Lab 8 : Animal Classification using Decision Tree

Name : Murali kumar R

Roll no : 225229120

Step 1 : [Create Dataset]

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

In [8]:
```
tree=pd.read_csv('animal.csv')
```

Step 2 : [Model building using ID3]

In [3]:
```
#1) import your dataset:
```

In [11]:
```
tree
```

Out[11]:

	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 [12]:
```
tree.shape
```

Out[12]:
```
(10, 5)
```

In [13]:
```
tree.size
```

Out[13]:
```
50
```

In [16]:
```
tree.describe()
```

Out[16]:

	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

In [20]:
```
tree.count()
```

Out[20]:
```
Toothed     10
hair        10
breathes    10
legs        10
species     10
dtype: int64
```

Create DT model using 'entropy' criterion

In [46]:
```
X = tree.drop(['species'],axis = 1)
```

In [45]:
```
y = tree['species']
```

In [24]:
```
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_state=0
```

In [28]:
```
from sklearn.tree import DecisionTreeClassifier
clf_entropy = DecisionTreeClassifier(criterion = "entropy")
```

Perform training and testing

In [29]:
```
clf_entropy.fit(X_train,y_train)
```

Out[29]:
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

In [34]:
```
y_pred=clf_entropy.predict(X_test)
y_pred
```

Out[34]:
```
array(['Reptile', 'Mammal', 'Mammal', 'Mammal'], dtype=object)
```

Print accuracy and classification report

In [32]:
```
from sklearn.metrics import accuracy_score
```

In [35]:
```
print ("Accuracy for ID3: ",accuracy_score(y_test,y_pred))
```

```
Accuracy for ID3:  0.75
```

In [36]:
```
from sklearn.metrics import classification_report
```

In [41]:
```
print(classification_report(y_test,y_pred))
```

```
             precision    recall  f1-score   support

     Mammal       0.67      1.00      0.80         2
    Reptile       1.00      0.50      0.67         2

avg / total       0.83      0.75      0.73         4
```

Interpret your results

In [47]:
```
from sklearn import tree
```

Visualize your DT model using graphviz

In [52]:
```
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 [56]:
```
!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="#e5813
97f"] ;
1 [label="samples = 2\nvalue = [0, 2]\nclass = Mammal", fillcolor="#399de5ff"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="samples = 4\nvalue = [4, 0]\nclass = Reptile", fillcolor="#e58139ff"] ;
0 -> 2 [labeldistance=2.5, labelangle=-45, headlabel="False"] ;
}
```

Step3 : [Create a Test Set]

In [65]:
```
set=pd.read_csv('atest.csv')
```

In [66]:
```
set
```

Out[66]:

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

In [70]:
```
test = set.drop(['species'],axis = 1)
```

In [71]:
```
test
```

Out[71]:

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

Step 4 : [Perform Predicition]

In [75]:
```
y_pred_test=clf_entropy.predict(test)
```

In [76]:
```
y_pred_test
```

Out[76]:
```
array(['Reptile', 'Mammal', 'Mammal'], dtype=object)
```

Step5 : [Build CART Decision Tree Model]

Now, you are going to build a new CART decision tree using criterion='gini'

In [86]:
```
clf_gini = DecisionTreeClassifier(criterion ="gini")
```

Train you model with full training data (No,train test spilt,this time)

In [87]:
```
clf_gini.fit(X,y)
```

Out[87]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

Predict samples for the test file

In [84]:
```
y_pred_gini=clf_gini.predict(test)
```

In [85]:
```
y_pred_gini
```

Out[85]:
```
array(['Reptile', 'Mammal', 'Reptile'], dtype=object)
```

Visualize your CART DT using graphviz

In [88]:
```
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 [89]:
```
!type tree2.dot
```

```
digraph Tree {
node [shape=box, style="filled", color="black"] ;
0 [label="hair <= 0.5\nsamples = 10\nvalue = [6, 4]\nclass = Reptile", fillcolor="#e581
3955"] ;
1 [label="samples = 4\nvalue = [0, 4]\nclass = Mammal", fillcolor="#399de5ff"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="samples = 6\nvalue = [6, 0]\nclass = Reptile", fillcolor="#e58139ff"] ;
0 -> 2 [labeldistance=2.5, labelangle=-45, headlabel="False"] ;
}
```

Step 6 : [Build DT with Zoo dataset]

Entropy

In [181]:
```
zoo=pd.read_csv('zoo.csv')
zoo.head()
```

Out[181]:

	animal_name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venom
0	aardvark	1	0	0	1	0	0	1	1	1	1	
1	antelope	1	0	0	1	0	0	0	1	1	1	
2	bass	0	0	1	0	0	1	1	1	1	0	
3	bear	1	0	0	1	0	0	1	1	1	1	
4	boar	1	0	0	1	0	0	1	1	1	1	

In [130]:
```
X1=zoo.drop(['animal_name'],axis=1)
```

In [131]:
```
y1=zoo['class_type']
```

In [132]:
```
from sklearn.model_selection import train_test_split
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.33, random_s
```

In [133]:
```
from sklearn.tree import DecisionTreeClassifier
zoo_entropy = DecisionTreeClassifier(criterion = "entropy")
```

In [134]:
```
zoo_entropy.fit(X1_train,y1_train)
```

Out[134]:
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

In [137]:
```
y1_pred=zoo_entropy.predict(X1_test)
```

In [138]:
```
y1_pred
```

Out[138]:
```
array([5, 4, 4, 1, 1, 1, 2, 4, 1, 1, 7, 1, 2, 7, 4, 6, 1, 6, 2, 4, 2, 3,
       1, 2, 1, 1, 1, 2, 4, 3, 4, 3, 3, 1], dtype=int64)
```

Train_acc

In [144]:
```
train_acc=zoo_entropy.predict(X1_train)
```

In [145]:
```
train_acc
```

Out[145]:
```
array([6, 1, 4, 2, 1, 1, 1, 4, 6, 2, 1, 7, 1, 6, 4, 1, 4, 2, 1, 4, 1, 1,
       2, 1, 2, 5, 1, 1, 1, 2, 2, 7, 1, 1, 7, 3, 1, 1, 1, 1, 7, 1, 2, 2,
       7, 7, 5, 2, 7, 7, 6, 1, 2, 4, 5, 6, 1, 2, 1, 2, 2, 1, 6, 1, 1, 1,
       1], dtype=int64)
```

In [146]:
```
train_acc=zoo_entropy.predict(X1_test)
```

In [147]:
```
train_acc
```

Out[147]:
```
array([5, 4, 4, 1, 1, 1, 2, 4, 1, 1, 7, 1, 2, 7, 4, 6, 1, 6, 2, 4, 2, 3,
       1, 2, 1, 1, 1, 2, 4, 3, 4, 3, 3, 1], dtype=int64)
```

In [158]:
```
accuracy_score(y1_test,y1_pred)
```

Out[158]:
```
1.0
```

In [160]:
```
print(classification_report(y1_test, y1_pred))
```

```
             precision    recall  f1-score   support

          1       1.00      1.00      1.00        12
          2       1.00      1.00      1.00         6
          3       1.00      1.00      1.00         4
          4       1.00      1.00      1.00         7
          5       1.00      1.00      1.00         1
          6       1.00      1.00      1.00         2
          7       1.00      1.00      1.00         2

avg / total       1.00      1.00      1.00        34
```

In [162]:
```
with open("tree1.dot","w") as f:
    f=tree.export_graphviz(zoo_entropy,
                           out_file=f,
                           max_depth=4,
                           impurity=False,
                           feature_names=X.columns.values,
                           class_names=['1','2','3','4','5','6','7','8'],
                           filled=True)
```

In [163]:
```
!type tree1.dot
```

```
digraph Tree {
node [shape=box, style="filled", color="black"] ;
0 [label="milk <= 0.5\nsamples = 67\nvalue = [29, 14, 1, 6, 3, 6, 8]\nclass = 1", fillc
olor="#e5813948"] ;
1 [label="class_type <= 4.5\nsamples = 38\nvalue = [0, 14, 1, 6, 3, 6, 8]\nclass = 2",
fillcolor="#b7e53933"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="class_type <= 2.5\nsamples = 21\nvalue = [0, 14, 1, 6, 0, 0, 0]\nclass = 2",
fillcolor="#b7e53988"] ;
1 -> 2 ;
3 [label="samples = 14\nvalue = [0, 14, 0, 0, 0, 0, 0]\nclass = 2", fillcolor="#b7e539f
f"] ;
2 -> 3 ;
4 [label="class_type <= 3.5\nsamples = 7\nvalue = [0, 0, 1, 6, 0, 0, 0]\nclass = 4", fi
llcolor="#39e5e2d4"] ;
2 -> 4 ;
5 [label="samples = 1\nvalue = [0, 0, 1, 0, 0, 0, 0]\nclass = 3", fillcolor="#39e54df
f"] ;
4 -> 5 ;
6 [label="samples = 6\nvalue = [0, 0, 0, 6, 0, 0, 0]\nclass = 4", fillcolor="#39e5e2f
f"] ;
4 -> 6 ;
7 [label="class_type <= 6.5\nsamples = 17\nvalue = [0, 0, 0, 0, 3, 6, 8]\nclass = 7", f
illcolor="#e539862e"] ;
1 -> 7 ;
8 [label="legs <= 5.0\nsamples = 9\nvalue = [0, 0, 0, 0, 3, 6, 0]\nclass = 6", fillcolo
r="#b139e57f"] ;
7 -> 8 ;
9 [label="samples = 3\nvalue = [0, 0, 0, 0, 3, 0, 0]\nclass = 5", fillcolor="#3956e5f
f"] ;
8 -> 9 ;
10 [label="samples = 6\nvalue = [0, 0, 0, 0, 0, 6, 0]\nclass = 6", fillcolor="#b139e5f
f"] ;
8 -> 10 ;
11 [label="samples = 8\nvalue = [0, 0, 0, 0, 0, 0, 8]\nclass = 7", fillcolor="#e53986f
f"] ;
7 -> 11 ;
12 [label="samples = 29\nvalue = [29, 0, 0, 0, 0, 0, 0]\nclass = 1", fillcolor="#e58139
ff"] ;
0 -> 12 [labeldistance=2.5, labelangle=-45, headlabel="False"] ;
}
```

Gini

In [192]:
```
zoo_gini = DecisionTreeClassifier(criterion ="gini")
```

In [193]:
```
zoo_gini.fit(X,y)
```

Out[193]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

In [194]:
```
y_pred_gini=zoo_gini.predict(X1_test)
```

In [195]:
```
y_pred_gini
```

Out[195]:
```
array([5, 4, 4, 1, 1, 1, 2, 4, 1, 1, 7, 1, 2, 7, 4, 6, 1, 6, 2, 4, 2, 3,
       1, 2, 1, 1, 1, 2, 4, 3, 4, 3, 3, 1], dtype=int64)
```

In [196]:
```
accuracy_score(y1_test,y_pred_gini)
```

Out[196]:
```
1.0
```

In [197]:
```
print(classification_report(y1_test, y1_pred))
```

```
             precision    recall  f1-score   support

          1       1.00      1.00      1.00        12
          2       1.00      1.00      1.00         6
          3       1.00      1.00      1.00         4
          4       1.00      1.00      1.00         7
          5       1.00      1.00      1.00         1
          6       1.00      1.00      1.00         2
          7       1.00      1.00      1.00         2

avg / total       1.00      1.00      1.00        34
```

In [198]:
```
with open("tree2.dot","w") as f:
    f=tree.export_graphviz(zoo_gini,
                           out_file=f,
                           max_depth=4,
                           impurity=False,
                           feature_names=X.columns.values,
                           class_names=['1','2','3','4','5','6','7','8'],
                           filled=True)
```

In [199]:
```
!type tree2.dot
```

```
digraph Tree {
node [shape=box, style="filled", color="black"] ;
0 [label="milk <= 0.5\nsamples = 101\nvalue = [41, 20, 5, 13, 4, 8, 10]\nclass = 1", fi
llcolor="#e5813942"] ;
1 [label="feathers <= 0.5\nsamples = 60\nvalue = [0, 20, 5, 13, 4, 8, 10]\nclass = 2",
fillcolor="#b7e53926"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="fins <= 0.5\nsamples = 40\nvalue = [0, 0, 5, 13, 4, 8, 10]\nclass = 4", fillc
olor="#39e5e21a"] ;
1 -> 2 ;
3 [label="class_type <= 6.5\nsamples = 27\nvalue = [0, 0, 5, 0, 4, 8, 10]\nclass = 7",
fillcolor="#e539861b"] ;
2 -> 3 ;
4 [label="legs <= 5.0\nsamples = 17\nvalue = [0, 0, 5, 0, 4, 8, 0]\nclass = 6", fillcol
or="#b139e540"] ;
3 -> 4 ;
5 [label="(...)", fillcolor="#C0C0C0"] ;
4 -> 5 ;
8 [label="(...)", fillcolor="#C0C0C0"] ;
4 -> 8 ;
9 [label="samples = 10\nvalue = [0, 0, 0, 0, 0, 0, 10]\nclass = 7", fillcolor="#e53986f
f"] ;
3 -> 9 ;
10 [label="samples = 13\nvalue = [0, 0, 0, 13, 0, 0, 0]\nclass = 4", fillcolor="#39e5e2
ff"] ;
2 -> 10 ;
11 [label="samples = 20\nvalue = [0, 20, 0, 0, 0, 0, 0]\nclass = 2", fillcolor="#b7e539
ff"] ;
1 -> 11 ;
12 [label="samples = 41\nvalue = [41, 0, 0, 0, 0, 0, 0]\nclass = 1", fillcolor="#e58139
ff"] ;
0 -> 12 [labeldistance=2.5, labelangle=-45, headlabel="False"] ;
}
```

In [ ]:
```

```