DECISION TREE - MUSHROOM CLASSIFICATION

Attribute Information: (classes: edible=e, poisonous=p)

cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s

cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s

cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r,pink=p,purple=u,red=e,white=w,yellow=y

bruises: bruises=t,no=f

odor: almond=a,anise=l,creosote=c,fishy=y,foul=f,musty=m,none=n,pungent=p,spicy=s

gill-attachment: attached=a,descending=d,free=f,notched=n

gill-spacing: close=c,crowded=w,distant=d

gill-size: broad=b,narrow=n

gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g, green=r,orange=o,pink=p,purple=u,red=e,white=w,yellow=y

stalk-shape: enlarging=e,tapering=t

stalk-root: bulbous=b,club=c,cup=u,equal=e,rhizomorphs=z,rooted=r,missing=?

stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s

stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s

stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y

stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y

veil-type: partial=p,universal=u

veil-color: brown=n,orange=o,white=w,yellow=y

ring-number: none=n,one=o,two=t

ring-type: cobwebby=c,evanescent=e,flaring=f,large=l,none=n,pendant=p,sheathing=s,zone=z

spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r,orange=o,purple=u,white=w,yellow=y

population: abundant=a,clustered=c,numerous=n,scattered=s,several=v,solitary=y

habitat: grasses=g,leaves=l,meadows=m,paths=p,urban=u,waste=w,woods=d

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

In [2]:
```
df=pd.read_csv(r"C:\Users\LOKESH B S\Downloads\mushrooms.csv")
df
```

Out[2]:

	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
0	p	x	s	n	t	p	f	c	n	k	...	s	w
1	e	x	s	y	t	a	f	c	b	k	...	s	w
2	e	b	s	w	t	l	f	c	b	n	...	s	w
3	p	x	y	w	t	p	f	c	n	n	...	s	w
4	e	x	s	g	f	n	f	w	b	k	...	s	w
...	...	...	...	...	...	...	...	...	...	...	...	...	...
8119	e	k	s	n	f	n	a	c	b	y	...	s	o
8120	e	x	s	n	f	n	a	c	b	y	...	s	o
8121	e	f	s	n	f	n	a	c	b	n	...	s	o
8122	p	k	y	n	f	y	f	c	n	b	...	k	w
8123	e	x	s	n	f	n	a	c	b	y	...	s	o

8124 rows × 23 columns

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	stal colo belov rir
0	p	x	s	n	t	p	f	c	n	k	...	s	w	
1	e	x	s	y	t	a	f	c	b	k	...	s	w	
2	e	b	s	w	t	l	f	c	b	n	...	s	w	
3	p	x	y	w	t	p	f	c	n	n	...	s	w	
4	e	x	s	g	f	n	f	w	b	k	...	s	w	

5 rows × 23 columns

In [4]:
```
df.tail()
```

Out[4]:

	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
8119	e	k	s	n	f	n	a	c	b	y	...	s	o
8120	e	x	s	n	f	n	a	c	b	y	...	s	o
8121	e	f	s	n	f	n	a	c	b	n	...	s	o
8122	p	k	y	n	f	y	f	c	n	b	...	k	w
8123	e	x	s	n	f	n	a	c	b	y	...	s	o

5 rows × 23 columns

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

Out[5]:
```
0
```

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

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

In [7]:
```
df.dtypes
```

Out[7]:
```
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 [8]:
```
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
```

In [9]:
```
df.describe()
```

Out[9]:

	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
count	8124	8124	8124	8124	8124	8124	8124	8124	8124	8124	...	8124	8124
unique	2	6	4	10	2	9	2	2	2	12	...	4	9
top	e	x	y	n	f	n	f	c	b	b	...	s	w
freq	4208	3656	3244	2284	4748	3528	7914	6812	5612	1728	...	4936	4464

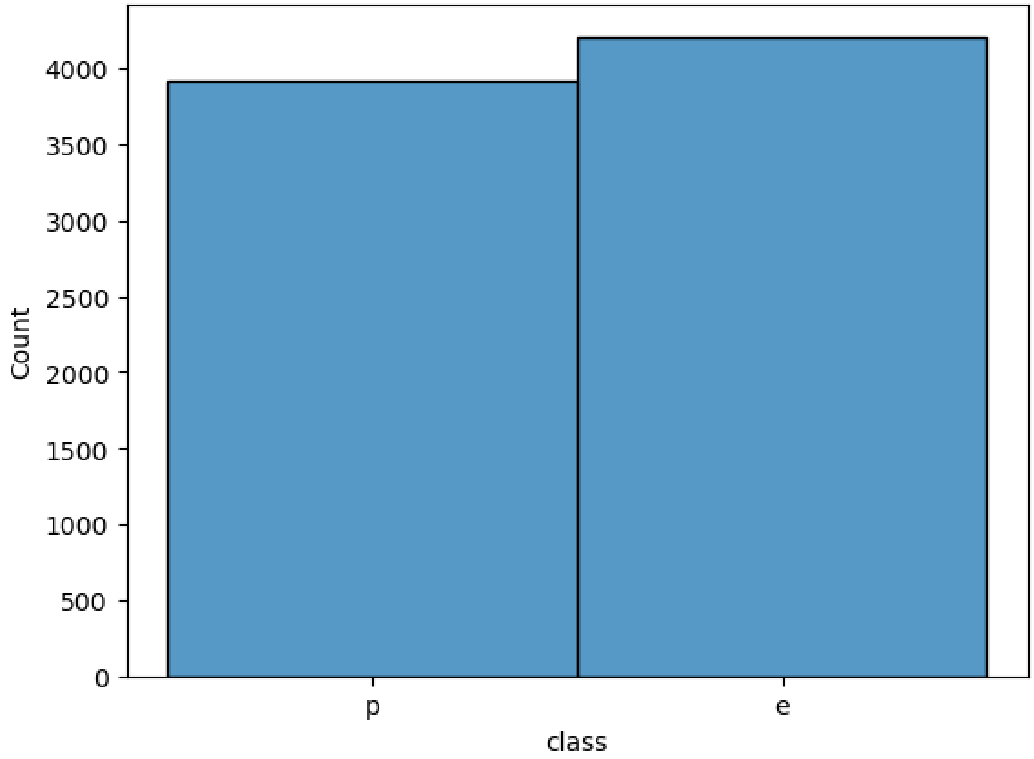
4 rows × 23 columns

In [10]:
```
df.shape
```

Out[10]: (8124, 23)

In [11]:
```
sns.histplot(df['class'])
```

Out[11]: <Axes: xlabel='class', ylabel='Count'>



In [12]:
```
x= df.drop(['class'], axis = 1)
x
```

Out[12]:

	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	...	stalk-surface-below-ring	stalk-color-above-ring
0	x	s	n	t	p	f	c	n	k	e	...	s	w
1	x	s	y	t	a	f	c	b	k	e	...	s	w
2	b	s	w	t	l	f	c	b	n	e	...	s	w
3	x	y	w	t	p	f	c	n	n	e	...	s	w
4	x	s	g	f	n	f	w	b	k	t	...	s	w
...	...	...	...	...	...	...	...	...	...	...	...	...	...
8119	k	s	n	f	n	a	c	b	y	e	...	s	o
8120	x	s	n	f	n	a	c	b	y	e	...	s	o
8121	f	s	n	f	n	a	c	b	n	e	...	s	o
8122	k	y	n	f	y	f	c	n	b	t	...	k	w
8123	x	s	n	f	n	a	c	b	y	e	...	s	o

8124 rows × 22 columns

In [13]:
```
y = df['class']
y
```

Out[13]:
```
0       p
1       e
2       e
3       p
4       e
       ..
8119    e
8120    e
8121    e
8122    p
8123    e
Name: class, Length: 8124, dtype: object
```

In [14]:
```
X = pd.get_dummies(x).astype(int)
X.head()
```

Out[14]:

	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	...
0	0	0	0	0	0	1	0	0	1	0	...
1	0	0	0	0	0	1	0	0	1	0	...
2	1	0	0	0	0	0	0	0	1	0	...
3	0	0	0	0	0	1	0	0	0	1	...
4	0	0	0	0	0	1	0	0	1	0	...

5 rows × 117 columns

In [15]:
```
X.head()
```

Out[15]:

	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	...
0	0	0	0	0	0	1	0	0	1	0	...
1	0	0	0	0	0	1	0	0	1	0	...
2	1	0	0	0	0	0	0	0	1	0	...
3	0	0	0	0	0	1	0	0	0	1	...
4	0	0	0	0	0	1	0	0	1	0	...

5 rows × 117 columns

In [16]:
```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
Y = encoder.fit_transform(y)
Y
```

Out[16]:
```
array([1, 0, 0, ..., 0, 1, 0])
```

In [17]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X,Y,  test_size=0.2, random_state=1
```

In [18]:
```
x_train.shape, x_test.shape
```

Out[18]:
```
((6499, 117), (1625, 117))
```

In [19]:
```
y_train.shape, y_test.shape
```
Out[19]:
```
((6499,), (1625,))
```

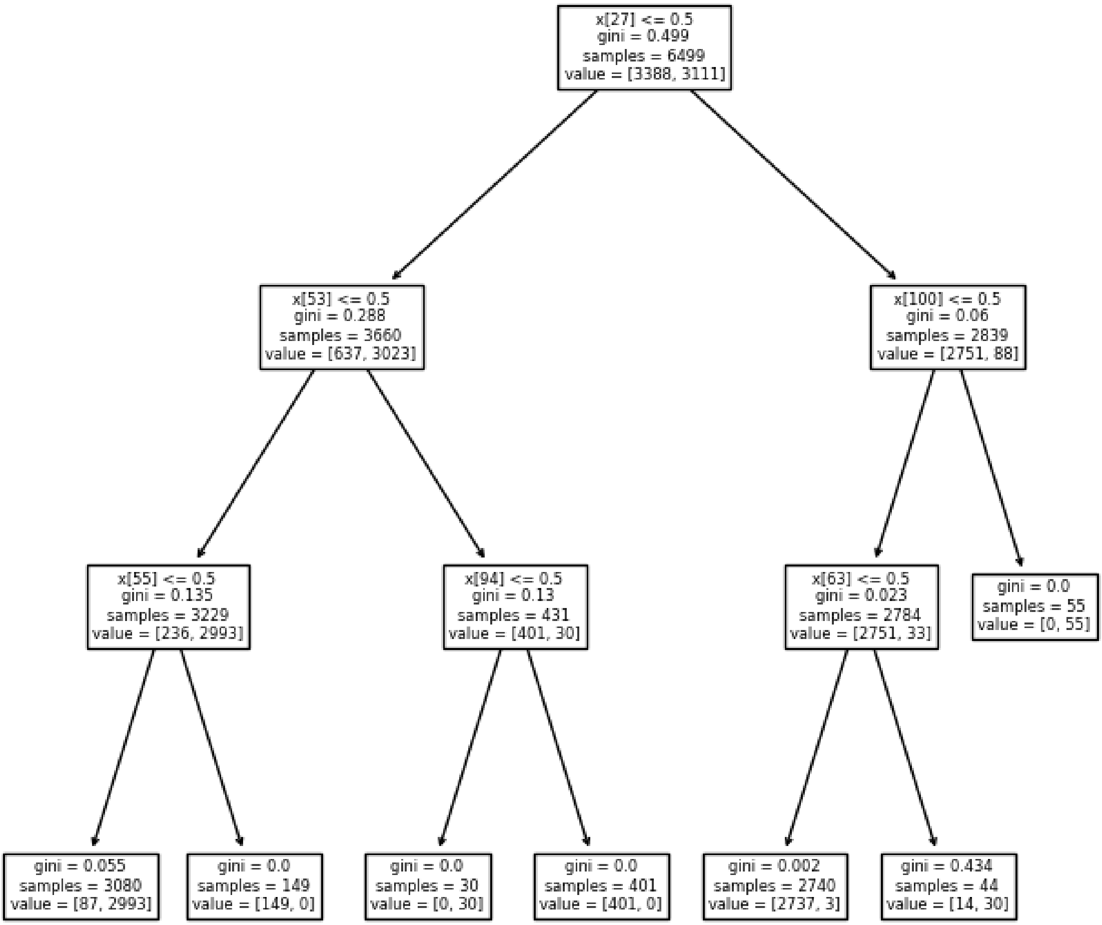
In [20]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.metrics import accuracy_score
```

Creating decision tree using GENIN INDEX

In [21]:
```
df_gini = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=0)
df_gini.fit(x_train, y_train)
```
Out[21]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, random_state=0)
```

In [22]:
```
plt.figure(figsize=(8,8))
tree.plot_tree(df_gini.fit(x_train, y_train))
```

Out[22]:
```
[Text(0.5769230769230769, 0.875, 'x[27] <= 0.5\ngini = 0.499\nsamples = 6499\nvalue = [3388, 3111]'),
 Text(0.3076923076923077, 0.625, 'x[53] <= 0.5\ngini = 0.288\nsamples = 3660\nvalue = [637, 3023]'),
 Text(0.15384615384615385, 0.375, 'x[55] <= 0.5\ngini = 0.135\nsamples = 3229\nvalue = [236, 2993]'),
 Text(0.07692307692307693, 0.125, 'gini = 0.055\nsamples = 3080\nvalue = [87, 2993]'),
 Text(0.23076923076923078, 0.125, 'gini = 0.0\nsamples = 149\nvalue = [149, 0]'),
 Text(0.46153846153846156, 0.375, 'x[94] <= 0.5\ngini = 0.13\nsamples = 431\nvalue = [401, 30]'),
 Text(0.38461538461538464, 0.125, 'gini = 0.0\nsamples = 30\nvalue = [0, 30]'),
 Text(0.5384615384615384, 0.125, 'gini = 0.0\nsamples = 401\nvalue = [401, 0]'),
 Text(0.8461538461538461, 0.625, 'x[100] <= 0.5\ngini = 0.06\nsamples = 2839\nvalue = [2751, 88]'),
 Text(0.7692307692307693, 0.375, 'x[63] <= 0.5\ngini = 0.023\nsamples = 2784\nvalue = [2751, 33]'),
 Text(0.6923076923076923, 0.125, 'gini = 0.002\nsamples = 2740\nvalue = [2737, 3]'),
 Text(0.8461538461538461, 0.125, 'gini = 0.434\nsamples = 44\nvalue = [14, 30]'),
 Text(0.9230769230769231, 0.375, 'gini = 0.0\nsamples = 55\nvalue = [0, 55]')]
```



In [23]:
```
y_pred_train_gini = df_gini.predict(x_train)
y_pred_train_gini
```

Out[23]:
```
array([0, 1, 1, ..., 1, 0, 0])
```

In [24]:
```
y_pred_test_gini = df_gini.predict(x_test)
y_pred_test_gini
```

Out[24]:
```
array([0, 0, 1, ..., 1, 1, 1])
```

In [26]:
```
print("accuracy score with criterian gini index is", accuracy_score(y_test, y_pred_test_
```

```
accuracy score with criterian gini index is 0.9901538461538462
```

In [27]:
```
print("accuracy score of training set is", accuracy_score(y_train, y_pred_train_gini))
```

```
accuracy score of training set is 0.983997538082782
```

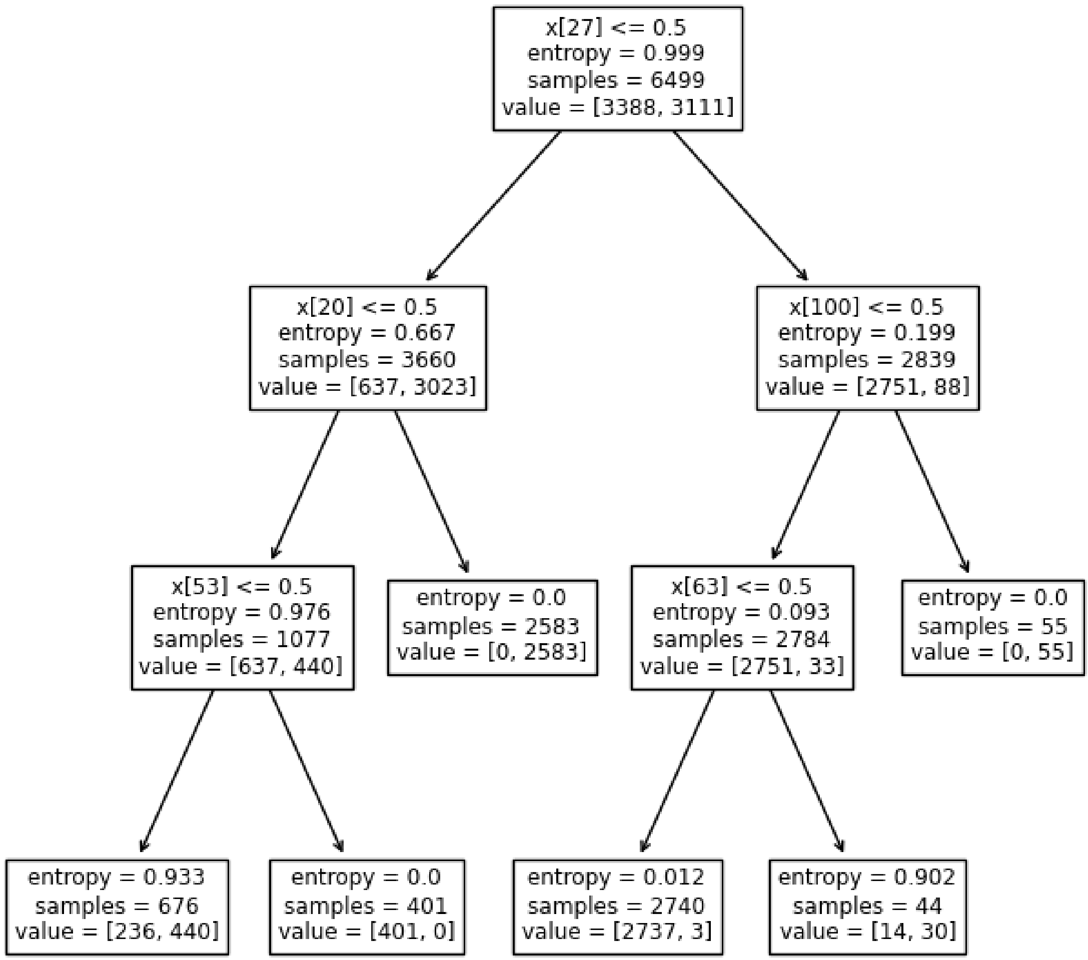
Creating decision tree using ENTROPY

In [28]:
```
df_ent = DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)
df_ent.fit(x_train, y_train)
```

Out[28]:
```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)
```

In [30]:
```
plt.figure(figsize=(8,8))
tree.plot_tree(df_ent.fit(x_train, y_train))
```

Out[30]:
```
[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, 44
0]'),
 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\nvalu
e = [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]')]
```



In [31]:
```
y_pred_test_ent = df_ent.predict(x_test)
y_pred_test_ent
```

Out[31]:
```
array([0, 1, 1, ..., 1, 0, 0])
```

In [32]:
```
y_pred_train_ent = df_ent.predict(x_train)
y_pred_train_ent
```

Out[32]:
```
array([0, 0, 1, ..., 1, 1, 1])
```

In [33]:
```
accuracy_score(y_test, y_pred_test_ent)
```

Out[33]:
```
0.9636923076923077
```

In [34]:
```
accuracy_score(y_train, y_pred_train_ent)
```

Out[34]:
```
0.9610709339898446
```

In [36]:
```
df_ent.score(x_test,y_test)
```

Out[36]:
```
0.9636923076923077
```

In [37]:
```
df_ent.score(x_train,y_train)
```

Out[37]:
```
0.9610709339898446
```

In [38]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import f1_score
```

In [39]:
```
conmat = confusion_matrix(y_test, y_pred_test_ent)
print(conmat)
```

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

In [42]:
```
print(classification_report(y_test, y_pred_test_ent))
```

```
              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
```

In [43]:
```
f1_score(y_test, y_pred_test_ent)
```

Out[43]:
```
0.9644364074743822
```