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

In [ ]:
```
data = pd.read_csv('WA_Fn-UseC_-HR-Employee-Attrition.csv')
data.head()
```

Out[3]:

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

5 rows × 35 columns

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

Out[6]:

	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

In [ ]:
```
data.isnull().sum()
```

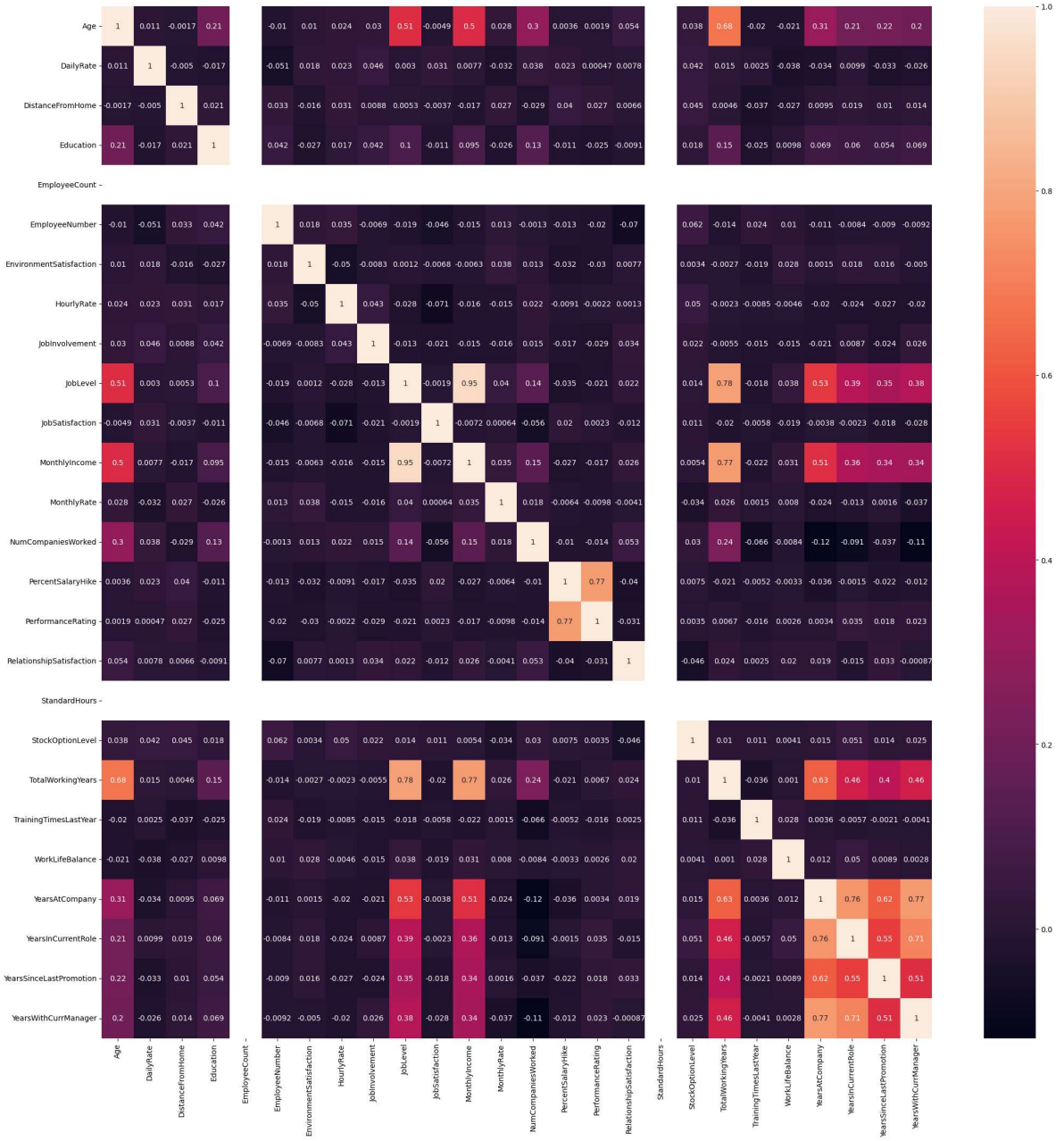
Out[7]:
```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              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 [ ]:
```
cor = data.corr()
```

```
<ipython-input-8-06847dd9a2e1>:1: FutureWarning: The default value of numeric
_only in DataFrame.corr is deprecated. In a future version, it will default t
o False. Select only valid columns or specify the value of numeric_only to si
lence this warning.
  cor = data.corr()
```

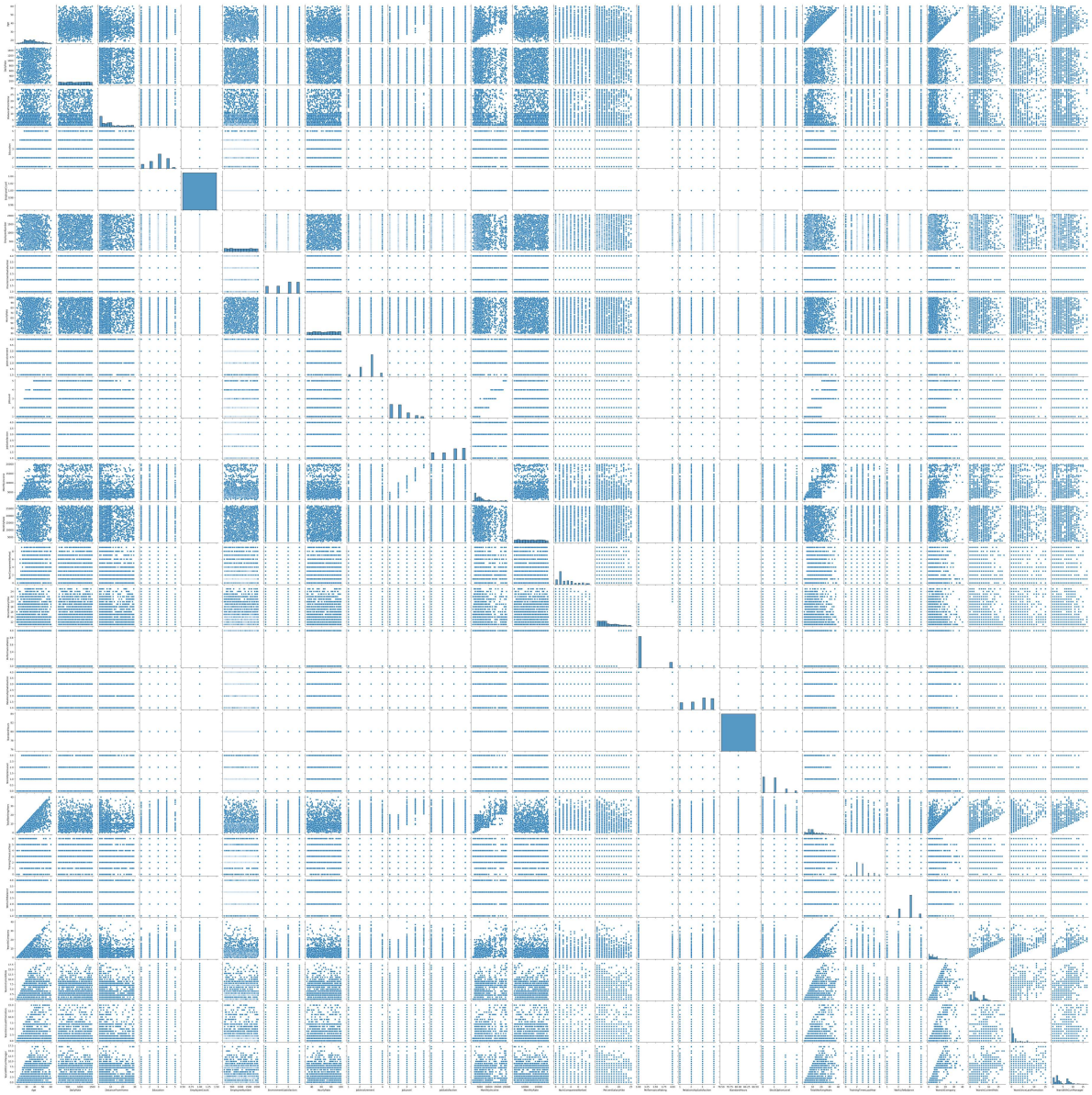
In [ ]:
```
fig, ax = plt.subplots(figsize=(25,25))
sns.heatmap(cor, annot=True)
```

Out[9]: <Axes: >



In [11]:
```
sns.pairplot(data)
```

Out[11]:
```
<seaborn.axisgrid.PairGrid at 0x7bcdd83535e0>
```



In [12]:
```
from sklearn.preprocessing import LabelEncoder
```

In [13]:
```
le=LabelEncoder()
```

In [14]:
```
data["BusinessTravel"]=le.fit_transform(data["BusinessTravel"])
```

In [15]:
```
data["Department"]=le.fit_transform(data["Department"])
```

In [17]:
```
data["EducationField"]=le.fit_transform(data["EducationField"])
```

In [16]:
```
data["Gender"]=le.fit_transform(data["Gender"])
```

In [18]:
```
data["JobRole"]=le.fit_transform(data["JobRole"])
```

In [19]:
```
data["MaritalStatus"]=le.fit_transform(data["MaritalStatus"])
```

In [20]:
```
data["Over18"]=le.fit_transform(data["Over18"])
```

In [21]:
```
data["OverTime"]=le.fit_transform(data["OverTime"])
```

In [22]:
```
data.head()
```

Out[22]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educat
0	41	Yes	2	1102	2	1	2	
1	49	No	1	279	1	8	1	
2	37	Yes	2	1373	1	2	2	
3	33	No	1	1392	1	3	4	
4	27	No	2	591	1	2	1	

5 rows × 35 columns

In [23]:
```
data.tail()
```

Out[23]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Edu
1465	36	No	1	884	1	23	2	
1466	39	No	2	613	1	6	1	
1467	27	No	2	155	1	4	3	
1468	49	No	1	1023	2	2	3	
1469	34	No	2	628	1	8	3	

5 rows × 35 columns

In [24]:
```
X=data.drop(columns=["EmployeeNumber","EmployeeCount","StandardHours","Attriti
```

In [25]:
```
y=data["Attrition"]
```

In [26]:
```
from sklearn.preprocessing import MinMaxScaler
ms=MinMaxScaler()
```

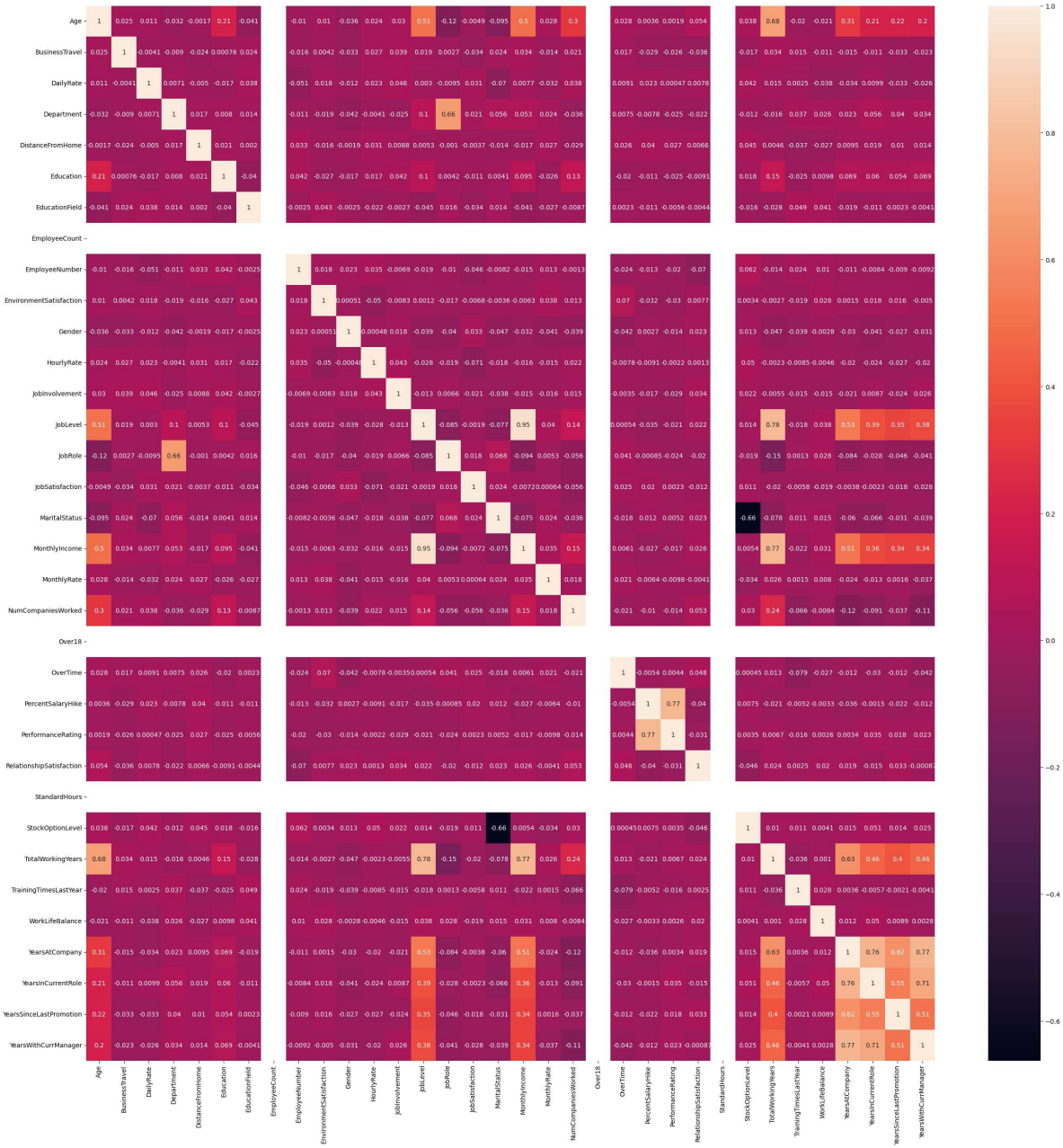
In [27]:
```
X_Scaled=ms.fit_transform(X)
```

In [28]:
```
cor=data.corr()
```

```
<ipython-input-28-410fe4458127>:1: FutureWarning: The default value of numeri
c_only in DataFrame.corr is deprecated. In a future version, it will default
to False. Select only valid columns or specify the value of numeric_only to s
ilence this warning.
  cor=data.corr()
```

In [29]:
```
fig, ax = plt.subplots(figsize=(30,30))
sns.heatmap(cor, annot=True)
```

Out[29]: <Axes: >



In [30]:
```
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,ran
```

In [31]:
```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state=0)
classifier.fit(x_train,y_train)
```

Out[31]:
```
LogisticRegression(random_state=0)
```
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 [32]:
```
from sklearn.metrics import accuracy_score,confusion_matrix
y_pred = classifier.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)*100
```

```
[[242   3]
 [ 32  17]]
```

Out[32]:
```
88.09523809523809
```

In [33]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_rep
```

In [34]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

          No       0.88      0.99      0.93       245
         Yes       0.85      0.35      0.49        49

    accuracy                           0.88       294
   macro avg       0.87      0.67      0.71       294
weighted avg       0.88      0.88      0.86       294
```

In [35]:
```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
```

In [36]:
```
dtc.fit(x_train,y_train)
```

Out[36]:
```
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 [37]:
```
from sklearn.metrics import accuracy_score,confusion_matrix
y_pred = dtc.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)*100
```

```
[[205  40]
 [ 34  15]]
```

Out[37]:
```
74.82993197278913
```

In [38]:
```
from sklearn import tree
plt.figure(figsize=(25,15))
tree.plot_tree(dtc,filled=True)
```

Out[38]:
```
[Text(0.319423577724359, 0.9722222222222222, 'x[23] <= 0.038\ngini = 0.269
\nsamples = 1176\nvalue = [988, 188]'),
 Text(0.07692307692307693, 0.9166666666666666, 'x[14] <= 0.75\ngini = 0.5
\nsamples = 78\nvalue = [39, 39]'),
 Text(0.04807692307692308, 0.8611111111111112, 'x[4] <= 0.554\ngini = 0.42
6\nsamples = 39\nvalue = [27, 12]'),
 Text(0.03205128205128205, 0.8055555555555556, 'x[13] <= 0.167\ngini = 0.3
12\nsamples = 31\nvalue = [25, 6]'),
 Text(0.019230769230769232, 0.75, 'x[18] <= 0.5\ngini = 0.49\nsamples = 7
\nvalue = [3, 4]'),
 Text(0.01282051282051282, 0.6944444444444444, 'x[2] <= 0.298\ngini = 0.37
5\nsamples = 4\nvalue = [3, 1]'),
 Text(0.00641025641025641, 0.6388888888888888, 'gini = 0.0\nsamples = 1\nv
alue = [0, 1]'),
 Text(0.019230769230769232, 0.6388888888888888, 'gini = 0.0\nsamples = 3\n
value = [3, 0]'),
 Text(0.02564102564102564, 0.6944444444444444, 'gini = 0.0\nsamples = 3\nv
alue = [0, 3]'),
 Text(0.04487179487179487, 0.75, 'x[17] <= 0.056\ngini = 0.153\nsamples =
```

In [39]:
```
from sklearn.model_selection import GridSearchCV
parameter={
 'criterion':['gini','entropy'],
  'splitter':['best','random'],
  'max_depth':[1,2,3,4,5,6,7,8,9,10],
  'max_features':['auto', 'sqrt', 'log2']

}
```

In [40]:
```
grid_search=GridSearchCV(estimator=dtc,param_grid=parameter,cv=5,scoring="accu
```

In [41]:
```
grid_search.fit(x_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: Futu
reWarning: `max_features='auto'` has been deprecated in 1.1 and will be re
moved in 1.3. To keep the past behaviour, explicitly set `max_features='sq
rt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: Futu
reWarning: `max_features='auto'` has been deprecated in 1.1 and will be re
moved in 1.3. To keep the past behaviour, explicitly set `max_features='sq
rt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: Futu
reWarning: `max_features='auto'` has been deprecated in 1.1 and will be re
moved in 1.3. To keep the past behaviour, explicitly set `max_features='sq
rt'`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: Futu
reWarning: `max_features='auto'` has been deprecated in 1.1 and will be re
moved in 1.3. To keep the past behaviour, explicitly set `max_features='sq
rt'`.
```

In [42]:
```
grid_search.best_params_
```

Out[42]:
```
{'criterion': 'gini',
 'max_depth': 5,
 'max_features': 'log2',
 'splitter': 'best'}
```

In [43]:
```
dtc_cv=DecisionTreeClassifier(criterion= 'entropy',
 max_depth= 4,
 max_features= 'sqrt',
 splitter= 'best')
dtc_cv.fit(x_train,y_train)
```

Out[43]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=4, max_features='sqrt')
```

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 [44]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

          No       0.86      0.84      0.85       245
         Yes       0.27      0.31      0.29        49

    accuracy                           0.75       294
   macro avg       0.57      0.57      0.57       294
weighted avg       0.76      0.75      0.75       294
```

In [45]:
```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 1000, criterion = 'entropy'
classifier.fit(x_train, y_train)
```

Out[45]:
```
RandomForestClassifier(criterion='entropy', n_estimators=1000, random_state=
0)
```

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 [46]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
y_pred = classifier.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[243   2]
 [ 41   8]]
```

Out[46]:
```
0.8537414965986394
```

In [47]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [48]:
```
rfc=RandomForestClassifier()
```

In [49]:
```
forest_params = [{'max_depth': list(range(10, 15)), 'max_features': list(range
```

In [50]:
```
rfc_cv=GridSearchCV(rfc,param_grid=forest_params,cv=10,scoring="accuracy")
```

In [51]:
```
rfc_cv.fit(x_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.p
y:378: 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 erro
r_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
---
50 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_vali
dation.py", line 686, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py",
line 340, in fit
    self._validate_params()
  File "/usr/local/lib/python3.10/dist-packages/sklearn/base.py", line 600, i
n _validate_params
    validate_parameter_constraints(
  File "/usr/local/lib/python3.10/dist-packages/sklearn/utils/_param_validati
on.py", line 97, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' par
ameter of RandomForestClassifier must be an int in the range [1, inf), a floa
t in the range (0.0, 1.0], a str among {'sqrt', 'auto' (deprecated), 'log2'}
or None. Got 0 instead.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_search.py:95
2: UserWarning: One or more of the test scores are non-finite: [       nan 0.
84608866 0.85374475 0.85713458 0.85969144 0.86222657
 0.85881501 0.86053165 0.85880776 0.86562364 0.85712734 0.86050992
 0.86051717 0.86308851        nan 0.84950746 0.85628712 0.86052441
 0.85882949 0.86306678 0.86307403 0.85456323 0.85370853 0.86392873
 0.85627264 0.85885122 0.85882225 0.85625815        nan 0.84864552
 0.85544691 0.85714182 0.8596842  0.86137911 0.86136462 0.85966971
 0.85966247 0.85965522 0.85883674 0.86221932 0.85882225 0.85882225
       nan 0.84695784 0.8613936  0.86138635 0.85712009 0.86224105
 0.85967695 0.85882949 0.85798204 0.86137187 0.86394321 0.86479067
 0.85541793 0.86308127        nan 0.85204983 0.85883674 0.85629436
 0.85967695 0.85798928 0.86053165 0.85964798 0.86222657 0.85966971
 0.86137911 0.86647834 0.85627264 0.86135014]
  warnings.warn(
```

Out[51]:
```
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 [52]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

          No       0.86      0.99      0.92       245
         Yes       0.80      0.16      0.27        49

    accuracy                           0.85       294
   macro avg       0.83      0.58      0.59       294
weighted avg       0.85      0.85      0.81       294
```