day49-decision-tree-implementation

November 21, 2023

Day49 Decision Tree Implementation By: Loga Aswin

[52]:
```
#import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn  as sns

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report, f1_score
```

[2]:
```
#load data
df = pd.read_csv('/content/mushrooms.csv')
```

Exploratory Data Analysis(EDA):

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

[3]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	…	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color
0	p	x	s	n	t	p	f	…	s	w	w	p	w
1	e	x	s	y	t	a	f	…	s	w	w	p	w
2	e	b	s	w	t	l	f	…	s	w	w	p	w
3	p	x	y	w	t	p	f	…	s	w	w	p	w
4	e	x	s	g	f	n	f	…	s	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 23 columns]
```

[5]:
```
df.isnull().sum()
```

[5]:
```
class                       0
cap-shape                   0
cap-surface                 0
cap-color                   0
bruises                     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
dtype: int64
```

[6]:
```
df['class'].unique()
```

[6]:
```
array(['p', 'e'], dtype=object)
```

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

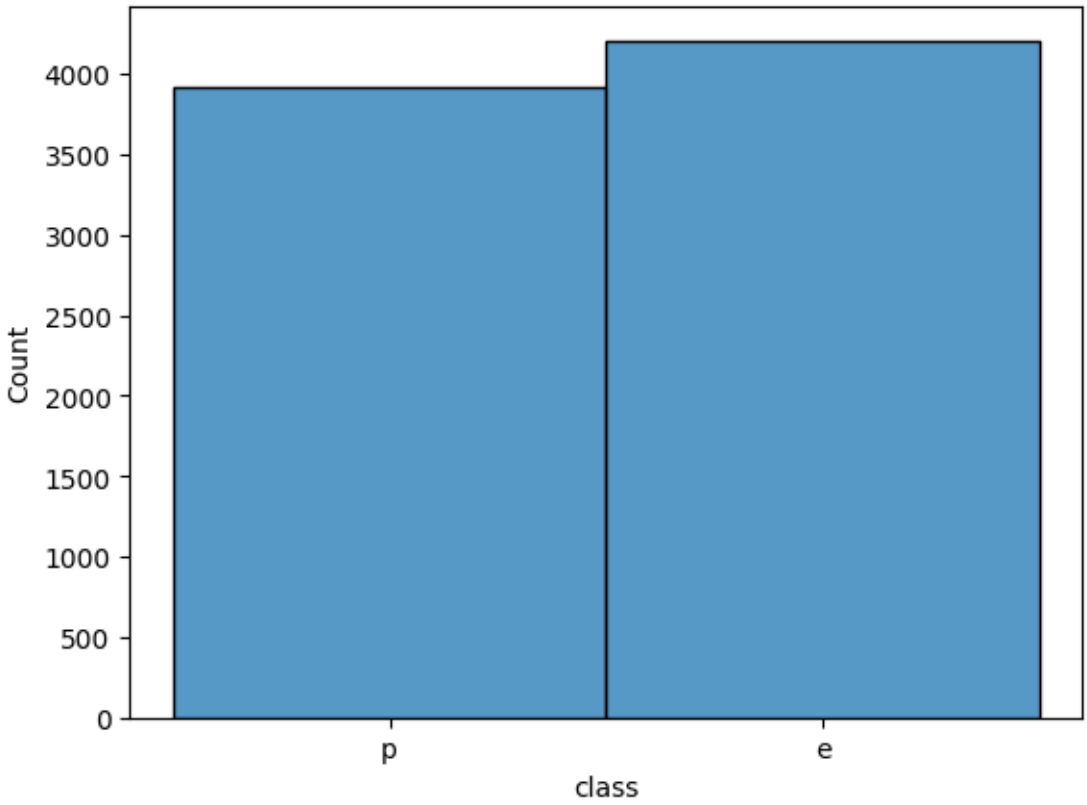
```
RangeIndex: 8124 entries, 0 to 8123
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   class                     8124 non-null   object
 1   cap-shape                 8124 non-null   object
 2   cap-surface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

[8]:
```
df.shape
```

[8]:
```
(8124, 23)
```

[9]:
```
sns.histplot(df['class'])
```

[9]:
```
<Axes: xlabel='class', ylabel='Count'>
```



Seprating Features and Targets:

[11]:
```
X = df.drop('class',axis=1)
y = df['class']
```

[12]:
```
X = pd.get_dummies(X)
X.head()
```

[12]:

	cap-shape_b	cap-shape_c	cap-shape_f	cap-shape_k	cap-shape_s	\
0	0	0	0	0	0	
1	0	0	0	0	0	
2	1	0	0	0	0	
3	0	0	0	0	0	
4	0	0	0	0	0	

	cap-shape_x	cap-surface_f	cap-surface_g	cap-surface_s	cap-surface_y	\
0	1	0	0	1	0	
1	1	0	0	1	0	
2	0	0	0	1	0	
3	1	0	0	0	1	
4	1	0	0	1	0	

```
   …  population_s  population_v  population_y  habitat_d  habitat_g  \
0  …             1             0             0          0          0
1  …             0             0             0          0          1
2  …             0             0             0          0          0
3  …             1             0             0          0          0
4  …             0             0             0          0          1

   habitat_l  habitat_m  habitat_p  habitat_u  habitat_w
0          0          0          0          1          0
1          0          0          0          0          0
2          0          1          0          0          0
3          0          0          0          1          0
4          0          0          0          0          0

[5 rows x 117 columns]
```

Label Encoding

[13]:
```
encoder = LabelEncoder()
y = encoder.fit_transform(y)
print(y)
```

```
[1 0 0 … 0 1 0]
```

Splitting into training and testing:

[14]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=1)
```

Creating Decision Tree using entropy:

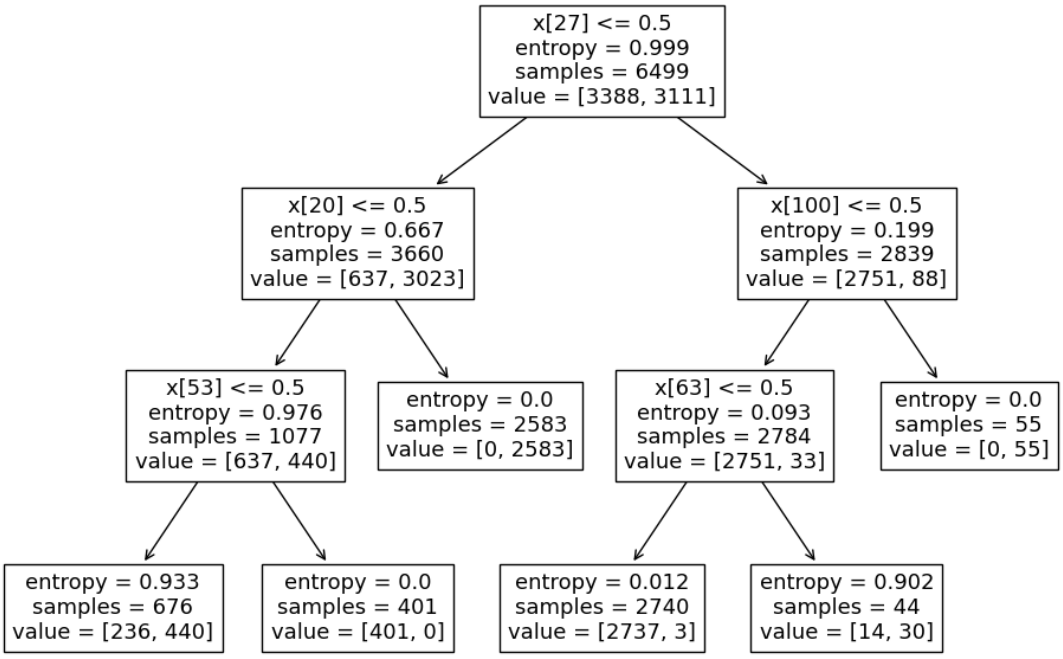
[18]:
```
clf = DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)

clf.fit(X_train, y_train)
```

[18]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)
```

[19]:
```
plt.figure(figsize=(12,8))
tree.plot_tree(clf_en.fit(X_train, y_train))
```

[19]:
```
[Text(0.5555555555555556, 0.875, 'x[27] <= 0.5\nentropy = 0.999\nsamples =
6499\nvalue = [3388, 3111]'),
 Text(0.3333333333333333, 0.625, 'x[20] <= 0.5\nentropy = 0.667\nsamples =
3660\nvalue = [637, 3023]'),
 Text(0.2222222222222222, 0.375, 'x[53] <= 0.5\nentropy = 0.976\nsamples =
1077\nvalue = [637, 440]'),
 Text(0.1111111111111111, 0.125, 'entropy = 0.933\nsamples = 676\nvalue = [236,
```

```
440]'),
 Text(0.3333333333333333, 0.125, 'entropy = 0.0\nsamples = 401\nvalue = [401,
0]'),
 Text(0.4444444444444444, 0.375, 'entropy = 0.0\nsamples = 2583\nvalue = [0,
2583]'),
 Text(0.7777777777777778, 0.625, 'x[100] <= 0.5\nentropy = 0.199\nsamples =
2839\nvalue = [2751, 88]'),
 Text(0.6666666666666666, 0.375, 'x[63] <= 0.5\nentropy = 0.093\nsamples =
2784\nvalue = [2751, 33]'),
 Text(0.5555555555555556, 0.125, 'entropy = 0.012\nsamples = 2740\nvalue =
[2737, 3]'),
 Text(0.7777777777777778, 0.125, 'entropy = 0.902\nsamples = 44\nvalue = [14,
30]'),
 Text(0.8888888888888888, 0.375, 'entropy = 0.0\nsamples = 55\nvalue = [0,
55]')]
```



[23]:
```
#Predict values
y_pred = clf.predict(X_test)
```

[24]:
```
#Predict values using x_train
y_pred_train = clf.predict(X_train)
```

Calculating accuracy_score from scikit_learn

[60]:
```
print('criterion entropy accuracy: {0:.2f}'. format(accuracy_score(y_test,␣
 ↪y_pred)*100))
print('Training set: {0:.2f}'. format(accuracy_score(y_train,␣
 ↪y_pred_train)*100))
```

```
criterion entropy accuracy: 96.37
Training set: 96.11
```

Calculating accuracy_score from model of the classifier

[61]:
```
print('Training set score: {0:.2f}'.format(clf_en.score(X_train, y_train)*100))
print('Test set score: {0:.2f}'.format(clf_en.score(X_test, y_test)*100))
```

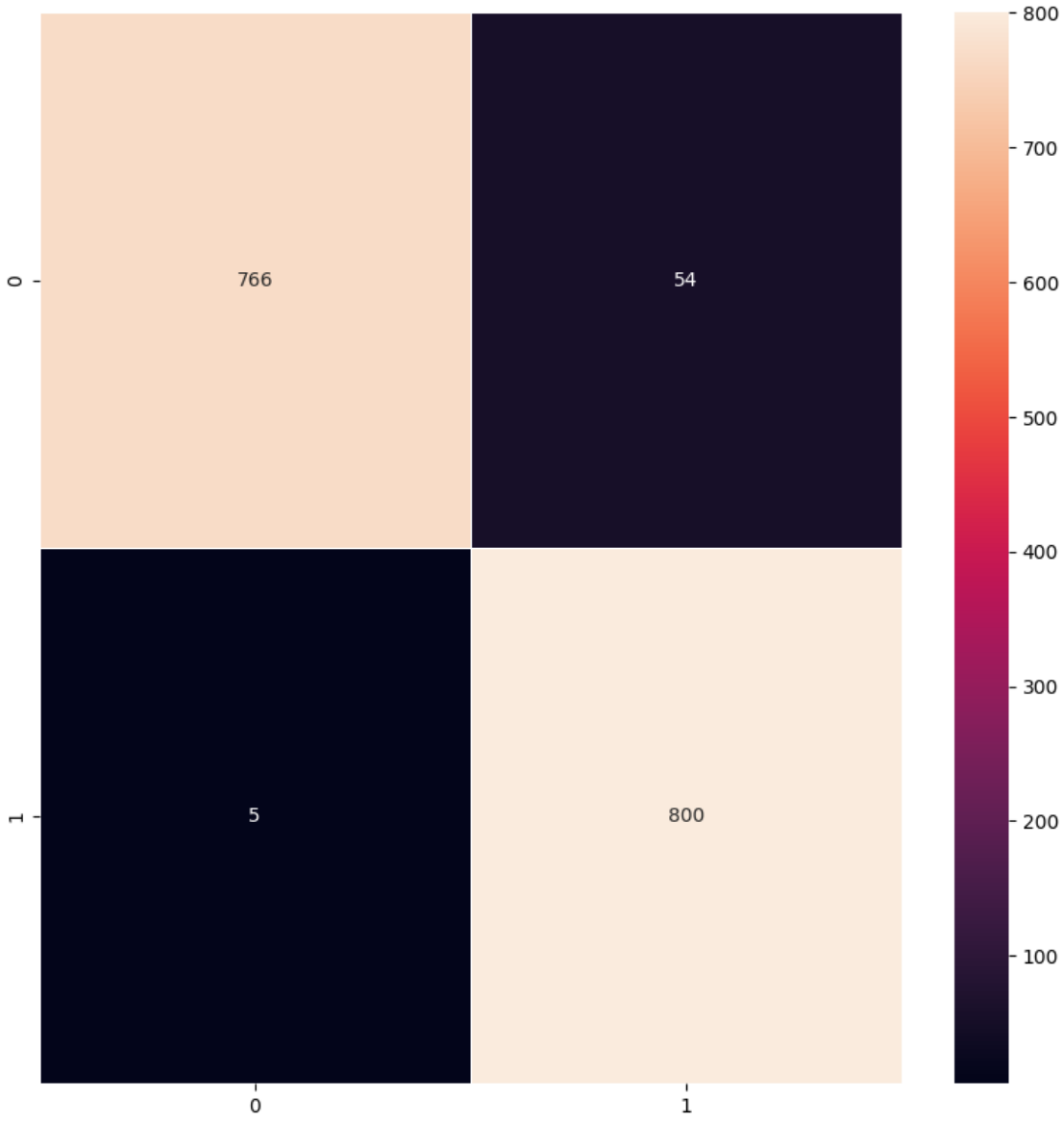
```
Training set score: 96.11
Test set score: 96.37
```

[51]:
```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[766  54]
 [  5 800]]
```

Confusion Matrix:

[48]:
```
plt.subplots(figsize=(10, 10))
sns.heatmap(cm, annot=True, linewidths=0.5,fmt= '.0f')
plt.show()
```



[49]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.99      0.93      0.96       820
           1       0.94      0.99      0.96       805

    accuracy                           0.96      1625
   macro avg       0.97      0.96      0.96      1625
weighted avg       0.97      0.96      0.96      1625
```

[62]:
```
f1_score = f1_score(y_test, y_pred_en)
print(f1_score)
```

```
0.9644364074743822
```