21MIS1152 Rajeev Sekar

Loading the dataset

```
import numpy as np
import pandas as pd
data = pd.read_csv("mushroom_csv.csv")

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8124 entries, 0 to 8123
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   cap-shape                 8124 non-null   object
 1   cap-surface               8124 non-null   object
 2   cap-color                 8124 non-null   object
 3   bruises%3F                8124 non-null   object
 4   odor                      8124 non-null   object
 5   gill-attachment           8124 non-null   object
 6   gill-spacing              8124 non-null   object
 7   gill-size                 8124 non-null   object
 8   gill-color                8124 non-null   object
 9   stalk-shape               8124 non-null   object
 10  stalk-root                5644 non-null   object
 11  stalk-surface-above-ring  8124 non-null   object
 12  stalk-surface-below-ring  8124 non-null   object
 13  stalk-color-above-ring    8124 non-null   object
 14  stalk-color-below-ring    8124 non-null   object
 15  veil-type                 8124 non-null   object
 16  veil-color                8124 non-null   object
 17  ring-number               8124 non-null   object
 18  ring-type                 8124 non-null   object
 19  spore-print-color         8124 non-null   object
 20  population                8124 non-null   object
 21  habitat                   8124 non-null   object
 22  class                     8124 non-null   object
dtypes: object(23)
memory usage: 1.4+ MB
```

All the features are catagorical and the target variable has 2 value: poisonous or edible

DATA PREPROCESSING

1. Separating features and target cols

```
X=data.drop('class',axis=1)
y=data['class']
X.head()
```

```
  cap-shape cap-surface cap-color bruises%3F odor gill-attachment  \
0         x           s         n          t    p               f
1         x           s         y          t    a               f
2         b           s         w          t    l               f
3         x           y         w          t    p               f
4         x           s         g          f    n               f

  gill-spacing gill-size gill-color stalk-shape  ... stalk-surface-
below-ring  \
0            c         n          k           e  ...
s
1            c         b          k           e  ...
s
2            c         b          n           e  ...
s
3            c         n          n           e  ...
s
4            w         b          k           t  ...
s

  stalk-color-above-ring stalk-color-below-ring veil-type veil-
color  \
0                      w                      w         p          w

1                      w                      w         p          w

2                      w                      w         p          w

3                      w                      w         p          w

4                      w                      w         p          w


  ring-number ring-type spore-print-color population habitat
0           o         p                 k          s       u
1           o         p                 n          n       g
2           o         p                 n          n       m
3           o         p                 k          s       u
4           o         e                 n          a       g

[5 rows x 22 columns]
```

Handling null values

```
data = data.replace('?', np.NaN)
data.isnull().sum()

cap-shape                    0
cap-surface                  0
cap-color                    0
```

```
bruises%3F                   0
odor                         0
gill-attachment              0
gill-spacing                 0
gill-size                    0
gill-color                   0
stalk-shape                  0
stalk-root                2480
stalk-surface-above-ring     0
stalk-surface-below-ring     0
stalk-color-above-ring       0
stalk-color-below-ring       0
veil-type                    0
veil-color                   0
ring-number                  0
ring-type                    0
spore-print-color            0
population                   0
habitat                      0
class                        0
dtype: int64
```

```
data["stalk-root"].fillna(data["stalk-root"].mode()[0],inplace = True)
```

```
data.isnull().sum()
```

```
cap-shape                  0
cap-surface                0
cap-color                  0
bruises%3F                 0
odor                       0
gill-attachment            0
gill-spacing               0
gill-size                  0
gill-color                 0
stalk-shape                0
stalk-root                 0
stalk-surface-above-ring   0
stalk-surface-below-ring   0
stalk-color-above-ring     0
stalk-color-below-ring     0
veil-type                  0
veil-color                 0
ring-number                0
ring-type                  0
spore-print-color          0
population                 0
habitat                    0
class                      0
dtype: int64
```

Label encoding catagorical data

```
from sklearn.preprocessing import LabelEncoder
Encoder_X = LabelEncoder()
for col in X.columns:
    X[col] = Encoder_X.fit_transform(X[col])
Encoder_y=LabelEncoder()
y = Encoder_y.fit_transform(y)
```

Splitting datset into test and train sets

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=1)
```

MODEL TRAINING: Decision tree: id3 algorithm (entropy) with tree depth of 6

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree


clf = DecisionTreeClassifier(criterion='entropy', max_depth=6,
random_state=0)
clf.fit(X_train, y_train)
```

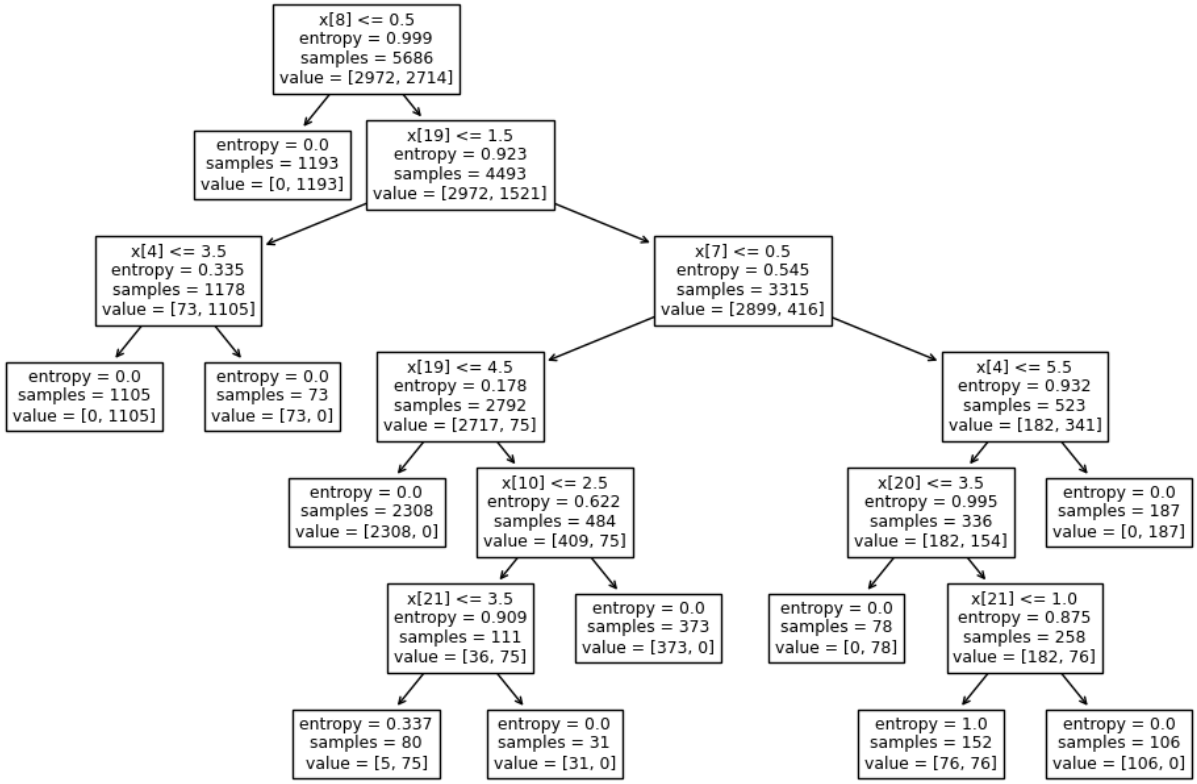
```
DecisionTreeClassifier(criterion='entropy', max_depth=6,
random_state=0)
```

Visualising the tree

```
plt.figure(figsize=(12,8))
tree.plot_tree(clf.fit(X_train, y_train))
```

```
[Text(0.3076923076923077, 0.9285714285714286, 'x[8] <= 0.5\nentropy =
0.999\nsamples = 5686\nvalue = [2972, 2714]'),
 Text(0.23076923076923078, 0.7857142857142857, 'entropy = 0.0\nsamples
= 1193\nvalue = [0, 1193]'),
 Text(0.38461538461538464, 0.7857142857142857, 'x[19] <= 1.5\nentropy
= 0.923\nsamples = 4493\nvalue = [2972, 1521]'),
 Text(0.15384615384615385, 0.6428571428571429, 'x[4] <= 3.5\nentropy =
0.335\nsamples = 1178\nvalue = [73, 1105]'),
 Text(0.07692307692307693, 0.5, 'entropy = 0.0\nsamples = 1105\nvalue
= [0, 1105]'),
 Text(0.23076923076923078, 0.5, 'entropy = 0.0\nsamples = 73\nvalue =
[73, 0]'),
 Text(0.6153846153846154, 0.6428571428571429, 'x[7] <= 0.5\nentropy =
0.545\nsamples = 3315\nvalue = [2899, 416]'),
 Text(0.38461538461538464, 0.5, 'x[19] <= 4.5\nentropy = 0.178\
```

```
nsamples = 2792\nvalue = [2717, 75]'),
 Text(0.3076923076923077, 0.35714285714285715, 'entropy = 0.0\nsamples
= 2308\nvalue = [2308, 0]'),
 Text(0.46153846153846156, 0.35714285714285715, 'x[10] <= 2.5\nentropy
= 0.622\nsamples = 484\nvalue = [409, 75]'),
 Text(0.38461538461538464, 0.21428571428571427, 'x[21] <= 3.5\nentropy
= 0.909\nsamples = 111\nvalue = [36, 75]'),
 Text(0.3076923076923077, 0.07142857142857142, 'entropy = 0.337\
nsamples = 80\nvalue = [5, 75]'),
 Text(0.46153846153846156, 0.07142857142857142, 'entropy = 0.0\
nsamples = 31\nvalue = [31, 0]'),
 Text(0.5384615384615384, 0.21428571428571427, 'entropy = 0.0\nsamples
= 373\nvalue = [373, 0]'),
 Text(0.8461538461538461, 0.5, 'x[4] <= 5.5\nentropy = 0.932\nsamples
= 523\nvalue = [182, 341]'),
 Text(0.7692307692307693, 0.35714285714285715, 'x[20] <= 3.5\nentropy
= 0.995\nsamples = 336\nvalue = [182, 154]'),
 Text(0.6923076923076923, 0.21428571428571427, 'entropy = 0.0\nsamples
= 78\nvalue = [0, 78]'),
 Text(0.8461538461538461, 0.21428571428571427, 'x[21] <= 1.0\nentropy
= 0.875\nsamples = 258\nvalue = [182, 76]'),
 Text(0.7692307692307693, 0.07142857142857142, 'entropy = 1.0\nsamples
= 152\nvalue = [76, 76]'),
 Text(0.9230769230769231, 0.07142857142857142, 'entropy = 0.0\nsamples
= 106\nvalue = [106, 0]'),
 Text(0.9230769230769231, 0.35714285714285715, 'entropy = 0.0\nsamples
= 187\nvalue = [0, 187]')]
```



MODEL EVALUATION:

checking training and test set score to ensure there is no overfitting, because DT tend to overfit sometimes

```
print('Training set score: {:.4f}'.format(clf.score(X_train,
y_train)))
print('Test set score: {:.4f}'.format(clf.score(X_test, y_test)))
```

```
Training set score: 0.9858
Test set score: 0.9774
```

=> model has not overfit

Classification report:

```
from sklearn.metrics import classification_report
y_pred=clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98      1236
           1       1.00      0.96      0.98      1202
```

```
    accuracy                           0.98      2438
   macro avg       0.98      0.98      0.98      2438
weighted avg       0.98      0.98      0.98      2438
```

MODEL INTERPRETATION:

=> only 96% of the samples predicted as poisonous actually are poisonous (precision of class 0 = 0.96)

=> only 96% of the samples actually belonging in edible class have been predicted correctly (recall of class 1 = 0.96)

=> In 98% of the cases, the model has predicted if the mushroom is edible or poisonous correctly