Name:P.Asha Belcilda

Rollno:225229104

Lab-8. Animal Classifications using Decision Trees

Step1.[Create dataset] ¶

In [1]:
```
import pandas as pd
```

In [2]:
```
df=pd.read_csv('data.csv')
```

In [3]:
```
df
```

Out[3]:

	Toothed	hair	breathes	legs	species
0	True	True	True	True	Mammal
1	True	True	True	True	Mammal
2	True	False	True	False	Reptile
3	False	True	True	True	Mammal
4	True	True	True	True	Mammal
5	True	True	True	True	Mammal
6	True	False	False	False	Reptile
7	True	False	True	False	Reptile
8	True	True	True	True	Mammal
9	False	False	True	True	Reptile

In [4]:
```
#head
df.head()
```

Out[4]:

	Toothed	hair	breathes	legs	species
0	True	True	True	True	Mammal
1	True	True	True	True	Mammal
2	True	False	True	False	Reptile
3	False	True	True	True	Mammal
4	True	True	True	True	Mammal

In [5]:
```
#shape
df.shape
```

Out[5]: (10, 5)

In [6]:
```
#size
df.size
```

Out[6]: 50

In [7]:
```
#desceibe
df.describe()
```

Out[7]:

	Toothed	hair	breathes	legs	species
count	10	10	10	10	10
unique	2	2	2	2	2
top	True	True	True	True	Mammal
freq	8	6	9	7	6

Step2.[Model building using ID3]

In [8]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [9]:
```
#create DT model using 'entropy' criterion
X = df.drop(['species'],axis = 1)
y = df['species']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_
```

In [10]:
```
from sklearn.tree import DecisionTreeClassifier
clf_entropy = DecisionTreeClassifier(criterion = "entropy")
```

In [11]:
```
#perform training and testing
clf_entropy =clf_entropy.fit(X_train,y_train)
y_pred_entropy = clf_entropy.predict(X_test)
y_pred_entropy
```

Out[11]: array(['Reptile', 'Mammal', 'Mammal', 'Mammal'], dtype=object)

In [12]:
```
#accuracy
from sklearn.metrics import accuracy_score
print ("Accuracy for ID3: ",accuracy_score(y_test,y_pred_entropy))
```

Accuracy for ID3: 0.75

In [13]:
```
#classification
from sklearn.metrics import classification_report
print("Classification Report of ID3 : ",classification_report(y_test, y_pred_entr
```

```
Classification Report of ID3 :                precision    recall  f1-score   s
upport

      Mammal       0.67      1.00      0.80         2
     Reptile       1.00      0.50      0.67         2

    accuracy                           0.75         4
   macro avg       0.83      0.75      0.73         4
weighted avg       0.83      0.75      0.73         4
```

In [14]:
```
#interpreting results
from sklearn import tree
```

In [15]:
```
#Visualilze your DT model using graphviz
with open('tree1.dot','w') as f:
 f = tree.export_graphviz(clf_entropy,
 out_file=f,
max_depth=4,
impurity=False,
feature_names = X.columns.values,
 class_names = ['Reptile','Mammal'],
 filled=True)
```

In [16]:
```
!type tree1.dot
```

```
digraph Tree {
node [shape=box, style="filled", color="black"] ;
0 [label="legs <= 0.5\nsamples = 6\nvalue = [4, 2]\nclass = Reptile", fillcolor
="#f2c09c"] ;
1 [label="samples = 2\nvalue = [0, 2]\nclass = Mammal", fillcolor="#399de5"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="samples = 4\nvalue = [4, 0]\nclass = Reptile", fillcolor="#e58139"] ;
0 -> 2 [labeldistance=2.5, labelangle=-45, headlabel="False"] ;
}
```

In [17]:
```
tree.plot_tree(clf_entropy)
```

Out[17]:
```
[Text(251.5,282.45,'X[3] <= 0.5\nentropy = 0.918\nsamples = 6\nvalue = [4,
2]'),
 Text(125.75,94.15,'entropy = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(377.25,94.15,'entropy = 0.0\nsamples = 4\nvalue = [4, 0]')]
```

Step3. [Create a Test Set]

In [19]:
```
test_file = pd.read_csv('test_file.csv')
```

In [20]:
```
test_file
```

Out[20]:

	Toothed	hair	breathes	legs
0	False	False	True	False
1	False	True	True	True
2	True	False	True	True

Step4. [Perform prediction]

In [21]:
```
y_pred_entropy=clf_entropy.predict(test_file)
y_pred_entropy
```

Out[21]:
```
array(['Reptile', 'Mammal', 'Mammal'], dtype=object)
```

Step5. [Build DT with zoo dataset]

In [22]:
```
clf_gini = DecisionTreeClassifier(criterion = "gini")
```

In [23]:
```
#Train model with full training data
clf_gini.fit(X,y)
```

Out[23]:
```
DecisionTreeClassifier()
```

In [24]:
```
#Predict Samples for the test file
y_pred_gini = clf_gini.predict(test_file)
y_pred_gini
```

Out[24]:
```
array(['Reptile', 'Mammal', 'Reptile'], dtype=object)
```

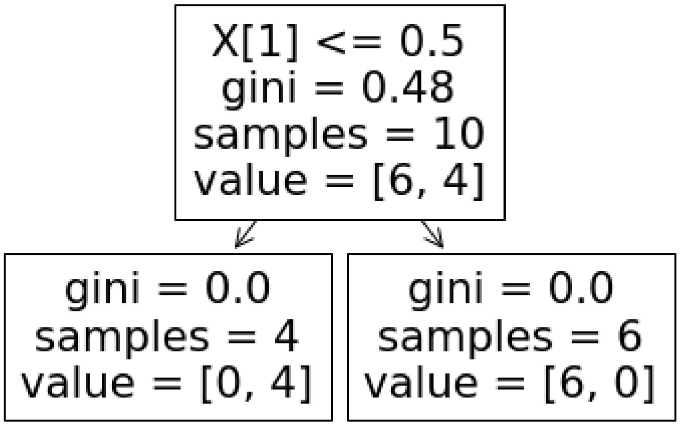
In [25]:
```
#Visualize your CART DT using graphviz
with open("tree2.dot",'w') as f:
 f= tree.export_graphviz(clf_gini, out_file=f, max_depth=4, impurity= False,
 feature_names = X.columns.values, class_names = ['Reptile','Mammal'],filled=True
```

In [26]:
```
!type tree2.dot
```

```
digraph Tree {
node [shape=box, style="filled", color="black"] ;
0 [label="hair <= 0.5\nsamples = 10\nvalue = [6, 4]\nclass = Reptile", fillcolo
r="#f6d5bd"] ;
1 [label="samples = 4\nvalue = [0, 4]\nclass = Mammal", fillcolor="#399de5"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="samples = 6\nvalue = [6, 0]\nclass = Reptile", fillcolor="#e58139"] ;
0 -> 2 [labeldistance=2.5, labelangle=-45, headlabel="False"] ;
}
```

In [27]:
```
tree.plot_tree(clf_gini)
```

Out[27]:
```
[Text(170.9,168.33,'X[1] <= 0.5\ngini = 0.48\nsamples = 10\nvalue = [6, 4]'),
 Text(85.45,56.11,'gini = 0.0\nsamples = 4\nvalue = [0, 4]'),
 Text(256.35,56.11,'gini = 0.0\nsamples = 6\nvalue = [6, 0]')]
```



Step-6. [Build DT with Zoo dataset]

In [28]:
```
zoo_df = pd.read_csv('zoo.data')
```

In [29]:
```
zoo_df
```

Out[29]:

	aardvark	1	0	0.1	1.1	0.2	0.3	1.2	1.3	1.4	1.5	0.4	0.5	4	0.6	0.7	1.6	1.7
0	antelope	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
1	bass	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	4
2	bear	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	1
3	boar	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
4	buffalo	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
5	calf	1	0	0	1	0	0	0	1	1	1	0	0	4	1	1	1	1
6	carp	0	0	1	0	0	1	0	1	1	0	0	1	0	1	1	0	4
7	catfish	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	4
8	cavy	1	0	0	1	0	0	0	1	1	1	0	0	4	0	1	0	1
9	cheetah	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
10	chicken	0	1	1	0	1	0	0	0	1	1	0	0	2	1	1	0	2
11	chub	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	4
12	clam	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	7
13	crab	0	0	1	0	0	1	1	0	0	0	0	0	4	0	0	0	7
14	crayfish	0	0	1	0	0	1	1	0	0	0	0	0	6	0	0	0	7
15	crow	0	1	1	0	1	0	1	0	1	1	0	0	2	1	0	0	2
16	deer	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
17	dogfish	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	1	4
18	dolphin	0	0	0	1	0	1	1	1	1	1	0	1	0	1	0	1	1
19	dove	0	1	1	0	1	0	0	0	1	1	0	0	2	1	1	0	2
20	duck	0	1	1	0	1	1	0	0	1	1	0	0	2	1	0	0	2
21	elephant	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
22	flamingo	0	1	1	0	1	0	0	0	1	1	0	0	2	1	0	1	2
23	flea	0	0	1	0	0	0	0	0	0	1	0	0	6	0	0	0	6
24	frog	0	0	1	0	0	1	1	1	1	1	0	0	4	0	0	0	5
25	frog	0	0	1	0	0	1	1	1	1	1	1	0	4	0	0	0	5
26	fruitbat	1	0	0	1	1	0	0	1	1	1	0	0	2	1	0	0	1
27	giraffe	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
28	girl	1	0	0	1	0	0	1	1	1	1	0	0	2	0	1	1	1
29	gnat	0	0	1	0	1	0	0	0	0	1	0	0	6	0	0	0	6
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
70	rhea	0	1	1	0	0	0	1	0	1	1	0	0	2	1	0	1	2
71	scorpion	0	0	0	0	0	0	1	0	0	1	1	0	8	1	0	0	7
72	seahorse	0	0	1	0	0	1	0	1	1	0	0	1	0	1	0	0	4
73	seal	1	0	0	1	0	1	1	1	1	1	0	1	0	0	0	1	1

	aardvark	1	0	0.1	1.1	0.2	0.3	1.2	1.3	1.4	1.5	0.4	0.5	4	0.6	0.7	1.6	1.7
74	sealion	1	0	0	1	0	1	1	1	1	1	0	1	2	1	0	1	1
75	seasnake	0	0	0	0	0	1	1	1	1	0	1	0	0	1	0	0	3
76	seawasp	0	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0	7
77	skimmer	0	1	1	0	1	1	1	0	1	1	0	0	2	1	0	0	2
78	skua	0	1	1	0	1	1	1	0	1	1	0	0	2	1	0	0	2
79	slowworm	0	0	1	0	0	0	1	1	1	1	0	0	0	1	0	0	3
80	slug	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	7
81	sole	0	0	1	0	0	1	0	1	1	0	0	1	0	1	0	0	4
82	sparrow	0	1	1	0	1	0	0	0	1	1	0	0	2	1	0	0	2
83	squirrel	1	0	0	1	0	0	0	1	1	1	0	0	2	1	0	0	1
84	starfish	0	0	1	0	0	1	1	0	0	0	0	0	5	0	0	0	7
85	stingray	0	0	1	0	0	1	1	1	1	0	1	1	0	1	0	1	4
86	swan	0	1	1	0	1	1	0	0	1	1	0	0	2	1	0	1	2
87	termite	0	0	1	0	0	0	0	0	0	1	0	0	6	0	0	0	6
88	toad	0	0	1	0	0	1	0	1	1	1	0	0	4	0	0	0	5
89	tortoise	0	0	1	0	0	0	0	0	1	1	0	0	4	1	0	1	3
90	tuatara	0	0	1	0	0	0	1	1	1	1	0	0	4	1	0	0	3
91	tuna	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	1	4
92	vampire	1	0	0	1	1	0	0	1	1	1	0	0	2	1	0	0	1
93	vole	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	0	1
94	vulture	0	1	1	0	1	0	1	0	1	1	0	0	2	1	0	1	2
95	wallaby	1	0	0	1	0	0	0	1	1	1	0	0	2	1	0	1	1
96	wasp	1	0	1	0	1	0	0	0	0	1	1	0	6	0	0	0	6
97	wolf	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
98	worm	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	7
99	wren	0	1	1	0	1	0	0	0	1	1	0	0	2	1	0	0	2

100 rows × 18 columns

In [30]:
```
#Split Zoo data into train and test sets
X_zoo = zoo_df.drop(['aardvark', '1.7'], axis = 1)
y_zoo = zoo_df['1.7']
```

In [31]:
```
X_train_zoo,X_test_zoo,y_train_zoo,y_test_zoo = train_test_split(X_zoo, y_zoo, te
```

In [32]:
```
#Create DT model using 'entropy' criterion
clf_entropy_zoo = DecisionTreeClassifier( criterion = "entropy")
clf_entropy_zoo.fit(X_train_zoo, y_train_zoo)
```

Out[32]:
```
DecisionTreeClassifier(criterion='entropy')
```

In [33]:
```
y_pred_entropy_zoo = clf_entropy_zoo.predict(X_test_zoo)
```

In [34]:
```
y_pred_entropy_zoo
```

Out[34]:
```
array([1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 7, 4, 1, 2, 5, 4, 1, 1, 5,
       1, 1, 7, 1, 4, 2, 2, 7, 4, 7, 3], dtype=int64)
```

In [35]:
```
#predict using train zoo
y_pred_train_zoo = clf_entropy_zoo.predict(X_train_zoo)
y_pred_train_zoo
```

Out[35]:
```
array([1, 5, 1, 1, 2, 1, 1, 4, 3, 2, 6, 1, 2, 4, 2, 6, 1, 4, 4, 1, 1, 1,
       6, 4, 1, 6, 7, 2, 1, 1, 2, 3, 4, 2, 7, 7, 3, 2, 6, 1, 1, 7, 1, 2,
       2, 4, 2, 5, 4, 4, 1, 6, 1, 2, 7, 5, 2, 6, 2, 1, 1, 1, 6, 1, 1, 1,
       1], dtype=int64)
```

Accuracy

In [36]:
```
print("Train Accuracy of Zoo Data :", accuracy_score(y_train_zoo, y_pred_train_zo
```

```
Train Accuracy of Zoo Data : 1.0
```

In [37]:
```
print("Test Accuracy of Zoo Data :", accuracy_score(y_test_zoo, y_pred_entropy_zo
```

```
Test Accuracy of Zoo Data : 0.9090909090909091
```

In [38]:
```
#Report for zoo data using entropy criterion
print("Report for zoo data using entropy criterion :\n", classification_report(y_
```

```
Report for zoo data using entropy criterion :
               precision    recall  f1-score   support

           1       0.88      0.93      0.90        15
           2       1.00      1.00      1.00         6
           3       1.00      0.50      0.67         2
           4       1.00      1.00      1.00         4
           5       0.50      1.00      0.67         1
           7       1.00      0.80      0.89         5

    accuracy                           0.91        33
   macro avg       0.90      0.87      0.85        33
weighted avg       0.93      0.91      0.91        33
```

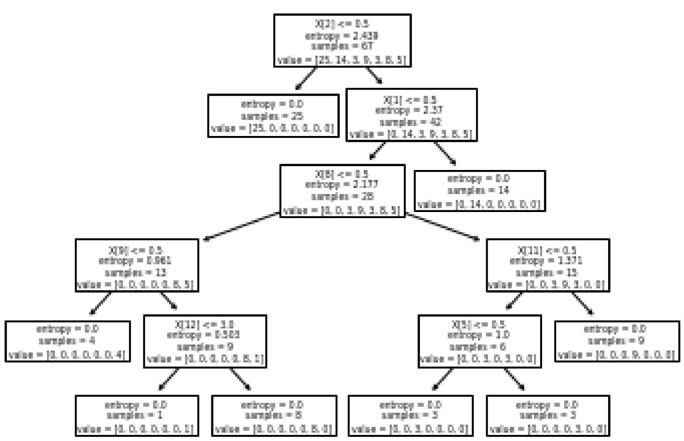
In [39]:
```
#Visualize ID3 DT using graphviz for zoo data
with open("tree1_zoo.dot",'w') as f:
 f= tree.export_graphviz(clf_entropy, out_file=f, max_depth=4, impurity= False,
 feature_names = X.columns.values,class_names = ['Reptile','Mammal'],filled=True)
```

In [40]:
```
!type tree1_zoo.dot
```

```
digraph Tree {
node [shape=box, style="filled", color="black"] ;
0 [label="legs <= 0.5\nsamples = 6\nvalue = [4, 2]\nclass = Reptile", fillcolor
="#f2c09c"] ;
1 [label="samples = 2\nvalue = [0, 2]\nclass = Mammal", fillcolor="#399de5"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="samples = 4\nvalue = [4, 0]\nclass = Reptile", fillcolor="#e58139"] ;
0 -> 2 [labeldistance=2.5, labelangle=-45, headlabel="False"] ;
}
```

In [41]:
```
tree.plot_tree(clf_entropy_zoo)
```

Out[41]:
```
[Text(170.9,205.737,'X[2] <= 0.5\nentropy = 2.439\nsamples = 67\nvalue = [25, 1
4, 3, 9, 3, 8, 5]'),
 Text(136.72,168.33,'entropy = 0.0\nsamples = 25\nvalue = [25, 0, 0, 0, 0, 0,
0]'),
 Text(205.08,168.33,'X[1] <= 0.5\nentropy = 2.37\nsamples = 42\nvalue = [0, 14,
3, 9, 3, 8, 5]'),
 Text(170.9,130.923,'X[8] <= 0.5\nentropy = 2.177\nsamples = 28\nvalue = [0, 0,
3, 9, 3, 8, 5]'),
 Text(68.36,93.5167,'X[9] <= 0.5\nentropy = 0.961\nsamples = 13\nvalue = [0, 0,
0, 0, 0, 8, 5]'),
 Text(34.18,56.11,'entropy = 0.0\nsamples = 4\nvalue = [0, 0, 0, 0, 0, 0, 4]'),
 Text(102.54,56.11,'X[12] <= 3.0\nentropy = 0.503\nsamples = 9\nvalue = [0, 0,
0, 0, 0, 8, 1]'),
 Text(68.36,18.7033,'entropy = 0.0\nsamples = 1\nvalue = [0, 0, 0, 0, 0, 0,
1]'),
 Text(136.72,18.7033,'entropy = 0.0\nsamples = 8\nvalue = [0, 0, 0, 0, 0, 8,
0]'),
 Text(273.44,93.5167,'X[11] <= 0.5\nentropy = 1.371\nsamples = 15\nvalue = [0,
0, 3, 9, 3, 0, 0]'),
 Text(239.26,56.11,'X[5] <= 0.5\nentropy = 1.0\nsamples = 6\nvalue = [0, 0, 3,
0, 3, 0, 0]'),
 Text(205.08,18.7033,'entropy = 0.0\nsamples = 3\nvalue = [0, 0, 3, 0, 0, 0,
0]'),
 Text(273.44,18.7033,'entropy = 0.0\nsamples = 3\nvalue = [0, 0, 0, 0, 3, 0,
0]'),
 Text(307.62,56.11,'entropy = 0.0\nsamples = 9\nvalue = [0, 0, 0, 9, 0, 0,
0]'),
 Text(239.26,130.923,'entropy = 0.0\nsamples = 14\nvalue = [0, 14, 0, 0, 0, 0,
0]')]
```



[Create DT model using 'gini' criterion]

In [42]:
```
clf_gini_zoo = DecisionTreeClassifier( criterion = "gini")
clf_gini_zoo.fit(X_train_zoo, y_train_zoo)
y_pred_gini_zoo = clf_gini_zoo.predict(X_test_zoo)
```

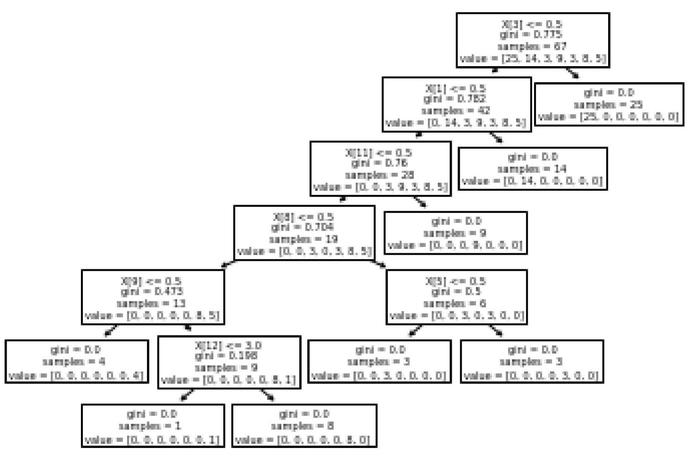
In [43]:
```
with open("tree2_zoo.dot",'w') as f:
 f= tree.export_graphviz(clf_gini, out_file=f, max_depth=4, impurity= False,
 feature_names = X.columns.values,class_names = ['Reptile','Mammal'],filled=True)
```

In [44]:
```
!type tree2_zoo.dot
```

```
digraph Tree {
node [shape=box, style="filled", color="black"] ;
0 [label="hair <= 0.5\nsamples = 10\nvalue = [6, 4]\nclass = Reptile", fillcolo
r="#f6d5bd"] ;
1 [label="samples = 4\nvalue = [0, 4]\nclass = Mammal", fillcolor="#399de5"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="samples = 6\nvalue = [6, 0]\nclass = Reptile", fillcolor="#e58139"] ;
0 -> 2 [labeldistance=2.5, labelangle=-45, headlabel="False"] ;
}
```

In [45]:
```
tree.plot_tree(clf_gini_zoo)
```

Out[45]:
```
[Text(265.844,208.409,'X[3] <= 0.5\ngini = 0.775\nsamples = 67\nvalue = [25, 1
4, 3, 9, 3, 8, 5]'),
 Text(227.867,176.346,'X[1] <= 0.5\ngini = 0.782\nsamples = 42\nvalue = [0, 14,
3, 9, 3, 8, 5]'),
 Text(189.889,144.283,'X[11] <= 0.5\ngini = 0.76\nsamples = 28\nvalue = [0, 0,
3, 9, 3, 8, 5]'),
 Text(151.911,112.22,'X[8] <= 0.5\ngini = 0.704\nsamples = 19\nvalue = [0, 0,
3, 0, 3, 8, 5]'),
 Text(75.9556,80.1571,'X[9] <= 0.5\ngini = 0.473\nsamples = 13\nvalue = [0, 0,
0, 0, 0, 8, 5]'),
 Text(37.9778,48.0943,'gini = 0.0\nsamples = 4\nvalue = [0, 0, 0, 0, 0, 0,
4]'),
 Text(113.933,48.0943,'X[12] <= 3.0\ngini = 0.198\nsamples = 9\nvalue = [0, 0,
0, 0, 0, 8, 1]'),
 Text(75.9556,16.0314,'gini = 0.0\nsamples = 1\nvalue = [0, 0, 0, 0, 0, 0,
1]'),
 Text(151.911,16.0314,'gini = 0.0\nsamples = 8\nvalue = [0, 0, 0, 0, 0, 8,
0]'),
 Text(227.867,80.1571,'X[5] <= 0.5\ngini = 0.5\nsamples = 6\nvalue = [0, 0, 3,
0, 3, 0, 0]'),
 Text(189.889,48.0943,'gini = 0.0\nsamples = 3\nvalue = [0, 0, 3, 0, 0, 0,
0]'),
 Text(265.844,48.0943,'gini = 0.0\nsamples = 3\nvalue = [0, 0, 0, 0, 3, 0,
0]'),
 Text(227.867,112.22,'gini = 0.0\nsamples = 9\nvalue = [0, 0, 0, 9, 0, 0, 0]'),
 Text(265.844,144.283,'gini = 0.0\nsamples = 14\nvalue = [0, 14, 0, 0, 0, 0,
0]'),
 Text(303.822,176.346,'gini = 0.0\nsamples = 25\nvalue = [25, 0, 0, 0, 0, 0,
0]')]
```



In [ ]:

In [ ]: