095435685, 0.40625, 'X[13] <= 0.724\ngini = 0.061\nsamples = 95\nvalue = [92, 3]),
Text(0.5286307053941909, 0.34375, 'gini = 0.0\nsamples = 76\nvalue = [76, 0]),
Text(0.541908713692946, 0.34375, 'X[5] <= -1.149\ngini = 0.266\nsamples = 19\nvalue = [16, 3]),
Text(0.5286307053941909, 0.28125, 'X[1] <= -1.404\ngini = 0.444\nsamples = 3\nvalue = [1, 2]),
Text(0.5219917012448133, 0.21875, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]),
Text(0.5352697095435685, 0.21875, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]),
Text(0.5551867219917013, 0.28125, 'X[23] <= 2.007\ngini = 0.117\nsamples = 16\nvalue = [15, 1]),
Text(0.5485477178423237, 0.21875, 'gini = 0.0\nsamples = 14\nvalue = [14, 0]),
Text(0.5618257261410788, 0.21875, 'X[9] <= 1.298\ngini = 0.5\nsamples = 2\nvalue = [1, 1]),
Text(0.5551867219917013, 0.15625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]),
Text(0.5684647302904564, 0.15625, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]),
Text(0.5244813278008299, 0.46875, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]),
Text(0.6215767634854772, 0.53125, 'X[14] <= -1.014\ngini = 0.053\nsamples = 295\nvalue = [287, 8]),
Text(0.5983402489626556, 0.46875, 'X[24] <= 2.58\ngini = 0.159\nsamples = 46\nvalue = [42, 4]),
Text(0.591701244813278, 0.40625, 'X[7] <= -1.52\ngini = 0.124\nsamples = 45\nvalue = [42, 3]),
Text(0.575103734439834, 0.34375, 'X[23] <= 0.903\ngini = 0.5\nsamples = 2\nvalue = [1, 1]),
Text(0.5684647302904564, 0.28125, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]),
Text(0.5817427385892117, 0.28125, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]),
Text(0.6082987551867219, 0.34375, 'X[19] <= 2.085\ngini = 0.089\nsamples = 43\nvalue = [41, 2]),
Text(0.5950207468879668, 0.28125, 'X[2] <= 0.593\ngini = 0.048\nsamples = 41\nvalue = [40, 1]),
Text(0.5883817427385892, 0.21875, 'gini = 0.0\nsamples = 34\nvalue = [34, 0]),
Text(0.6016597510373444, 0.21875, 'X[2] <= 0.84\ngini = 0.245\nsamples = 7\nvalue = [6, 1]),
Text(0.5950207468879668, 0.15625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]),

```

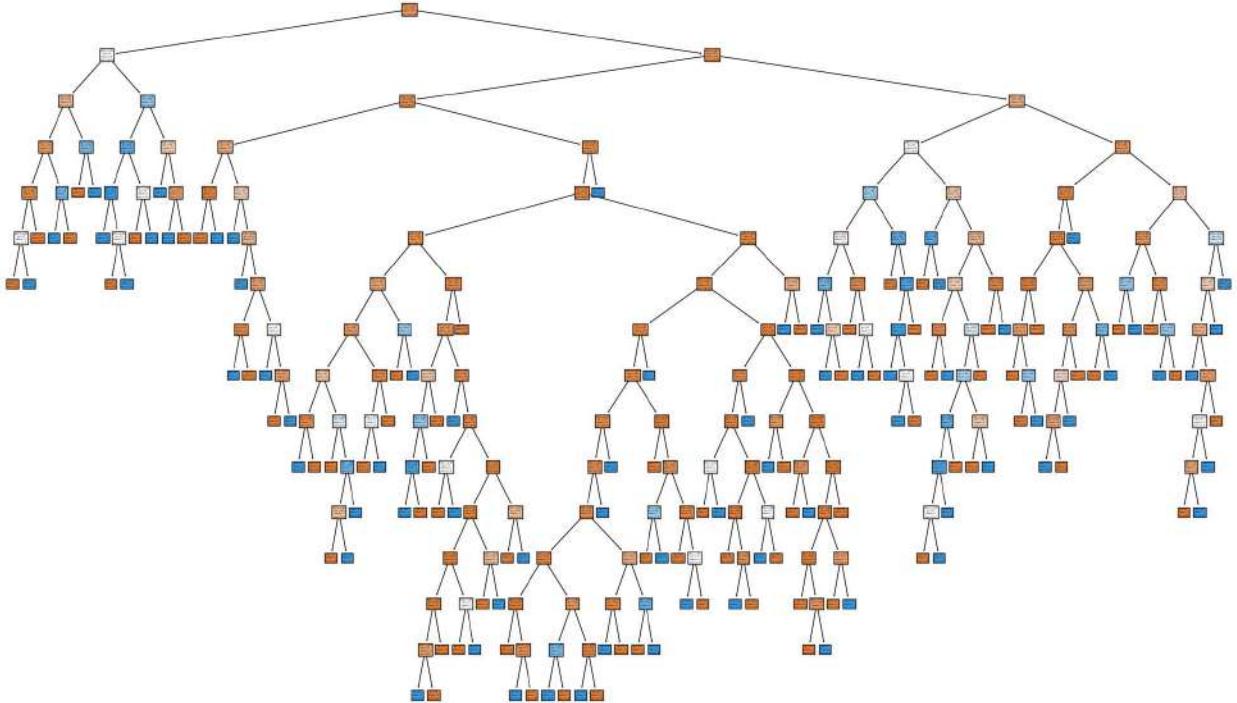
```
Text(0.6082987551867219, 0.15625, 'gini = 0.0\nsamples = 6\nvalue = [6, 0']),
Text(0.6215767634854772, 0.28125, 'X[9] <= 1.298\ngini = 0.5\nsamples = 2\nvalue =
[1, 1']),
Text(0.6149377593360996, 0.21875, 'gini = 0.0\nsamples = 1\nvalue = [0, 1']),
Text(0.6282157676348548, 0.21875, 'gini = 0.0\nsamples = 1\nvalue = [1, 0']),
Text(0.6049792531120332, 0.40625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1']),
Text(0.6448132780082988, 0.46875, 'X[11] <= -0.943\ngini = 0.032\nsamples = 249\nvalue =
[245, 4']),
Text(0.6282157676348548, 0.40625, 'X[5] <= -0.566\ngini = 0.32\nsamples = 5\nvalue =
[4, 1']),
Text(0.6215767634854772, 0.34375, 'gini = 0.0\nsamples = 1\nvalue = [0, 1']),
Text(0.6348547717842323, 0.34375, 'gini = 0.0\nsamples = 4\nvalue = [4, 0']),
Text(0.6614107883817427, 0.40625, 'X[1] <= -1.685\ngini = 0.024\nsamples = 244\nvalue =
[241, 3']),
Text(0.6481327800829876, 0.34375, 'X[12] <= 0.787\ngini = 0.278\nsamples = 6\nvalue =
[5, 1']),
Text(0.64149377593361, 0.28125, 'gini = 0.0\nsamples = 5\nvalue = [5, 0']),
Text(0.6547717842323652, 0.28125, 'gini = 0.0\nsamples = 1\nvalue = [0, 1']),
Text(0.674688796680498, 0.34375, 'X[16] <= -1.122\ngini = 0.017\nsamples = 238\nvalue =
[236, 2']),
Text(0.6680497925311203, 0.28125, 'X[21] <= 1.046\ngini = 0.073\nsamples = 53\nvalue =
[51, 2']),
Text(0.6547717842323652, 0.21875, 'X[37] <= 0.85\ngini = 0.041\nsamples = 48\nvalue =
[47, 1']),
Text(0.6481327800829876, 0.15625, 'gini = 0.0\nsamples = 42\nvalue = [42, 0']),
Text(0.6614107883817427, 0.15625, 'X[14] <= 0.079\ngini = 0.278\nsamples = 6\nvalue =
[5, 1']),
Text(0.6547717842323652, 0.09375, 'gini = 0.0\nsamples = 5\nvalue = [5, 0']),
Text(0.6680497925311203, 0.09375, 'gini = 0.0\nsamples = 1\nvalue = [0, 1']),
Text(0.6813278008298755, 0.21875, 'X[14] <= 0.216\ngini = 0.32\nsamples = 5\nvalue =
[4, 1']),
Text(0.674688796680498, 0.15625, 'gini = 0.0\nsamples = 4\nvalue = [4, 0']),
Text(0.6879668049792531, 0.15625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1']),
Text(0.6813278008298755, 0.28125, 'gini = 0.0\nsamples = 185\nvalue = [185, 0']),
Text(0.64149377593361, 0.59375, 'X[35] <= -0.204\ngini = 0.408\nsamples = 7\nvalue =
[5, 2']),
Text(0.6348547717842323, 0.53125, 'gini = 0.0\nsamples = 2\nvalue = [0, 2']),
Text(0.6481327800829876, 0.53125, 'gini = 0.0\nsamples = 5\nvalue = [5, 0']),
Text(0.483298755186722, 0.71875, 'gini = 0.0\nsamples = 1\nvalue = [0, 1']),
Text(0.8244813278008298, 0.84375, 'X[11] <= -0.533\ngini = 0.385\nsamples = 300\nvalue =
[222, 78']),
Text(0.7385892116182573, 0.78125, 'X[18] <= -0.345\ngini = 0.5\nsamples = 96\nvalue =
[49, 47']),
Text(0.7045643153526971, 0.71875, 'X[2] <= -0.456\ngini = 0.459\nsamples = 42\nvalue =
[15, 27']),
Text(0.6813278008298755, 0.65625, 'X[5] <= -0.275\ngini = 0.499\nsamples = 23\nvalue =
[12, 11']),
Text(0.6680497925311203, 0.59375, 'X[12] <= 0.245\ngini = 0.355\nsamples = 13\nvalue =
[3, 10']),
Text(0.6614107883817427, 0.53125, 'gini = 0.0\nsamples = 8\nvalue = [0, 8']),
Text(0.674688796680498, 0.53125, 'X[6] <= -0.661\ngini = 0.48\nsamples = 5\nvalue =
[3, 2']),
Text(0.6680497925311203, 0.46875, 'gini = 0.0\nsamples = 2\nvalue = [0, 2']),
Text(0.6813278008298755, 0.46875, 'gini = 0.0\nsamples = 3\nvalue = [3, 0']),
Text(0.6946058091286307, 0.59375, 'X[43] <= 1.922\ngini = 0.18\nsamples = 10\nvalue =
[9, 1']),
Text(0.6879668049792531, 0.53125, 'gini = 0.0\nsamples = 8\nvalue = [8, 0']),
Text(0.7012448132780082, 0.53125, 'X[0] <= -0.375\ngini = 0.5\nsamples = 2\nvalue =
[1, 1']),
Text(0.6946058091286307, 0.46875, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]),
```

```
Text(0.7078838174273859, 0.46875, 'gini = 0.0\nsamples = 1\nvalue = [1, 0']),
Text(0.7278008298755186, 0.65625, 'X[7] <= -1.077\ngini = 0.266\nsamples = 19\nvalue
= [3, 16']'),
Text(0.7211618257261411, 0.59375, 'gini = 0.0\nsamples = 1\nvalue = [1, 0']),
Text(0.7344398340248963, 0.59375, 'X[19] <= 0.35\ngini = 0.198\nsamples = 18\nvalue
= [2, 16']'),
Text(0.7278008298755186, 0.53125, 'X[24] <= 1.339\ngini = 0.111\nsamples = 17\nvalue
= [1, 16']'),
Text(0.7211618257261411, 0.46875, 'gini = 0.0\nsamples = 15\nvalue = [0, 15']'),
Text(0.7344398340248963, 0.46875, 'X[1] <= -1.003\ngini = 0.5\nsamples = 2\nvalue =
[1, 1']'),
Text(0.7278008298755186, 0.40625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1']'),
Text(0.7410788381742739, 0.40625, 'gini = 0.0\nsamples = 1\nvalue = [1, 0']'),
Text(0.7410788381742739, 0.53125, 'gini = 0.0\nsamples = 1\nvalue = [1, 0']'),
Text(0.7726141078838175, 0.71875, 'X[0] <= -1.141\ngini = 0.466\nsamples = 54\nvalue
= [34, 20']'),
Text(0.754356846473029, 0.65625, 'X[5] <= -1.14\ngini = 0.245\nsamples = 7\nvalue =
[1, 6']'),
Text(0.7477178423236515, 0.59375, 'gini = 0.0\nsamples = 1\nvalue = [1, 0']'),
Text(0.7609958506224066, 0.59375, 'gini = 0.0\nsamples = 6\nvalue = [0, 6']'),
Text(0.7908713692946058, 0.65625, 'X[1] <= 0.419\ngini = 0.418\nsamples = 47\nvalue
= [33, 14']'),
Text(0.7742738589211619, 0.59375, 'X[1] <= -1.236\ngini = 0.482\nsamples = 32\nvalue
= [19, 13']'),
Text(0.7609958506224066, 0.53125, 'X[22] <= 0.407\ngini = 0.18\nsamples = 10\nvalue
= [9, 1']'),
Text(0.754356846473029, 0.46875, 'gini = 0.0\nsamples = 9\nvalue = [9, 0']'),
Text(0.7676348547717843, 0.46875, 'gini = 0.0\nsamples = 1\nvalue = [0, 1']'),
Text(0.787551867219917, 0.53125, 'X[12] <= 1.329\ngini = 0.496\nsamples = 22\nvalue
= [10, 12']'),
Text(0.7809128630705394, 0.46875, 'X[41] <= 0.755\ngini = 0.465\nsamples = 19\nvalue
= [7, 12']'),
Text(0.7676348547717843, 0.40625, 'X[7] <= 1.531\ngini = 0.298\nsamples = 11\nvalue
= [2, 9']'),
Text(0.7609958506224066, 0.34375, 'X[12] <= -1.011\ngini = 0.18\nsamples = 10\nvalue
= [1, 9']'),
Text(0.754356846473029, 0.28125, 'X[20] <= -0.232\ngini = 0.5\nsamples = 2\nvalue =
[1, 1']'),
Text(0.7477178423236515, 0.21875, 'gini = 0.0\nsamples = 1\nvalue = [1, 0']'),
Text(0.7609958506224066, 0.21875, 'gini = 0.0\nsamples = 1\nvalue = [0, 1']'),
Text(0.7676348547717843, 0.28125, 'gini = 0.0\nsamples = 8\nvalue = [0, 8']'),
Text(0.7742738589211619, 0.34375, 'gini = 0.0\nsamples = 1\nvalue = [1, 0']'),
Text(0.7941908713692946, 0.40625, 'X[14] <= -0.057\ngini = 0.469\nsamples = 8\nvalue
= [5, 3']'),
Text(0.787551867219917, 0.34375, 'gini = 0.0\nsamples = 5\nvalue = [5, 0']'),
Text(0.8008298755186722, 0.34375, 'gini = 0.0\nsamples = 3\nvalue = [0, 3']'),
Text(0.7941908713692946, 0.46875, 'gini = 0.0\nsamples = 3\nvalue = [3, 0']'),
Text(0.8074688796680498, 0.59375, 'X[13] <= 2.325\ngini = 0.124\nsamples = 15\nvalue
= [14, 1']'),
Text(0.8008298755186722, 0.53125, 'gini = 0.0\nsamples = 14\nvalue = [14, 0']'),
Text(0.8141078838174274, 0.53125, 'gini = 0.0\nsamples = 1\nvalue = [0, 1']'),
Text(0.9103734439834025, 0.78125, 'X[45] <= 0.387\ngini = 0.258\nsamples = 204\nvalu
e = [173, 31']'),
Text(0.8639004149377594, 0.71875, 'X[11] <= 2.837\ngini = 0.138\nsamples = 147\nvalu
e = [136, 11']'),
Text(0.8572614107883817, 0.65625, 'X[2] <= 0.655\ngini = 0.128\nsamples = 146\nvalue
= [136, 10']'),
Text(0.8340248962655602, 0.59375, 'X[22] <= -0.736\ngini = 0.038\nsamples = 104\nvalu
e = [102, 2']'),
Text(0.8273858921161825, 0.53125, 'X[32] <= 0.397\ngini = 0.32\nsamples = 10\nvalue
```

```

= [8, 2']),
Text(0.820746887966805, 0.46875, 'gini = 0.0\nsamples = 7\nvalue = [7, 0']),
Text(0.8340248962655602, 0.46875, 'X[1] <= 0.274\ngini = 0.444\nsamples = 3\nvalue =
[1, 2']),
Text(0.8273858921161825, 0.40625, 'gini = 0.0\nsamples = 1\nvalue = [1, 0']),
Text(0.8406639004149378, 0.40625, 'gini = 0.0\nsamples = 2\nvalue = [0, 2']),
Text(0.8406639004149378, 0.53125, 'gini = 0.0\nsamples = 94\nvalue = [94, 0']),
Text(0.8804979253112033, 0.59375, 'X[31] <= 1.262\ngini = 0.308\nsamples = 42\nvalue
= [34, 8']),
Text(0.8672199170124482, 0.53125, 'X[6] <= -0.203\ngini = 0.229\nsamples = 38\nvalue
= [33, 5']),
Text(0.8605809128630706, 0.46875, 'X[13] <= 0.924\ngini = 0.486\nsamples = 12\nvalue
= [7, 5']),
Text(0.8539419087136929, 0.40625, 'X[5] <= -1.067\ngini = 0.346\nsamples = 9\nvalue
= [7, 2']),
Text(0.8473029045643153, 0.34375, 'gini = 0.0\nsamples = 2\nvalue = [0, 2']),
Text(0.8605809128630706, 0.34375, 'gini = 0.0\nsamples = 7\nvalue = [7, 0']),
Text(0.8672199170124482, 0.40625, 'gini = 0.0\nsamples = 3\nvalue = [0, 3']),
Text(0.8738589211618257, 0.46875, 'gini = 0.0\nsamples = 26\nvalue = [26, 0]),
Text(0.8937759336099586, 0.53125, 'X[2] <= 1.21\ngini = 0.375\nsamples = 4\nvalue =
[1, 3']),
Text(0.8871369294605809, 0.46875, 'gini = 0.0\nsamples = 1\nvalue = [1, 0']),
Text(0.9004149377593361, 0.46875, 'gini = 0.0\nsamples = 3\nvalue = [0, 3']),
Text(0.870539419087137, 0.65625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1']),
Text(0.9568464730290457, 0.71875, 'X[42] <= 0.67\ngini = 0.456\nsamples = 57\nvalue
= [37, 20']),
Text(0.9269709543568465, 0.65625, 'X[5] <= -1.458\ngini = 0.238\nsamples = 29\nvalue
= [25, 4']),
Text(0.9136929460580913, 0.59375, 'X[5] <= -1.637\ngini = 0.444\nsamples = 3\nvalue
= [1, 2']),
Text(0.9070539419087137, 0.53125, 'gini = 0.0\nsamples = 1\nvalue = [1, 0']),
Text(0.9203319502074688, 0.53125, 'gini = 0.0\nsamples = 2\nvalue = [0, 2']),
Text(0.9402489626556016, 0.59375, 'X[24] <= 1.183\ngini = 0.142\nsamples = 26\nvalue
= [24, 2']),
Text(0.9336099585062241, 0.53125, 'gini = 0.0\nsamples = 23\nvalue = [23, 0]),
Text(0.9468879668049792, 0.53125, 'X[1] <= -0.616\ngini = 0.444\nsamples = 3\nvalue
= [1, 2']),
Text(0.9402489626556016, 0.46875, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]),
Text(0.9535269709543569, 0.46875, 'gini = 0.0\nsamples = 1\nvalue = [1, 0']),
Text(0.9867219917012449, 0.65625, 'X[24] <= -0.214\ngini = 0.49\nsamples = 28\nvalue
= [12, 16']),
Text(0.9800829875518672, 0.59375, 'X[2] <= 1.765\ngini = 0.48\nsamples = 20\nvalue =
[12, 8']),
Text(0.9734439834024896, 0.53125, 'X[22] <= -1.063\ngini = 0.415\nsamples = 17\nvalue
= [12, 5']),
Text(0.966804979253112, 0.46875, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]),
Text(0.9800829875518672, 0.46875, 'X[16] <= -0.196\ngini = 0.32\nsamples = 15\nvalue
= [12, 3']),
Text(0.9734439834024896, 0.40625, 'X[13] <= 0.724\ngini = 0.5\nsamples = 6\nvalue =
[3, 3']),
Text(0.966804979253112, 0.34375, 'X[3] <= 0.573\ngini = 0.375\nsamples = 4\nvalue =
[3, 1']),
Text(0.9601659751037345, 0.28125, 'gini = 0.0\nsamples = 3\nvalue = [3, 0']),
Text(0.9734439834024896, 0.28125, 'gini = 0.0\nsamples = 1\nvalue = [0, 1']),
Text(0.9800829875518672, 0.34375, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]),
Text(0.9867219917012449, 0.40625, 'gini = 0.0\nsamples = 9\nvalue = [9, 0]),
Text(0.9867219917012449, 0.53125, 'gini = 0.0\nsamples = 3\nvalue = [0, 3]),
Text(0.9933609958506224, 0.59375, 'gini = 0.0\nsamples = 8\nvalue = [0, 8]')

```



```
In [72]: 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']
}
```

```
In [73]: grid_search=GridSearchCV(estimator=dtc,param_grid=parameter,cv=5,scoring="accuracy")
```

```
In [74]: grid_search.fit(x_train,y_train)
```

```
Out[74]: GridSearchCV(cv=5, estimator=DecisionTreeClassifier(),
param_grid={'criterion': ['gini', 'entropy'],
'max_depth': [1, 2, 3, 4, 5],
'max_features': ['auto', 'sqrt', 'log2'],
'splitter': ['best', 'random']},
scoring='accuracy')
```

```
In [75]: grid_search.best_params_
```

```
Out[75]: {'criterion': 'entropy',
'max_depth': 4,
'max_features': 'auto',
'splitter': 'best'}
```

```
In [76]: dtc_cv=DecisionTreeClassifier(criterion= 'entropy',
max_depth=3,
max_features='sqrt',
splitter='best')
dtc_cv.fit(x_train,y_train)
```

```
Out[76]: DecisionTreeClassifier(criterion='entropy', max_depth=3, max_features='sqrt')
```

```
In [77]: y_pred=dtc_cv.predict(x_test)
```

```
In [78]: print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
No	0.84	0.98	0.90	245
Yes	0.33	0.04	0.07	49
accuracy			0.83	294
macro avg	0.59	0.51	0.49	294
weighted avg	0.75	0.83	0.77	294

Random Forest

```
In [79]: from sklearn.ensemble import RandomForestClassifier  
rfc=RandomForestClassifier()
```

```
In [80]: rfc.fit(x_train,y_train)
```

```
Out[80]: RandomForestClassifier()
```

```
In [81]: y_pred2=dtc.predict(x_test)
```

```
In [82]: y pred2
```

```
In [83]: y_test
```

```
Out[83]: 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
```

```
In [84]: forest_params = [{ 'max_depth': list(range(10, 15)), 'max_features': list(range(0,14))}]
```

```
In [85]: rfc_cv=GridSearchCV(rfc,param_grid=forest_params,cv=10,scoring='accuracy')
```

```
In [86]: rfc_cv.fit(x_train,y_train)
```

```
C:\Users\admin\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:37
2: FitFailedWarning:
50 fits failed out of a total of 700.
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:
-----
50 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\admin\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\admin\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py", line 450, in fit
    trees = Parallel(
      File "C:\Users\admin\anaconda3\lib\site-packages\joblib\parallel.py", line 1863, in __call__
        return output if self.return_generator else list(output)
  File "C:\Users\admin\anaconda3\lib\site-packages\joblib\parallel.py", line 1792, in _get_sequential_output
    res = func(*args, **kwargs)
  File "C:\Users\admin\anaconda3\lib\site-packages\sklearn\utils\fixes.py", line 216, in __call__
    return self.function(*args, **kwargs)
  File "C:\Users\admin\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py", line 185, in _parallel_build_trees
    tree.fit(X, y, sample_weight=curr_sample_weight, check_input=False)
  File "C:\Users\admin\anaconda3\lib\site-packages\sklearn\tree\_classes.py", line 937, in fit
    super().fit(
      File "C:\Users\admin\anaconda3\lib\site-packages\sklearn\tree\_classes.py", line 308, in fit
      raise ValueError("max_features must be in (0, n_features]")
ValueError: max_features must be in (0, n_features)

    warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\admin\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:969: UserWarning: One or more of the test scores are non-finite: [       nan  0.84948573  0.84950022  0.85034043  0.85459945  0.85801101
  0.85628712  0.8605389  0.85885122  0.85798204  0.86052441  0.8596842
  0.86138635  0.85796031       nan  0.84693611  0.85290453  0.85714907
  0.85882949  0.85884398  0.85799652  0.85798204  0.85543242  0.8596842
  0.8596842  0.85883674  0.8613936  0.85542518       nan  0.84948573
  0.85204983  0.85630161  0.85798928  0.85628712  0.85543966  0.86222657
  0.85883674  0.86309576  0.86053165  0.86223381  0.86308127  0.85797479
       nan  0.84779806  0.85204983  0.85800377  0.85883674  0.86055338
  0.85800377  0.86052441  0.8605389  0.85712009  0.85797479  0.85798928
  0.86053165  0.86137187       nan  0.84863827  0.85545415  0.85629436
  0.85884398  0.85798928  0.85373751  0.85799652  0.85799652  0.85712734
  0.86140808  0.85712734  0.86222657  0.85967695]
    warnings.warn(
Out[86]: GridSearchCV(cv=10, estimator=RandomForestClassifier(),
                     param_grid=[{'max_depth': [10, 11, 12, 13, 14],
                                  'max_features': [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
                                                      12, 13]}],
                     scoring='accuracy')
```

```
In [87]: y_pred=rfc_cv.predict(x_test)
```

```
In [88]: from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_
```

```
In [89]: accuracy_score(y_test,y_pred2)
```

```
Out[89]: 0.7687074829931972
```

```
In [90]: confusion_matrix(y_test,y_pred2)
```

```
Out[90]: array([[210, 35],  
                 [ 33, 16]], dtype=int64)
```

```
In [91]: pd.crosstab(y_test,y_pred2)
```

```
Out[91]:   col_0  No  Yes
```

Attrition

No	210	35
----	-----	----

Yes	33	16
-----	----	----

```
In [92]: print(classification_report(y_test,y_pred2))
```

	precision	recall	f1-score	support
No	0.86	0.86	0.86	245
Yes	0.31	0.33	0.32	49
accuracy			0.77	294
macro avg	0.59	0.59	0.59	294
weighted avg	0.77	0.77	0.77	294

```
In [93]: rfc_cv.best_params_
```

```
Out[93]: {'max_depth': 12, 'max_features': 9}
```

```
In [94]: probability=rfc.predict_proba(x_test)[:,1]
```

```
In [95]: probability
```

```
Out[95]: array([0.13, 0.05, 0.18, 0.15, 0.71, 0.36, 0.3 , 0.11, 0.11, 0.13, 0.06,
 0.11, 0.07, 0.47, 0.1 , 0.02, 0.16, 0.08, 0.11, 0.16, 0.56, 0.07,
 0.05, 0.11, 0.39, 0.21, 0.06, 0.12, 0.57, 0.07, 0.06, 0.13, 0.13,
 0.16, 0.07, 0.09, 0.16, 0.14, 0.03, 0.25, 0.19, 0.08, 0.11, 0.2 ,
 0.05, 0.36, 0.31, 0.08, 0.65, 0.33, 0.26, 0.48, 0.14, 0.21, 0.41,
 0.14, 0.14, 0.1 , 0.05, 0.26, 0.03, 0.16, 0.11, 0.15, 0.37, 0.12,
 0.21, 0.14, 0.09, 0.16, 0.1 , 0.3 , 0.12, 0.09, 0.1 , 0.11, 0.1 ,
 0.04, 0.39, 0.05, 0.05, 0.11, 0.23, 0.1 , 0.18, 0.08, 0.09, 0.25,
 0.02, 0.04, 0.65, 0.1 , 0.13, 0.26, 0.06, 0.07, 0.1 , 0.32, 0.14,
 0.13, 0.27, 0.24, 0.36, 0.05, 0.07, 0.03, 0.09, 0.47, 0.23, 0.27,
 0.2 , 0.12, 0.04, 0.07, 0.15, 0.16, 0.1 , 0.19, 0.04, 0.01, 0.11,
 0.03, 0.04, 0.58, 0.12, 0.16, 0.02, 0.06, 0.11, 0.12, 0.01, 0.29,
 0.5 , 0.21, 0.18, 0.14, 0.35, 0.2 , 0.14, 0.16, 0.1 , 0.24, 0.09,
 0.15, 0.24, 0.17, 0.06, 0.05, 0.15, 0.15, 0.1 , 0.23, 0.13, 0.23,
 0.15, 0.12, 0.11, 0.21, 0.12, 0.09, 0.23, 0.07, 0.28, 0.51, 0.1 ,
 0.14, 0.23, 0.06, 0.11, 0.04, 0.12, 0.05, 0.1 , 0.25, 0.07, 0.43,
 0.12, 0.25, 0.21, 0.09, 0.04, 0.02, 0.01, 0.37, 0.07, 0.03, 0.37,
 0.07, 0.2 , 0.13, 0.24, 0.5 , 0.2 , 0.04, 0.1 , 0.08, 0.06, 0.04,
 0.46, 0.12, 0.29, 0.19, 0.16, 0.04, 0.11, 0.17, 0.35, 0.07, 0.08,
 0.01, 0.1 , 0.16, 0.09, 0.23, 0.1 , 0.08, 0.06, 0.25, 0.11, 0.37,
 0.17, 0.36, 0.38, 0.17, 0.11, 0.08, 0.13, 0.27, 0.52, 0.12, 0.03,
 0.31, 0.05, 0.08, 0.05, 0.26, 0.28, 0.03, 0.08, 0.1 , 0.21, 0.13,
 0.1 , 0.13, 0.15, 0.09, 0.05, 0.32, 0.06, 0.15, 0.2 , 0.14, 0.24,
 0.07, 0.18, 0.14, 0.08, 0.61, 0.23, 0.31, 0.16, 0.08, 0.13, 0.11,
 0.1 , 0.22, 0.06, 0.28, 0.24, 0.03, 0.06, 0.02, 0.09, 0.1 , 0.07,
 0.05, 0.13, 0.32, 0.09, 0.19, 0.32, 0.1 , 0.11, 0.24, 0.17, 0.22,
 0.01, 0.1 , 0.1 , 0.12, 0.13, 0.28, 0.09, 0.11])
```

In [105...]

```
# Import the necessary Libraries
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

logreg_roc_auc = roc_auc_score(y_test, lr.predict_proba(x_test)[:, 1])

# Calculate the ROC AUC score for Decision Tree Classifier
dt_roc_auc = roc_auc_score(y_test, grid_search.predict_proba(x_test)[:, 1])

# Compute ROC curve for Logistic Regression
logreg_fpr, logreg_tpr, _ = roc_curve(y_test, lr.predict_proba(x_test)[:, 1])

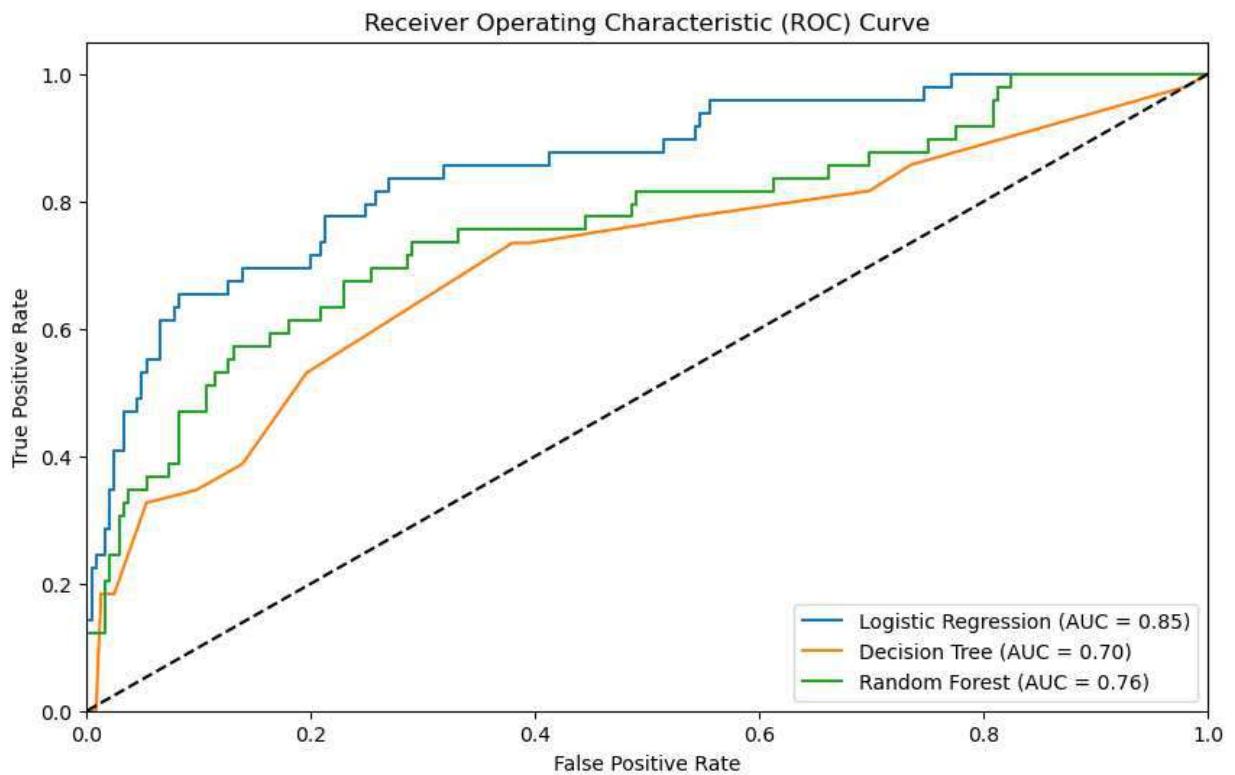
# Compute ROC curve for Decision Tree Classifier
dt_fpr, dt_tpr, _ = roc_curve(y_test, grid_search.predict_proba(x_test)[:, 1])

# Calculate the ROC AUC score for Random Forest
rf_roc_auc = roc_auc_score(y_test, rfc_cv.predict_proba(x_test)[:, 1])

# Compute ROC curve for Random Forest
rf_fpr, rf_tpr, _ = roc_curve(y_test, rfc_cv.predict_proba(x_test)[:, 1])

# Plot ROC AUC curves
plt.figure(figsize=(10, 6))
plt.plot(logreg_fpr, logreg_tpr, label='Logistic Regression (AUC = %0.2f)' % logreg_roc_auc)
plt.plot(dt_fpr, dt_tpr, label='Decision Tree (AUC = %0.2f)' % dt_roc_auc)
plt.plot(rf_fpr, rf_tpr, label='Random Forest (AUC = %0.2f)' % rf_roc_auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
```

```
plt.legend(loc='lower right')  
plt.show()
```



In []: