Get underatanding about data set

Company wants to automate the loan eligibility process (real time) based on customer detail provided while filling online application form.These details are Gender, maretial status,Education,Number of dependants,income,loan eligibility,Credit history and others.

1.Customer_ID

2.Gender

3.Married

4.Dependents

5.Education

6.Self_Employed

7.Applicant_Income

8.Coapplicant_Income

9.Loan_Amount

10.Loan_Amount_Term

11.Credit_History

12.Property_Area

13.Loan_Status

Import Library

```
import pandas as pd
```

```
import numpy as np
```

Double-click (or enter) to edit

Import CSV as DataFrame

```
df = pd.read_csv(r'https://github.com/YBI-Foundation/Dataset/raw/main/Loan%20Eligibi
```

Get the first five rows of DataFrame

```
df.head()
```

	Customer_ID	Gender	Married	Dependents	Education	Self_Employed	Applicant_Inc
0	569	Female	No	0	Graduate	No	2
1	15	Male	Yes	2	Graduate	No	1
2	95	Male	No	0	Not Graduate	No	3
3	134	Male	Yes	0	Graduate	Yes	3
4	556	Male	Yes	1	Graduate	No	5

Get Information of DataFrame

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Customer_ID         614 non-null    int64  
 1   Gender              614 non-null    object 
 2   Married             614 non-null    object 
 3   Dependents          614 non-null    int64  
 4   Education           614 non-null    object 
 5   Self_Employed       614 non-null    object 
 6   Applicant_Income    614 non-null    int64  
 7   Coapplicant_Income  614 non-null    float64
 8   Loan_Amount         614 non-null    int64  
 9   Loan_Amount_Term    614 non-null    int64  
 10  Credit_History      614 non-null    int64  
 11  Property_Area       614 non-null    object 
 12  Loan_Status         614 non-null    object 
dtypes: float64(1), int64(6), object(6)
memory usage: 62.5+ KB
```

Get the summary Statistics

```
df.describe()
```

	Customer_ID	Dependents	Applicant_Income	Coapplicant_Income	Loan_Amount	L
count	614.000000	614.000000	614.000000	614.000000	614.000000	
mean	307.500000	0.856678	5403.459283	1621.245798	142.022801	
std	177.390811	1.216651	6109.041673	2926.248369	87.083089	
min	1.000000	0.000000	150.000000	0.000000	9.000000	
25%	154.250000	0.000000	2877.500000	0.000000	98.000000	
50%	307.500000	0.000000	3812.500000	1188.500000	125.000000	

Get Column Names

```
df.shape
```

```
(614, 13)
```

Get the shape of the data Frame

```
df.shape
```

```
(614, 13)
```

Get the Unique values(Class or Lavel) in y Variable

```
df['Loan_Status'].value_counts()
```

```
Y    422
N    192
Name: Loan_Status, dtype: int64
```

```
df.groupby('Loan_Status').mean()
```

	Customer_ID	Dependents	Applicant_Income	Coapplicant_Income	Loan_Amo
Loan_Status					
N	304.406250	0.864583	5446.078125	1877.807292	143.869
Y	308.907583	0.853081	5384.068720	1504.516398	141.182

Get catagories and counts of catagorical variables

```
df['Gender'].value_counts()
```

```
Male      499
Female    115
Name: Gender, dtype: int64
```

```
df['Married'].value_counts()
```

```
Yes    399
No     215
Name: Married, dtype: int64
```

```
df['Education'].value_counts()
```

```
Graduate        480
Not Graduate    134
Name: Education, dtype: int64
```

```
df['Self_Employed'].value_counts()
```

```
No     523
Yes     91
Name: Self_Employed, dtype: int64
```

```
df['Property_Area'].value_counts()
```

```
Semiurban    233
Urban        202
Rural        179
Name: Property_Area, dtype: int64
```

Get Encoding of Catagorical Features

```
df.replace({'Gender':{'Male':0,
'Female':1}},inplace = True)
```

```
df.replace({'Married':{'Yes':1,
'No':0}},inplace = True)
```

```
df.replace({'Education':{'Graduate':1,
'Not Graduate':0}},inplace = True)
```

```
df.replace({'Self_Employed':{'Yes':1,
'No':0}},inplace = True)
```

```
df.replace({'Property_Area':{'Urban':1,
'Semiurban':1,'Rural':0}},inplace = True)
```

Define y (Dependent or label or target variable) and X (independent or features or attribute Variables)

```
y = df ['Loan_Status']
```

```
y.shape
```

```
(614,)
```

```
y
```

```
0      N
1      Y
2      Y
3      Y
4      Y
      ..
609    N
610    N
611    N
612    Y
613    N
Name: Loan_Status, Length: 614, dtype: object
```

```
df.columns
```

```
Index(['Customer_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'Applicant_Income', 'Coapplicant_Income',
       'Loan_Amount', 'Loan_Amount_Term', 'Credit_History', 'Property_Area',
       'Loan_Status'],
      dtype='object')
```

```
X = df[[ 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'Applicant_Income', 'Coapplicant_Income',
       'Loan_Amount', 'Loan_Amount_Term', 'Credit_History', 'Property_Area']]
```

```
X
```

	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Coappl
0	1	0	0	1	0	2378	
1	0	1	2	1	0	1299	
2	0	0	0	0	0	3620	
3	0	1	0	1	1	3459	
4	0	1	1	1	0	5468	
...	...	...	...	...	...	...	
609	0	1	2	1	0	2947	

614 rows × 11 columns

Get train_test_split

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state =254
```

```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

```
((429, 11), (185, 11), (429,), (185,))
```

```
X_train
```

	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Coappl
1	0	1	2	1	0	1299	
238	0	1	1	1	0	3750	
438	0	1	0	0	1	2609	
210	0	1	2	1	0	9703	
388	0	1	2	1	0	3340	
...	...	...	...	...	...	...	
328	0	0	0	0	1	5800	
308	0	1	4	0	1	3333	
200	0	0	0	0	0	3833	
611	0	1	2	1	1	6633	
351	0	1	0	1	0	3087	

429 rows × 11 columns

Standardization of Train and test features

```
X_train_std = X_train[['Applicant_Income','Coapplicant_Income','Loan_Amount','Loan_A
X_test_std = X_test[['Applicant_Income','Coapplicant_Income','Loan_Amount','Loan_Amo
```

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

```
ss = StandardScaler()
```

```
X_train_std = ss.fit_transform(X_train_std)
X_test_std = ss.fit_transform(X_test_std)
```

```
X_train_std
```

```
array([[-0.80571566, -0.19393576, -1.57777937, -3.19876573],
       [-0.30020187, -0.61240927, -0.33240483,  0.30729073],
       [-0.53553081,  0.71661022,  0.28399266, -2.32225162],
       ...,
       [-0.28308328, -0.61240927, -0.40788208,  0.30729073],
       [ 0.29441106, -0.61240927, -1.47714304,  0.30729073],
       [-0.43694428,  0.23918048, -0.08081402,  0.30729073]])
```

```
X_test_std
```

```
array([[-4.47541939e-01,  4.31781875e-01,  3.60655789e-01,
         2.93944575e-01],
       [-3.51366900e-01, -8.77897020e-03, -6.94690237e-01,
        -2.19293407e+00],
       [-3.04846142e-01,  8.53268429e-02, -4.11310286e-01,
         2.93944575e-01],
       [-1.84326044e-01,  3.23923230e-02,  1.84764785e-01,
         2.93944575e-01],
       [-2.23736116e-01, -4.74295720e-02,  2.62938565e-01,
         2.93944575e-01],
       [-3.95236216e-01,  1.77752195e-01, -1.57245502e-01,
         2.93944575e-01],
       [-2.34462405e-01, -4.74826806e-01, -1.67017224e-01,
         2.93944575e-01],
       [-3.24129358e-01,  3.71285281e-01, -3.99848319e-02,
         2.93944575e-01],
       [-2.18433232e-01,  1.01014052e+00, -3.02131095e-02,
         2.93944575e-01],
       [ 8.16618128e-02, -4.74826806e-01,  6.24492296e-01,
         2.93944575e-01],
       [-3.17380233e-01,  2.92303616e-01, -3.03821339e-01,
         2.93944575e-01],
       [-5.36822576e-02, -4.74826806e-01,  2.62938565e-01,
         2.93944575e-01],
       [-4.63812152e-01,  6.54413883e-02, -6.55603347e-01,
```

```
        2.93944575e-01],
      [-1.90352049e-01,  1.92596267e-01, -1.37702057e-01,
        2.93944575e-01],
      [-3.45581936e-01, -4.74826806e-01, -7.92407462e-01,
        2.93944575e-01],
      [-2.76041839e-01,  6.77409250e-01,  1.06591005e-01,
        2.93944575e-01],
      [-3.05689783e-01, -2.19116665e-01, -3.52679951e-01,
       -3.51926934e+00],
      [-4.06203545e-01,  1.90355652e-01, -1.08555914e+00,
        2.93944575e-01],
      [-2.93155693e-01, -1.24730776e-01, -1.27930334e-01,
        2.93944575e-01],
      [-3.70650116e-01,  6.20804664e-02,  4.79606703e-02,
        2.93944575e-01],
      [-5.32749648e-01, -4.74826806e-01, -1.03670052e+00,
        2.93944575e-01],
      [ 8.91797913e-01, -4.74826806e-01, -9.87841911e-01,
        2.93944575e-01],
      [-4.70561278e-01,  2.07440338e-01, -9.86151667e-02,
        2.93944575e-01],
      [ 1.77234251e-01,  1.85905337e+00,  3.31340622e-01,
       -5.35014973e-01],
      [-3.43774134e-01,  4.03494116e-01, -1.13441775e+00,
        2.93944575e-01],
      [-2.41452571e-01,  4.49426715e-01,  8.87378040e-03,
       -2.19293407e+00],
      [ 4.17671847e-01, -4.74826806e-01,  4.58373014e-01,
       -2.19293407e+00],
      [ 8.74715330e-03, -4.74826806e-01, -7.90717218e-02,
        2.93944575e-01],
      [-4.05239384e-01,  5.12670968e+00, -3.72223396e-01,
```

```
X_train[['Applicant_Income','Coapplicant_Income','Loan_Amount','Loan_Amount_Term']]=
```

```
X_train
```

	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Coappl
1	0	1	2	1	0	-0.300202	

```
X_train = X_train.fillna(0)
```

438	0	1	0	0	1	NaN	

```
X_train
```

	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Coappl
1	0	1	2	1	0	-0.300202	
238	0	1	1	1	0	-0.166141	
438	0	1	0	0	1	0.000000	
210	0	1	2	1	0	0.638226	
388	0	1	2	1	0	-0.436532	
...	...	...	...	...	...	...	
328	0	0	0	0	1	-0.351764	
308	0	1	4	0	1	0.043614	
200	0	0	0	0	0	-0.002999	
611	0	1	2	1	1	0.000000	
351	0	1	0	1	0	-0.058273	

429 rows × 11 columns

```
X_test[['Applicant_Income','Coapplicant_Income','Loan_Amount','Loan_Amount_Term']]=p
```

```
X_test
```

	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Coappl
464	1	1	0	0	0	NaN	
52	0	1	4	0	0	1.085112	
136	0	1	2	0	0	9.055588	

```
X_test = X_test.fillna(0)
```

```
X_test
```

	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Coappl
464	1	1	0	0	0	0.000000	
52	0	1	4	0	0	1.085112	
136	0	1	2	0	0	9.055588	
424	0	1	4	1	0	0.000000	
443	0	1	2	1	0	0.000000	
...	...	...	...	...	...	...	
177	1	1	0	1	0	-0.540463	
570	0	1	1	1	0	0.000000	
240	1	0	0	1	0	0.000000	
315	0	1	2	1	0	0.000000	
280	0	1	0	0	0	0.000000	

185 rows × 11 columns

```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

```
((429, 11), (185, 11), (429,), (185,))
```

Get the model train

```
from sklearn.tree import DecisionTreeClassifier
```

```
dt = DecisionTreeClassifier()
```

```
dt.fit(X_train,y_train)
```

```
DecisionTreeClassifier()
```

Get model Prediction

```
y_pred = dt.predict(X_test)
```

```
y_pred.shape
```

```
(185,)
```

```
y_pred
```

```
array(['N', 'N', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'Y', 'Y', 'N', 'N', 'Y', 'N', 'Y', 'N', 'N', 'N',
       'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y',
       'Y', 'Y', 'N', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y',
       'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y',
       'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y',
       'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y',
       'N', 'N', 'Y', 'Y', 'N', 'N', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y',
       'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'N',
       'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'N',
       'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'N', 'N',
       'N', 'N', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'N', 'Y', 'N',
       'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y',
       'Y', 'Y', 'N'], dtype=object)
```

Get Probability of each predicted value

```
dt.predict_proba(X_test)
```

```
       [1.        , 0.        ],
       [0.        , 1.        ],
       [1.        , 0.        ],
       [0.35294118, 0.64705882],
       [0.        , 1.        ],
       [0.        , 1.        ],
       [0.        , 1.        ],
       [0.        , 1.        ],
       [1.        , 0.        ],
       [1.        , 0.        ],
       [1.        , 0.        ],
       [0.        , 1.        ],
       [0.        , 1.        ],
       [0.        , 1.        ],
       [0.        , 1.        ],
       [1.        , 0.        ],
       [1.        , 0.        ],
       [0.        , 1.        ],
       [0.35294118, 0.64705882],
       [0.        , 1.        ],
       [0.        , 1.        ],
```

```
       [0.        , 1.        ],
       [0.42857143, 0.57142857],
       [0.5       , 0.5       ],
       [0.5       , 0.5       ],
       [0.        , 1.        ],
       [0.        , 1.        ],
       [1.        , 0.        ],
       [1.        , 0.        ],
       [1.        , 0.        ],
       [1.        , 0.        ],
       [0.5       , 0.5       ],
       [0.33333333, 0.66666667],
       [0.        , 1.        ],
       [0.4       , 0.6       ],
       [1.        , 0.        ],
       [0.        , 1.        ],
       [1.        , 0.        ],
       [0.        , 1.        ],
       [1.        , 0.        ],
       [0.        , 1.        ],
       [1.        , 0.        ],
       [0.        , 1.        ],
       [0.        , 1.        ],
       [0.        , 1.        ],
       [1.        , 0.        ],
       [0.30769231, 0.69230769],
       [0.5       , 0.5       ],
       [0.        , 1.        ],
       [0.35294118, 0.64705882],
       [0.07692308, 0.92307692],
       [0.35294118, 0.64705882],
       [1.        , 0.        ],
       [0.        , 1.        ],
       [0.30769231, 0.69230769],
       [0.25      , 0.75      ],
       [0.07692308, 0.92307692],
       [1.        , 0.        ]])
```

Get model evaluation

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

```
confusion_matrix(y_pred,y_test)
```

```
array([[33, 28],
       [25, 99]])
```

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

```
              precision    recall  f1-score   support

           N       0.54      0.57      0.55        58
           Y       0.80      0.78      0.79       127
```

```
    accuracy                           0.71       185
   macro avg       0.67      0.67      0.67       185
weighted avg       0.72      0.71      0.72       185
```

Get Decission_tree_plot

```
from sklearn.tree import plot_tree
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(30,30))
plot_tree(dt,filled = True)
```

```
[Text(0.24828890489913544, 0.9736842105263158, 'X[9] <= 0.5\ngini = 0.43\nsamples = 4
 Text(0.08069164265129683, 0.9210526315789473, 'X[5] <= 2.199\ngini = 0.161\nsamples
 Text(0.05763688760806916, 0.868421052631579, 'X[6] <= -0.027\ngini = 0.116\nsamples
 Text(0.04610951008645533, 0.8157894736842105, 'gini = 0.0\nsamples = 21\nvalue = [2
 Text(0.069164265129683, 0.8157894736842105, 'X[6] <= 0.117\ngini = 0.165\nsamples =
 Text(0.05763688760806916, 0.7631578947368421, 'X[6] <= 0.106\ngini = 0.238\nsamples
 Text(0.04610951008645533, 0.7105263157894737, 'X[10] <= 0.5\ngini = 0.191\nsamples =
 Text(0.0345821325648415, 0.6578947368421053, 'gini = 0.0\nsamples = 9\nvalue = [9, 
 Text(0.05763688760806916, 0.6578947368421053, 'X[6] <= 0.04\ngini = 0.266\nsamples =
 Text(0.0345821325648415, 0.6052631578947368, 'X[2] <= 0.5\ngini = 0.208\nsamples = 
 Text(0.023054755043227664, 0.5526315789473685, 'gini = 0.0\nsamples = 7\nvalue = [7,
 Text(0.04610951008645533, 0.5526315789473685, 'X[3] <= 0.5\ngini = 0.32\nsamples = 
 Text(0.023054755043227664, 0.5, 'X[1] <= 0.5\ngini = 0.5\nsamples = 2\nvalue = [1, 
 Text(0.011527377521613832, 0.4473684210526316, 'gini = 0.0\nsamples = 1\nvalue = [1,
 Text(0.0345821325648415, 0.4473684210526316, 'gini = 0.0\nsamples = 1\nvalue = [0, 
 Text(0.069164265129683, 0.5, 'X[1] <= 0.5\ngini = 0.219\nsamples = 8\nvalue = [7, 1]
 Text(0.05763688760806916, 0.4473684210526316, 'gini = 0.0\nsamples = 1\nvalue = [0,
 Text(0.08069164265129683, 0.4473684210526316, 'gini = 0.0\nsamples = 7\nvalue = [7,
 Text(0.08069164265129683, 0.6052631578947368, 'X[6] <= 0.076\ngini = 0.5\nsamples =
 Text(0.069164265129683, 0.5526315789473685, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]
 Text(0.09221902017291066, 0.5526315789473685, 'gini = 0.0\nsamples = 1\nvalue = [1,
 Text(0.069164265129683, 0.7105263157894737, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]
 Text(0.08069164265129683, 0.7631578947368421, 'gini = 0.0\nsamples = 15\nvalue = [1
 Text(0.1037463976945245, 0.868421052631579, 'X[7] <= -0.88\ngini = 0.444\nsamples =
 Text(0.09221902017291066, 0.8157894736842105, 'gini = 0.0\nsamples = 1\nvalue = [1,
 Text(0.11527377521613832, 0.8157894736842105, 'gini = 0.0\nsamples = 2\nvalue = [0,
 Text(0.4158861671469741, 0.9210526315789473, 'X[5] <= -0.53\ngini = 0.319\nsamples =
 Text(0.14985590778097982, 0.868421052631579, 'X[3] <= 0.5\ngini = 0.085\nsamples = 
 Text(0.138328530259366, 0.8157894736842105, 'X[4] <= 0.5\ngini = 0.245\nsamples = 1
 Text(0.12680115273775217, 0.7631578947368421, 'X[5] <= -0.622\ngini = 0.142\nsample
 Text(0.11527377521613832, 0.7105263157894737, 'X[5] <= -0.664\ngini = 0.444\nsample
 Text(0.1037463976945245, 0.6578947368421053, 'gini = 0.0\nsamples = 2\nvalue = [0, 
 Text(0.12680115273775217, 0.6578947368421053, 'gini = 0.0\nsamples = 1\nvalue = [1,
 Text(0.138328530259366, 0.7105263157894737, 'gini = 0.0\nsamples = 10\nvalue = [0, 
 Text(0.14985590778097982, 0.7631578947368421, 'gini = 0.0\nsamples = 1\nvalue = [1,
 Text(0.16138328530259366, 0.8157894736842105, 'gini = 0.0\nsamples = 31\nvalue = [0,
 Text(0.6819164265129684, 0.868421052631579, 'X[5] <= 0.1\ngini = 0.345\nsamples = 3
 Text(0.45605187319884727, 0.8157894736842105, 'X[5] <= -0.097\ngini = 0.378\nsample
 Text(0.2478386167146974, 0.7631578947368421, 'X[5] <= -0.392\ngini = 0.295\nsamples
 Text(0.1786743515850144, 0.7105263157894737, 'X[6] <= 0.231\ngini = 0.418\nsamples =
 Text(0.14985590778097982, 0.6578947368421053, 'X[5] <= -0.405\ngini = 0.269\nsample
 Text(0.138328530259366, 0.6052631578947368, 'X[7] <= -0.735\ngini = 0.159\nsamples =
 Text(0.12680115273775217, 0.5526315789473685, 'X[0] <= 0.5\ngini = 0.444\nsamples =
 Text(0.11527377521613832, 0.5, 'gini = 0.0\nsamples = 4\nvalue = [0, 4]'),
 Text(0.138328530259366, 0.5, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(0.14985590778097982, 0.5526315789473685, 'gini = 0.0\nsamples = 17\nvalue = [0,
 Text(0.16138328530259366, 0.6052631578947368, 'gini = 0.0\nsamples = 2\nvalue = [2,
 Text(0.207492795389049, 0.6578947368421053, 'X[6] <= 0.636\ngini = 0.486\nsamples =
 Text(0.1844380403458213, 0.6052631578947368, 'X[4] <= 0.5\ngini = 0.278\nsamples = 
 Text(0.1729106628242075, 0.5526315789473685, 'gini = 0.0\nsamples = 5\nvalue = [5, 
 Text(0.19596541786743515, 0.5526315789473685, 'gini = 0.0\nsamples = 1\nvalue = [0,
 Text(0.23054755043227665, 0.6052631578947368, 'X[5] <= -0.524\ngini = 0.444\nsample
 Text(0.21902017291066284, 0.5526315789473685, 'gini = 0.0\nsamples = 2\nvalue = [2,
 Text(0.2420749279538905, 0.5526315789473685, 'gini = 0.0\nsamples = 4\nvalue = [0, 
 Text(0.3170028818443804, 0.7105263157894737, 'X[5] <= -0.288\ngini = 0.214\nsamples
 Text(0.2881844380403458, 0.6578947368421053, 'X[5] <= -0.355\ngini = 0.062\nsamples
 Text(0.276657060518732, 0.6052631578947368, 'X[5] <= -0.359\ngini = 0.153\nsamples =
 Text(0.26512968299711814, 0.5526315789473685, 'gini = 0.0\nsamples = 11\nvalue = [0,
 Text(0.2881844380403458, 0.5526315789473685, 'gini = 0.0\nsamples = 1\nvalue = [1, 
 Text(0.29971181556195964, 0.6052631578947368, 'gini = 0.0\nsamples = 19\nvalue = [0,
```