Mushroom Classification

Aarsh Chaube

RA1911027010105

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

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

In [3]:
```
df.head()
```

Out[3]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color
0	p	x	s	n	t	p	f	c	n	k	...	s	w	w	p	w
1	e	x	s	y	t	a	f	c	b	k	...	s	w	w	p	w
2	e	b	s	w	t	l	f	c	b	n	...	s	w	w	p	w
3	p	x	y	w	t	p	f	c	n	n	...	s	w	w	p	w
4	e	x	s	g	f	n	f	w	b	k	...	s	w	w	p	w

5 rows × 23 columns

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

Out[4]:
```
(8124, 23)
```

In [5]:
```
df.dtypes
```

Out[5]:
```
class                       object
cap-shape                   object
cap-surface                 object
cap-color                   object
bruises                     object
odor                        object
gill-attachment             object
gill-spacing                object
gill-size                   object
gill-color                  object
stalk-shape                 object
stalk-root                  object
stalk-surface-above-ring    object
stalk-surface-below-ring    object
stalk-color-above-ring      object
stalk-color-below-ring      object
veil-type                   object
veil-color                  object
ring-number                 object
ring-type                   object
spore-print-color           object
population                  object
habitat                     object
dtype: object
```

In [6]:
```
#inspecting unique values
for column in df.columns:
    print('Unique value in',column+":",df[column].unique())
```

```
Unique value in class: ['p' 'e']
Unique value in cap-shape: ['x' 'b' 's' 'f' 'k' 'c']
Unique value in cap-surface: ['s' 'y' 'f' 'g']
Unique value in cap-color: ['n' 'y' 'w' 'g' 'e' 'p' 'b' 'u' 'c' 'r']
Unique value in bruises: ['t' 'f']
Unique value in odor: ['p' 'a' 'l' 'n' 'f' 'c' 'y' 's' 'm']
Unique value in gill-attachment: ['f' 'a']
Unique value in gill-spacing: ['c' 'w']
Unique value in gill-size: ['n' 'b']
Unique value in gill-color: ['k' 'n' 'g' 'p' 'w' 'h' 'u' 'e' 'b' 'r' 'y' 'o']
Unique value in stalk-shape: ['e' 't']
Unique value in stalk-root: ['e' 'c' 'b' 'r' '?']
Unique value in stalk-surface-above-ring: ['s' 'f' 'k' 'y']
Unique value in stalk-surface-below-ring: ['s' 'f' 'y' 'k']
Unique value in stalk-color-above-ring: ['w' 'g' 'p' 'n' 'b' 'e' 'o' 'c' 'y']
Unique value in stalk-color-below-ring: ['w' 'p' 'g' 'b' 'n' 'e' 'y' 'o' 'c']
Unique value in veil-type: ['p']
Unique value in veil-color: ['w' 'n' 'o' 'y']
Unique value in ring-number: ['o' 't' 'n']
Unique value in ring-type: ['p' 'e' 'l' 'f' 'n']
Unique value in spore-print-color: ['k' 'n' 'u' 'h' 'w' 'r' 'o' 'y' 'b']
Unique value in population: ['s' 'n' 'a' 'v' 'y' 'c']
Unique value in habitat: ['u' 'g' 'm' 'd' 'p' 'w' 'l']
```

since the above data is categorical we convert it into numerical data

One Hot encoding

In [7]:
```
y = df['class']

y[y=='p'] = 1  # p (poisonous) is encoded as 1
y[y=='e'] = 0  # e (edible) is encoded as 0

y = y.astype(np.uint8) # store as unsigned 8bit integer
```

In [8]:
```
X = df.drop('class', axis=1)
X.columns
```

Out[8]:
```
Index(['cap-shape', 'cap-surface', 'cap-color', 'bruises', 'odor',
       'gill-attachment', 'gill-spacing', 'gill-size', 'gill-color',
       'stalk-shape', 'stalk-root', 'stalk-surface-above-ring',
       'stalk-surface-below-ring', 'stalk-color-above-ring',
       'stalk-color-below-ring', 'veil-type', 'veil-color', 'ring-number',
       'ring-type', 'spore-print-color', 'population', 'habitat'],
      dtype='object')
```

In [9]:
```
# one hot encode X features , each attribute will get converted into a series of 0s and 1s

X_encoded = pd.get_dummies(X)
X_encoded.columns
```

Out[9]:
```
Index(['cap-shape_b', 'cap-shape_c', 'cap-shape_f', 'cap-shape_k',
       'cap-shape_s', 'cap-shape_x', 'cap-surface_f', 'cap-surface_g',
       'cap-surface_s', 'cap-surface_y',
       ...
       'population_s', 'population_v', 'population_y', 'habitat_d',
       'habitat_g', 'habitat_l', 'habitat_m', 'habitat_p', 'habitat_u',
       'habitat_w'],
      dtype='object', length=117)
```

In [10]:
```
X_encoded.head(5)
```

Out[10]:

	cap-shape_b	cap-shape_c	cap-shape_f	cap-shape_k	cap-shape_s	cap-shape_x	cap-surface_f	cap-surface_g	cap-surface_s	cap-surface_y	...	population_s	p
0	0	0	0	0	0	1	0	0	1	0	...	1	
1	0	0	0	0	0	1	0	0	1	0	...	0	
2	1	0	0	0	0	0	0	0	1	0	...	0	
3	0	0	0	0	0	1	0	0	0	1	...	1	
4	0	0	0	0	0	1	0	0	1	0	...	0	

5 rows × 117 columns

Building a Decision Tree Classifier

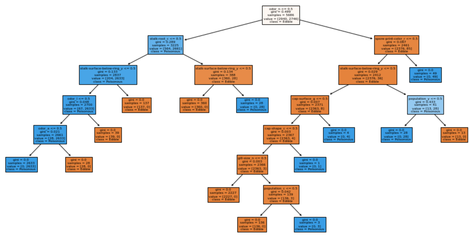
In [18]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
```

In [19]:
```
#splitting data into training and testing
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y,
                                                    test_size=0.3,
                                                    random_state=4)
```

In [20]:
```
dt = DecisionTreeClassifier()
dt = dt.fit(X_train, y_train)
```

In [21]:
```
# Visualize decision tree

plt.figure(figsize=(16,8))
plot_tree(dt,
          filled=True,
          class_names=['Edible', 'Poisonous'],
          feature_names=X_encoded.columns);
```



Doing a 5 fold cross validation

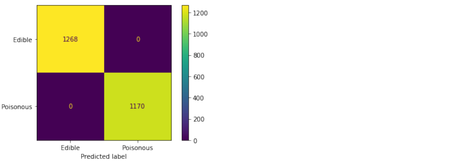
In [22]:
```
scores = cross_val_score(dt, X_train, y_train, cv=5)

for score in scores:
    print('Accuracy: ', round(score,4))
```

```
Accuracy:  1.0
Accuracy:  1.0
Accuracy:  1.0
Accuracy:  1.0
Accuracy:  1.0
```

Plotting confusion matrix

In [23]:
```
#model performance on test set
import seaborn as sns
plot_confusion_matrix(dt, X_test, y_test, display_labels=['Edible', 'Poisonous']);
```



In [ ]:
```
```