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

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

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

5 rows × 35 columns

```
data.tail()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educ
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	EmployeeNumber	EnvironmentSatisfaction	HourlyRate
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000

8 rows × 26 columns

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

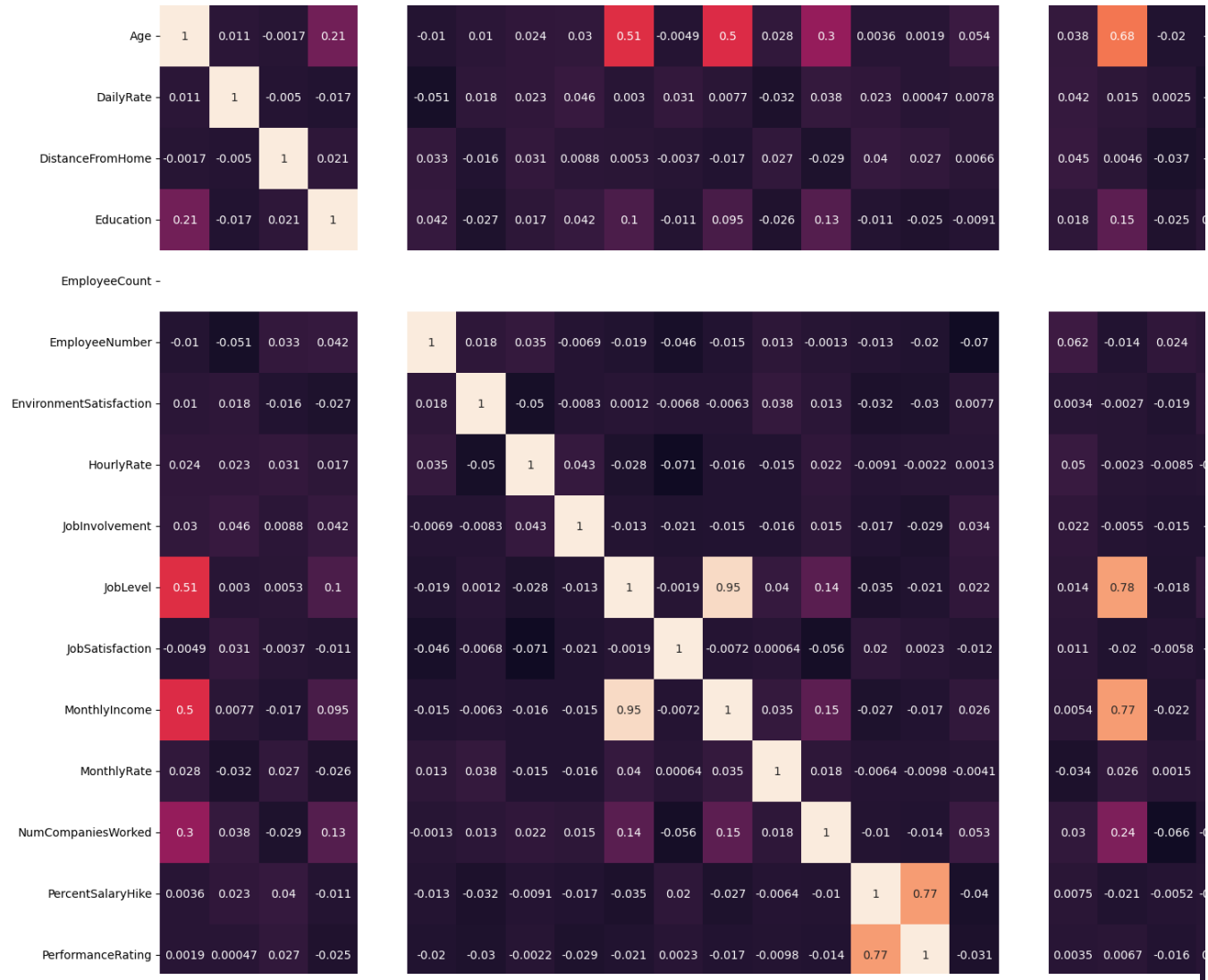
```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              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-7-06847dd9a2e1>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ver
  cor = data.corr()
```

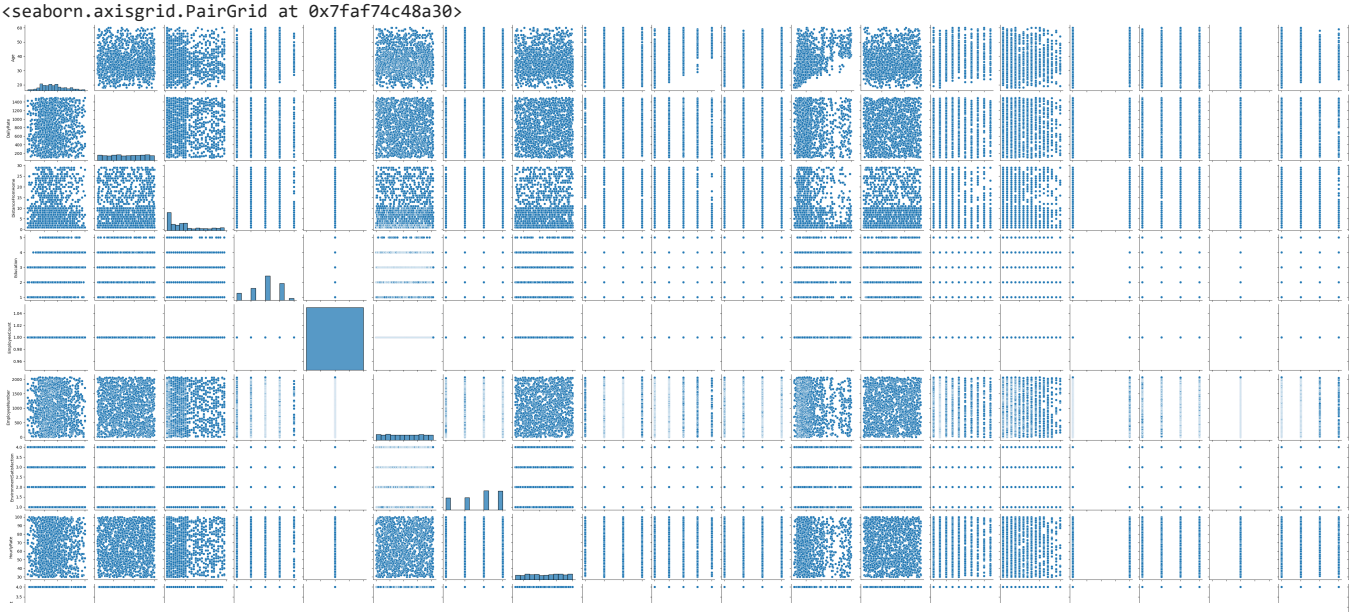
```
fig, ax = plt.subplots(figsize=(25,25))
sns.heatmap(cor, annot=True)
```

```
<Axes: >
```




```
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x7faf74c48a30>
```



```
from sklearn.preprocessing import LabelEncoder
```

```
le=LabelEncoder()
```

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

```
data["BusinessTravel"]=le.fit_transform(data["BusinessTravel"])
data["Department"]=le.fit_transform(data["Department"])
data["EducationField"]=le.fit_transform(data["EducationField"])
data["Gender"]=le.fit_transform(data["Gender"])
data["JobRole"]=le.fit_transform(data["JobRole"])
data["MaritalStatus"]=le.fit_transform(data["MaritalStatus"])
data["Over18"]=le.fit_transform(data["Over18"])
data["OverTime"]=le.fit_transform(data["OverTime"])
```

```
data.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber
0	41	Yes	2	1102	2	1	2	1	1	1
1	49	No	1	279	1	8	1	1	1	2
2	37	Yes	2	1373	1	2	2	4	1	4
3	33	No	1	1392	1	3	4	1	1	5
4	27	No	2	591	1	2	1	3	1	7

5 rows × 35 columns

```
data.tail()
```

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

5 rows × 35 columns

```
X=data.drop(columns=["EmployeeNumber","EmployeeCount","StandardHours","Attrition","Over18"],axis=1)
```

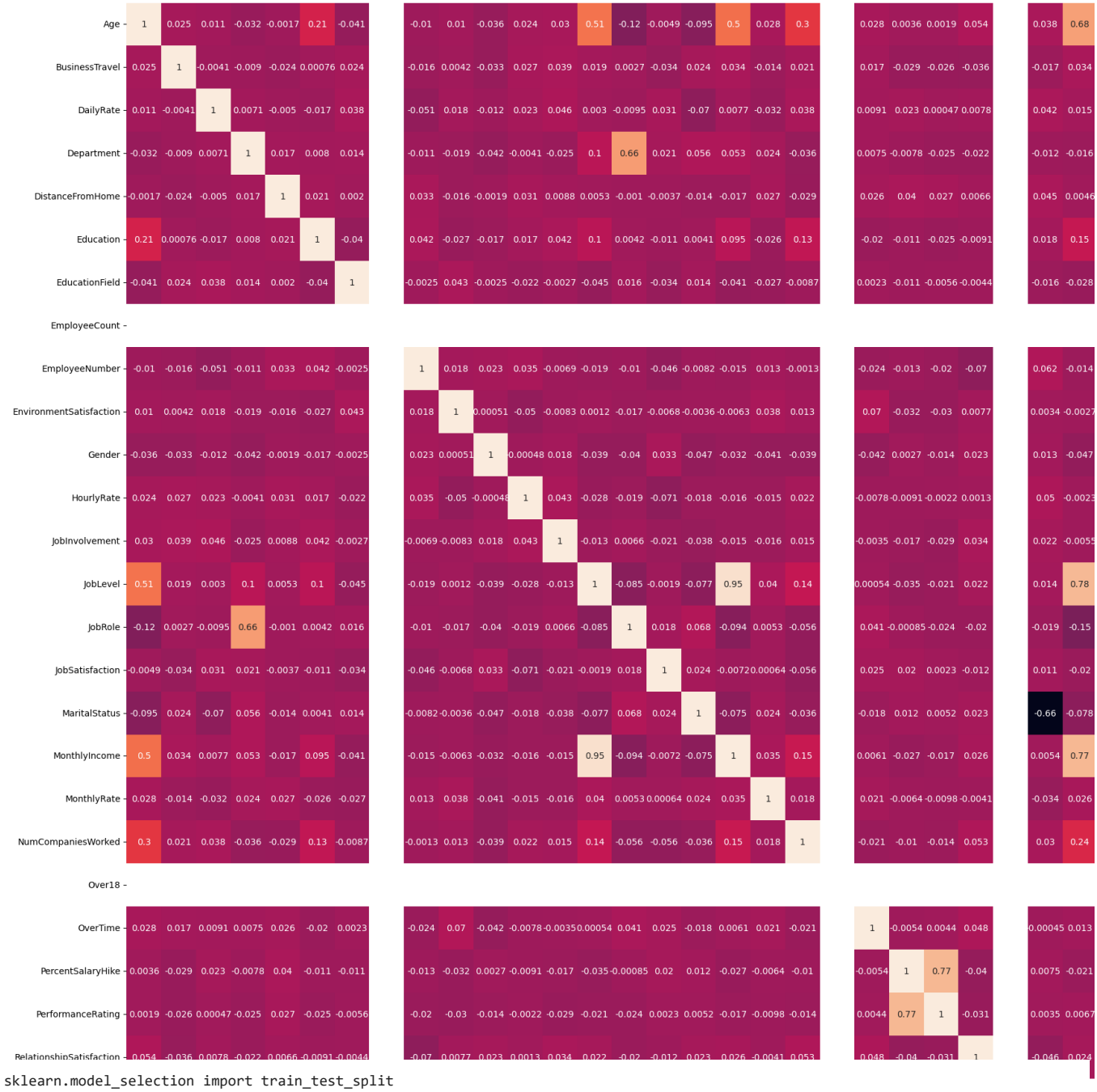
```
y=data["Attrition"]
```

```
from sklearn.preprocessing import MinMaxScaler
ms=MinMaxScaler()


X_Scaled=ms.fit_transform(X)


cor=data.corr()
```

```
<ipython-input-20-410fe4458127>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ve
  cor=data.corr()
```

```
fig, ax = plt.subplots(figsize=(30,30))
sns.heatmap(cor, annot=True)
```

```
<Axes: >
```




```
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)
```

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state=0)
classifier.fit(x_train,y_train)
```

```
LogisticRegression
LogisticRegression(random_state=0)
```

```
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]]
88.09523809523809
```

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

```
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
```

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

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

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
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
```

```
[[209  36]
 [ 32  17]]
76.87074829931973
```

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

```
[Text(0.3236328125, 0.9722222222222222, 'x[23] <= 0.038\ngini = 0.269\nsamples = 1176\nvalue = [988, 188]'),
 Text(0.07333333333333333, 0.9166666666666666, 'x[14] <= 0.75\ngini = 0.5\nsamples = 78\nvalue = [39, 39]'),
 Text(0.043333333333333335, 0.8611111111111112, 'x[4] <= 0.554\ngini = 0.426\nsamples = 39\nvalue = [27, 12]'),
 Text(0.02666666666666667, 0.8055555555555556, 'x[13] <= 0.167\ngini = 0.312\nsamples = 31\nvalue = [25, 6]'),
 Text(0.013333333333333334, 0.75, 'x[15] <= 0.057\ngini = 0.49\nsamples = 7\nvalue = [3, 4]'),
 Text(0.006666666666666667, 0.6944444444444444, 'gini = 0.0\nsamples = 3\nvalue = [0, 3]'),
 Text(0.02, 0.6944444444444444, 'x[14] <= 0.25\ngini = 0.375\nsamples = 4\nvalue = [3, 1]'),
 Text(0.013333333333333334, 0.6388888888888888, 'gini = 0.0\nsamples = 3\nvalue = [3, 0]'),
 Text(0.02666666666666667, 0.6388888888888888, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.04, 0.75, 'x[17] <= 0.056\ngini = 0.153\nsamples = 24\nvalue = [22, 2]'),
 Text(0.03333333333333333, 0.6944444444444444, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.04666666666666667, 0.6944444444444444, 'x[7] <= 0.167\ngini = 0.083\nsamples = 23\nvalue = [22, 1]'),
 Text(0.04, 0.6388888888888888, 'x[12] <= 0.5\ngini = 0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(0.03333333333333333, 0.5833333333333334, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.04666666666666667, 0.5833333333333334, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.05333333333333334, 0.6388888888888888, 'gini = 0.0\nsamples = 21\nvalue = [21, 0]'),
 Text(0.06, 0.8055555555555556, 'x[19] <= 0.679\ngini = 0.375\nsamples = 8\nvalue = [2, 6]'),
 Text(0.05333333333333334, 0.75, 'gini = 0.0\nsamples = 6\nvalue = [0, 6]'),
 Text(0.06666666666666667, 0.75, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(0.10333333333333333, 0.8611111111111112, 'x[9] <= 0.364\ngini = 0.426\nsamples = 39\nvalue = [12, 27]'),
 Text(0.08666666666666667, 0.8055555555555556, 'x[25] <= 0.167\ngini = 0.133\nsamples = 14\nvalue = [1, 13]'),
```

```
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']
}
```

```
Text(0.10666666666666667, 0.5277777777777778, 'gini = 0.0\nsamples = 10\nvalue = [10, 0]'),
```

```
grid_search=GridSearchCV(estimator=dtc,param_grid=parameter,cv=5,scoring="accuracy")
```

```
Text(0.13333333333333333, 0.75, 'x[2] <= 0.162\ngini = 0.219\nsamples = 8\nvalue = [1, 7]'),
```

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

```
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1
  warnings.warn(
```

```
grid_search.best_params_
```

```
{'criterion': 'gini',
 'max_depth': 3,
 'max_features': 'sqrt',
 'splitter': 'best'}
```

```
  warnings.warn(
```

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

```
DecisionTreeClassifier(criterion='entropy', max_depth=4, max_features='sqrt')
```

```
  warnings.warn(
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

          No       0.87      0.85      0.86       245
         Yes       0.32      0.35      0.33        49

    accuracy                           0.77       294
   macro avg       0.59      0.60      0.60       294
weighted avg       0.78      0.77      0.77       294
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1
```

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 1000, criterion = 'entropy', random_state = 0)
classifier.fit(x_train, y_train)
```

```
RandomForestClassifier(criterion='entropy', n_estimators=1000, random_state=0)
```

```
  warnings.warn(
```

```
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]]
0.8537414965986394
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1
```

```
from sklearn.ensemble import RandomForestClassifier
  warnings.warn(
```

```
rfc=RandomForestClassifier()
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1
```

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

```
rfc_cv=GridSearchCV(rfc,param_grid=forest_params,cv=10,scoring="accuracy")
/usr/local/lib/python3.10/dist-packages/sklearn/tree/_classes.py:269: FutureWarning: `max_features='auto'` has been deprecated in 1
```

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

```
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py: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 error_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/_validation.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, in _validate_params
    validate_parameter_constraints(
  File "/usr/local/lib/python3.10/dist-packages/sklearn/utils/_param_validation.py", line 97, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter of RandomForestClassifier must be an int in the

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_search.py:952: UserWarning: One or more of the test scores are non
 0.86223381 0.86223381 0.85967695 0.86308127 0.86137187 0.85797479
```

```
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
```