Sonal Shukla (CS23176)

Lab-4 Assignment-1 : To implement a Decision Tree regression model on the Developer Performance dataset and analyze how different tree depths and splitting criteria influence model performance.

```
import pandas as pd
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
```

```
from google .colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
path= "/content/drive/MyDrive/ML_DATASETS/Developer_Performance_dataset_1000 - Developer_Performance_dataset_1000.csv"
df=pd.read_csv(path)
df.head(5)
```

	Hours_Coding	Lines_of_Code	Bugs_Found	Bugs_Fixed	AI_Usage_Hours	Sleep_Hours	Cognitive_Load	Coffee_Intake	Stress_L
0	7	416	9	7	6	5.9	92	7	
1	4	269	16	13	5	5.1	85	2	
2	11	439	3	0	2	6.2	38	2	
3	8	472	15	9	4	4.2	26	5	
4	5	265	19	16	5	8.1	82	6	

Next steps: Generate code with df New interactive sheet

```
df.describe()
```

	Hours_Coding	Lines_of_Code	Bugs_Found	Bugs_Fixed	AI_Usage_Hours	Sleep_Hours	Cognitive_Load	Coffee_Intake	Str
count	1000.00000	1000.00000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	10
mean	5.84000	356.23400	9.876000	7.153000	2.961000	6.465800	56.933000	3.368000	
std	3.15854	188.15535	5.796052	5.468226	2.021278	1.439529	21.767506	2.342234	
min	1.00000	26.00000	0.000000	0.000000	0.000000	4.000000	20.000000	0.000000	
25%	3.00000	209.50000	5.000000	2.000000	1.000000	5.200000	38.000000	1.000000	
50%	6.00000	332.00000	10.000000	7.000000	3.000000	6.400000	57.000000	3.000000	
75%	9.00000	480.50000	15.000000	12.000000	5.000000	7.700000	76.000000	5.000000	
max	11.00000	993.00000	19.000000	19.000000	6.000000	9.000000	94.000000	7.000000	1

```
df.shape
```

```
(1000, 13)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 13 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Hours_Coding         1000 non-null   int64  
 1   Lines_of_Code        1000 non-null   int64  
 2   Bugs_Found           1000 non-null   int64  
 3   Bugs_Fixed           1000 non-null   int64  
 4   AI_Usage_Hours       1000 non-null   int64  
 5   Sleep_Hours          1000 non-null   float64
 6   Cognitive_Load       1000 non-null   int64  
 7   Coffee_Intake        1000 non-null   int64  
 8   Stress_Level         1000 non-null   int64  
 9   Task_Duration_Hours  1000 non-null   float64
 10  Commits              1000 non-null   int64  
 11  Errors               1000 non-null   int64  
 12  Task_Success_Rate    1000 non-null   int64  
dtypes: float64(2), int64(11)
memory usage: 101.7 KB
```

```
df.columns
```

```
Index(['Hours_Coding', 'Lines_of_Code', 'Bugs_Found', 'Bugs_Fixed',
       'AI_Usage_Hours', 'Sleep_Hours', 'Cognitive_Load', 'Coffee_Intake',
       'Stress_Level', 'Task_Duration_Hours', 'Commits', 'Errors',
       'Task_Success_Rate'],
      dtype='object')
```

```
df.isnull()
```

	Hours_Coding	Lines_of_Code	Bugs_Found	Bugs_Fixed	AI_Usage_Hours	Sleep_Hours	Cognitive_Load	Coffee_Intake	Stress_
0	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	
995	False	False	False	False	False	False	False	False	
996	False	False	False	False	False	False	False	False	
997	False	False	False	False	False	False	False	False	
998	False	False	False	False	False	False	False	False	
999	False	False	False	False	False	False	False	False	

1000 rows × 13 columns

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

	0
Hours_Coding	False
Lines_of_Code	False
Bugs_Found	False
Bugs_Fixed	False
AI_Usage_Hours	False
Sleep_Hours	False
Cognitive_Load	False
Coffee_Intake	False
Stress_Level	False
Task_Duration_Hours	False
Commits	False
Errors	False
Task_Success_Rate	False

dtype: bool

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

```
np.False_
```

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

	0
Hours_Coding	0
Lines_of_Code	0
Bugs_Found	0
Bugs_Fixed	0
AI_Usage_Hours	0
Sleep_Hours	0
Cognitive_Load	0
Coffee_Intake	0
Stress_Level	0
Task_Duration_Hours	0
Commits	0
Errors	0
Task_Success_Rate	0

dtype: int64

```
df[df.isnull().any(axis=1)]
```

Hours_Coding	Lines_of_Code	Bugs_Found	Bugs_Fixed	AI_Usage_Hours	Sleep_Hours	Cognitive_Load	Coffee_Intake	Stress_Lev

```
df.columns
```

```
Index(['Hours_Coding', 'Lines_of_Code', 'Bugs_Found', 'Bugs_Fixed',
       'AI_Usage_Hours', 'Sleep_Hours', 'Cognitive_Load', 'Coffee_Intake',
       'Stress_Level', 'Task_Duration_Hours', 'Commits', 'Errors',
       'Task_Success_Rate'],
      dtype='object')
```

```
df.shape
```

```
(1000, 13)
```

```
before_rows = df.shape[0]
df=df.dropna()
after_rows = df.shape[0]

print("Before rows: ",before_rows)
print("After Rows :",after_rows)
print("The number of  droped rows are :", before_rows-after_rows)
```

```
Before rows:  1000
After Rows : 1000
The number of  droped rows are : 0
```

```
df.shape
```

```
(1000, 13)
```

```
clean_data = df.copy()
```

```
clean_data.columns
```

```
Index(['Hours_Coding', 'Lines_of_Code', 'Bugs_Found', 'Bugs_Fixed',
       'AI_Usage_Hours', 'Sleep_Hours', 'Cognitive_Load', 'Coffee_Intake',
       'Stress_Level', 'Task_Duration_Hours', 'Commits', 'Errors',
       'Task_Success_Rate'],
      dtype='object')
```

```
clean_data.columns
```

```
Index(['Hours_Coding', 'Lines_of_Code', 'Bugs_Found', 'Bugs_Fixed',
       'AI_Usage_Hours', 'Sleep_Hours', 'Cognitive_Load', 'Coffee_Intake',
       'Stress_Level', 'Task_Duration_Hours', 'Commits', 'Errors',
       'Task_Success_Rate'],
      dtype='object')
```

```
clean_data.shape
```

```
(1000, 13)
```

```
X=clean_data.iloc[:,:-1]
print(X)
```

```
     Hours_Coding  Lines_of_Code  Bugs_Found  Bugs_Fixed  AI_Usage_Hours  \
0               7            416           9           7               6   
1               4            269          16          13               5   
2              11            439           3           0               2   
3               8            472          15           9               4   
4               5            265          19          16               5   
..            ...            ...         ...         ...             ...   
995            10            660          14          13               0   
996             9            484          13          11               1   
997             2            128           4           0               3   
998             8            266           5           3               1   
999             8            604          16          15               3   

     Sleep_Hours  Cognitive_Load  Coffee_Intake  Stress_Level  \
0            5.9              92              7            99   
1            5.1              85              2           100   
2            6.2              38              2            55   
3            4.2              26              5            30   
4            8.1              82              6            82   
..           ...             ...            ...           ...   
995          5.5              21              2            30   
996          8.8              34              0            39   
997          8.7              42              3            49   
998          5.7              48              0            63   
999          6.9              26              4            30   

     Task_Duration_Hours  Commits  Errors  
0                   10.5       20       3  
1                    9.5       17       8  
2                   18.3       35       2  
3                   12.6       28       4  
4                    7.0       25       9  
..                   ...      ...     ...  
995                  5.2       20       5  
996                  4.7       37       1  
997                  2.8        5       7  
998                 11.8        9       0  
999                  4.4       11       1  

[1000 rows x 12 columns]
```

```
y=clean_data.iloc[:,-1]
print(y)
```

```
0      34
1      36
2      79
3      94
4      33
       ..
995    80
996    78
997    74
998    51
999    90
Name: Task_Success_Rate, Length: 1000, dtype: int64
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,train_size=0.7,test_size=0.3,random_state=42)
```

```
X_train
```

	Hours_Coding	Lines_of_Code	Bugs_Found	Bugs_Fixed	AI_Usage_Hours	Sleep_Hours	Cognitive_Load	Coffee_Intake	Stress_
541	3	178	0	0	1	8.1	28	0	
440	6	226	5	2	2	5.2	21	2	
482	5	388	4	1	6	5.4	68	1	
422	1	176	8	3	3	6.4	84	3	
778	5	340	17	15	0	6.3	24	4	
...	...	...	...	...	...	...	...	...	
106	8	236	17	15	3	6.0	61	4	
270	8	228	5	3	2	8.8	23	5	
860	1	102	19	14	2	5.9	51	3	
435	7	189	17	12	0	8.1	26	0	
102	10	282	4	0	2	8.4	76	0	

700 rows × 12 columns

Next steps: Generate code with `X_train` New interactive sheet

```
X_test
```

	Hours_Coding	Lines_of_Code	Bugs_Found	Bugs_Fixed	AI_Usage_Hours	Sleep_Hours	Cognitive_Load	Coffee_Intake	Stress_
521	3	234	5	3	2	8.4	63	5	
737	2	254	5	1	6	6.6	86	4	
740	5	266	13	9	4	6.4	84	3	
660	5	140	3	0	0	8.1	60	3	
411	8	810	2	0	6	8.0	43	5	
...	...	...	...	...	...	...	...	...	
468	4	168	8	6	3	4.4	60	5	
935	4	210	2	0	2	7.9	34	0	
428	9	646	8	2	2	8.3	94	7	
7	3	234	5	1	6	5.8	78	2	
155	7	224	19	15	1	8.0	54	2	

300 rows × 12 columns

Next steps: Generate code with `X_test` New interactive sheet

```
y_train
```

	Task_Success_Rate
541	85
440	80
482	50
422	30
778	74
...	...
106	43
270	79
860	52
435	84
102	30

700 rows × 1 columns

dtype: int64

```
y_test
```

	Task_Success_Rate
521	39
737	36
740	36
660	49
411	82
...	...
468	49
935	77
428	30
7	40
155	49

300 rows × 1 columns

dtype: int64

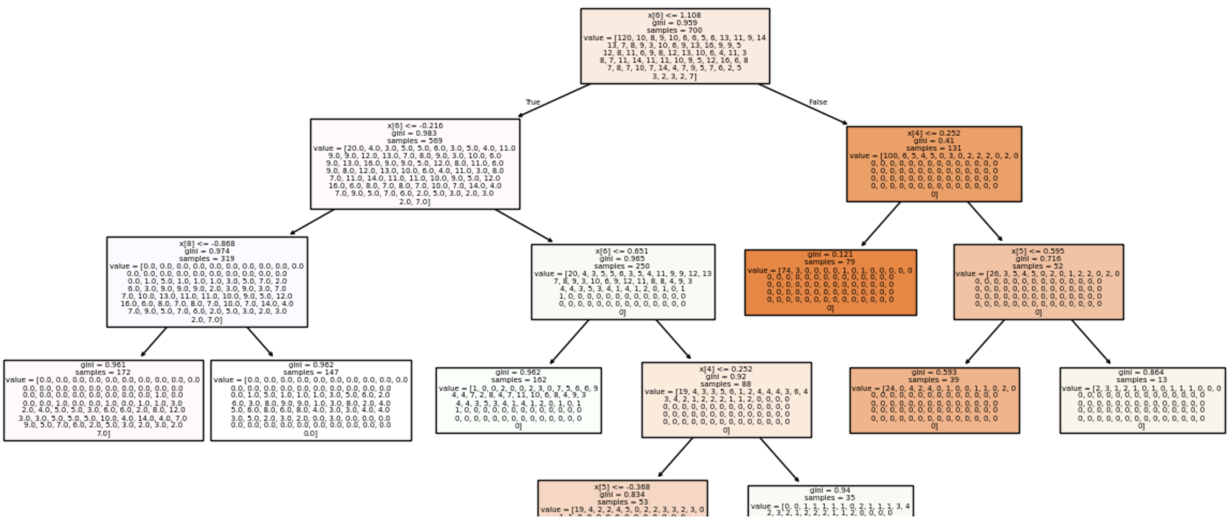
```
from sklearn.preprocessing import StandardScaler
scaling = StandardScaler()
X_train = scaling.fit_transform(X_train)
X_test = scaling.transform(X_test)
```

```
humidity_classifier = DecisionTreeClassifier(max_leaf_nodes= 10,random_state=0)
humidity_classifier.fit(X_train,y_train)
```

```
▾              DecisionTreeClassifier              i ?
DecisionTreeClassifier(max_leaf_nodes=10, random_state=0)
```

```
import matplotlib.pyplot as plt
from sklearn import tree
plt.figure(figsize=(15,10))
tree.plot_tree(humidity_classifier,filled=True)
```

```
[Text(0.5416666666666666, 0.9285714285714286, 'x[6] <= 1.108\ngini = 0.959\nsamples = 700\nvalue = [120, 10, 8, 9, 10, 6,
6, 5, 6, 13, 11, 9, 14\n13, 7, 8, 9, 3, 10, 6, 9, 13, 16, 9, 9, 5\n12, 8, 11, 6, 9, 8, 12, 13, 10, 6, 4, 11, 3\n8, 7, 11,
14, 11, 11, 10, 9, 5, 12, 16, 6, 8\n7, 8, 7, 10, 7, 14, 4, 7, 9, 5, 7, 6, 2, 5\n3, 2, 3, 2, 7]'),
 Text(0.3333333333333333, 0.7857142857142857, 'x[6] <= -0.216\ngini = 0.983\nsamples = 569\nvalue = [20.0, 4.0, 3.0, 5.0,
5.0, 6.0, 3.0, 5.0, 4.0, 11.0\n9.0, 9.0, 12.0, 13.0, 7.0, 8.0, 9.0, 3.0, 10.0, 6.0\n9.0, 13.0, 16.0, 9.0, 9.0, 5.0, 12.0,
8.0, 11.0, 6.0\n9.0, 8.0, 12.0, 13.0, 10.0, 6.0, 4.0, 11.0, 3.0, 8.0\n7.0, 11.0, 14.0, 11.0, 11.0, 10.0, 9.0, 5.0,
12.0\n16.0, 6.0, 8.0, 7.0, 8.0, 7.0, 10.0, 7.0, 14.0, 4.0\n7.0, 9.0, 5.0, 7.0, 6.0, 2.0, 5.0, 3.0, 2.0, 3.0\n2.0, 7.0]'),
 Text(0.4375, 0.8571428571428572, 'True  '),
 Text(0.16666666666666666, 0.6428571428571429, 'x[8] <= -0.868\ngini = 0.974\nsamples = 319\nvalue = [0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0\n0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0\n0.0, 1.0, 5.0, 1.0, 1.0, 1.0, 3.0, 5.0,
7.0, 2.0\n6.0, 3.0, 9.0, 9.0, 9.0, 2.0, 3.0, 9.0, 3.0, 7.0\n7.0, 10.0, 13.0, 11.0, 11.0, 10.0, 9.0, 5.0, 12.0\n16.0, 6.0,
8.0, 7.0, 8.0, 7.0, 10.0, 7.0, 14.0, 4.0\n7.0, 9.0, 5.0, 7.0, 6.0, 2.0, 5.0, 3.0, 2.0, 3.0\n2.0, 7.0]'),
 Text(0.08333333333333333, 0.5, 'gini = 0.961\nsamples = 172\nvalue = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0\n0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0\n0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0\n0.0, 0.0, 1.0,
0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 3.0\n2.0, 4.0, 5.0, 5.0, 3.0, 6.0, 6.0, 2.0, 8.0, 12.0\n3.0, 3.0, 5.0, 5.0, 5.0, 10.0, 4.0,
14.0, 4.0, 7.0\n9.0, 5.0, 7.0, 6.0, 2.0, 5.0, 3.0, 2.0, 3.0, 2.0\n7.0]'),
 Text(0.25, 0.5, 'gini = 0.962\nsamples = 147\nvalue = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0\n0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0\n0.0, 1.0, 5.0, 1.0, 1.0, 1.0, 3.0, 5.0, 6.0, 2.0\n6.0, 3.0, 8.0, 9.0, 9.0, 1.0, 3.0,
8.0, 2.0, 4.0\n5.0, 6.0, 8.0, 6.0, 8.0, 4.0, 3.0, 3.0, 4.0, 4.0\n3.0, 5.0, 2.0, 3.0, 2.0, 0.0, 3.0, 0.0, 0.0, 0.0\n0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0\n0.0]'),
 Text(0.5, 0.6428571428571429, 'x[6] <= 0.651\ngini = 0.965\nsamples = 250\nvalue = [20, 4, 3, 5, 5, 6, 3, 5, 4, 11, 9, 9,
12, 13\n7, 8, 9, 3, 10, 6, 9, 12, 11, 8, 8, 4, 9, 3\n4, 4, 3, 5, 3, 4, 1, 4, 1, 2, 0, 1, 0, 1\n1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0]'),
 Text(0.4166666666666667, 0.5, 'gini = 0.962\nsamples = 162\nvalue = [1, 0, 0, 2, 0, 0, 2, 3, 0, 7, 5, 6, 6, 9\n4, 4, 7, 2,
8, 4, 7, 11, 10, 6, 8, 4, 9, 3\n4, 4, 3, 5, 3, 4, 1, 4, 1, 2, 0, 1, 0, 1\n1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0]'),
 Text(0.5833333333333334, 0.5, 'x[4] <= 0.252\ngini = 0.92\nsamples = 88\nvalue = [19, 4, 3, 3, 5, 6, 1, 2, 4, 4, 4, 3, 6,
4\n3, 4, 2, 1, 2, 2, 2, 1, 1, 2, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0]'),
 Text(0.5, 0.35714285714285715, 'x[5] <= -0.368\ngini = 0.834\nsamples = 53\nvalue = [19, 4, 2, 2, 4, 5, 0, 2, 2, 3, 3, 2,
3, 0\n1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0]'),
 Text(0.4166666666666667, 0.21428571428571427, 'x[4] <= -0.745\ngini = 0.653\nsamples = 21\nvalue = [11, 0, 0, 0, 1, 5, 0,
1, 0, 0, 0, 0, 2, 0\n1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0]'),
 Text(0.3333333333333333, 0.07142857142857142, 'gini = 0.298\nsamples = 11\nvalue = [9, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0]'),
 Text(0.5, 0.07142857142857142, 'gini = 0.8\nsamples = 10\nvalue = [2, 0, 0, 0, 1, 3, 0, 1, 0, 0, 0, 0, 2, 0\n1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0]'),
 Text(0.5833333333333334, 0.21428571428571427, 'gini = 0.877\nsamples = 32\nvalue = [8, 4, 2, 2, 3, 0, 0, 1, 2, 3, 3, 2, 1,
0\n0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0]'),
 Text(0.6666666666666666, 0.35714285714285715, 'gini = 0.94\nsamples = 35\nvalue = [0, 0, 1, 1, 1, 1, 1, 0, 2, 1, 1, 1, 3,
4\n2, 3, 2, 1, 2, 2, 2, 1, 1, 2, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0]'),
 Text(0.75, 0.7857142857142857, 'x[4] <= 0.252\ngini = 0.41\nsamples = 131\nvalue = [100, 6, 5, 4, 5, 0, 3, 0, 2, 2, 2, 0,
2, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0]'),
 Text(0.6458333333333333, 0.8571428571428572, '  False'),
 Text(0.6666666666666666, 0.6428571428571429, 'gini = 0.121\nsamples = 79\nvalue = [74, 3, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0]'),
 Text(0.8333333333333334, 0.6428571428571429, 'x[5] <= 0.595\ngini = 0.716\nsamples = 52\nvalue = [26, 3, 5, 4, 5, 0, 2, 0,
1, 2, 2, 0, 2, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0]'),
 Text(0.75, 0.5, 'gini = 0.593\nsamples = 39\nvalue = [24, 0, 4, 2, 4, 0, 1, 0, 0, 1, 1, 0, 2, 0\n0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0\n0]'),
 Text(0.9166666666666666, 0.5, 'gini = 0.864\nsamples = 13\nvalue = [2, 3, 1, 2, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0\n0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0]')]
```



```
y_pred = humidity_classifier.predict(X_test)
print(y_pred)
```

```
[51 30 30 51 63 30 51 63 51 30 51 51 63 87 51 30 51 87 87 87 63 43 51 51
 87 51 87 43 63 30 30 87 51 87 87 63 87 87 51 51 30 87 87 51 87 63 30 51
 51 63 51 51 87 51 51 63 63 63 30 51 30 63 63 87 30 87 30 51 30 51 87 87
 30 30 30 87 51 30 35 51 51 87 43 51 87 43 63 63 51 63 87 51 30 30 63 51
 30 30 51 51 87 63 63 43 51 63 87 87 30 30 35 51 63 51 87 30 30 63 51 51
 30 30 51 43 30 87 30 51 63 30 43 87 87 87 63 30 51 87 87 51 87 63 87 30
 30 63 30 31 87 51 63 30 51 51 87 51 51 87 35 63 51 51 63 51 31 63 30 35
 63 87 51 30 30 51 43 63 51 30 87 51 87 63 51 30 63 87 51 63 63 63 30 51
 51 30 63 51 30 51 30 87 30 87 51 63 63 30 87 63 63 30 63 87 63 30 51 51
 31 30 63 87 87 63 63 63 30 63 30 87 43 51 87 30 30 30 51 30 30 30 30 31
 87 30 87 63 51 31 51 30 87 51 51 30 30 51 63 30 43 63 87 51 63 43 51 30
 63 30 51 30 87 63 87 51 51 30 63 30 30 30 51 51 43 63 63 63 31 43 63 30
 87 30 63 87 30 51 63 51 87 30 43 51]
```

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

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

```
[[51  2  1 ...  0  0  0]
 [ 2  0  1 ...  0  0  0]
 [ 0  0  0 ...  0  0  0]
 ...
 [ 0  0  0 ...  0  0  0]
 [ 0  0  0 ...  0  0  0]
 [ 0  0  0 ...  0  0  0]]
```

```
accuracy = accuracy_score(y_pred,y_test)
print(accuracy)
```

```
0.18666666666666668
```

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

```
              precision    recall  f1-score   support

          30       0.89      0.68      0.77        75
          31       0.00      0.00      0.00         6
          32       0.00      0.00      0.00         0
          33       0.00      0.00      0.00         0
```