Data Collection. o Collect the dataset or Create the dataset • Data Preprocessing. o Import the Libraries. o Importing the dataset. o Checking for Null Values. o Data Visualization. o Outlier Detection o Splitting Dependent and Independent variables o- Encoding o Feature Scaling. o Splitting Data into Train and Test. • 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 • Application Building o Create an HTML file o Build a Python Code

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

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

```
df.head()
```

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

5 rows × 35 columns

```
df.shape
```

```
(1470, 35)
```

```
df.StockOptionLevel.value_counts()
```

```
0    631
1    596
2    158
3     85
Name: StockOptionLevel, dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64
 6   Education                 1470 non-null   int64
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64
 9   EmployeeNumber            1470 non-null   int64
 10  EnvironmentSatisfaction   1470 non-null   int64
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64
 13  JobInvolvement            1470 non-null   int64
 14  JobLevel                  1470 non-null   int64
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64
 19  MonthlyRate               1470 non-null   int64
 20  NumCompaniesWorked        1470 non-null   int64
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
 24  PerformanceRating         1470 non-null   int64
 25  RelationshipSatisfaction  1470 non-null   int64
 26  StandardHours             1470 non-null   int64
 27  StockOptionLevel          1470 non-null   int64
 28  TotalWorkingYears         1470 non-null   int64
```

```
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

```
df.describe()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyR
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.00
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.89
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.32
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.00
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.00
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.00
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.75
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.00

8 rows × 26 columns

```
df.isnull().any()
```

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

```
df.isnull().sum()
```

```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
```

```
JobRole                   0
JobSatisfaction           0
MaritalStatus             0
MonthlyIncome             0
MonthlyRate               0
NumCompaniesWorked        0
Over18                    0
OverTime                  0
PercentSalaryHike         0
PerformanceRating         0
RelationshipSatisfaction  0
StandardHours             0
StockOptionLevel          0
TotalWorkingYears         0
TrainingTimesLastYear     0
WorkLifeBalance           0
YearsAtCompany            0
YearsInCurrentRole        0
YearsSinceLastPromotion   0
YearsWithCurrManager      0
dtype: int64
```

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

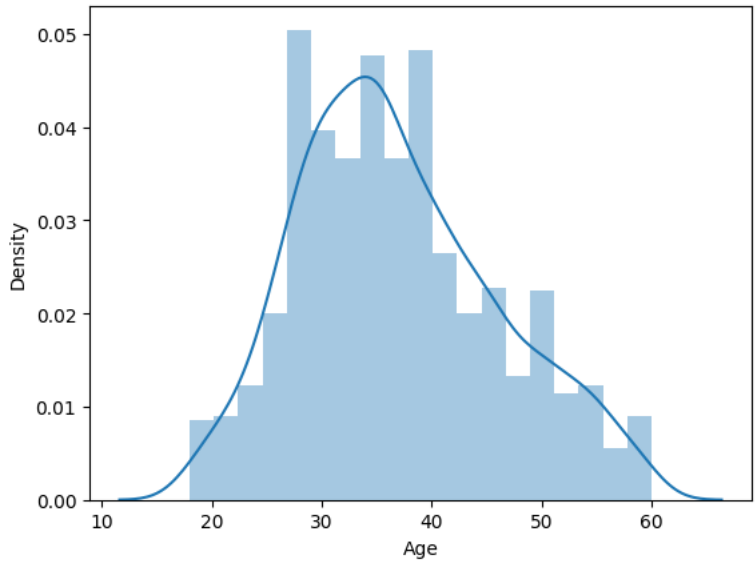
```
<ipython-input-12-cf0334540b62>:1: UserWarning:

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

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

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

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



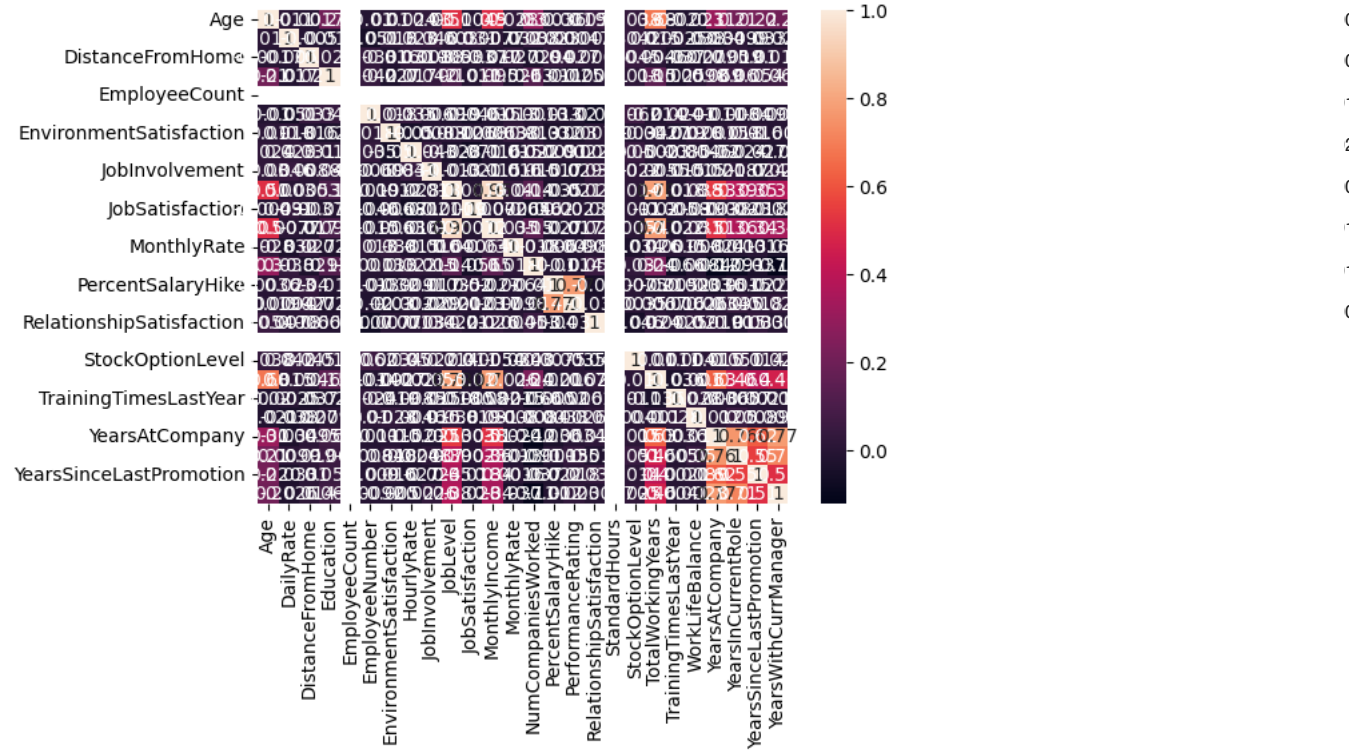

```
df.corr()
```

```
<ipython-input-13-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a
  df.corr()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfac
Age	1.000000	0.010661	-0.001686	0.208034	NaN	-0.010145	0.0
DailyRate	0.010661	1.000000	-0.004985	-0.016806	NaN	-0.050990	0.0
DistanceFromHome	-0.001686	-0.004985	1.000000	0.021042	NaN	0.032916	-0.0
Education	0.208034	-0.016806	0.021042	1.000000	NaN	0.042070	-0.0
EmployeeCount	NaN	NaN	NaN	NaN	NaN	NaN	
EmployeeNumber	-0.010145	-0.050990	0.032916	0.042070	NaN	1.000000	0.0
EnvironmentSatisfaction	0.010146	0.018355	-0.016075	-0.027128	NaN	0.017621	1.0
HourlyRate	0.024287	0.023381	0.031131	0.016775	NaN	0.035179	-0.0
JobInvolvement	0.029820	0.046135	0.008783	0.042438	NaN	-0.006888	-0.0
JobLevel	0.509604	0.002966	0.005303	0.101589	NaN	-0.018519	0.0
JobSatisfaction	-0.004892	0.030571	-0.003669	-0.011296	NaN	-0.046247	-0.0
MonthlyIncome	0.497855	0.007707	-0.017014	0.094961	NaN	-0.014829	-0.0
MonthlyRate	0.028051	-0.032182	0.027473	-0.026084	NaN	0.012648	0.0
NumCompaniesWorked	0.299635	0.038153	-0.029251	0.126317	NaN	-0.001251	0.0
PercentSalaryHike	0.003634	0.022704	0.040235	-0.011111	NaN	-0.012944	-0.0

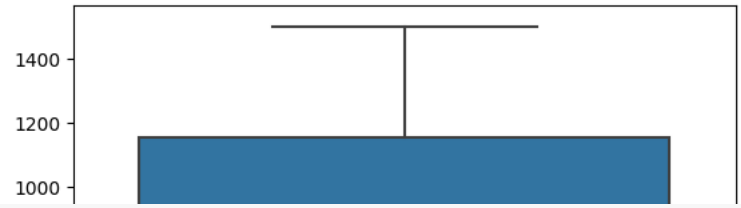
```
sns.heatmap(df.corr() , annot = True)
```

```
<ipython-input-14-19b7fba43d4c>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In
  sns.heatmap(df.corr() , annot = True)
<Axes: >
```



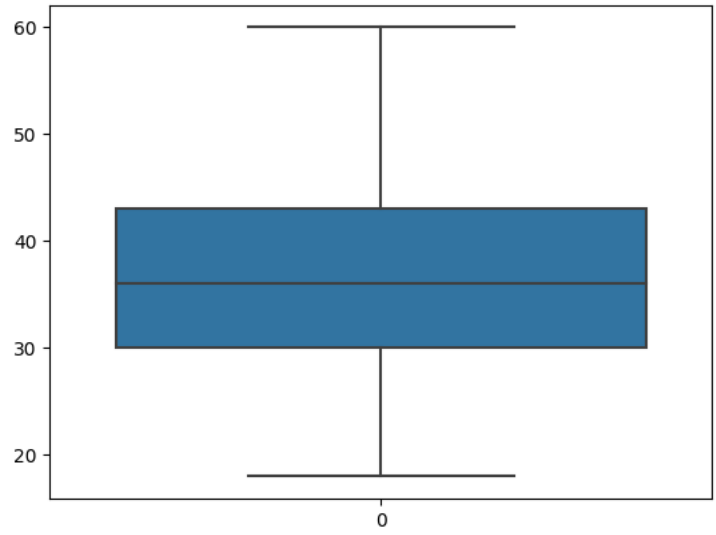

```
sns.boxplot(df.DailyRate)
```

```
<Axes: >
```




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

```
<Axes: >
```




```
df.head()
```

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

5 rows × 35 columns

```
x = df.iloc[:,:4]
x.head()
```

	Age	Attrition	BusinessTravel	DailyRate
0	41	Yes	Travel_Rarely	1102
1	49	No	Travel_Frequently	279
2	37	Yes	Travel_Rarely	1373
3	33	No	Travel_Frequently	1392
4	27	No	Travel_Rarely	591

```
y = df.StockOptionLevel
y.head()
```

```
0    0
1    1
2    0
3    0
4    1
Name: StockOptionLevel, dtype: int64
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
x.Attrition = le.fit_transform(x.Attrition)
x.head()
```

	Age	Attrition	BusinessTravel	DailyRate
0	41	1	Travel_Rarely	1102
1	49	0	Travel_Frequently	279
2	37	1	Travel_Rarely	1373
3	33	0	Travel_Frequently	1392
4	27	0	Travel_Rarely	591

```
le = LabelEncoder()
x.BusinessTravel = le.fit_transform(x.BusinessTravel)
x.head()
```

	Age	Attrition	BusinessTravel	DailyRate
0	41	1	2	1102
1	49	0	1	279
2	37	1	2	1373
3	33	0	1	1392
4	27	0	2	591

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

	Age	Attrition	BusinessTravel	DailyRate
0	0.547619	1.0	1.0	0.715820
1	0.738095	0.0	0.5	0.126700
2	0.452381	1.0	1.0	0.909807
3	0.357143	0.0	0.5	0.923407
4	0.214286	0.0	1.0	0.350036
...	...	...	...	...
1465	0.428571	0.0	0.5	0.559771
1466	0.500000	0.0	1.0	0.365784
1467	0.214286	0.0	1.0	0.037938
1468	0.738095	0.0	0.5	0.659270
1469	0.380952	0.0	1.0	0.376521

1470 rows × 4 columns

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

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

```
((1176, 4), (294, 4), (1176,), (294,))
```

```
x_train.head()
```

	Age	Attrition	BusinessTravel	DailyRate
1002	0.642857	0.0	1.0	0.607915

Logistic regression

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

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

```
▾ LogisticRegression
LogisticRegression()
```

```
pred = model.predict(x_test)
```

```
pred
```

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

```
y_test
```

```
442     0
1091    0
981     1
785     1
1332    0
       ..
1439    2
481     1
124     0
198     0
1229    1
Name: StockOptionLevel, Length: 294, dtype: int64
```

```
df
```

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

```
model.predict(ms.transform([[0.357143,0.0,0.5,0.923407]]))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but MinMa
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but Logis
  warnings.warn(
array([0])
```

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

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

```
0.48299319727891155
```

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

```
array([[72, 55,  0,  0],
       [56, 70,  0,  0],
       [12, 17,  0,  0],
       [ 7,  5,  0,  0]])
```

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

col_0	0	1
StockOptionLevel		
0	72	55
1	56	70
2	12	17
3	7	5

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

```
              precision    recall  f1-score   support

           0       0.49      0.57      0.53       127
           1       0.48      0.56      0.51       126
           2       0.00      0.00      0.00        29
           3       0.00      0.00      0.00        12

    accuracy                           0.48       294
   macro avg       0.24      0.28      0.26       294
weighted avg       0.42      0.48      0.45       294

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F
  _warn_prf(average, modifier, msg_start, len(result))
```

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

```
0.37704638, 0.25474628, 0.39212757, 0.48998872, 0.36423451,
0.47675607, 0.27529761, 0.37631384, 0.22763938, 0.47588246,
0.41186374, 0.40322484, 0.20308884, 0.41755869, 0.49406593,
0.32449296, 0.46289286, 0.39567549, 0.4568658 , 0.36679296,
0.39639401, 0.49833031, 0.48254698, 0.33404629, 0.49284674,
0.40651582, 0.41672567, 0.4753446 , 0.37650721, 0.47087929,
0.3818264 , 0.43843179, 0.34549193, 0.31224176, 0.4145689 ,
0.46418122, 0.41166599, 0.42824331, 0.42744064, 0.38709819,
0.23978081, 0.36786845, 0.35316342, 0.18654034, 0.2120848 ,
0.5464013 , 0.29032349, 0.4126091 , 0.47089042, 0.2702083 ,
0.42973823, 0.41521412, 0.32885513, 0.37706001, 0.46085173,
0.45856421, 0.36106089, 0.43606135, 0.36033219, 0.4493689 ,
0.51142985, 0.28933998, 0.24729348, 0.45154839, 0.25633732,
```

```
       0.35874969, 0.52426839, 0.55812476, 0.23798133, 0.44232166,
       0.4759602 , 0.40957438, 0.43000671, 0.38269376, 0.37436428,
       0.393337  , 0.53418509, 0.54200145, 0.42252918, 0.37666595,
       0.31115076, 0.4234845 , 0.41759473, 0.41743188, 0.38067797,
       0.43036209, 0.41348247, 0.3862818 , 0.42829305, 0.39334226,
       0.24389453, 0.27524722, 0.49659614, 0.51217095, 0.41536427,
       0.51237664, 0.45147217, 0.38348501, 0.50354111, 0.42822098,
       0.37803376, 0.44542215, 0.54556825, 0.40341246, 0.36999256,
       0.45346718, 0.37536645, 0.45135752, 0.50475678, 0.41125781,
       0.47666215, 0.39174442, 0.39017922, 0.22388436, 0.52991052,
       0.53875244, 0.52885558, 0.41117769, 0.46378499, 0.36508042,
       0.46370068, 0.28228228, 0.2711845 , 0.37046887, 0.54204887,
       0.46345639, 0.21254482, 0.50503544, 0.38794113, 0.43251129,
       0.46265583, 0.44661362, 0.37796196, 0.38673975, 0.52430746,
       0.33019464, 0.44825008, 0.45497594, 0.26233078, 0.35549572,
       0.38115736, 0.38523308, 0.40289597, 0.31050269, 0.53585229,
       0.49022334, 0.50599687, 0.40118256, 0.39944439, 0.37694233,
       0.28769715, 0.43524988, 0.35834837, 0.24310768, 0.5388659 ,
       0.4574191 , 0.41050837, 0.35458345, 0.43315299, 0.46340716,
       0.44992134, 0.35866514, 0.54027371, 0.37191693, 0.29033496,
       0.33196845, 0.39371539, 0.39684214, 0.38233935, 0.50103299,
       0.38302139, 0.45748894, 0.49874452, 0.3517435 , 0.51834644,
       0.39828366, 0.56545729, 0.40217817, 0.25788292, 0.35426626,
       0.37819206, 0.20631961, 0.40150258, 0.40446605, 0.3607504 ,
       0.48274959, 0.40562294, 0.53022358, 0.23131845, 0.37556291,
       0.39383319, 0.41219961, 0.21352296, 0.42063161, 0.37969579,
       0.38107731, 0.3613703 , 0.50504306, 0.38739066, 0.42481061,
       0.42780344, 0.41790206, 0.45610417, 0.22744096, 0.34421021,
       0.43552882, 0.47013808, 0.40441442, 0.45806973, 0.3779163 ,
       0.45709677, 0.38148689, 0.33027823, 0.52017576, 0.3526529 ,
       0.52386752, 0.5273219 , 0.35931882, 0.20473947, 0.35990531,
       0.43273755, 0.42979931, 0.38711668, 0.45590059, 0.42938181,
       0.4016247 , 0.38967641, 0.48275064, 0.37195487, 0.51773987,
       0.29730218, 0.35662129, 0.52416294, 0.39683313, 0.47303644,
       0.39565078, 0.40380083, 0.45177485, 0.38674601, 0.39435472,
       0.42419323, 0.43791179, 0.21220383, 0.46139059, 0.4296156 ,
       0.49436096, 0.50037494, 0.23329727, 0.37128198, 0.24701072,
       0.4144757 , 0.3993662 , 0.40381922, 0.39649409, 0.39916637,
       0.50479319, 0.39651705, 0.21091855, 0.26673986, 0.44039745,
       0.4104556 , 0.39113026, 0.38678767, 0.4907775 , 0.44881472,
       0.46128229, 0.41010836, 0.48868465, 0.52609987, 0.36496048,
       0.20327885, 0.37711095, 0.34774677, 0.48018454, 0.53267315,
       0.41492672, 0.40368745, 0.23839733, 0.48261862, 0.41950735,
       0.40515757, 0.23781019, 0.43509554, 0.44168999])
```

```
y_test
```

```
442     0
1091    0
981     1
785     1
1332    0
       ..
1439    2
481     1
124     0
198     0
1229    1
Name: StockOptionLevel, Length: 294, dtype: int64
```

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

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

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

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

```
array([0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       3, 0, 1, 1, 0, 0, 0, 3, 1, 1, 0, 1, 1, 1, 1, 1, 1, 3, 1, 0, 0, 0,
       2, 0, 0, 1, 0, 3, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 2, 0, 2, 2, 1, 1,
       1, 1, 1, 0, 1, 2, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 3, 1, 1,
       1, 1, 0, 0, 1, 1, 1, 1, 2, 1, 1, 0, 0, 0, 1, 0, 2, 1, 1, 1, 0, 1,
       3, 1, 1, 2, 1, 0, 0, 1, 1, 1, 2, 0, 1, 3, 2, 1, 2, 0, 1, 2, 2, 2,
       0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 2, 2, 0,
       2, 2, 1, 1, 0, 2, 1, 1, 2, 0, 1, 1, 0, 3, 2, 1, 1, 1, 1, 0, 0, 0,
       0, 0, 1, 1, 2, 1, 1, 0, 0, 0, 3, 1, 1, 0, 0, 1, 3, 3, 0, 1, 1, 0,
       0, 0, 3, 2, 0, 1, 0, 3, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 2, 0, 2,
       1, 1, 3, 1, 1, 1, 1, 0, 0, 0, 0, 0, 2, 1, 0, 0, 1, 1, 0, 1, 1, 0,
       1, 1, 2, 3, 0, 1, 1, 0, 0, 0, 1, 2, 0, 0, 1, 0, 1, 0, 0, 1, 0, 2,
```

```
       0, 0, 2, 0, 0, 3, 0, 1, 2, 0, 0, 1, 2, 0, 1, 0, 1, 3, 1, 1, 0, 1,
       1, 0, 0, 1, 0, 0, 1, 3])
```

```
y_test
```

```
442     0
1091    0
981     1
785     1
1332    0
       ..
1439    2
481     1
124     0
198     0
1229    1
Name: StockOptionLevel, Length: 294, dtype: int64
```

```
df
```

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

1470 rows × 35 columns

```
dtc.predict(ms.transform([[0.357143,0.0,0.5,0.923407]]))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but MinMa
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but Decis
  warnings.warn(
array([0])
```

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

```
0.3979591836734694
```

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

```
array([[57, 53,  8,  9],
       [43, 56, 19,  8],
       [11, 13,  4,  1],
       [ 5,  6,  1,  0]])
```

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

col_0	0	1	2	3
StockOptionLevel				
0	57	53	8	9
1	43	56	19	8

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

```
                precision    recall  f1-score   support

             0       0.49      0.45      0.47       127
             1       0.44      0.44      0.44       126
             2       0.12      0.14      0.13        29
             3       0.00      0.00      0.00        12

      accuracy                           0.40       294
     macro avg       0.26      0.26      0.26       294
  weighted avg       0.41      0.40      0.40       294
```

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

```
# roc_curve
fpr , tpr , threshsholds = roc_curve(y_test , probability)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-52-17b0a15ba9ba> in <cell line: 2>()
      1 # roc_curve
----> 2 fpr , tpr , threshsholds = roc_curve(y_test , probability)

 1 frames
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_ranking.py in _binary_clf_curve(y_true, y_score, pos_label, sam
    747     y_type = type_of_target(y_true, input_name="y_true")
    748     if not (y_type == "binary" or (y_type == "multiclass" and pos_label is not None)):
--> 749         raise ValueError("{0} format is not supported".format(y_type))
    750
    751     check_consistent_length(y_true, y_score, sample_weight)

ValueError: multiclass format is not supported
```

SEARCH STACK OVERFLOW

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