Name : Arul kumar ARK

Roll No. : 225229103

Lab : 8

```
Animal Classification Using Decision Trees
```

Step : 1

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

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

In [3]:
```
data
```

Out[3]:

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

In [4]:
```
data.shape
```

Out[4]: (10, 5)

In [5]:
```
data.describe()
```

Out[5]:

	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

Step : 2

In [6]:
```
X=data.drop(['species'],axis=1)
```

In [7]:
```
X
```

Out[7]:

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

In [8]:
```
y=data['species'].values
```

In [9]:
```
y
```

Out[9]:
```
array(['Mammal', 'Mammal', 'Raptile', 'Mammal', 'Mammal', 'Mammal',
       'Raptile', 'Raptile', 'Mammal', 'Raptile'], dtype=object)
```

In [10]:
```
from sklearn.model_selection import train_test_split
```

In [11]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.40,random_state=0)
```

In [12]:
```
X_train.shape
```

Out[12]:
```
(6, 4)
```

In [13]:
```
X_test.shape
```

Out[13]:
```
(4, 4)
```

In [14]:
```
y_train.shape
```

Out[14]:
```
(6,)
```

In [15]:
```
y_test.shape
```

Out[15]:
```
(4,)
```

In [16]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [17]:
```
data_entropy = DecisionTreeClassifier(criterion ="entropy")
data_entropy.fit(X_train,y_train)
```

Out[17]:
```
DecisionTreeClassifier(criterion='entropy')
```

In [18]:
```
y_pred = data_entropy.predict(X_test)
```

In [19]:
```
y_pred
```

Out[19]:
```
array(['Raptile', 'Mammal', 'Mammal', 'Raptile'], dtype=object)
```

In [21]:
```
from sklearn.metrics import accuracy_score,classification_report
```

In [22]:
```
acc = accuracy_score(y_test, y_pred)
print("Accuracy score :",acc)
```

```
Accuracy score : 1.0
```

In [23]:
```
clf_report= classification_report(y_test, y_pred)
print("Classification report: ",clf_report)
```

```
Classification report:                precision    recall  f1-score   support

      Mammal       1.00      1.00      1.00         2
     Raptile       1.00      1.00      1.00         2

    accuracy                           1.00         4
   macro avg       1.00      1.00      1.00         4
weighted avg       1.00      1.00      1.00         4
```

In [24]:
```
from sklearn import tree
```

In [26]:
```
with open("tree1.dot",'w') as f:
    f= tree.export_graphviz(data_entropy,out_file=f,max_depth=4,impurity= False,
                            feature_names = X.columns.values,class_names=['Reptile','Mammal'],filled=True)
```

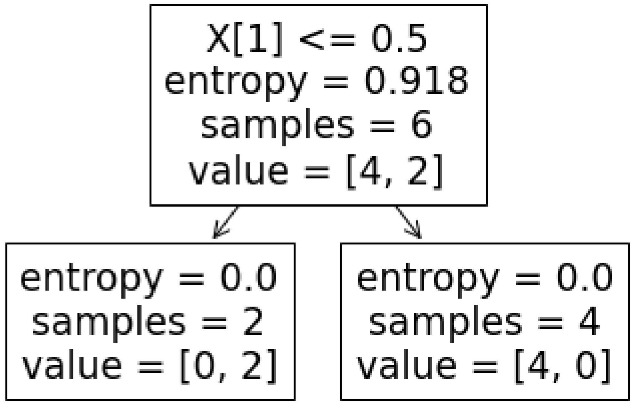
In [27]:
```
!type tree1.dot
```

```
digraph Tree {
node [shape=box, style="filled", color="black"] ;
0 [label="hair <= 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 [29]:
```
tree.plot_tree(data_entropy)
```

```
Matplotlib is building the font cache; this may take a moment.
```

Out[29]:
```
[Text(167.4, 163.07999999999998, 'X[1] <= 0.5\nentropy = 0.918\nsamples = 6\nvalue = [4, 2]'),
 Text(83.7, 54.360000000000014, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(251.10000000000002, 54.360000000000014, 'entropy = 0.0\nsamples = 4\nvalue = [4, 0]')]
```



step : 3

In [32]:
```
d_test=pd.read_csv("animals_test.csv")
```

In [33]:
```
d_test
```

Out[33]:

	Name	toothed	hair	breathes	legs	species
0	Turtile	False	False	True	False	Raptile
1	Blue whales	False	True	True	True	Mammal
2	Crocodile	True	False	True	True	Raptile

In [34]:
```
test_x=d_test.drop(['species','Name'],axis=1)
```

In [35]:
```
test_x
```

Out[35]:

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

Step : 4

In [36]:
```
y_pred_test=data_entropy.predict(test_x)
```

In [37]:
```
y_pred_test
```

Out[37]:
```
array(['Raptile', 'Mammal', 'Raptile'], dtype=object)
```

Step : 5

In [38]:
```
d_gini = DecisionTreeClassifier(criterion ="gini")
d_gini.fit(X,y)
```

Out[38]:
```
DecisionTreeClassifier()
```

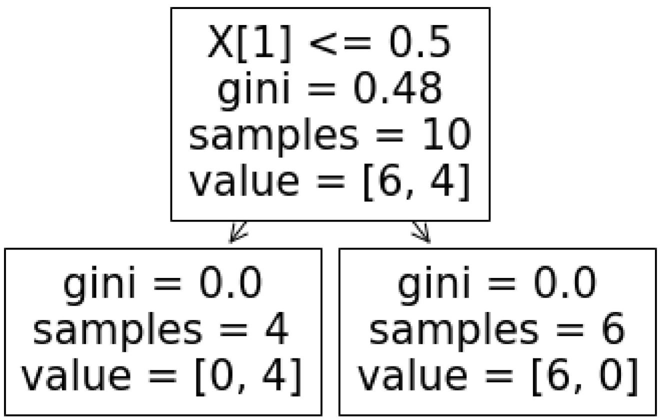
In [39]:
```
y_pred_test=d_gini.predict(test_x)
```

In [40]:
```
y_pred_test
```

Out[40]:
```
array(['Raptile', 'Mammal', 'Raptile'], dtype=object)
```

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

Out[41]:
```
[Text(167.4, 163.07999999999998, 'X[1] <= 0.5\ngini = 0.48\nsamples = 10\nvalue = [6, 4]'),
 Text(83.7, 54.360000000000014, 'gini = 0.0\nsamples = 4\nvalue = [0, 4]'),
 Text(251.10000000000002, 54.360000000000014, 'gini = 0.0\nsamples = 6\nvalue = [6, 0]')]
```



Step : 6

In [42]:
```
d_zoo=pd.read_csv("zoo.data")
```

In [43]:
```
d_zoo
```

Out[43]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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 [44]:
```
d_zoo.shape
```

Out[44]:
```
(100, 18)
```

In [45]:
```
d_zoo.describe()
```

Out[45]:

	1	0	0.1	1.1	0.2	0.3	1.2	1.3	1.4	1.5	0.4	
count	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.00	100.000000	100.000000	100.00000	100.00000	10
mean	0.420000	0.200000	0.590000	0.400000	0.240000	0.360000	0.55	0.600000	0.820000	0.79000	0.08000	
std	0.496045	0.402015	0.494311	0.492366	0.429235	0.482418	0.50	0.492366	0.386123	0.40936	0.27266	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.00000	0.00000	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	1.000000	1.00000	0.00000	
50%	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.00	1.000000	1.000000	1.00000	0.00000	
75%	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	1.00	1.000000	1.000000	1.00000	0.00000	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00	1.000000	1.000000	1.00000	1.00000	

In [47]:
```
X=d_zoo.drop(['aardvark','1.7'],axis=1)
```

In [49]:
```
X[:5]
```

Out[49]:

	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
0	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1
1	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0
2	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1
3	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1
4	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1

In [53]:
```
d_zoo.describe()
```

Out[53]:

	1	0	0.1	1.1	0.2	0.3	1.2	1.3	1.4	1.5	0.4
count	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.00	100.000000	100.000000	100.00000	100.00000
mean	0.420000	0.200000	0.590000	0.400000	0.240000	0.360000	0.55	0.600000	0.820000	0.79000	0.08000
std	0.496045	0.402015	0.494311	0.492366	0.429235	0.482418	0.50	0.492366	0.386123	0.40936	0.27266
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	0.000000	0.00000	0.00000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	0.000000	1.000000	1.00000	0.00000
50%	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.00	1.000000	1.000000	1.00000	0.00000
75%	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	1.00	1.000000	1.000000	1.00000	0.00000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00	1.000000	1.000000	1.00000	1.00000

10

In [54]:
```
y=d_zoo['1.7'].values
```

In [55]:
```
y
```

Out[55]:
```
array([1, 4, 1, 1, 1, 1, 4, 4, 1, 1, 2, 4, 7, 7, 7, 2, 1, 4, 1, 2, 2, 1,
       2, 6, 5, 5, 1, 1, 1, 6, 1, 1, 2, 4, 1, 1, 2, 4, 6, 6, 2, 6, 2, 1,
       1, 7, 1, 1, 1, 1, 6, 5, 7, 1, 1, 2, 2, 2, 2, 4, 4, 3, 1, 1, 1, 1,
       1, 1, 1, 1, 2, 7, 4, 1, 1, 3, 7, 2, 2, 3, 7, 4, 2, 1, 7, 4, 2, 6,
       5, 3, 3, 4, 1, 1, 2, 1, 6, 1, 7, 2], dtype=int64)
```

In [56]:
```
X1_train, X1_test, y1_train, y1_test = train_test_split(X, y, test_size=0.33,random_state=0)
```

In [60]:
```
X1_train.shape
```

Out[60]:
```
(67, 16)
```

In [61]:
```
X1_test.shape
```

Out[61]:
```
(33, 16)
```

In [62]:
```
y1_train.shape
```

Out[62]:
```
(67,)
```

In [63]:
```
y1_test.shape
```

Out[63]:
```
(33,)
```

In [64]:
```
zoo_entropy = DecisionTreeClassifier(criterion ="entropy")
zoo_entropy.fit(X1_train,y1_train)
```

Out[64]:
```
DecisionTreeClassifier(criterion='entropy')
```

In [65]:
```
y1_pred = zoo_entropy.predict(X1_test)
```

In [66]:
```
y1_pred
```

Out[66]:
```
array([1, 2, 1, 2, 5, 1, 1, 1, 1, 1, 1, 1, 2, 7, 4, 1, 2, 5, 4, 1, 1, 1,
       1, 6, 7, 1, 4, 2, 2, 7, 4, 7, 3], dtype=int64)
```

Accuracy

In [69]:
```
train_acc=zoo_entropy.predict(X1_train)
train_acc
```

Out[69]:
```
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)
```

In [70]:
```
print("Train Accuracy:", accuracy_score(y1_train, zoo_entropy.predict(X1_train)))
```

```
Train Accuracy: 1.0
```

In [71]:
```
test_acc=zoo_entropy.predict(X1_test)
test_acc
```

Out[71]:
```
array([1, 2, 1, 2, 5, 1, 1, 1, 1, 1, 1, 1, 2, 7, 4, 1, 2, 5, 4, 1, 1, 1,
       1, 6, 7, 1, 4, 2, 2, 7, 4, 7, 3], dtype=int64)
```

In [72]:
```
print("Test Accuracy:", accuracy_score(y1_test, zoo_entropy.predict(X1_test)))
```

```
Test Accuracy: 0.9393939393939394
```

In [73]:
```
acc = accuracy_score(y1_test, y1_pred)
print("Accuracy score :",acc)
```

```
Accuracy score : 0.9393939393939394
```

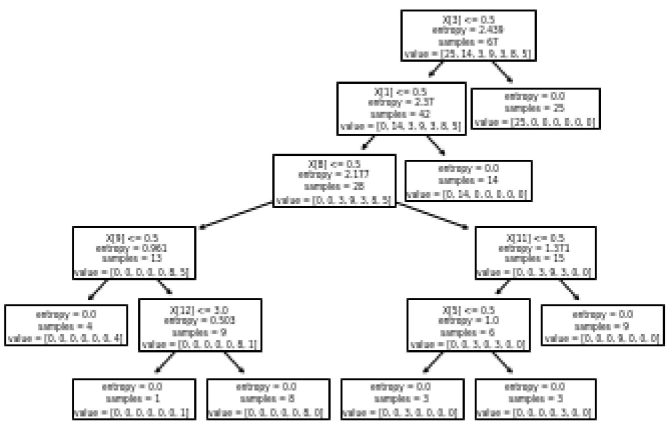
In [74]:
```
clf_report= classification_report(y1_test, y1_pred)
print("Classification report: ",clf_report)
```

```
Classification report:                precision    recall  f1-score   support

           1       1.00      1.00      1.00        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
           6       0.00      0.00      0.00         0
           7       1.00      0.80      0.89         5

    accuracy                           0.94        33
   macro avg       0.79      0.76      0.75        33
weighted avg       0.98      0.94      0.95        33


C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Re
call and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` para
meter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Re
call and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` para
meter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Re
call and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` para
meter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

ID3

In [75]:
```
tree.plot_tree(zoo_entropy)
```

Out[75]:
```
[Text(234.36, 199.32, 'X[3] <= 0.5\nentropy = 2.439\nsamples = 67\nvalue = [25, 14, 3, 9, 3, 8, 5]'),
 Text(200.88000000000002, 163.07999999999998, 'X[1] <= 0.5\nentropy = 2.37\nsamples = 42\nvalue = [0, 14, 3,
9, 3, 8, 5]'),
 Text(167.40000000000003, 126.83999999999999, 'X[8] <= 0.5\nentropy = 2.177\nsamples = 28\nvalue = [0, 0, 3,
9, 3, 8, 5]'),
 Text(66.96000000000001, 90.6, 'X[9] <= 0.5\nentropy = 0.961\nsamples = 13\nvalue = [0, 0, 0, 0, 0, 8, 5]'),
 Text(33.480000000000004, 54.359999999999985, 'entropy = 0.0\nsamples = 4\nvalue = [0, 0, 0, 0, 0, 0, 4]'),
 Text(100.44000000000001, 54.359999999999985, 'X[12] <= 3.0\nentropy = 0.503\nsamples = 9\nvalue = [0, 0, 0,
0, 0, 8, 1]'),
 Text(66.96000000000001, 18.119999999999976, 'entropy = 0.0\nsamples = 1\nvalue = [0, 0, 0, 0, 0, 0, 1]'),
 Text(133.92000000000002, 18.119999999999976, 'entropy = 0.0\nsamples = 8\nvalue = [0, 0, 0, 0, 0, 8, 0]'),
 Text(267.84000000000003, 90.6, 'X[11] <= 0.5\nentropy = 1.371\nsamples = 15\nvalue = [0, 0, 3, 9, 3, 0, 0]'),
 Text(234.36, 54.359999999999985, 'X[5] <= 0.5\nentropy = 1.0\nsamples = 6\nvalue = [0, 0, 3, 0, 3, 0, 0]'),
 Text(200.88000000000002, 18.119999999999976, 'entropy = 0.0\nsamples = 3\nvalue = [0, 0, 3, 0, 0, 0, 0]'),
 Text(267.84000000000003, 18.119999999999976, 'entropy = 0.0\nsamples = 3\nvalue = [0, 0, 0, 0, 3, 0, 0]'),
 Text(301.32000000000005, 54.359999999999985, 'entropy = 0.0\nsamples = 9\nvalue = [0, 0, 0, 9, 0, 0, 0]'),
 Text(234.36, 126.83999999999999, 'entropy = 0.0\nsamples = 14\nvalue = [0, 14, 0, 0, 0, 0, 0]'),
 Text(267.84000000000003, 163.07999999999998, 'entropy = 0.0\nsamples = 25\nvalue = [25, 0, 0, 0, 0, 0, 0]')]
```



Gini

In [77]:
```
X2_train, X2_test, y2_train, y2_test = train_test_split(X, y, test_size=0.33,random_state=0)
```

In [78]:
```
zoo2_entropy = DecisionTreeClassifier(criterion ="gini")
zoo2_entropy.fit(X2_train,y2_train)
```

Out[78]:
```
DecisionTreeClassifier()
```

In [79]:
```
y2_pred = zoo2_entropy.predict(X2_test)
```

In [80]:
```
y2_pred
```

Out[80]:
```
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 [83]:
```
train_acc=zoo2_entropy.predict(X2_train)
train_acc
```

Out[83]:
```
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)
```

In [84]:
```
print("Train Accuracy:", accuracy_score(y2_train, zoo2_entropy.predict(X2_train)))
```

```
Train Accuracy: 1.0
```

In [86]:
```
test_acc=zoo2_entropy.predict(X2_train)
test_acc
```

Out[86]:
```
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)
```

In [87]:
```
print("Test Accuracy:", accuracy_score(y2_test, zoo2_entropy.predict(X2_test)))
```

```
Test Accuracy: 0.9090909090909091
```

In [88]:
```
acc = accuracy_score(y2_test, y2_pred)
print("Accuracy score :",acc)
```

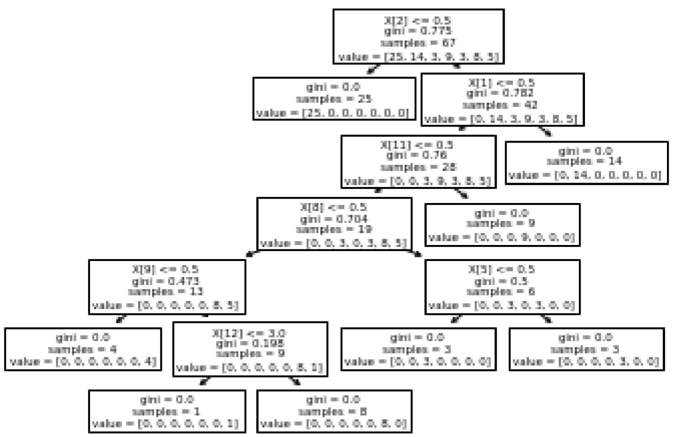
```
Accuracy score : 0.9090909090909091
```

In [89]:
```
clf_report= classification_report(y2_test, y2_pred)
print("Classification report: ",clf_report)
```

```
Classification report:                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 [90]:
```
tree.plot_tree(zoo2_entropy)
```

Out[90]:
```
[Text(209.25, 201.90857142857143, 'X[2] <= 0.5\ngini = 0.775\nsamples = 67\nvalue = [25, 14, 3, 9, 3, 8, 5]'),
 Text(167.4, 170.84571428571428, 'gini = 0.0\nsamples = 25\nvalue = [25, 0, 0, 0, 0, 0, 0]'),
 Text(251.10000000000002, 170.84571428571428, 'X[1] <= 0.5\ngini = 0.782\nsamples = 42\nvalue = [0, 14, 3, 9, 3, 8, 5]'),
 Text(209.25, 139.78285714285715, 'X[11] <= 0.5\ngini = 0.76\nsamples = 28\nvalue = [0, 0, 3, 9, 3, 8, 5]'),
 Text(167.4, 108.72, 'X[8] <= 0.5\ngini = 0.704\nsamples = 19\nvalue = [0, 0, 3, 0, 3, 8, 5]'),
 Text(83.7, 77.65714285714284, 'X[9] <= 0.5\ngini = 0.473\nsamples = 13\nvalue = [0, 0, 0, 0, 0, 8, 5]'),
 Text(41.85, 46.59428571428572, 'gini = 0.0\nsamples = 4\nvalue = [0, 0, 0, 0, 0, 0, 4]'),
 Text(125.55000000000001, 46.59428571428572, 'X[12] <= 3.0\ngini = 0.198\nsamples = 9\nvalue = [0, 0, 0, 0, 0, 8, 1]'),
 Text(83.7, 15.531428571428563, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 0, 0, 0, 0, 1]'),
 Text(167.4, 15.531428571428563, 'gini = 0.0\nsamples = 8\nvalue = [0, 0, 0, 0, 0, 8, 0]'),
 Text(251.10000000000002, 77.65714285714284, 'X[5] <= 0.5\ngini = 0.5\nsamples = 6\nvalue = [0, 0, 3, 0, 3, 0, 0]'),
 Text(209.25, 46.59428571428572, 'gini = 0.0\nsamples = 3\nvalue = [0, 0, 3, 0, 0, 0, 0]'),
 Text(292.95, 46.59428571428572, 'gini = 0.0\nsamples = 3\nvalue = [0, 0, 0, 0, 3, 0, 0]'),
 Text(251.10000000000002, 108.72, 'gini = 0.0\nsamples = 9\nvalue = [0, 0, 0, 9, 0, 0, 0]'),
 Text(292.95, 139.78285714285715, 'gini = 0.0\nsamples = 14\nvalue = [0, 14, 0, 0, 0, 0, 0]')]
```



In [ ]: