

```
In [1]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
In [2]: data=pd.read_csv("Employee-Attrition.csv")
```

```
In [3]: data.head()
```

Out[3]:

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

5 rows × 35 columns

```
In [4]: data.tail()
```

Out[4]:

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

5 rows × 35 columns

In [5]: `data.info()`

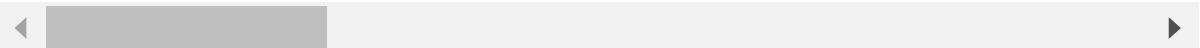
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Age              1470 non-null    int64  
 1   Attrition        1470 non-null    object  
 2   BusinessTravel   1470 non-null    object  
 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 [6]: `data.describe()`

Out[6]:

|              | Age         | DailyRate   | DistanceFromHome | Education   | EmployeeCount | EmployeeNu |
|--------------|-------------|-------------|------------------|-------------|---------------|------------|
| <b>count</b> | 1470.000000 | 1470.000000 | 1470.000000      | 1470.000000 | 1470.0        | 1470.0     |
| <b>mean</b>  | 36.923810   | 802.485714  | 9.192517         | 2.912925    | 1.0           | 1024.8     |
| <b>std</b>   | 9.135373    | 403.509100  | 8.106864         | 1.024165    | 0.0           | 602.0      |
| <b>min</b>   | 18.000000   | 102.000000  | 1.000000         | 1.000000    | 1.0           | 1.0        |
| <b>25%</b>   | 30.000000   | 465.000000  | 2.000000         | 2.000000    | 1.0           | 491.2      |
| <b>50%</b>   | 36.000000   | 802.000000  | 7.000000         | 3.000000    | 1.0           | 1020.5     |
| <b>75%</b>   | 43.000000   | 1157.000000 | 14.000000        | 4.000000    | 1.0           | 1555.7     |
| <b>max</b>   | 60.000000   | 1499.000000 | 29.000000        | 5.000000    | 1.0           | 2068.0     |

8 rows × 26 columns



## Handling the null values

In [7]: `data.isnull().any()`

Out[7]:

|                          |       |
|--------------------------|-------|
| 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 [8]: data.isnull().sum()
```

```
Out[8]: 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
```

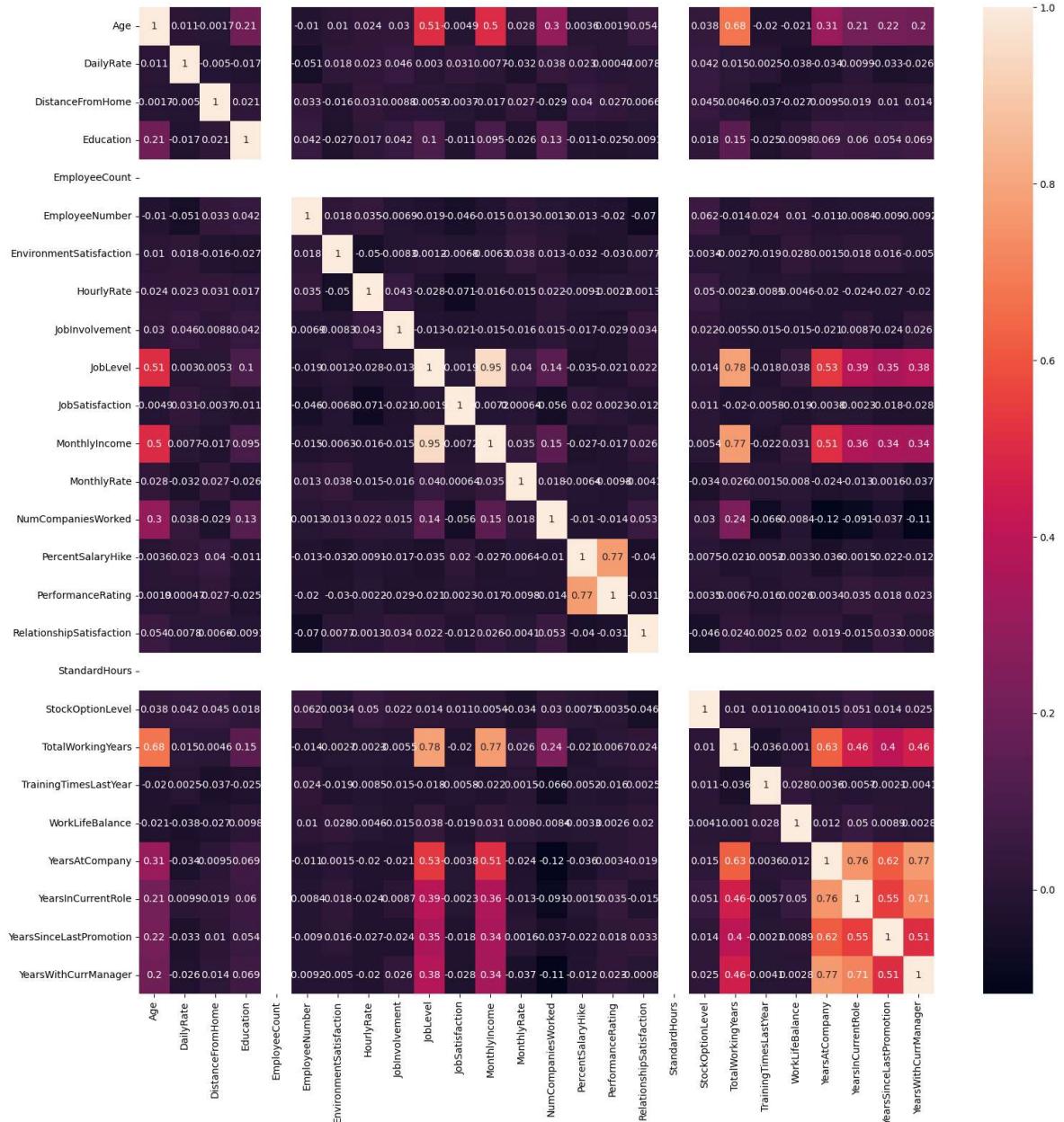
```
In [9]: cor=data.corr()
```

```
C:\Users\pichi\AppData\Local\Temp\ipykernel_10044\1426905697.py:1: FutureWarning:  
The default value of numeric_only in DataFrame.corr is deprecated. In  
a future version, it will default to False. Select only valid columns or spe  
cify the value of numeric_only to silence this warning.
```

```
cor=data.corr()
```

```
In [10]: fig=plt.figure(figsize=(18,18))
sns.heatmap(cor,annot=True)
```

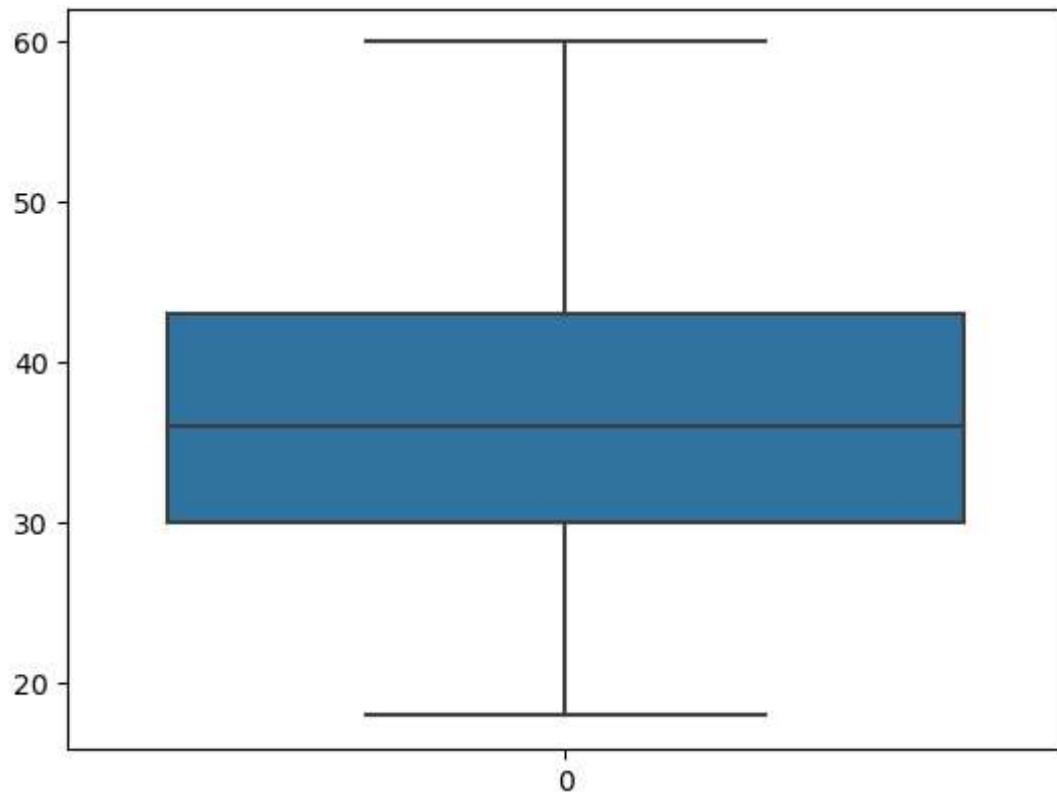
Out[10]: <Axes: >



## outliers

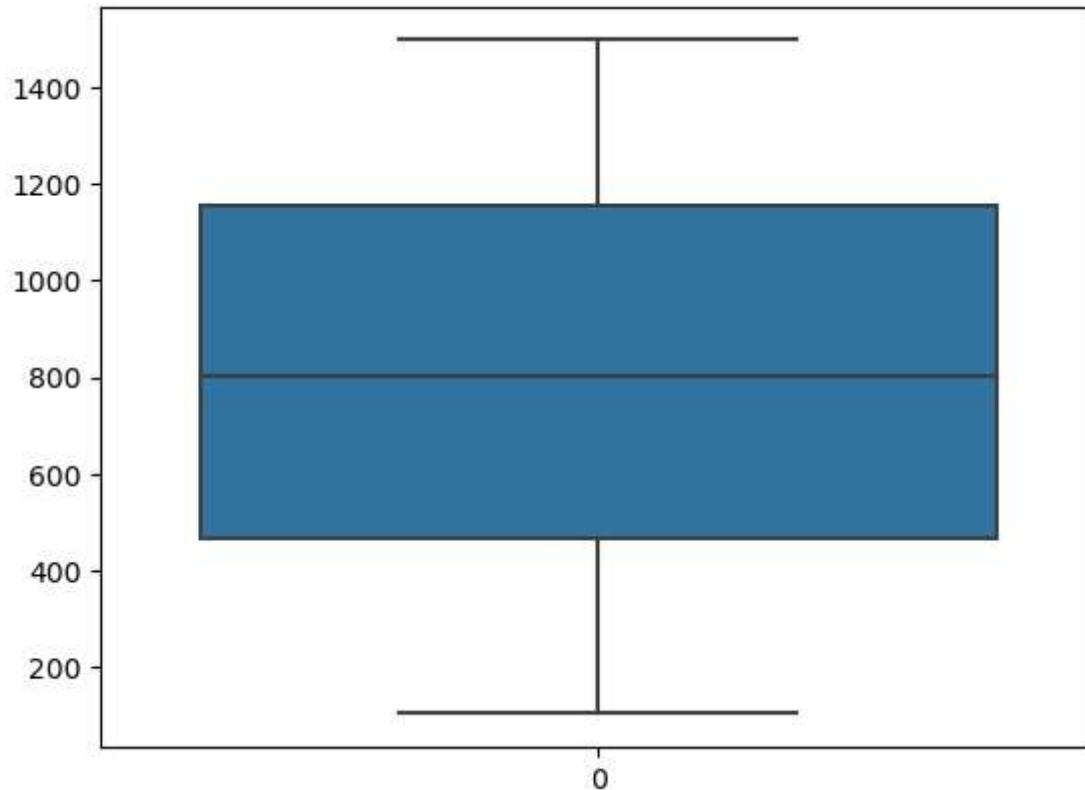
In [11]: `sns.boxplot(data["Age"])`

Out[11]: <Axes: >



In [12]: `sns.boxplot(data["DailyRate"])`

Out[12]: <Axes: >

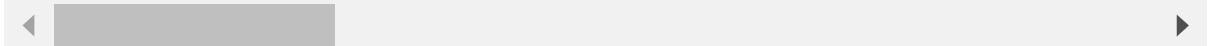


In [13]: `data.describe()`

Out[13]:

|              | Age         | DailyRate   | DistanceFromHome | Education   | EmployeeCount | EmployeeNu |
|--------------|-------------|-------------|------------------|-------------|---------------|------------|
| <b>count</b> | 1470.000000 | 1470.000000 | 1470.000000      | 1470.000000 | 1470.0        | 1470.0     |
| <b>mean</b>  | 36.923810   | 802.485714  | 9.192517         | 2.912925    | 1.0           | 1024.8     |
| <b>std</b>   | 9.135373    | 403.509100  | 8.106864         | 1.024165    | 0.0           | 602.0      |
| <b>min</b>   | 18.000000   | 102.000000  | 1.000000         | 1.000000    | 1.0           | 1.0        |
| <b>25%</b>   | 30.000000   | 465.000000  | 2.000000         | 2.000000    | 1.0           | 491.2      |
| <b>50%</b>   | 36.000000   | 802.000000  | 7.000000         | 3.000000    | 1.0           | 1020.5     |
| <b>75%</b>   | 43.000000   | 1157.000000 | 14.000000        | 4.000000    | 1.0           | 1555.7     |
| <b>max</b>   | 60.000000   | 1499.000000 | 29.000000        | 5.000000    | 1.0           | 2068.0     |

8 rows × 26 columns

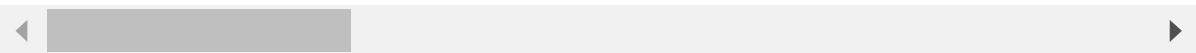


In [14]: `data.head()`

Out[14]:

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

5 rows × 35 columns

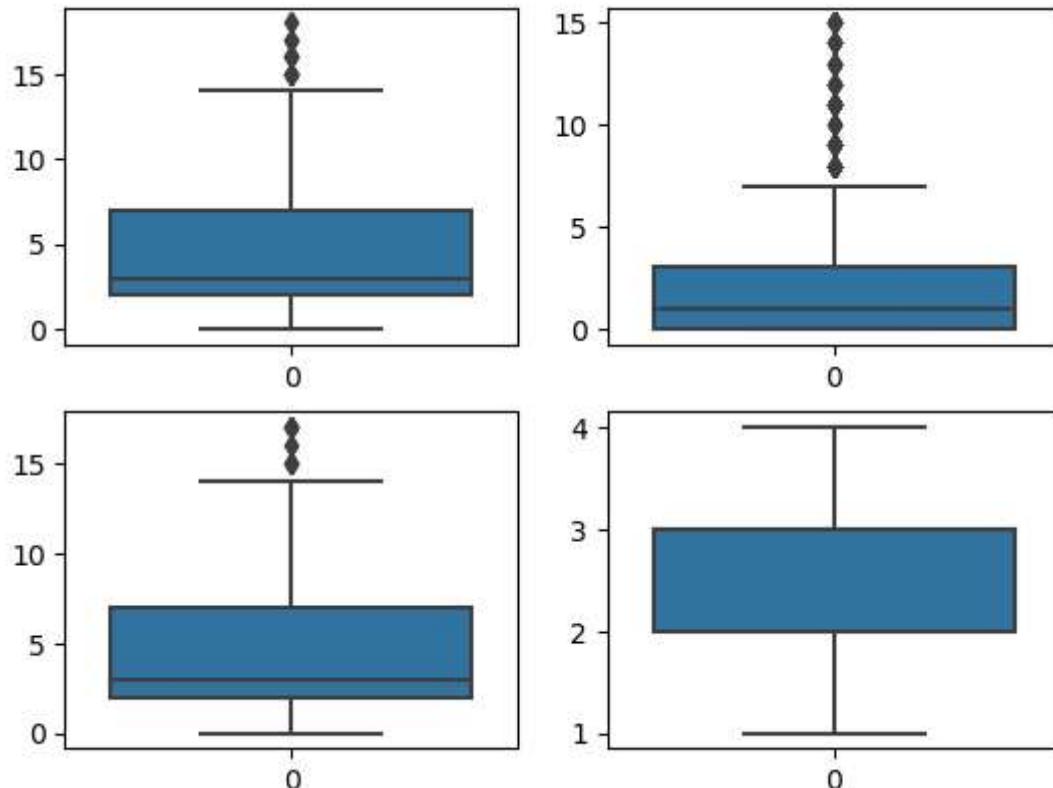


In [ ]:

In [15]:

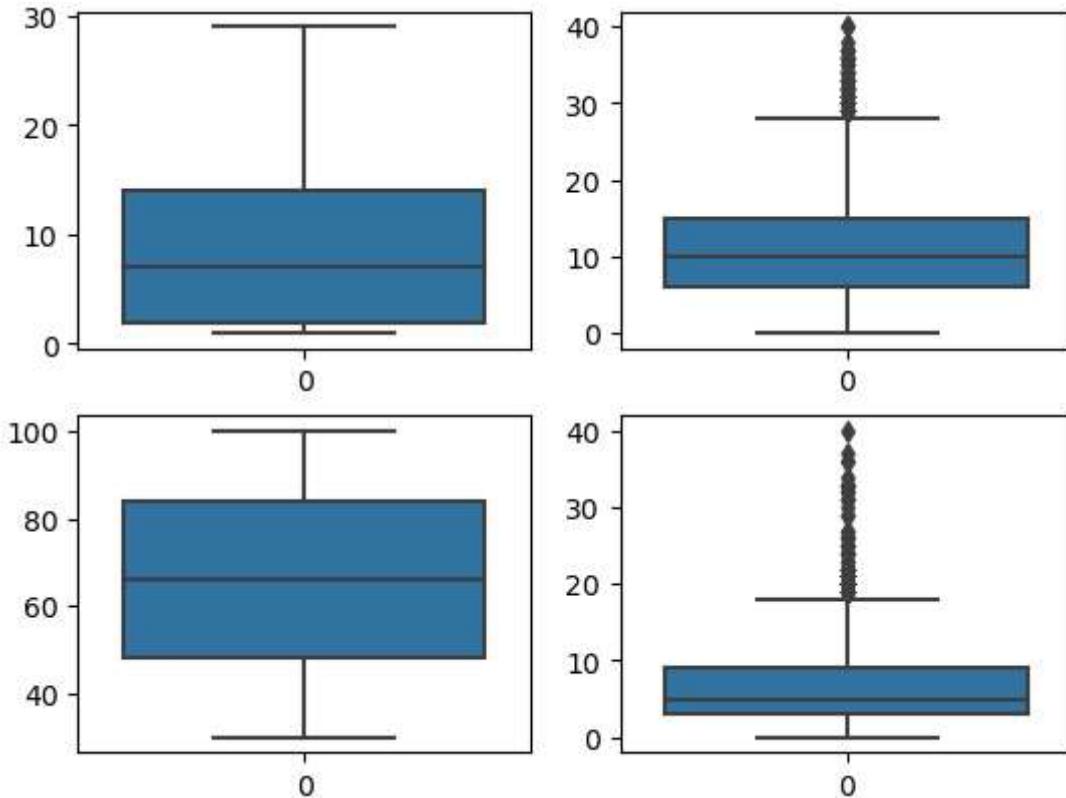
```
fig, axes = plt.subplots(2,2)
sns.boxplot(data=data[ "YearsInCurrentRole"],ax=axes[0,0])
sns.boxplot(data=data[ "YearsSinceLastPromotion"],ax=axes[0,1])
sns.boxplot(data=data[ "YearsWithCurrManager"],ax=axes[1,0])
sns.boxplot(data=data[ "WorkLifeBalance"],ax=axes[1,1])
```

Out[15]: <Axes: >



```
In [16]: fig, axes = plt.subplots(2,2)
sns.boxplot(data=data[ "DistanceFromHome"],ax=axes[0,0])
sns.boxplot(data=data[ "TotalWorkingYears"],ax=axes[0,1])
sns.boxplot(data=data[ "HourlyRate"],ax=axes[1,0])
sns.boxplot(data=data[ "YearsAtCompany"],ax=axes[1,1])
```

Out[16]: <Axes: >



## Handling the outliers

```
In [17]: YearsInCurrentRole_q1 = data.YearsInCurrentRole.quantile(0.25)
YearsInCurrentRole_q3 = data.YearsInCurrentRole.quantile(0.75)
IQR_YearsInCurrentRole=YearsInCurrentRole_q3-YearsInCurrentRole_q1
upperlimit_YearsInCurrentRole=YearsInCurrentRole_q3+1.5*IQR_YearsInCurrentRole
lower_limit_YearsInCurrentRole =YearsInCurrentRole_q1-1.5*IQR_YearsInCurrentRole
median_YearsInCurrentRole=data[ "YearsInCurrentRole"].median()
data[ 'YearsInCurrentRole' ] = np.where(
    (data[ 'YearsInCurrentRole' ] > upperlimit_YearsInCurrentRole),
    median_YearsInCurrentRole,
    data[ 'YearsInCurrentRole' ]
)
```

```
In [18]: YearsSinceLastPromotion_q1 = data.YearsSinceLastPromotion.quantile(0.25)
YearsSinceLastPromotion_q3 = data.YearsSinceLastPromotion.quantile(0.75)
IQR_YearsSinceLastPromotion=YearsSinceLastPromotion_q3-YearsSinceLastPromotion_q1
upperlimit_YearsSinceLastPromotion=YearsSinceLastPromotion_q3+1.5*IQR_YearsSinceLastPromotion
lower_limit_YearsSinceLastPromotion =YearsSinceLastPromotion_q1-1.5*IQR_YearsSinceLastPromotion
median_YearsSinceLastPromotion=data[ "YearsSinceLastPromotion"].median()
data[ 'YearsSinceLastPromotion' ] = np.where(
    (data[ 'YearsSinceLastPromotion' ] > upperlimit_YearsSinceLastPromotion),
    median_YearsSinceLastPromotion,
    data[ 'YearsSinceLastPromotion' ]
)
```

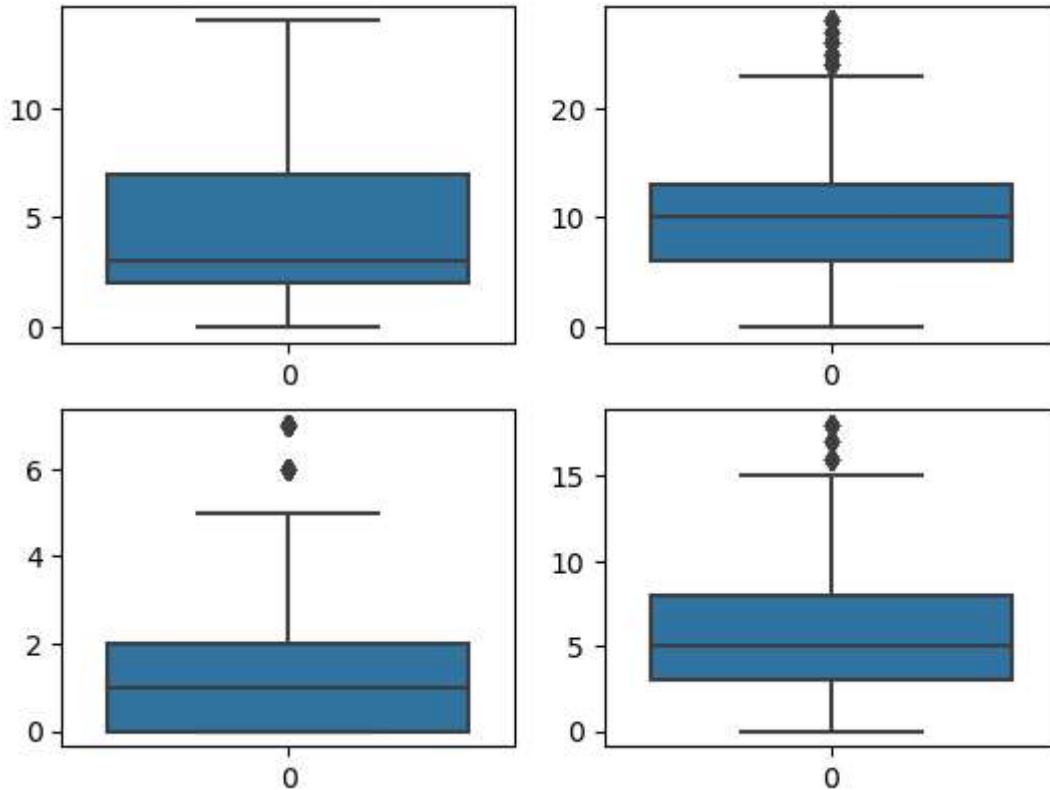
```
In [19]: YearsWithCurrManager_q1 = data.YearsWithCurrManager.quantile(0.25)
YearsWithCurrManager_q3 = data.YearsWithCurrManager.quantile(0.75)
IQR_YearsWithCurrManager=YearsWithCurrManager_q3-YearsWithCurrManager_q1
upperlimit_YearsWithCurrManager=YearsWithCurrManager_q3+1.5*IQR_YearsWithCurrManager
lower_limit_YearsWithCurrManager =YearsWithCurrManager_q1-1.5*IQR_YearsWithCurrManager
median_YearsWithCurrManager=data[ "YearsWithCurrManager"].median()
data[ 'YearsWithCurrManager' ] = np.where(
    (data[ 'YearsWithCurrManager' ] > upperlimit_YearsWithCurrManager),
    median_YearsWithCurrManager,
    data[ 'YearsWithCurrManager' ]
)
```

```
In [20]: TotalWorkingYears_q1 = data.TotalWorkingYears.quantile(0.25)
TotalWorkingYears_q3 = data.TotalWorkingYears.quantile(0.75)
IQR_TotalWorkingYears=TotalWorkingYears_q3-TotalWorkingYears_q1
upperlimit_TotalWorkingYears=TotalWorkingYears_q3+1.5*IQR_TotalWorkingYears
lower_limit_TotalWorkingYears=TotalWorkingYears_q1-1.5*IQR_TotalWorkingYears
median_TotalWorkingYears=data[ "TotalWorkingYears"].median()
data[ 'TotalWorkingYears' ] = np.where(
    (data[ 'TotalWorkingYears' ] > upperlimit_TotalWorkingYears),
    median_TotalWorkingYears,
    data[ 'TotalWorkingYears' ]
)
```

```
In [21]: YearsAtCompany_q1 = data.YearsAtCompany.quantile(0.25)
YearsAtCompany_q3 = data.YearsAtCompany.quantile(0.75)
IQR_YearsAtCompany=YearsAtCompany_q3-YearsAtCompany_q1
upperlimit_YearsAtCompany=YearsAtCompany_q3+1.5*IQR_YearsAtCompany
lower_limit_YearsAtCompany=YearsAtCompany_q1-1.5*IQR_YearsAtCompany
median_YearsAtCompany=data[ "YearsAtCompany"].median()
data[ 'YearsAtCompany' ] = np.where(
    (data[ 'YearsAtCompany' ] > upperlimit_YearsAtCompany),
    median_YearsAtCompany,
    data[ 'YearsAtCompany' ]
)
```

```
In [22]: fig, axes = plt.subplots(2,2)
sns.boxplot(data=data[ "YearsWithCurrManager"],ax=axes[0,0])
sns.boxplot(data=data[ "TotalWorkingYears"],ax=axes[0,1])
sns.boxplot(data=data[ "YearsSinceLastPromotion"],ax=axes[1,0])
sns.boxplot(data=data[ "YearsAtCompany"],ax=axes[1,1])
```

Out[22]: <Axes: >



In [23]: data.head()

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

5 rows × 35 columns

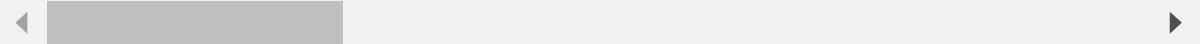
```
In [24]: data.drop("EducationField",axis=1,inplace=True)
```

```
In [25]: data.head(2)
```

```
Out[25]:
```

|   | Age | Attrition | BusinessTravel    | DailyRate | Department             | DistanceFromHome | Education | Em |
|---|-----|-----------|-------------------|-----------|------------------------|------------------|-----------|----|
| 0 | 41  | Yes       | Travel_Rarely     | 1102      | Sales                  |                  | 1         | 2  |
| 1 | 49  | No        | Travel_Frequently | 279       | Research & Development |                  | 8         | 1  |

2 rows × 34 columns



```
In [26]: data["BusinessTravel"].unique()
```

```
Out[26]: array(['Travel_Rarely', 'Travel_Frequently', 'Non-Travel'], dtype=object)
```

## splitting the data

```
In [27]: y=data["Attrition"]
```

```
In [28]: y.head()
```

```
Out[28]:
```

|   |     |
|---|-----|
| 0 | Yes |
| 1 | No  |
| 2 | Yes |
| 3 | No  |
| 4 | No  |

Name: Attrition, dtype: object

```
In [29]: data.drop("Attrition",axis=1,inplace=True)
```

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

Out[30]:

|   | Age | BusinessTravel    | DailyRate | Department             | DistanceFromHome | Education | EmployeeCount |
|---|-----|-------------------|-----------|------------------------|------------------|-----------|---------------|
| 0 | 41  | Travel_Rarely     | 1102      | Sales                  | 1                | 2         |               |
| 1 | 49  | Travel_Frequently | 279       | Research & Development | 8                | 1         |               |
| 2 | 37  | Travel_Rarely     | 1373      | Research & Development | 2                | 2         |               |
| 3 | 33  | Travel_Frequently | 1392      | Research & Development | 3                | 4         |               |
| 4 | 27  | Travel_Rarely     | 591       | Research & Development | 2                | 1         |               |

5 rows × 33 columns

## Encoding

In [31]: `from sklearn.preprocessing import LabelEncoder`

In [32]: `le=LabelEncoder()`

In [33]: `data["BusinessTravel"]=le.fit_transform(data["BusinessTravel"])`

In [34]: `data["Department"]=le.fit_transform(data["Department"])`

In [35]: `data["Gender"]=le.fit_transform(data["Gender"])`

In [36]: `y=le.fit_transform(y)`

In [37]: `y`

Out[37]: `array([1, 0, 1, ..., 0, 0, 0])`

In [38]: `data["JobRole"]=le.fit_transform(data["JobRole"])`

In [39]: `data["Over18"]=le.fit_transform(data["Over18"])`

```
In [40]: data[ "MaritalStatus" ]=le.fit_transform(data[ "MaritalStatus" ])
```

```
In [41]: data[ "OverTime" ]=le.fit_transform(data[ "OverTime" ])
```

```
In [42]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 33 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Age              1470 non-null    int64  
 1   BusinessTravel   1470 non-null    int32  
 2   DailyRate        1470 non-null    int64  
 3   Department       1470 non-null    int32  
 4   DistanceFromHome 1470 non-null    int64  
 5   Education        1470 non-null    int64  
 6   EmployeeCount   1470 non-null    int64  
 7   EmployeeNumber   1470 non-null    int64  
 8   EnvironmentSatisfaction 1470 non-null    int64  
 9   Gender            1470 non-null    int32  
 10  HourlyRate       1470 non-null    int64  
 11  JobInvolvement  1470 non-null    int64  
 12  JobLevel         1470 non-null    int64  
 13  JobRole          1470 non-null    int32  
 14  JobSatisfaction 1470 non-null    int64  
 15  MaritalStatus    1470 non-null    int32  
 16  MonthlyIncome    1470 non-null    int64  
 17  MonthlyRate      1470 non-null    int64  
 18  NumCompaniesWorked 1470 non-null    int64  
 19  Over18           1470 non-null    int32  
 20  OverTime          1470 non-null    int32  
 21  PercentSalaryHike 1470 non-null    int64  
 22  PerformanceRating 1470 non-null    int64  
 23  RelationshipSatisfaction 1470 non-null    int64  
 24  StandardHours    1470 non-null    int64  
 25  StockOptionLevel 1470 non-null    int64  
 26  TotalWorkingYears 1470 non-null    float64 
 27  TrainingTimesLastYear 1470 non-null    int64  
 28  WorkLifeBalance   1470 non-null    int64  
 29  YearsAtCompany   1470 non-null    float64 
 30  YearsInCurrentRole 1470 non-null    float64 
 31  YearsSinceLastPromotion 1470 non-null    float64 
 32  YearsWithCurrManager 1470 non-null    float64 
dtypes: float64(5), int32(7), int64(21)
memory usage: 338.9 KB
```

## train test split

```
In [43]: from sklearn.model_selection import train_test_split  
x_train,x_test,y_train,y_test=train_test_split(data,y,test_size=0.3,random_st  
  
In [44]: x_train.shape,x_test.shape,y_train.shape,y_test.shape  
  
Out[44]: ((1029, 33), (441, 33), (1029,), (441,))
```

## Feature Scaling

```
In [45]: from sklearn.preprocessing import StandardScaler  
  
In [46]: sc=StandardScaler()  
  
In [47]: x_train=sc.fit_transform(x_train)  
  
In [48]: x_test=sc.fit_transform(x_test)
```

## Building the model

### Multi-Linear Regression

```
In [49]: from sklearn.linear_model import LinearRegression  
  
In [50]: lr = LinearRegression()  
  
In [51]: lr.fit(x_train,y_train)  
  
Out[51]: LinearRegression()  
  
In a Jupyter environment, please rerun this cell to show the HTML representation or  
trust the notebook.  
On GitHub, the HTML representation is unable to render, please try loading this page  
with nbviewer.org.
```

```
In [52]: lr.coef_ #slope(m)
```

```
Out[52]: array([-3.54940447e-02,  7.88352347e-05, -1.70825038e-02,  3.46389690e-02,
   2.44612841e-02,  3.65668214e-03,  4.16333634e-17, -9.46820520e-03,
  -4.11203734e-02,  1.06338881e-02, -2.97662154e-03, -3.84864283e-02,
 -1.52927977e-02, -1.57839139e-02, -3.67252862e-02,  3.35765928e-02,
 -5.90043558e-03,  5.81099165e-03,  3.78471890e-02, -6.93889390e-18,
  9.55263279e-02, -2.55800078e-02,  2.01844797e-02, -2.64773510e-02,
 -1.21430643e-17, -1.79286106e-02, -3.30529386e-02, -1.09247807e-02,
 -3.10631611e-02, -2.47887717e-02, -1.10177742e-02,  2.11897289e-02,
 -6.60823991e-03])
```

```
In [53]: lr.intercept_ #(c)
```

```
Out[53]: 0.16229348882410102
```

```
In [54]: y_pred = lr.predict(x_test)
```

```
In [55]: y_pred
```

```
Out[55]: array([ 1.30302477e-01,  2.17626230e-01,  3.46282415e-01,  5.41382549e-0
 3,
 4.99292896e-01,  1.01628868e-01,  3.44742777e-01,  1.23994945e-0
 1,
-1.60694945e-01,  4.02435622e-01,  1.44159172e-01,  2.67416840e-0
 1,
-4.62559536e-02,  5.58671849e-01,  2.81858700e-01,  1.53537792e-0
 2,
 1.78573363e-01,  2.77532834e-01,  9.37121052e-02,  2.17571624e-0
 1,
 2.65936178e-01,  1.41499184e-02,  8.36251186e-02,  9.58849826e-0
 2,
 5.09869963e-01,  2.94764240e-01,  7.85819529e-02,  1.26647773e-0
 1,
 5.05518902e-01,  8.48456917e-02, -7.97229275e-02,  2.15516993e-0
 2,
 1.08079105e-01,  3.65998400e-01,  1.24517362e-01,  5.13682786e-0
 2,
 1.06749689e-01,  6.07640778e-02,  6.66425313e-02,  4.81312859e-0
 ~
```

In [56]: y\_test

# Logistic Regression

```
In [57]: from sklearn.linear_model import LogisticRegression
```

```
In [58]: lg=LogisticRegression()
```

```
In [59]: lg.fit(x_train,y_train)
```

Out[59]: LogisticRegression()

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**

**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

```
In [60]: y_pred_lg=lg.predict(x_test)
```

```
In [61]: y_pred
```

```
Out[61]: array([ 1.30302477e-01,  2.17626230e-01,  3.46282415e-01,  5.41382549e-01,
   3,
   4.99292896e-01,  1.01628868e-01,  3.44742777e-01,  1.23994945e-01,
   1,
   -1.60694945e-01,  4.02435622e-01,  1.44159172e-01,  2.67416840e-01,
   1,
   -4.62559536e-02,  5.58671849e-01,  2.81858700e-01,  1.53537792e-01,
   2,
   1.78573363e-01,  2.77532834e-01,  9.37121052e-02,  2.17571624e-01,
   1,
   2.65936178e-01,  1.41499184e-02,  8.36251186e-02,  9.58849826e-01,
   2,
   5.09869963e-01,  2.94764240e-01,  7.85819529e-02,  1.26647773e-01,
   1,
   5.05518902e-01,  8.48456917e-02,  -7.97229275e-02,  2.15516993e-01,
   2,
   1.08079105e-01,  3.65998400e-01,  1.24517362e-01,  5.13682786e-01,
   2,
   1.06749689e-01,  6.07640778e-02,  6.66425313e-02,  4.81312859e-01])
```

In [62]: y\_test

```
In [63]: score = lg.score(x_test, y_test)
        print(score)
```

0.8820861678004536

## confusion matrix

```
In [64]: from sklearn import metrics
cm = metrics.confusion_matrix(y_test,y_pred_lg)
print(cm)
```

```
[[366  5]
 [ 47 23]]
```

## Ridge and Lasso

```
In [65]: from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
```

```
In [66]: rg=Ridge()
```

```
In [67]: parametres={"alpha":[1,2,3,5,10,20,30,40,60,70,80,90]}
ridgecv=GridSearchCV(rg,parametres,scoring="neg_mean_squared_error",cv=5)
ridgecv.fit(x_train,y_train)
```

```
Out[67]: GridSearchCV(cv=5, estimator=Ridge(),
                      param_grid={'alpha': [1, 2, 3, 5, 10, 20, 30, 40, 60, 70, 80, 90]},
                      scoring='neg_mean_squared_error')
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**

**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

```
In [68]: print(ridgecv.best_params_)
```

```
{'alpha': 90}
```

```
In [69]: print(ridgecv.best_score_)
```

```
-0.11390621139234183
```

```
In [70]: y_pred_rg=ridgecv.predict(x_test)
```

In [71]: y\_pred\_rg

```
Out[71]: array([ 1.34413485e-01,  2.22561818e-01,  3.41692977e-01,  3.88209867e-01,
   3,
   4.84617338e-01,  1.16361483e-01,  3.30449743e-01,  1.27358807e-01,
   1,
   -1.34442619e-01,  3.77692888e-01,  1.33001445e-01,  2.69898751e-01,
   1,
   -2.54707392e-02,  5.25771894e-01,  2.67543514e-01,  2.78725024e-01,
   2,
   1.82233111e-01,  2.78896415e-01,  9.12689699e-02,  2.11494641e-01,
   1,
   2.70103341e-01,  8.44922044e-03,  8.74746722e-02,  1.05348798e-01,
   1,
   4.87749940e-01,  2.83080512e-01,  8.80556209e-02,  1.23817268e-01,
   1,
   4.82185624e-01,  9.34824523e-02,  -7.16448509e-02,  4.07003104e-01,
   2,
   1.08437994e-01,  3.42151399e-01,  1.22270929e-01,  6.85889862e-01,
   2,
   1.06690533e-01,  7.08689637e-02,  7.51570276e-02,  6.05829413e-01])
```

```
In [72]: y_test
```

```
In [73]: from sklearn import metrics  
print(metrics.r2_score(y_test,y_pred_rg))  
print(metrics.r2_score(y_train,ridgecv.predict(x_train)))
```

$0.21073458438815906$   
 $0.2061567210285109$

## Lasso

```
In [74]: from sklearn.linear_model import Lasso  
from sklearn.model_selection import GridSearchCV
```

```
In [75]: la=Ridge()
```

```
In [76]: parametres={"alpha":[1,2,3,5,10,20,30,40,60,70,80,90]}  
ridgecv=GridSearchCV(la,parametres,scoring="neg_mean_squared_error",cv=5)  
ridgecv.fit(x_train,y_train)
```

```
Out[76]: GridSearchCV(cv=5, estimator=Ridge(),  
param_grid={'alpha': [1, 2, 3, 5, 10, 20, 30, 40, 60, 70, 80, 90]},  
scoring='neg_mean_squared_error')
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**

**On GitHub, the HTML representation is unable to render, please try loading this page with [nbviewer.org](#).**

```
In [77]: print(ridgecv.best_params_)
```

```
{'alpha': 90}
```

```
In [78]: print(ridgecv.best_score_)
```

```
-0.11390621139234183
```

```
In [79]: y_pred_la=ridgecv.predict(x_test)
```

```
In [80]: y_pred_la
```

```
Out[80]: array([ 1.34413485e-01,  2.22561818e-01,  3.41692977e-01,  3.88209867e-0
3,
       4.84617338e-01,  1.16361483e-01,  3.30449743e-01,  1.27358807e-0
1,
      -1.34442619e-01,  3.77692888e-01,  1.33001445e-01,  2.69898751e-0
1,
      -2.54707392e-02,  5.25771894e-01,  2.67543514e-01,  2.78725024e-0
2,
      1.82233111e-01,  2.78896415e-01,  9.12689699e-02,  2.11494641e-0
1,
      2.70103341e-01,  8.44922044e-03,  8.74746722e-02,  1.05348798e-0
1,
      4.87749940e-01,  2.83080512e-01,  8.80556209e-02,  1.23817268e-0
1,
      4.82185624e-01,  9.34824523e-02, -7.16448509e-02,  4.07003104e-0
2,
      1.08437994e-01,  3.42151399e-01,  1.22270929e-01,  6.85889862e-0
2,
      1.06690533e-01,  7.08689637e-02,  7.51570276e-02,  6.05829413e-0
~
```

```
In [81]: from sklearn import metrics
print(metrics.r2_score(y_test,y_pred_la))
print(metrics.r2_score(y_train,ridgecv.predict(x_train)))
```

```
0.21073458438815906
0.2061567210285109
```

## Decision Tree

```
In [82]: from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
```

```
In [83]: dtc.fit(x_train,y_train)
```

```
Out[83]: DecisionTreeClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [84]: pred=dtc.predict(x_test)
```

In [85]: pred

In [86]: y\_test

```
In [87]: #Accuracy score
```

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

```
In [88]: accuracy_score(y_test,pred)
```

```
Out[88]: 0.7619047619047619
```

```
In [89]: confusion_matrix(y_test,pred)
```

```
Out[89]: array([[309,  62],  
                 [ 43,  27]], dtype=int64)
```

```
In [90]: pd.crosstab(y_test,pred)
```

```
Out[90]: col_0    0    1  
row_0  
-----  
0    309   62  
1     43   27
```

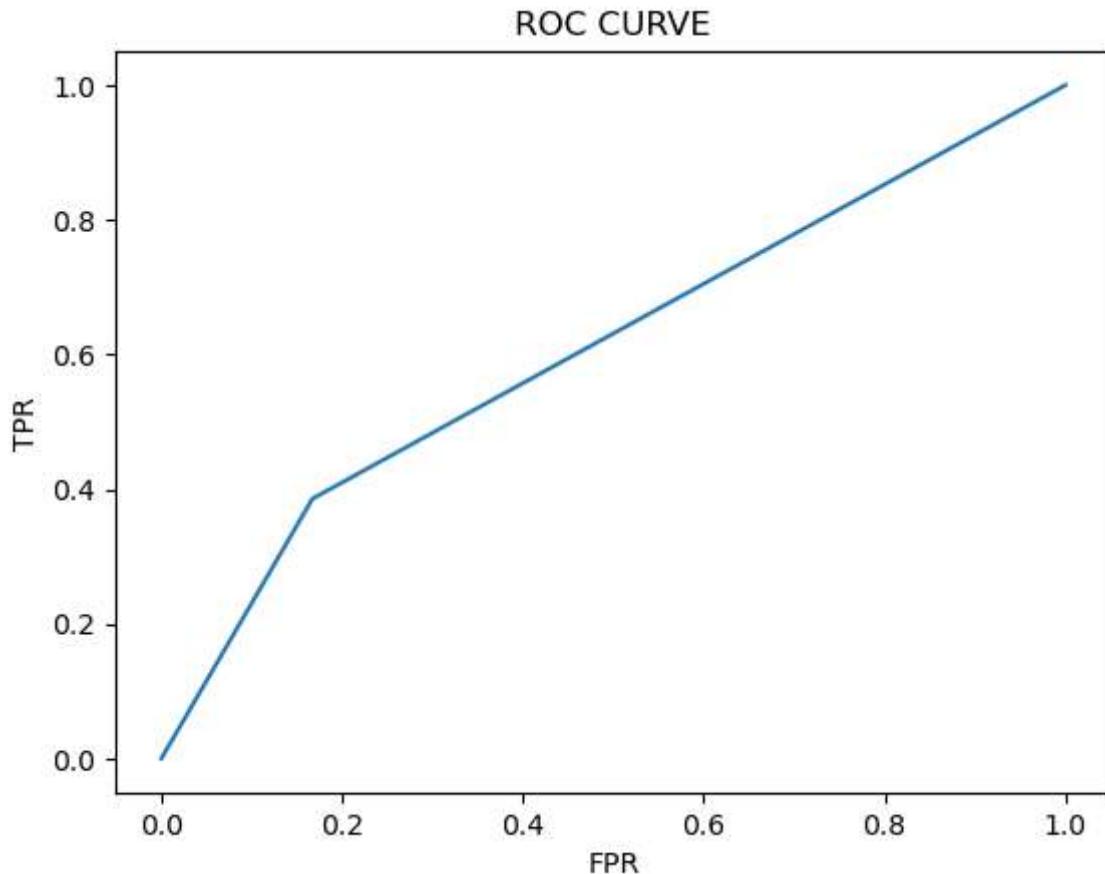
```
In [91]: print(classification_report(y_test,pred))
```

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0            | 0.88      | 0.83   | 0.85     | 371     |
| 1            | 0.30      | 0.39   | 0.34     | 70      |
| accuracy     |           |        | 0.76     | 441     |
| macro avg    | 0.59      | 0.61   | 0.60     | 441     |
| weighted avg | 0.79      | 0.76   | 0.77     | 441     |

```
In [92]: probability=dtc.predict_proba(x_test)[:,1]
```

```
In [93]: # roc_curve  
fpr,tpr,thresholds = roc_curve(y_test,probability)
```

```
In [94]: plt.plot(fpr,tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC CURVE')
plt.show()
```



## Random Forest

```
In [95]: from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
```

```
In [96]: forest_params = [{ 'max_depth': list(range(10, 15)), 'max_features': list(range(10, 15)) }]
```

```
In [97]: from sklearn.model_selection import GridSearchCV
```

```
In [98]: rfc_cv= GridSearchCV(rfc,param_grid=forest_params,cv=10,scoring="accuracy")
```

In [99]: rfc\_cv.fit(x\_train,y\_train)

```
C:\Users\pichi\AppData\Roaming\Python\Python310\site-packages\sklearn\model_selection\_validation.py:425: 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 err  
or_score='raise'.
```

Below are more details about the failures:

---

---

50 fits failed with the following error:

Traceback (most recent call last):

```
  File "C:\Users\pichi\AppData\Roaming\Python\Python310\site-packages\sklearn\model_selection\_validation.py", line 732, in _fit_and_score  
    estimator.fit(X_train, y_train, **fit_params)  
  File "C:\Users\pichi\AppData\Roaming\Python\Python310\site-packages\sklearn\base.py", line 1144, in wrapper  
    estimator._validate_params()  
  File "C:\Users\pichi\AppData\Roaming\Python\Python310\site-packages\sklearn\base.py", line 637, in _validate_params  
    validate_parameter_constraints()  
  File "C:\Users\pichi\AppData\Roaming\Python\Python310\site-packages\sklearn\utils\_param_validation.py", line 95, in validate_parameter_constraints  
    raise InvalidParameterError(  
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter of RandomForestClassifier must be an int in the range [1, inf), a float in the range (0.0, 1.0], a str among {'sqrt', 'log2'} or None. Got 0 instead.
```

```
    warnings.warn(some_fits_failed_message, FitFailedWarning)  
C:\Users\pichi\AppData\Roaming\Python\Python310\site-packages\sklearn\model_selection\_search.py:976: UserWarning: One or more of the test scores are non-finite: [      nan  0.84354654  0.8483914   0.85128498  0.84935275  0.85711974  
  0.85032362  0.84644013  0.8483914   0.85421664  0.85128498  0.84840091  
  0.85032362  0.85421664           nan  0.84840091  0.84936227  0.85130402  
  0.85130402  0.8541976   0.85323625  0.85421664  0.84838188  0.84546926  
  0.84351799  0.85032362  0.84643061  0.85226537           nan  0.84644965  
  0.85227489  0.84936227  0.85131354  0.85517799  0.85422616  0.85810965  
  0.86198363  0.84740148  0.85226537  0.8483914   0.84740148  0.84545022  
           nan  0.84353703  0.84644965  0.85130402  0.85518751  0.84936227  
  0.85615839  0.85323625  0.85226537  0.84935275  0.85032362  0.84935275  
  0.84838188  0.85225585           nan  0.84449838  0.85130402  0.85130402  
  0.85422616  0.85324576  0.85227489  0.84838188  0.84935275  0.85518751  
  0.85227489  0.85323625  0.85322673  0.84449838]  
  warnings.warn(
```

```
Out[99]: 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 a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with [nbviewer.org](#).

```
In [100]: pred=rfc_cv.predict(x_test)
```

```
In [101]: print(classification_report(y_test,pred))
```

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0            | 0.87      | 0.99   | 0.93     | 371     |
| 1            | 0.80      | 0.23   | 0.36     | 70      |
| accuracy     |           |        | 0.87     | 441     |
| macro avg    | 0.84      | 0.61   | 0.64     | 441     |
| weighted avg | 0.86      | 0.87   | 0.84     | 441     |

```
In [102]: rfc_cv.best_params_
```

```
Out[102]: {'max_depth': 12, 'max_features': 8}
```

```
In [103]: rfc_cv.best_score_
```

```
Out[103]: 0.8619836284028175
```

```
In [ ]:
```