LAB3

August 25, 2019

[1]:
```
from sklearn import preprocessing
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_graphviz
from subprocess import call
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import pandas as pd
```

[2]:
```
Outlook = ['Rainy', 'Rainy', 'Overcast', 'Sunny', 'Sunny', 'Sunny',␣
 ↪'Overcast','Rainy', 'Rainy', 'Sunny', 'Rainy','Overcast', 'Overcast',␣
 ↪'Sunny']
Temperature = ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool','Mild',␣
 ↪'Cool', 'Mild', 'Mild', 'Mild', 'Hot', 'Mild']
Humidity = ['High', 'High', 'High', 'High', 'Normal', 'Normal',␣
 ↪'Normal','High', 'Normal', 'Normal', 'Normal', 'High', 'Normal', 'High']
Wind = ['False', 'True', 'False', 'False', 'False', 'True', 'True','False',␣
 ↪'False', 'False', 'True', 'True', 'False', 'True']
Play = ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No','Yes', 'Yes', 'Yes',␣
 ↪'Yes', 'Yes', 'No']
```

[3]:
```
le = preprocessing.LabelEncoder()
Outlook_encoded = le.fit_transform(Outlook)
Outlook_name_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
print("Outllok mapping:",Outlook_name_mapping)
Temperature_encoded = le.fit_transform(Temperature)
Temperature_name_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
print("Temperature mapping:",Temperature_name_mapping)
Humidity_encoded = le.fit_transform(Humidity)
Humidity_name_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
print("Humidity mapping:",Humidity_name_mapping)
Wind_encoded = le.fit_transform(Wind)
Wind_name_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
print("Wind mapping:",Wind_name_mapping)
Play_encoded = le.fit_transform(Play)
Play_name_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
print("Play mapping:",Play_name_mapping)
print("\n\n")
print("Weather:" ,Outlook_encoded)
```

```
print("Temerature:" ,Temperature_encoded)
print("Humidity:" ,Humidity_encoded)
print("Wind:" ,Wind_encoded)
print("Play:" ,Play_encoded)
```

```
Outllok mapping: {'Overcast': 0, 'Rainy': 1, 'Sunny': 2}
Temperature mapping: {'Cool': 0, 'Hot': 1, 'Mild': 2}
Humidity mapping: {'High': 0, 'Normal': 1}
Wind mapping: {'False': 0, 'True': 1}
Play mapping: {'No': 0, 'Yes': 1}
```

```
Weather: [1 1 0 2 2 2 0 1 1 2 1 0 0 2]
Temerature: [1 1 1 2 0 0 0 2 0 2 2 2 1 2]
Humidity: [0 0 0 0 1 1 1 0 1 1 1 0 1 0]
Wind: [0 1 0 0 0 1 1 0 0 0 1 1 0 1]
Play: [0 0 1 1 1 0 1 0 1 1 1 1 1 0]
```

[4]:
```
features=tuple(zip(Outlook_encoded,Temperature_encoded,Humidity_encoded,Wind_encoded))
print("Features:",features)
```

```
Features: ((1, 1, 0, 0), (1, 1, 0, 1), (0, 1, 0, 0), (2, 2, 0, 0), (2, 0, 1, 0),
(2, 0, 1, 1), (0, 0, 1, 1), (1, 2, 0, 0), (1, 0, 1, 0), (2, 2, 1, 0), (1, 2, 1,
1), (0, 2, 0, 1), (0, 1, 1, 0), (2, 2, 0, 1))
```

[5]:
```
clf_entropy = DecisionTreeClassifier(criterion = "entropy")
clf_entropy.fit(features,Play_encoded)
```

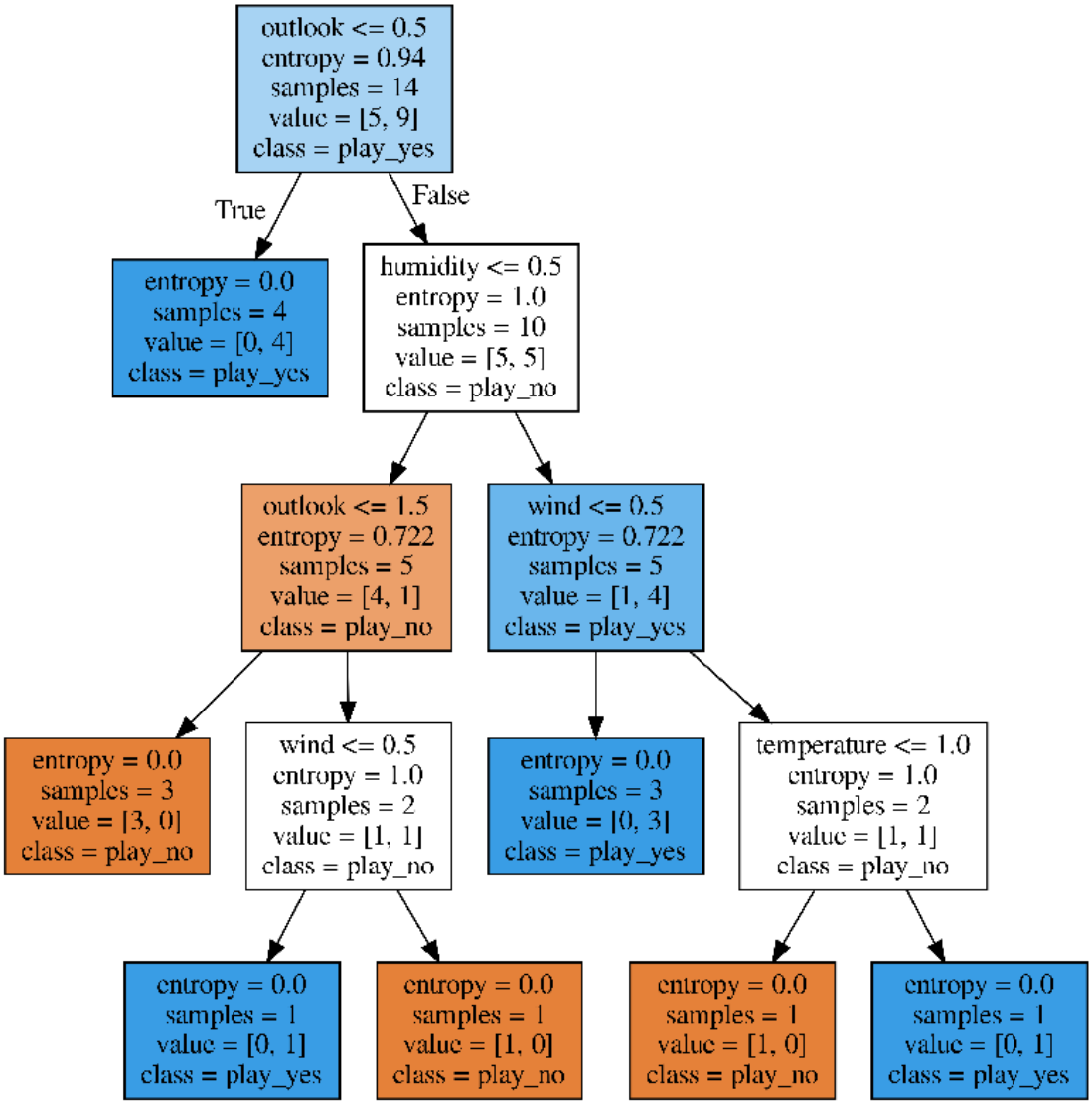
[5]:
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

[6]:
```
predicted= clf_entropy.predict([[1,2,1,0],[2,0,0,1]])
print("Predicted Play Values:", predicted)
```

```
Predicted Play Values: [1 0]
```

[7]:
```
export_graphviz(clf_entropy,out_file='tree_entropy.dot',
feature_names=['outlook','temperature','humidity','wind'],
class_names=['play_no','play_yes'],
filled=True)
call(['dot', '-Tpng', 'tree_entropy.dot', '-o', 'tree_entropy.png',␣
 ↪'-Gdpi=600'])
```

```
plt.figure(figsize = (14, 18))
plt.imshow(plt.imread('tree_entropy.png'))
plt.axis('off');
plt.show();
```



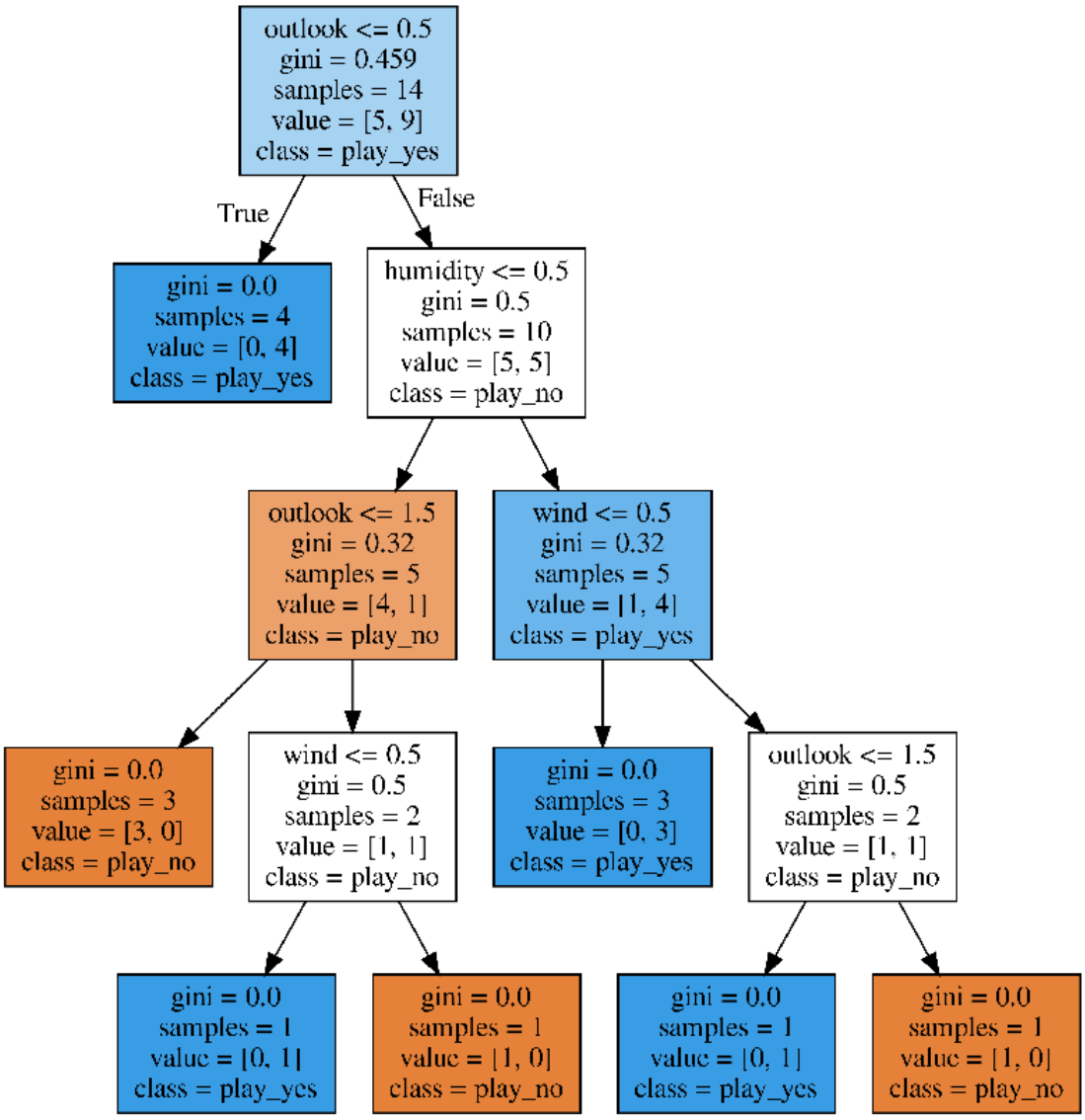
[8]:
```
clf_gini = DecisionTreeClassifier(criterion = "gini")
clf_gini.fit(features,Play_encoded)
```

[8]:
```
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')
```

[9]:
```
predicted= clf_gini.predict([[1,2,1,0],[2,0,0,1]])
print("Predicted Play Values:", predicted)
```

```
Predicted Play Values: [1 0]
```

[10]:
```
export_graphviz(clf_gini,out_file='tree_gini.dot',
feature_names=['outlook','temperature','humidity','wind'],
class_names=['play_no','play_yes'],
filled=True)
call(['dot', '-Tpng', 'tree_gini.dot', '-o', 'tree_gini.png', '-Gdpi=600'])
plt.figure(figsize = (14, 18))
plt.imshow(plt.imread('tree_gini.png'))
plt.axis('off');
plt.show();
```



[11]:
```
df = pd.read_csv('Dataset/zoo.csv')
df = df.drop(columns=['animal_name'])
df.head()
```

[11]:

	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	\
0	1	0	0	1	0	0	1	1	1	
1	1	0	0	1	0	0	0	1	1	
2	0	0	1	0	0	1	1	1	1	
3	1	0	0	1	0	0	1	1	1	
4	1	0	0	1	0	0	1	1	1	

	breathes	venomous	fins	legs	tail	domestic	catsize	type
0	1	0	0	4	0	0	1	1
1	1	0	0	4	1	0	1	1
2	0	0	1	0	1	0	0	4
3	1	0	0	4	0	0	1	1
4	1	0	0	4	1	0	1	1

[12]:
```
target = df['type']
df = df.drop(columns = ['type'])
df.head()
```

[12]:

	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	\
0	1	0	0	1	0	0	1	1	1	
1	1	0	0	1	0	0	0	1	1	
2	0	0	1	0	0	1	1	1	1	
3	1	0	0	1	0	0	1	1	1	
4	1	0	0	1	0	0	1	1	1	

	breathes	venomous	fins	legs	tail	domestic	catsize
0	1	0	0	4	0	0	1
1	1	0	0	4	1	0	1
2	0	0	1	0	1	0	0
3	1	0	0	4	0	0	1
4	1	0	0	4	1	0	1

[13]:
```
clf_entropy1 = DecisionTreeClassifier(criterion='entropy')
clf_entropy1.fit(df,target)
```

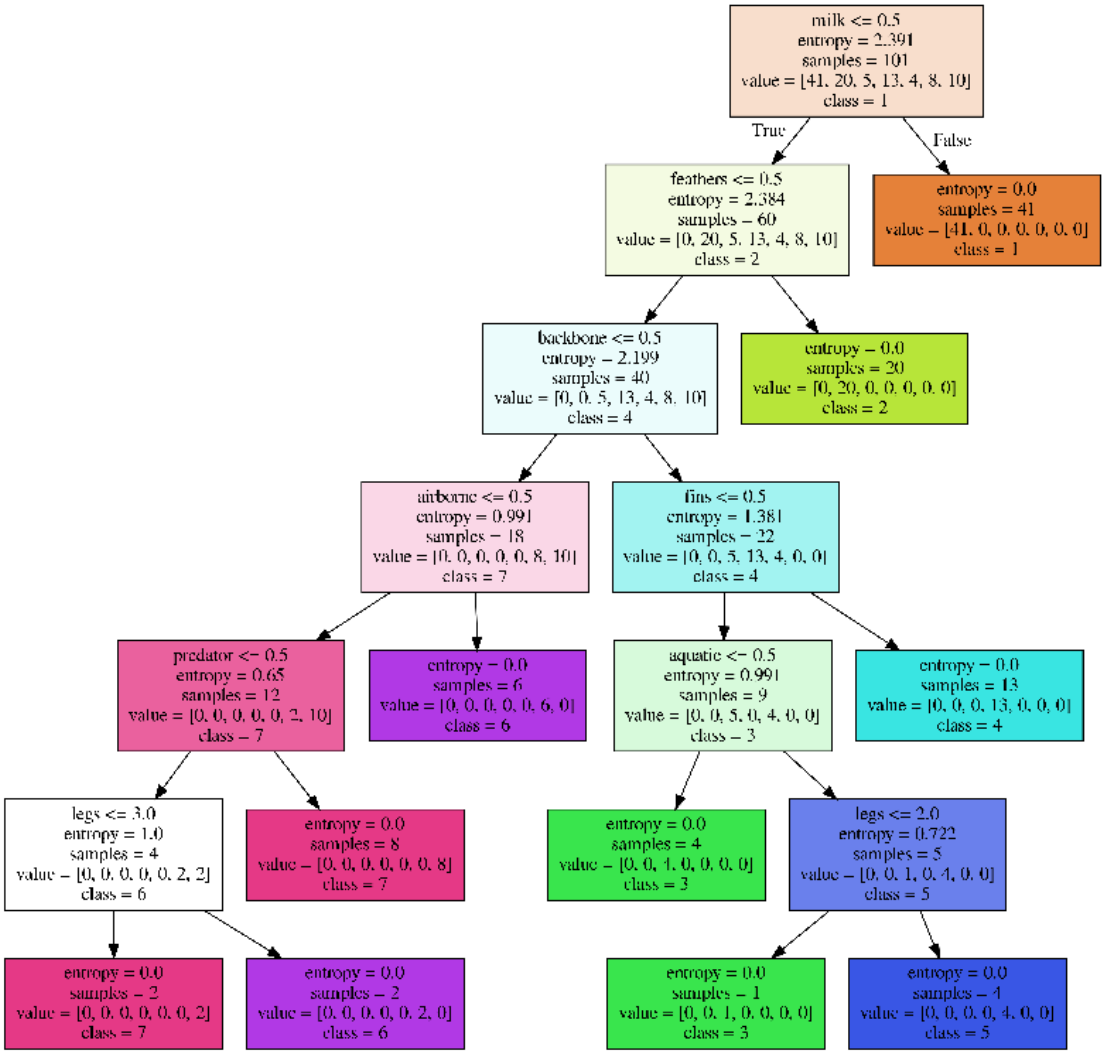
[13]:
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

[14]:
```
predicted= clf_entropy1.
 ↪predict([[0,0,0,0,0,0,0,1,1,0,0,0,1,1,0,0],[0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0]])
```

```
print("Predicted Play Values:", predicted)
```

```
Predicted Play Values: [3 7]
```

[15]:
```
export_graphviz(
    clf_entropy1,out_file='zoo_tree_entropy.dot',
    feature_names=df.columns,
    class_names=['1','2','3','4','5','6','7'],
    filled=True)
call(['dot', '-Tpng', 'zoo_tree_entropy.dot', '-o', 'zoo_tree_entropy.png',␣
 ↪'-Gdpi=600'])
plt.figure(figsize = (14, 18))
plt.imshow(plt.imread('zoo_tree_entropy.png'))
plt.axis('off');
plt.show();
```



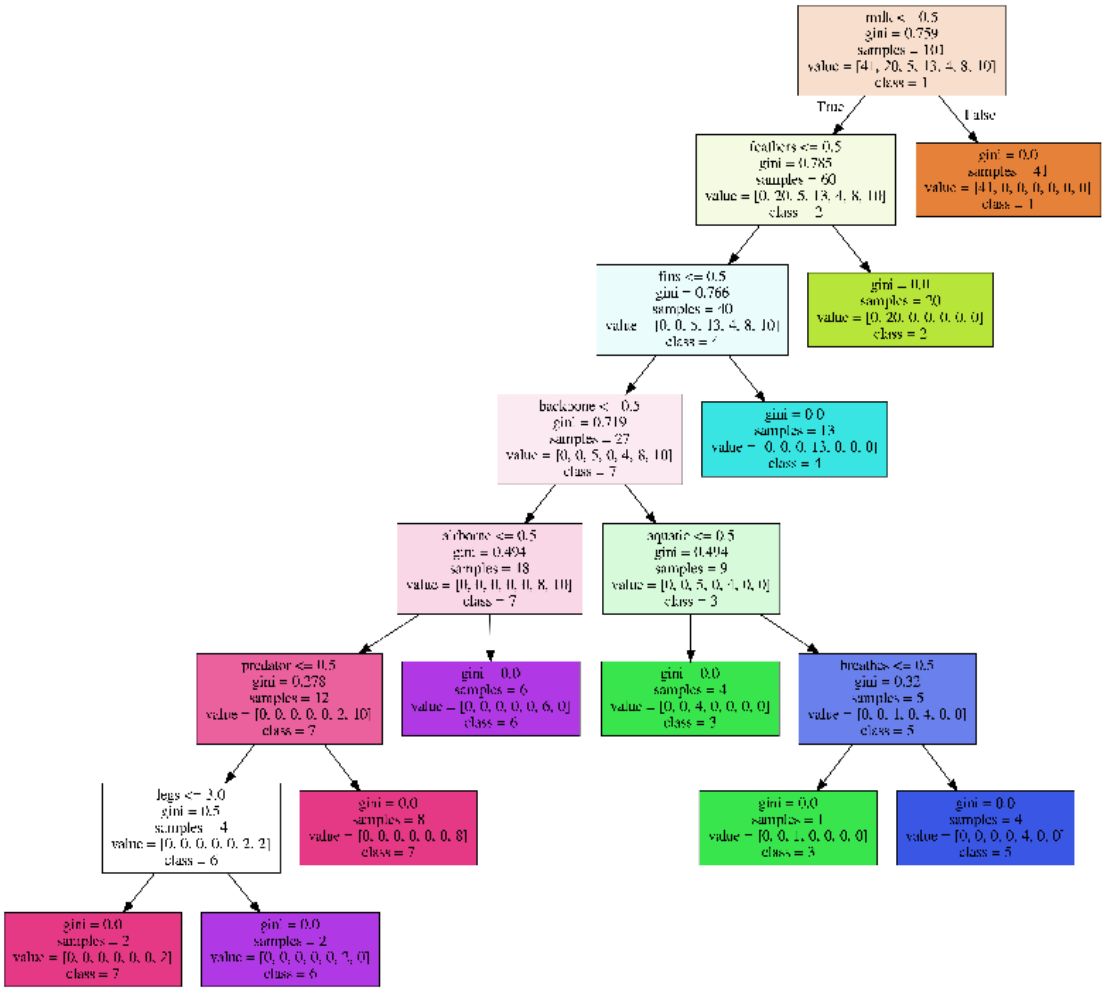
[16]:
```
clf_gini1 = DecisionTreeClassifier(criterion='gini')
clf_gini1.fit(df,target)
```

[16]:
```
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')
```

[17]:
```
predictions = clf_gini1.
 ↪predict([[0,0,0,0,0,0,0,1,1,0,0,0,1,1,0,0],[0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0]])
print('Predictions: ',predictions)
```

```
Predictions:  [3 7]
```

[18]:
```
export_graphviz(
    clf_gini1,out_file='zoo_tree_gini.dot',
    feature_names=df.columns,
    class_names=['1','2','3','4','5','6','7'],
    filled=True)
call(['dot', '-Tpng', 'zoo_tree_gini.dot', '-o', 'zoo_tree_gini.png',␣
 ↪'-Gdpi=600'])
plt.figure(figsize = (14, 18))
plt.imshow(plt.imread('zoo_tree_gini.png'))
plt.axis('off');
plt.show();
```


milk <= 0.5
gini = 0.759
samples = 101
value = [41, 20, 5, 13, 4, 8, 10]
class = 1
True
False
feathers <= 0.5
gini = 0.785
samples = 60
value = [0, 20, 5, 13, 4, 8, 10]
class = 2
gini = 0.0
samples = 41
value = [41, 0, 0, 0, 0, 0, 0]
class = 1
fins <= 0.5
gini = 0.766
samples = 40
value = [0, 0, 5, 13, 4, 8, 10]
class = 7
gini = 0.0
samples = 20
value = [0, 20, 0, 0, 0, 0, 0]
class = 2
backbone <= 0.5
gini = 0.719
samples = 27
value = [0, 0, 5, 0, 4, 8, 10]
class = 7
gini = 0.0
samples = 13
value = [0, 0, 0, 13, 0, 0, 0]
class = 4
airborne <= 0.5
gini = 0.494
samples = 18
value = [0, 0, 0, 0, 0, 8, 10]
class = 7
aquatic <= 0.5
gini = 0.494
samples = 9
value = [0, 0, 5, 0, 4, 0, 0]
class = 3
predator <= 0.5
gini = 0.278
samples = 12
value = [0, 0, 0, 0, 0, 2, 10]
class = 7
gini = 0.0
samples = 6
value = [0, 0, 0, 0, 0, 6, 0]
class = 6
gini = 0.0
samples = 4
value = [0, 0, 4, 0, 0, 0, 0]
class = 3
breathes <= 0.5
gini = 0.32
samples = 5
value = [0, 0, 1, 0, 4, 0, 0]
class = 5
legs <= 3.0
gini = 0.5
samples = 4
value = [0, 0, 0, 0, 0, 2, 2]
class = 6
gini = 0.0
samples = 8
value = [0, 0, 0, 0, 0, 0, 8]
class = 7
gini = 0.0
samples = 1
value = [0, 0, 1, 0, 0, 0, 0]
class = 3
gini = 0.0
samples = 4
value = [0, 0, 0, 0, 4, 0, 0]
class = 5
gini = 0.0
samples = 2
value = [0, 0, 0, 0, 0, 0, 2]
class = 7
gini = 0.0
samples = 2
value = [0, 0, 0, 0, 0, 2, 0]
class = 6