

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

```
data=pd.read_csv("/content/employee_attrition.csv")
```

```
data.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education
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

```
data.tail()
```

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

5 rows × 35 columns

```
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

```

```
data.describe()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNu
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.00
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.86
std	9.135373	403.509100	8.106864	1.024165	0.0	602.02
min	18.000000	102.000000	1.000000	1.000000	1.0	1.00
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.25
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.50
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.75
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.00

8 rows × 26 columns

```
data.isnull().any()
```

Age	False
Attrition	False
BusinessTravel	False
DailyRate	False
Department	False
DistanceFromHome	False
Education	False
EducationField	False
EmployeeCount	False
EmployeeNumber	False
EnvironmentSatisfaction	False
Gender	False
HourlyRate	False
JobInvolvement	False
JobLevel	False
JobRole	False
JobSatisfaction	False
MaritalStatus	False
MonthlyIncome	False
MonthlyRate	False
NumCompaniesWorked	False
Over18	False
Overtime	False
PercentSalaryHike	False
PerformanceRating	False
RelationshipSatisfaction	False
StandardHours	False
StockOptionLevel	False
TotalWorkingYears	False
TrainingTimesLastYear	False
WorkLifeBalance	False
YearsAtCompany	False
YearsInCurrentRole	False
YearsSinceLastPromotion	False
YearsWithCurrManager	False

dtype: bool

```
data.isnull().sum()

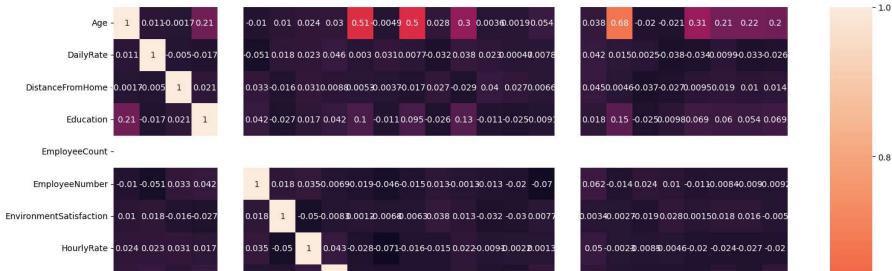
Age          0
Attrition    0
BusinessTravel 0
DailyRate     0
Department   0
DistanceFromHome 0
Education     0
EducationField 0
EmployeeCount 0
EmployeeNumber 0
EnvironmentSatisfaction 0
Gender        0
HourlyRate    0
JobInvolvement 0
JobLevel      0
JobRole       0
JobSatisfaction 0
MaritalStatus 0
MonthlyIncome  0
MonthlyRate    0
NumCompaniesWorked 0
Over18        0
OverTime      0
PercentSalaryHike 0
PerformanceRating 0
RelationshipSatisfaction 0
StandardHours 0
StockOptionLevel 0
TotalWorkingYears 0
TrainingTimesLastYear 0
WorkLifeBalance 0
YearsAtCompany 0
YearsInCurrentRole 0
YearsSinceLastPromotion 0
YearsWithCurrManager 0
dtype: int64

cor=data.corr()

<ipython-input-9-410fe4458127>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version,
cor=data.corr()

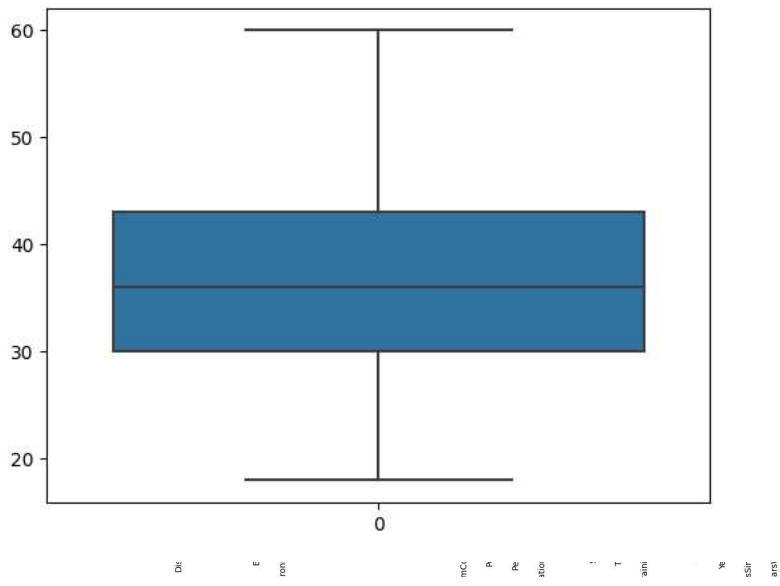
fig=plt.figure(figsize=(18,18))
sns.heatmap(cor,annot=True)
```

```
<Axes: >
```



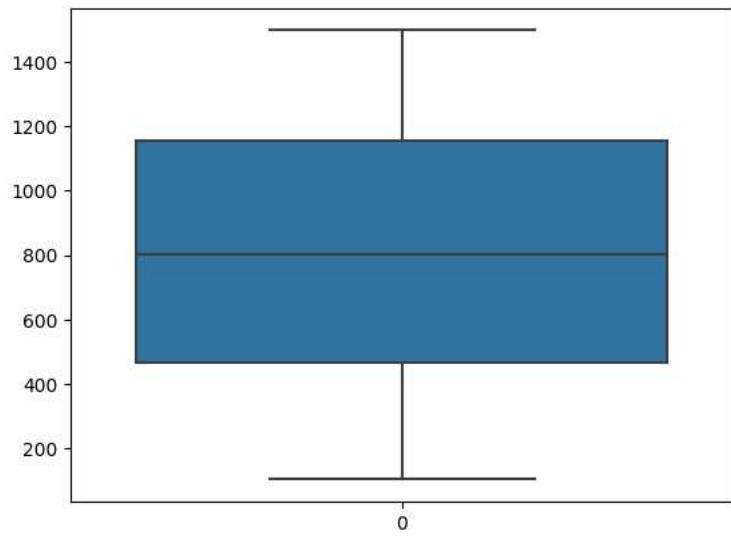
```
sns.boxplot(data["Age"])
```

```
<Axes: >
```



```
sns.boxplot(data["DailyRate"])
```

```
<Axes: >
```



```
data.describe()
```

```

      Age DailyRate DistanceFromHome Education EmployeeCount EmployeeNu
count 1470.000000 1470.000000 1470.000000 1470.000000 1470.0 1470.00
mean 36.923810 802.485714 9.192517 2.912925 1.0 1024.86
std 9.135373 403.509100 8.106864 1.024165 0.0 602.02
min 18.000000 102.000000 1.000000 1.000000 1.0 1.00

```

```
data.head()
```

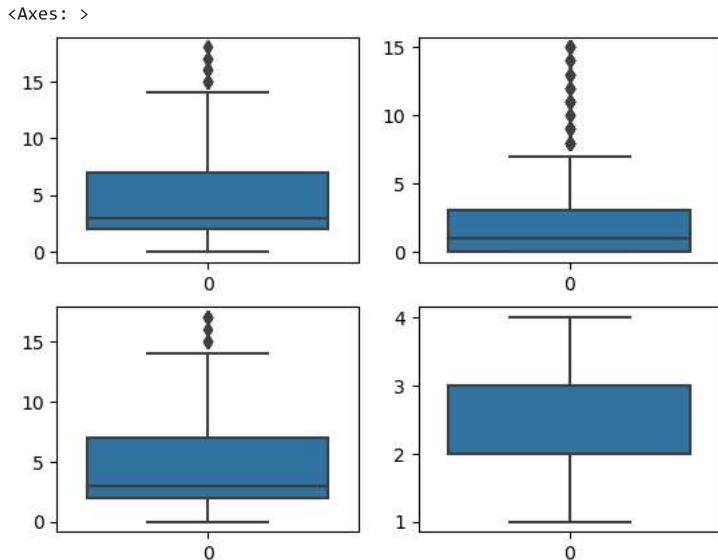
	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education
0	41	Yes	Travel_Rarely	1102	Sales	1	2
1	49	No	Travel_Frequently	279	Research & Development	8	1
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2
3	33	No	Travel_Frequently	1392	Research & Development	3	4
4	27	No	Travel_Rarely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

```
5 rows × 35 columns
```

```

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])

```

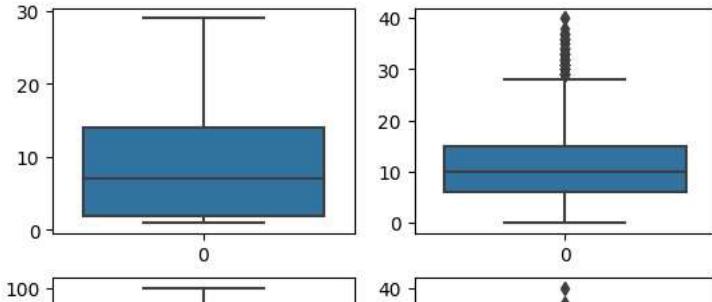


```

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])

```

<Axes: >



```
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']
)

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']
)

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']
)

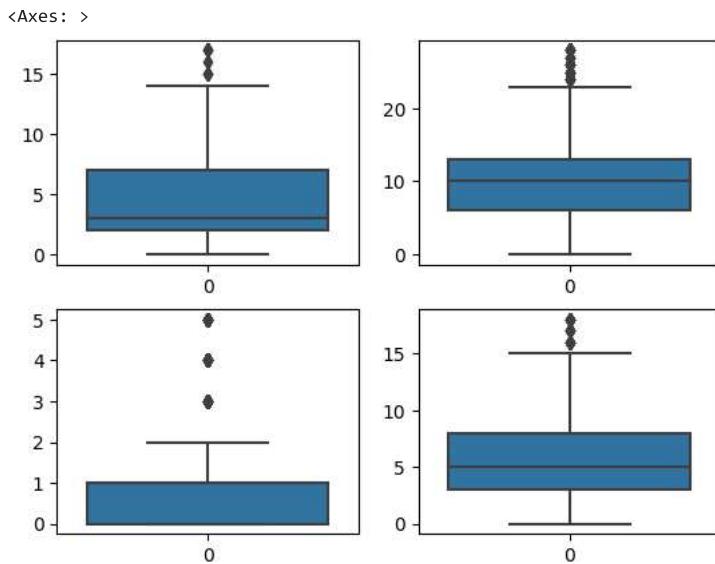
TotalWorkingYears_q1 = data.TotalWorkingYears.quantile(0.25)
TotalWorkingYears_q3 = data.TotalWorkingYears.quantile(0.75)
IQR_TotalWorkingYears=TotalWorkingYears_q3-TotalWorkingYears_q1
upperlimit_TotalWorkingYears=TotalWorkingYears_q3+1.5*IQR_TotalWorkingYears
lower_limit_TotalWorkingYears=TotalWorkingYears_q1-1.5*IQR_TotalWorkingYears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

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']
)
```

```

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])

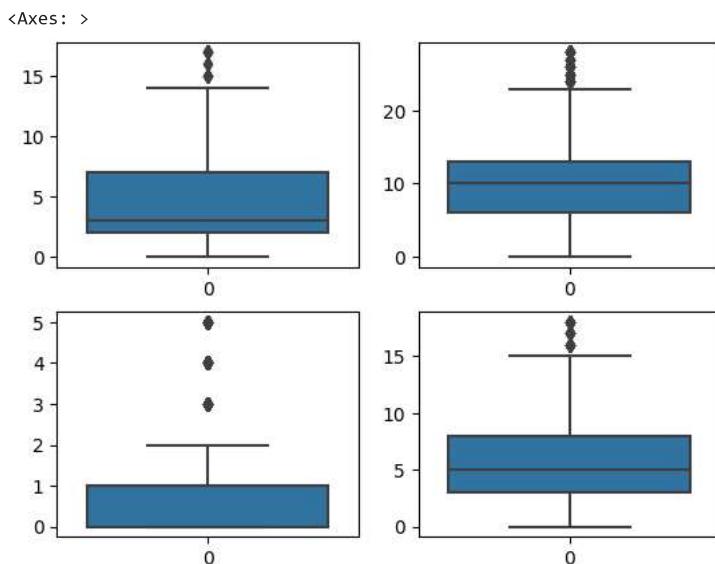
```



```

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])

```



```
data.drop("EducationField",axis=1,inplace=True)
```

```
data.head(2)
```

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

2 rows x 34 columns

```

data["BusinessTravel"].unique()

array(['Travel_Rarely', 'Travel_Frequently', 'Non-Travel'], dtype=object)

y=data["Attrition"]

y.head()

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

from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

data["BusinessTravel"]=le.fit_transform(data["BusinessTravel"])

data["Department"]=le.fit_transform(data["Department"])

data["Gender"]=le.fit_transform(data["Gender"])

y=le.fit_transform(y)

y

array([1, 0, 1, ..., 0, 0, 0])

data["JobRole"]=le.fit_transform(data["JobRole"])

data["Over18"]=le.fit_transform(data["Over18"])

data["MaritalStatus"]=le.fit_transform(data["MaritalStatus"])

data["OverTime"]=le.fit_transform(data["OverTime"])

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    int64  
 2   DailyRate        1470 non-null    int64  
 3   Department       1470 non-null    int64  
 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    int64  
 10  HourlyRate       1470 non-null    int64  
 11  JobInvolvement   1470 non-null    int64  
 12  JobLevel          1470 non-null    int64  
 13  JobRole           1470 non-null    int64  
 14  JobSatisfaction  1470 non-null    int64  
 15  MaritalStatus    1470 non-null    int64  
 16  MonthlyIncome    1470 non-null    int64  
 17  MonthlyRate      1470 non-null    int64  
 18  NumCompaniesWorked 1470 non-null    int64  
 19  Over18           1470 non-null    int64  
 20  OverTime          1470 non-null    int64  
 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 int64
dtypes: float64(4), int64(29)
memory usage: 379.1 KB
```

```
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_state=0)
```

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

```
((1029, 33), (441, 33), (1029,), (441,))
```

```
from sklearn.preprocessing import StandardScaler
```

```
sc=StandardScaler()
```

```
x_train=sc.fit_transform(x_train)
```

```
x_test=sc.fit_transform(x_test)
```

```
from sklearn.linear_model import LinearRegression
```

```
lr = LinearRegression()
```

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

```
▼ LinearRegression
  LinearRegression()
```

```
lr.coef_ #slope(m)
```

```
array([-3.55090313e-02,  3.16244127e-04, -1.77022589e-02,  3.46724269e-02,
       2.53882188e-02,  3.13002894e-03,  5.52509427e-16, -8.59732842e-03,
      -4.10446928e-02,  1.02773618e-02, -4.06315361e-03, -3.80151975e-02,
     -2.00117551e-02, -1.57446441e-02, -3.68204626e-02,  3.36047208e-02,
    -9.62856806e-04,  5.50205989e-03,  3.77654663e-02,  6.93889390e-18,
   9.55217842e-02, -2.52786287e-02,  1.96862504e-02, -2.58732161e-02,
   6.93889390e-18, -1.85032283e-02, -3.16965250e-02, -1.07118136e-02,
  -3.02436253e-02, -1.94083929e-02, -5.21067440e-03, -4.15175251e-03,
  -6.96034764e-03])
```

```
lr.intercept_ #(c)
```

```
0.16229348882410102
```

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

```
y_pred
```

```
array([[ 1.39746077e-01,  2.35860972e-01,  3.13729082e-01,  2.85601806e-02,
        4.90297011e-01,  1.04459627e-01,  3.75833184e-01,  1.41094459e-01,
       -1.19772088e-01,  3.62837491e-01,  1.10689933e-01,  2.84156593e-01,
      -3.84861295e-02,  5.42851762e-01,  3.10247224e-01, -3.9395890e-02,
       1.96607482e-01,  2.90240094e-01,  1.03469851e-01,  1.55710926e-01,
       2.75054762e-01, -4.62456972e-03,  4.09235420e-02,  7.85174805e-02,
      5.16693064e-01,  3.24070599e-01,  9.40294303e-02,  1.54256039e-01,
      5.04829398e-01,  7.11967422e-02, -9.22093977e-02,  3.70286128e-02,
     1.08865295e-01,  3.27645636e-01,  1.24827949e-01,  7.16328826e-02,
     1.21859754e-01,  6.51509433e-02,  3.90311688e-02,  5.44633507e-02,
    -1.70864754e-02, -1.73035474e-02,  7.72818919e-02, -4.52271002e-02,
     6.39094706e-03,  4.33301337e-01,  3.50304419e-01, -1.88150435e-01,
```

5.53488427e-01,	4.14438976e-01,	1.81986512e-01,	4.57526736e-01,
8.79294131e-02,	3.70081256e-01,	5.01443110e-01,	2.79225433e-01,
-3.33463339e-02,	3.19171962e-01,	7.07112536e-03,	2.55514746e-01,
-1.54430141e-02,	2.80576806e-01,	9.75758483e-02,	1.25611582e-01,
3.73786179e-01,	1.21873410e-02,	3.64646168e-01,	1.88382489e-01,
1.21814084e-01,	1.01979721e-01,	1.19343258e-02,	3.23367502e-01,
1.38663668e-01,	1.37718010e-01,	2.23526586e-01,	9.75536420e-02,
7.51574501e-02,	2.29453281e-01,	2.57693616e-01,	4.07305913e-03,
-5.66292840e-02,	-3.66798546e-02,	1.89304023e-01,	-2.03827776e-02,
-1.88676869e-02,	1.33555998e-01,	4.98462197e-02,	-2.57766037e-04,
6.75125162e-02,	2.52160875e-01,	3.17577731e-01,	1.67113160e-01,
3.39388237e-01,	2.42015199e-01,	-4.35865734e-02,	2.25073307e-01,
3.14193773e-01,	2.94915919e-01,	1.05348804e-01,	6.87196816e-02,
2.78628352e-01,	5.10425173e-01,	3.10659912e-01,	1.98079711e-02,
1.68005564e-01,	-1.18684696e-02,	4.46873671e-02,	2.19195312e-01,
9.54442779e-02,	1.64033379e-01,	1.30843456e-01,	5.75598171e-02,
-3.36393804e-02,	1.29256028e-01,	1.21414594e-01,	4.77728036e-02,
6.35085699e-02,	2.44558798e-01,	-7.46272431e-02,	-1.25246030e-01,
2.21685807e-01,	5.04493305e-02,	9.03416497e-02,	6.17676588e-01,
4.15328476e-05,	-1.74673544e-03,	-1.44909599e-01,	2.15253036e-01,
2.80527807e-01,	2.03020173e-01,	-1.48893113e-02,	2.64297202e-01,
4.41383832e-01,	4.18182405e-01,	1.81802443e-01,	4.24062174e-01,
5.09190482e-01,	1.36946123e-01,	1.40945289e-01,	3.71415605e-01,
2.35186850e-01,	1.57822203e-01,	2.47182357e-01,	2.51754086e-01,
2.74732050e-01,	1.62282102e-02,	1.62374785e-01,	-1.50571631e-01,
2.05556833e-01,	2.86631553e-01,	1.12713858e-01,	2.93765936e-01,
5.40011968e-02,	3.56817803e-01,	1.25134988e-01,	1.41757516e-02,
-2.11756304e-02,	1.33786727e-01,	-8.57122955e-02,	-7.81757898e-02,
3.53060584e-01,	-1.23234531e-01,	1.89978261e-01,	6.20968452e-01,
2.04106577e-01,	3.18374568e-01,	1.98978852e-01,	1.87747657e-01,
-2.38290139e-02,	-6.10645791e-02,	6.76347231e-03,	7.94472458e-02,
1.59637328e-01,	1.63247013e-01,	-7.988686251e-02,	1.92599111e-01,
2.01601196e-01,	2.19286252e-01,	1.18585045e-01,	2.04167787e-01,
1.77752754e-02,	1.60421610e-01,	-7.44003860e-02,	5.69715685e-01,
6.03751335e-02,	4.87404857e-02,	3.10938374e-01,	1.25222690e-01,
5.57587496e-01,	1.46037140e-01,	3.79850618e-01,	3.73689557e-01,
2.61553365e-01,	6.01177645e-02,	1.06346090e-01,	2.22285097e-01,
7.88345959e-02,	-4.61703652e-02,	3.46232300e-01,	4.35829190e-02,
2.26755898e-01,	2.57035453e-01,	4.82234963e-01,	1.47363152e-01,
2.99348177e-01,	7.90026717e-02,	4.41919583e-01,	-7.03130285e-02,
1.70465615e-01,	-3.64019741e-02,	1.80248625e-01,	1.87576444e-01,
6.22900421e-02,	1.15210649e-01,	1.44513501e-01,	-1.62872871e-02,
3.09165173e-02,	1.00101971e-01,	3.56698613e-02,	2.31696043e-01,
1.78672479e-01,	2.46734954e-01,	5.44481292e-01,	1.40752308e-01,
3.84893890e-01,	-8.42859330e-02,	1.48274367e-01,	3.00019986e-01,
2.04726776e-02,	4.16200626e-02,	3.15555621e-02,	2.51612626e-02,

y_test

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

```
lg=LogisticRegression()
```

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

```
    ▾ LogisticRegression  
        LogisticRegression()
```

```

y_pred_lg=lg.predict(x_test)

y_pred

array([[ 1.39746077e-01,   2.35860972e-01,   3.13729082e-01,   2.85601806e-02,
       4.90297011e-01,   1.04459627e-01,   3.75833184e-01,   1.41094459e-01,
      -1.19772088e-01,   3.62837491e-01,   1.10689933e-01,   2.84156593e-01,
      -3.84861295e-02,   5.42851762e-01,   3.10247224e-01,  -3.93958990e-02,
       1.96607482e-01,   2.90240094e-01,   1.03469851e-01,   1.55710926e-01,
       2.75054762e-01,  -4.62456972e-03,   4.09235420e-02,   7.85174805e-02,
      5.16693064e-01,   3.24070599e-01,   9.40294303e-02,   1.54256039e-01,
      5.04829398e-01,   7.11967422e-02,  -9.22093977e-02,   3.70286128e-02,
     1.08865295e-01,   3.27645636e-01,   1.24827949e-01,   7.16328826e-02,
     1.21859754e-01,   6.51509433e-02,   3.90311688e-02,   5.44633507e-02,
    -1.70864754e-02,  -1.73035474e-02,   7.72818919e-02,  -4.52271002e-02,
     6.39094706e-03,   4.33301337e-01,   3.50304419e-01,  -1.88150435e-01,
     5.53488427e-01,   4.14438976e-01,   1.81986512e-01,   4.57526736e-01,
     8.79294131e-02,   3.70081256e-01,   5.01443110e-01,   2.79225433e-01,
    -3.33463339e-02,   3.19171962e-01,   7.07112536e-03,  2.55514746e-01,
    -1.54430141e-02,   2.80576806e-01,   9.75758483e-02,  1.25611582e-01,
     3.73786179e-01,   1.21873410e-02,   3.64646168e-01,   1.88382489e-01,
     1.21814084e-01,   1.01979721e-01,   1.19343258e-02,   3.23367502e-01,
     1.38663668e-01,   1.37718010e-01,   2.23526586e-01,   9.75536420e-02,
     7.51574501e-02,   2.29453281e-01,   2.57693616e-01,   4.07305913e-03,
    -5.66292840e-02,  -3.66798546e-02,   1.89304023e-01,  -2.03827776e-02,
    -1.88676869e-02,   1.33555998e-01,   4.98462197e-02,  -2.57766037e-04,
     6.75125162e-02,   2.52160875e-01,   3.17577731e-01,   1.67113160e-01,
     3.39388237e-01,   2.42015199e-01,  -4.35865734e-02,  2.25073307e-01,
     3.14193773e-01,   2.94915919e-01,   1.05348804e-01,   6.87196816e-02,
     2.78628352e-01,   5.10425173e-01,   3.10659912e-01,   1.98079711e-02,
     1.68005564e-01,  -1.18684696e-02,   4.46873671e-02,  2.19195312e-01,
     9.54442779e-02,   1.64033379e-01,   1.30843456e-01,   5.75598171e-02,
    -3.36393804e-02,   1.29256028e-01,   1.21414594e-01,   4.77728036e-02,
     6.35085699e-02,   2.44558798e-01,  -7.46272431e-02,  -1.25246030e-01,
     2.21685807e-01,   5.04493305e-02,   9.03416497e-02,   6.17676588e-01,
     4.15328476e-05,  -1.74673544e-03,  -1.44909599e-01,  2.15253036e-01,
     2.80527807e-01,   2.03020173e-01,  -1.48893113e-02,  2.64297202e-01,
     4.41383832e-01,   4.18182405e-01,   1.81802443e-01,   4.24062174e-01,
     5.09190482e-01,   1.36946123e-01,   1.40945289e-01,   3.71415605e-01,
     2.35186850e-01,   1.57822203e-01,   2.47182357e-01,   2.51754866e-01,
     2.74732050e-01,   1.62282102e-02,   1.62374785e-01,  -1.50571631e-01,
     2.05556833e-01,   2.86631553e-01,   1.12713858e-01,   2.93765936e-01,
     5.40011968e-02,   3.56817803e-01,   1.25134988e-01,   1.41757516e-02,
    -2.11756304e-02,   1.33786727e-01,  -8.57122955e-02,  -7.81757898e-02,
     3.53060584e-01,  -1.23234531e-01,   1.89978261e-01,   6.20968452e-01,
     2.04106577e-01,   3.18374568e-01,   1.98978852e-01,   1.87747657e-01,
    -2.38290139e-02,  -6.10645791e-02,   6.76347231e-03,   7.94472458e-02,
     1.59637328e-01,   1.63247013e-01,  -7.98686251e-02,  1.92599111e-01,
     2.01601196e-01,   2.19286252e-01,   1.18585045e-01,   2.04167778e-01,
     1.77752754e-02,   1.60421610e-01,  -7.44003860e-02,   5.69715685e-01,
     6.03751335e-02,   4.87404857e-02,   3.10938374e-01,   1.25222690e-01,
     5.57587496e-01,   1.46037140e-01,   3.79850618e-01,   3.73689557e-01,
     2.61553365e-01,   6.01177645e-02,   1.06346090e-01,   2.22285097e-01,
     7.88345959e-02,  -4.61703652e-02,   3.46232300e-01,   4.35829190e-02,
     2.26755898e-01,   2.57035453e-01,   4.82234963e-01,   1.47363152e-01,
     2.99348177e-01,   7.90026717e-02,   4.41919583e-01,  -7.03130285e-02,
     1.70465615e-01,  -3.64019741e-02,   1.80248625e-01,   1.87576444e-01,
     6.22900421e-02,   1.15210649e-01,   1.44513501e-01,  -1.62872871e-02,
     3.09165173e-02,   1.00101971e-01,   3.56698613e-02,   2.31696043e-01,
     1.78672479e-01,   2.46734954e-01,   5.44481292e-01,   1.40752308e-01,
     3.84893890e-01,  -8.42859330e-02,   1.48274367e-01,   3.00019986e-01,
     3.04703759e-01,  -4.14000508e-02,  -2.15559924e-02,   3.52185267e-01,
  ]])

y_test

```

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

```
score = lg.score(x_test, y_test)
print(score)
```

```
0.8752834467120182
```

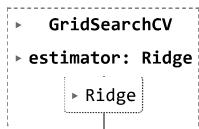
```
from sklearn import metrics
cm = metrics.confusion_matrix(y_test,y_pred_lg)
print(cm)
```

```
[[364  7]
 [ 48 22]]
```

```
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
```

```
rg=Ridge()
```

```
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)
```



```
print(ridgecv.best_params_)
```

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

```
print(ridgecv.best_score_)
```

```
-0.11407236809490047
```

```
y_pred_rg=ridgecv.predict(x_test)
```

```
y_pred_rg
```

```
array([ 1.43006327e-01,  2.40378141e-01,  3.11765437e-01,  2.34082267e-02,
       4.76281083e-01,  1.19904456e-01,  3.58214683e-01,  1.42524589e-01,
      -9.74745513e-02,  3.38140509e-01,  1.02825036e-01,  2.84369659e-01,
     -2.06162330e-02,  5.09484716e-01,  2.93717014e-01, -2.34288385e-02,
      1.98084122e-01,  2.91964884e-01,  1.02215131e-01,  1.57911429e-01,
      2.78486682e-01, -6.73170356e-03,  4.99598146e-02,  8.87505933e-02,
      4.95477102e-01,  3.09029528e-01,  1.00400854e-01,  1.48380335e-01,
      4.80986577e-01,  7.92550922e-02, -8.54138258e-02,  5.75072366e-02,
      1.09469599e-01,  3.05306331e-01,  1.22683770e-01,  8.72594892e-02,
      1.21894430e-01,  7.43542652e-02,  4.84808828e-02,  6.57208982e-02,
     -3.08502977e-03,  4.13721895e-03,  7.91345665e-02, -4.46724300e-02,
      1.52889763e-02,  4.22120422e-01,  3.26879320e-01, -1.58829835e-01,
      5.35099285e-01,  4.03487632e-01,  1.81397041e-01,  4.38170000e-01,
      9.14599593e-02,  3.47792094e-01,  4.77884673e-01,  2.75375118e-01,
     -1.81427076e-02,  3.06176633e-01,  2.31038146e-02,  2.48819114e-01,
     -3.84435620e-06,  2.71908742e-01,  9.54913585e-02,  1.23698952e-01,
      3.61787991e-01,  2.18385063e-02,  3.47218525e-01,  1.88261865e-01,
      1.24376833e-01,  1.16438880e-01,  1.29110556e-02,  3.09160319e-01,
      1.40447398e-01,  1.42389152e-01,  2.13249433e-01,  9.28483488e-02,
      8.69910824e-02,  2.20281257e-01,  2.56637833e-01,  1.01217283e-02,
     -4.76911729e-02, -3.28796702e-02,  1.99264346e-01, -1.70843966e-03,
     -1.03158063e-02,  1.41866434e-01,  4.18013343e-02,  2.24051855e-02,
      6.31082673e-02,  2.42163132e-01,  3.20356944e-01,  1.52672067e-01,
      3.23802985e-01,  2.36496694e-01, -3.63044096e-02,  2.24390582e-01,
      3.08221104e-01,  2.91477250e-01,  1.26122372e-01,  7.46305846e-02,
      2.80154727e-01,  4.84906765e-01,  2.95991299e-01,  3.03976854e-02,
      1.61017594e-01,  1.23582960e-02,  5.09758927e-02,  2.18895649e-01,
      1.04321900e-01,  1.665585806e-01,  1.36936901e-01,  6.37239069e-02,
     -2.08362199e-02,  1.28874711e-01,  1.38384061e-01,  6.03325324e-02,
      6.98153995e-02,  2.35859274e-01, -6.00329803e-02, -9.89108149e-02,
```

2.21511165e-01,	6.18359953e-02,	9.30277504e-02,	5.80386279e-01,
1.93564028e-02,	-9.32565016e-03,	-1.13041517e-01,	2.14546253e-01,
2.65779499e-01,	1.96429567e-01,	-1.08999488e-02,	2.61329516e-01,
4.21812609e-01,	3.91748783e-01,	1.83843557e-01,	4.10582146e-01,
4.84774967e-01,	1.41769091e-01,	1.39685141e-01,	3.53000662e-01,
2.28650831e-01,	1.52119497e-01,	2.26633418e-01,	2.41441574e-01,
2.64871106e-01,	2.41869977e-02,	1.52073280e-01,	-1.35819910e-01,
1.92607185e-01,	2.78710159e-01,	1.20581084e-01,	2.77163315e-01,
6.64696991e-02,	3.23551671e-01,	1.08952975e-01,	3.32287782e-02,
-1.58934100e-02,	1.43172846e-01,	-7.51313060e-02,	-5.97542932e-02,
3.26655464e-01,	-1.07957393e-01,	1.98412126e-01,	5.88459119e-01,
1.98920783e-01,	3.15497726e-01,	1.98639285e-01,	1.86730985e-01,
4.28513757e-04,	-4.72964961e-02,	1.29554081e-02,	9.32848407e-02,
1.69632702e-01,	1.65048162e-01,	-6.09104671e-02,	2.05156328e-01,
1.98878469e-01,	2.04776680e-01,	1.15330474e-01,	1.95407168e-01,
3.36338182e-02,	1.49667696e-01,	-6.63171254e-02,	5.31367132e-01,
6.34709828e-02,	6.16263893e-02,	2.993337375e-01,	1.24323764e-01,
5.31942655e-01,	1.57595244e-01,	3.69883154e-01,	3.65107098e-01,
2.49525088e-01,	8.12709655e-02,	1.15422787e-01,	2.17234975e-01,
8.53718601e-02,	-3.80121481e-02,	3.35711304e-01,	5.52671608e-02,
2.23085245e-01,	2.59407637e-01,	4.57360629e-01,	1.39033495e-01,
2.82233093e-01,	8.92098864e-02,	4.29197571e-01,	-6.55813190e-02,
1.60831493e-01,	-2.08025590e-02,	1.72479271e-01,	1.84144786e-01,
6.67305380e-02,	1.23829784e-01,	1.56293428e-01,	-1.30616773e-03,
1.31738963e-02,	1.13297330e-01,	4.29784379e-02,	2.40407291e-01,
1.62972288e-01,	2.53123540e-01,	5.24296551e-01,	1.37744588e-01,
3.80316507e-01,	-6.15864973e-02,	1.54002234e-01,	2.91703845e-01,
3.062776472e-01,	2.56527790e-02,	1.20247929e-02,	2.39772532e-01,

y_test

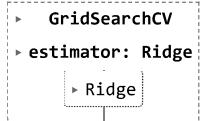
```
from sklearn import metrics
print(metrics.r2_score(y_test,y_pred_rg))
print(metrics.r2_score(y_train,ridgecv.predict(x_train)))
```

0.2042360987236791
 0.20382458288698668

```
from sklearn.linear_model import Lasso  
from sklearn.model_selection import GridSearchCV
```

```
la=Ridge()
```

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



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



```

print(classification_report(y_test,pred))

      precision    recall  f1-score   support

       0          0.87      0.87      0.87      371
       1          0.30      0.30      0.30       70

  accuracy                           0.78      441
   macro avg      0.59      0.59      0.59      441
weighted avg      0.78      0.78      0.78      441

```

```

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

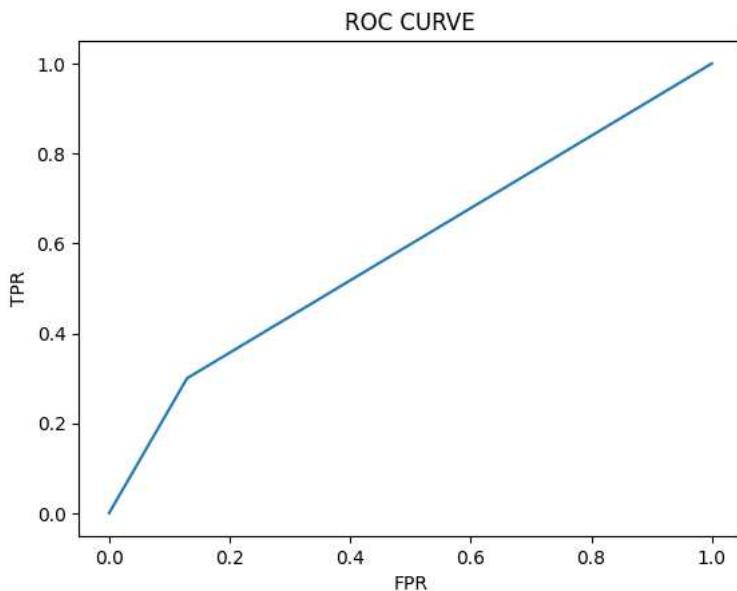
fpr,tpr,thresholds = roc_curve(y_test,probability)

```

```

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

```



```

from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()

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

from sklearn.model_selection import GridSearchCV

rfc_cv= GridSearchCV(rfc,param_grid=forest_params,cv=10,scoring="accuracy")

```