In [1]:

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import confusion_matrix, classification_report
```

In [2]:

```
df = pd.read_csv('heart_v2.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	age	sex	BP	cholestrol	heart disease
0	70	1	130	322	1
1	67	0	115	564	0
2	57	1	124	261	1
3	64	1	128	263	0
4	74	0	120	269	0

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 270 entries, 0 to 269
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   age            270 non-null    int64
 1   sex            270 non-null    int64
 2   BP             270 non-null    int64
 3   cholestrol     270 non-null    int64
 4   heart disease  270 non-null    int64
dtypes: int64(5)
memory usage: 10.7 KB
```

In [5]:

```
df.columns
```

Out[5]:

```
Index(['age', 'sex', 'BP', 'cholestrol', 'heart disease'], dtype='object')
```

In [6]:

```
df.describe()
```

Out[6]:

	age	sex	BP	cholestrol	heart disease
count	270.000000	270.000000	270.000000	270.000000	270.000000
mean	54.433333	0.677778	131.344444	249.659259	0.444444
std	9.109067	0.468195	17.861608	51.686237	0.497827
min	29.000000	0.000000	94.000000	126.000000	0.000000
25%	48.000000	0.000000	120.000000	213.000000	0.000000
50%	55.000000	1.000000	130.000000	245.000000	0.000000
75%	61.000000	1.000000	140.000000	280.000000	1.000000
max	77.000000	1.000000	200.000000	564.000000	1.000000

In [7]:

```
df.isnull().sum()
```

Out[7]:

```
age              0
sex              0
BP               0
cholestrol       0
heart disease    0
dtype: int64
```

In [8]:

```
df['heart disease'].value_counts()
```

Out[8]:

```
0    150
1    120
Name: heart disease, dtype: int64
```

In [9]:

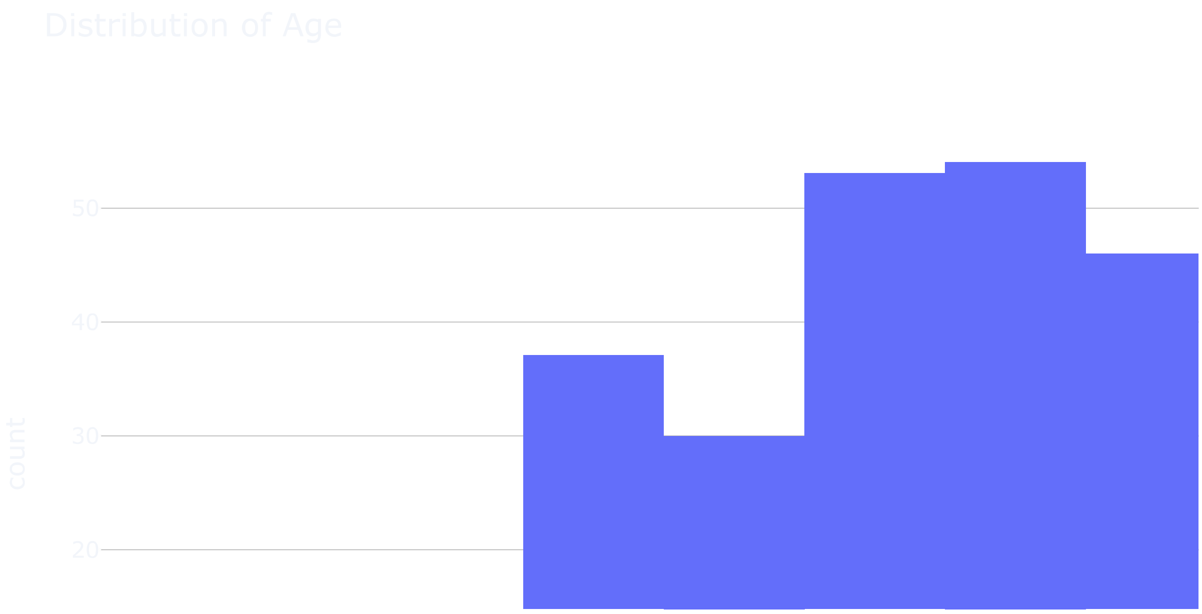
```
df.shape
```

Out[9]:

```
(270, 5)
```

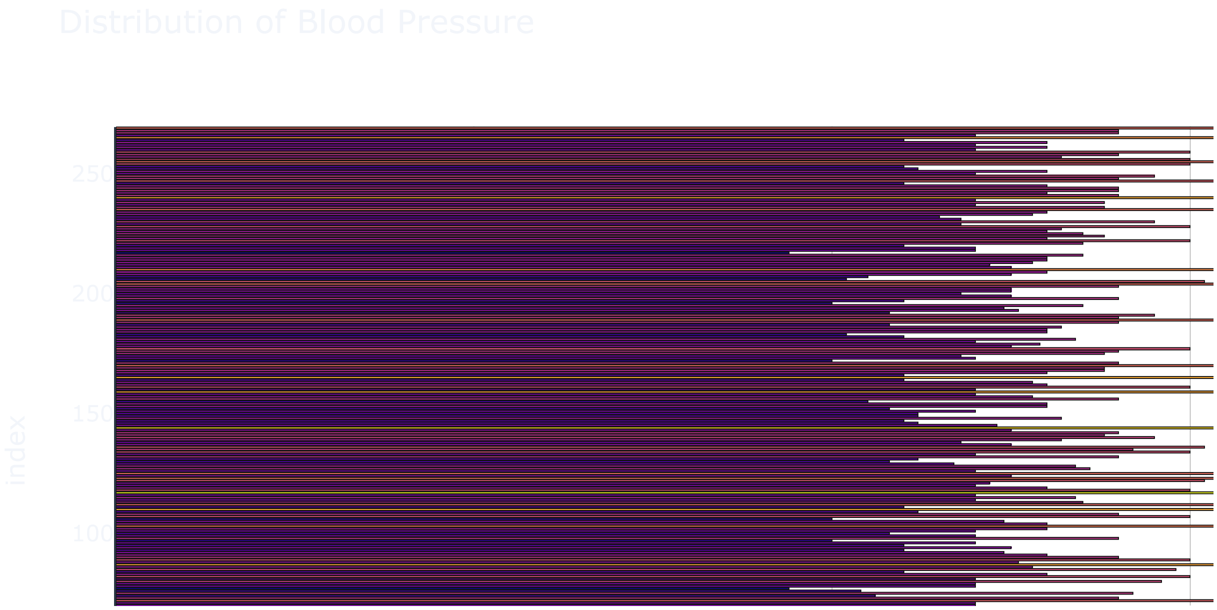
In [10]:

```
fig = px.histogram(df, x='age', nbins=20, title='Distribution of Age', labels={'age': '
fig.show()
```

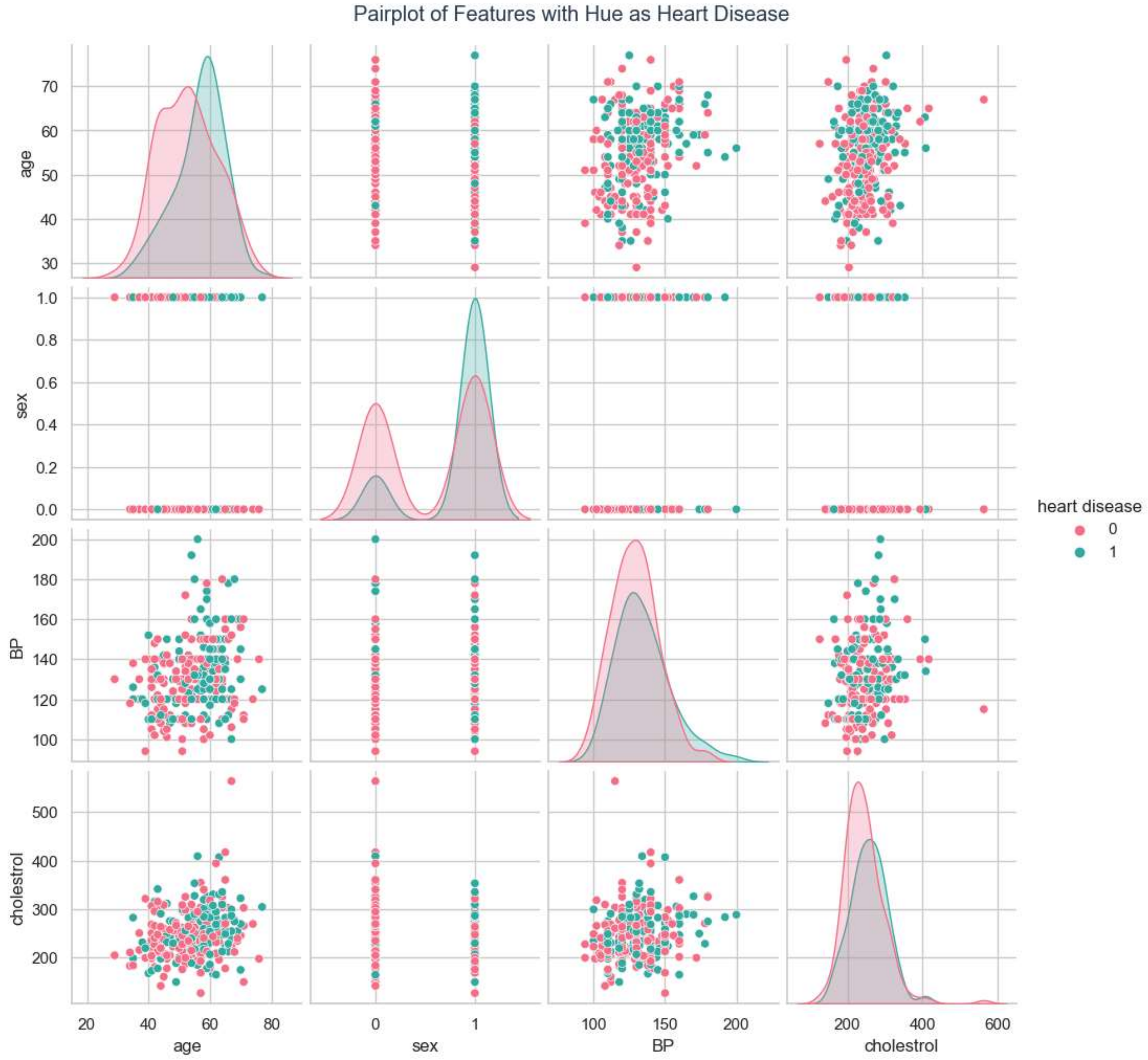


In [11]:

```
# Blood Pressure Distribution
fig = px.bar(df, x='BP', title='Distribution of Blood Pressure', color='BP', template='
fig.show()
```



In [12]:
```
sns.set_theme(style="whitegrid")
sns.pairplot(df, hue='heart disease', palette='husl')
plt.suptitle('Pairplot of Features with Hue as Heart Disease', y=1.02, color='#2c3e50')
plt.show()
```



In [13]:
```
# Putting feature variable to X
X = df.drop('heart disease',axis=1)

# Putting response variable to y
y = df['heart disease']
```

In [14]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, random_state=
```

In [15]:
```
X_train.shape, X_test.shape
```

Out[15]:
```
((189, 4), (81, 4))
```

In [16]:

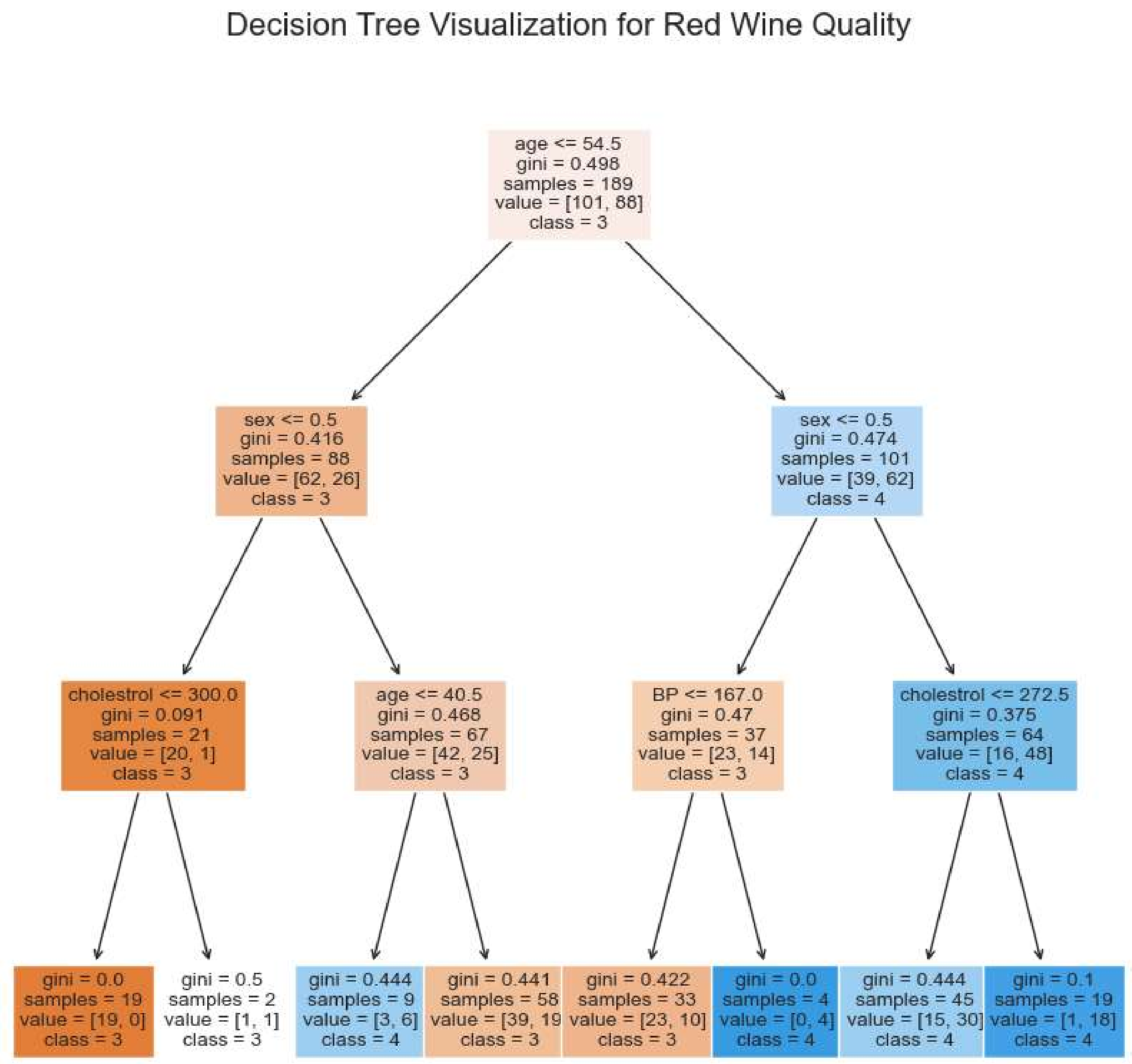
```
from sklearn.tree import DecisionTreeClassifier, plot_tree
dt = DecisionTreeClassifier(max_depth=3)
dt.fit(X_train, y_train)
```

Out[16]:

```
▾    DecisionTreeClassifier   ⓘ ?
                                (https://scikit-
                                learn.org/1.4/modules/generated/sklearn.tree.DecisionTreeClassifi
DecisionTreeClassifier(max_depth=3)
```

In [17]:

```
plt.figure(figsize=(10, 10))
plot_tree(dt, filled=True, feature_names=X.columns, class_names=['3', '4', '5', '6', '7
plt.title("Decision Tree Visualization for Red Wine Quality", fontsize=16)
plt.show()
```



In [18]:

```
y_train = dt.predict(X_train)
y_pred = dt.predict(X_test)
```

In [19]:

```
from sklearn.metrics import classification_report,confusion_matrix, accuracy_score
class_report = classification_report(y_test,y_pred)
print(class_report)
```

```
              precision    recall  f1-score   support

           0       0.66      0.71      0.69        49
           1       0.50      0.44      0.47        32

    accuracy                           0.60        81
   macro avg       0.58      0.58      0.58        81
weighted avg       0.60      0.60      0.60        81
```

In [20]:

```
print(accuracy_score(y_test, y_pred))
```

```
0.6049382716049383
```

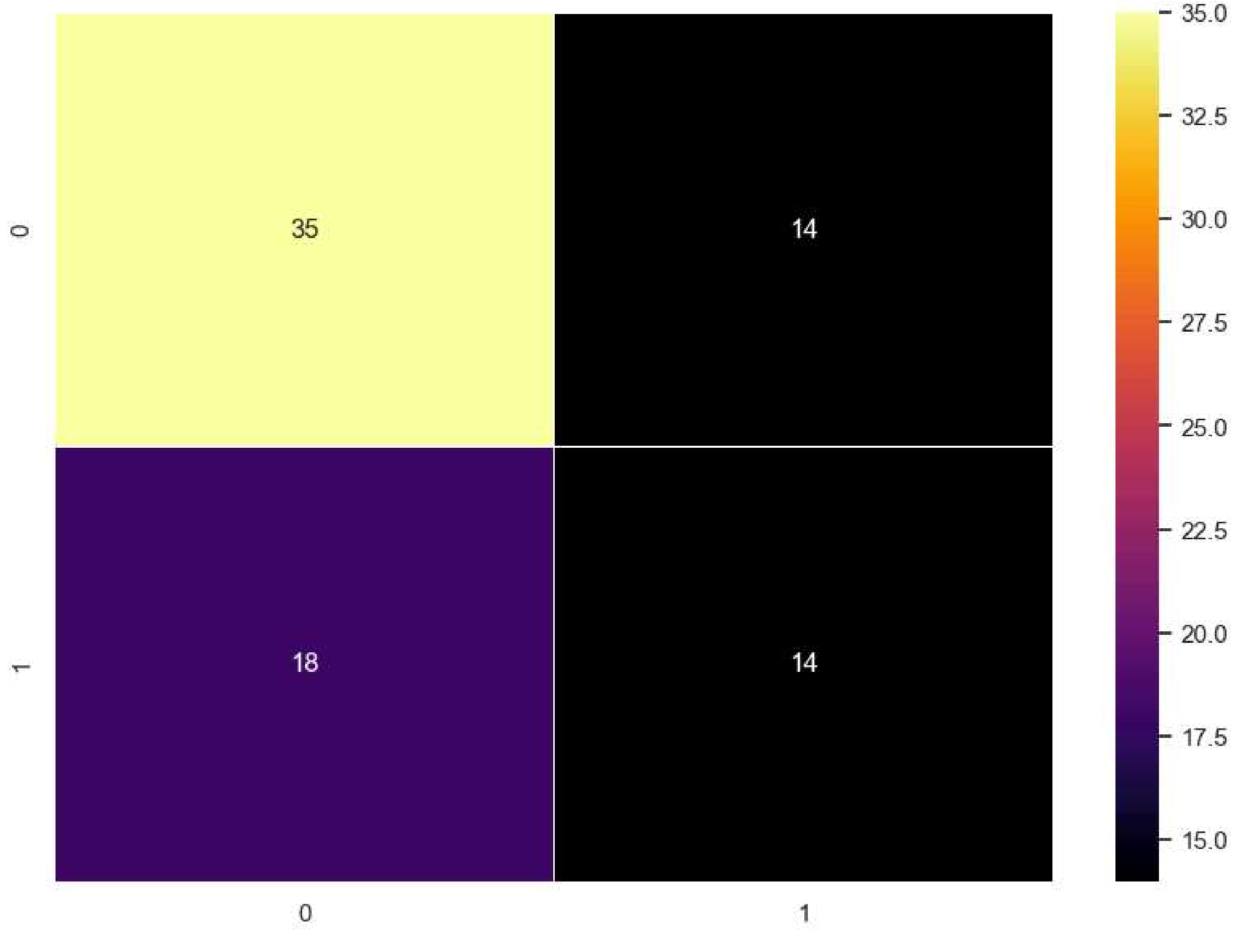
In [21]:

```
matrix=confusion_matrix(y_test,y_pred)
print(matrix)
```

```
[[35 14]
 [18 14]]
```

In [22]:
```
plt.figure(figsize = (10,7))
sns.heatmap(matrix, annot=True, cmap='inferno', linewidths=.5, fmt='g')
```

Out[22]:
```
<Axes: >
```



Mushroom dataset

In [23]:
```
#Load data
df = pd.read_csv("mushrooms.csv")
```

In [24]:

```
df.head()
```

Out[24]:

	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 [25]:

```
df.isnull().sum()
```

Out[25]:

```
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
```

In [26]:

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

Out[26]:

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

In [27]:

```
df.info()
```

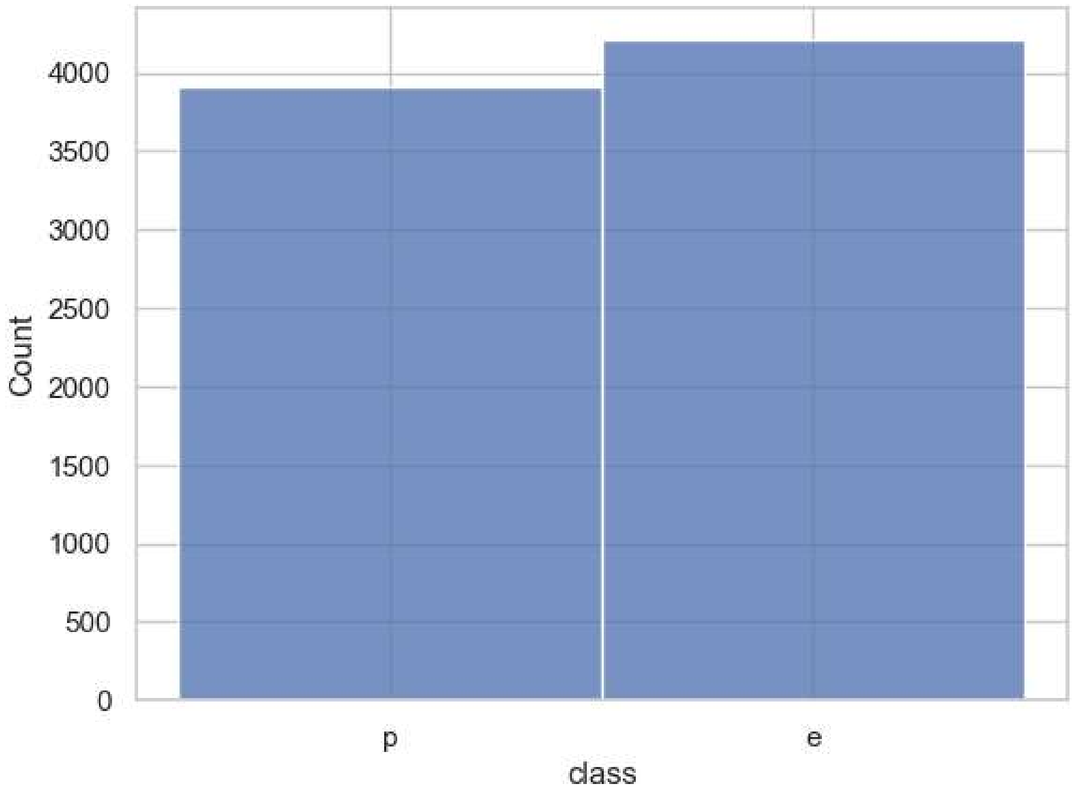
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8124 entries, 0 to 8123
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   class                     8124 non-null   object
 1   cap-shape                 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 [28]:

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

Out[28]:

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



Seprating Features and Targets:

In [29]:

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

In [30]:

```
X = pd.get_dummies(X)
X.head()
```

Out[30]:

	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

Label Encoding

In [31]:

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
y = encoder.fit_transform(y)
print(y)
```

```
[1 0 0 ... 0 1 0]
```

Splitting into training and testing:

In [32]:

```
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:

In [33]:

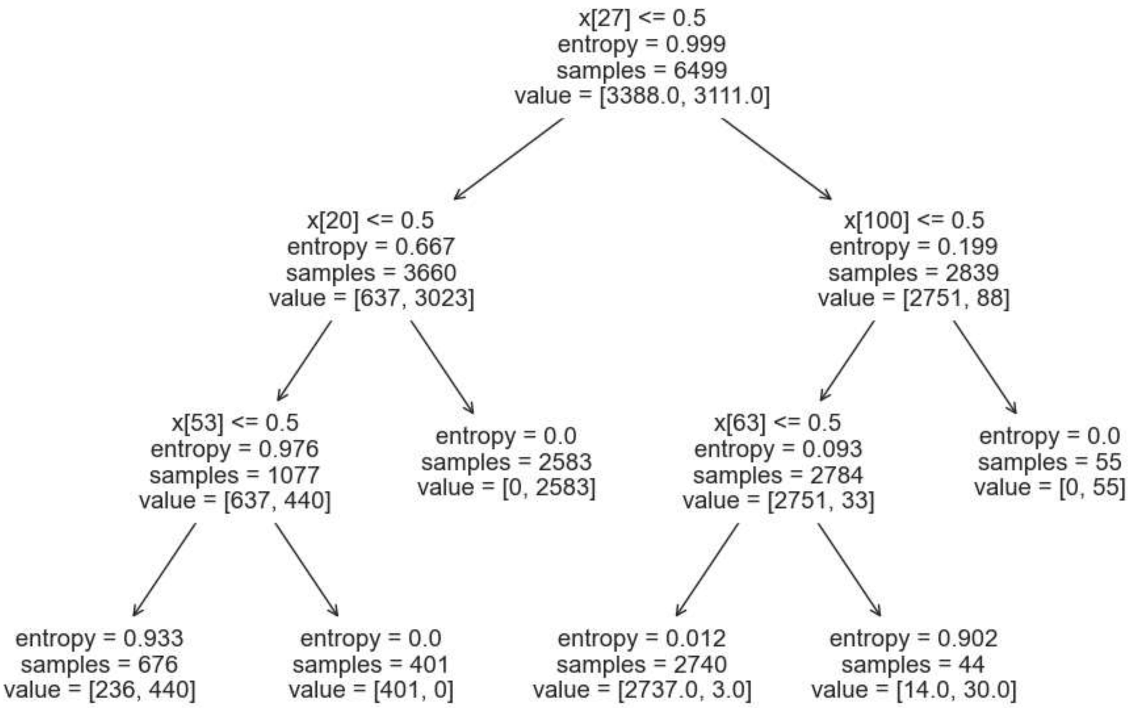
```
clf = DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)

clf.fit(X_train, y_train)
```

Out[33]:

```
▾                    DecisionTreeClassifier                    i ?
                                                               (https://scikit-
                                                               learn.org/1.4/module
DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)
```

In [34]:
```
from sklearn import tree
plt.figure(figsize=(12,8))
tree.plot_tree(clf.fit(X_train, y_train))
```

Out[34]:
```
[Text(0.5555555555555556, 0.875, 'x[27] <= 0.5\nentropy = 0.999\nsamples = 6499\nvalue
= [3388.0, 3111.0]'),
 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 = [2737.0, 3.
0]'),
 Text(0.7777777777777778, 0.125, 'entropy = 0.902\nsamples = 44\nvalue = [14.0, 30.
0]'),
 Text(0.8888888888888888, 0.375, 'entropy = 0.0\nsamples = 55\nvalue = [0, 55]')]
```



In [35]:
```
#Predict values
y_pred = clf.predict(X_test)
```

In [36]:
```
#Predict values using x_train
y_pred_train = clf.predict(X_train)
```

Calculating accuracy_score from scikit_learn

In [37]:
```
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

In [38]:
```
print('Training set score: {0:.2f}'.format(clf.score(X_train, y_train)*100))
print('Test set score: {0:.2f}'.format(clf.score(X_test, y_test)*100))
```

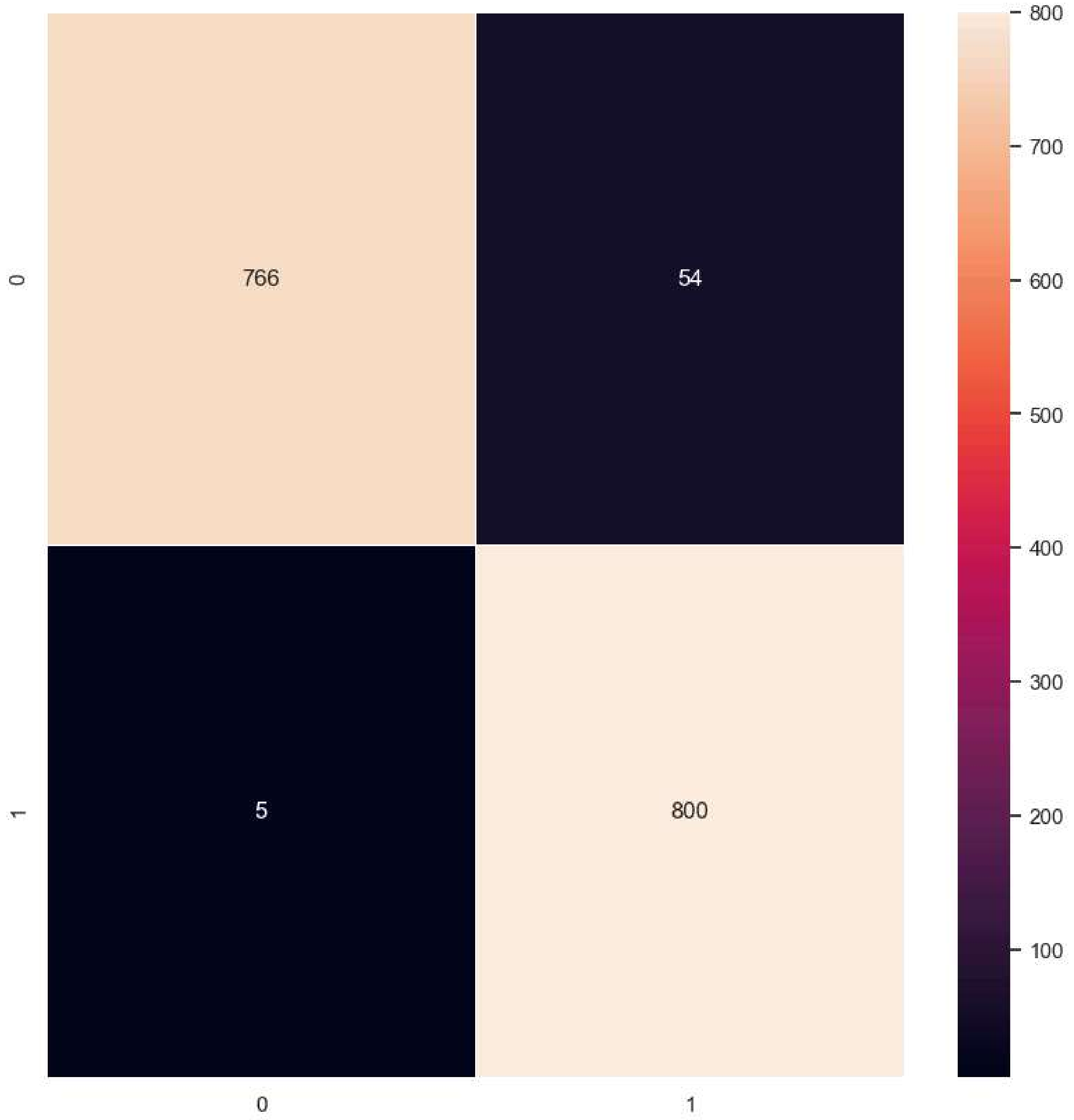
```
Training set score: 96.11
Test set score: 96.37
```

In [39]:
```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

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

In [40]:

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



In [41]:

```
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
```

In [42]:

```
from sklearn.metrics import  f1_score
f1_score = f1_score(y_test, y_pred)
print(f1_score)
```

```
0.9644364074743822
```

In [ ]: