In [1]:
```python
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```python
df=pd.read_csv(r'C:\Users\sharma17\Downloads\ml\classification dataset\HeartDisease.csv')
```

In [3]:
```python
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]:
```python
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
```

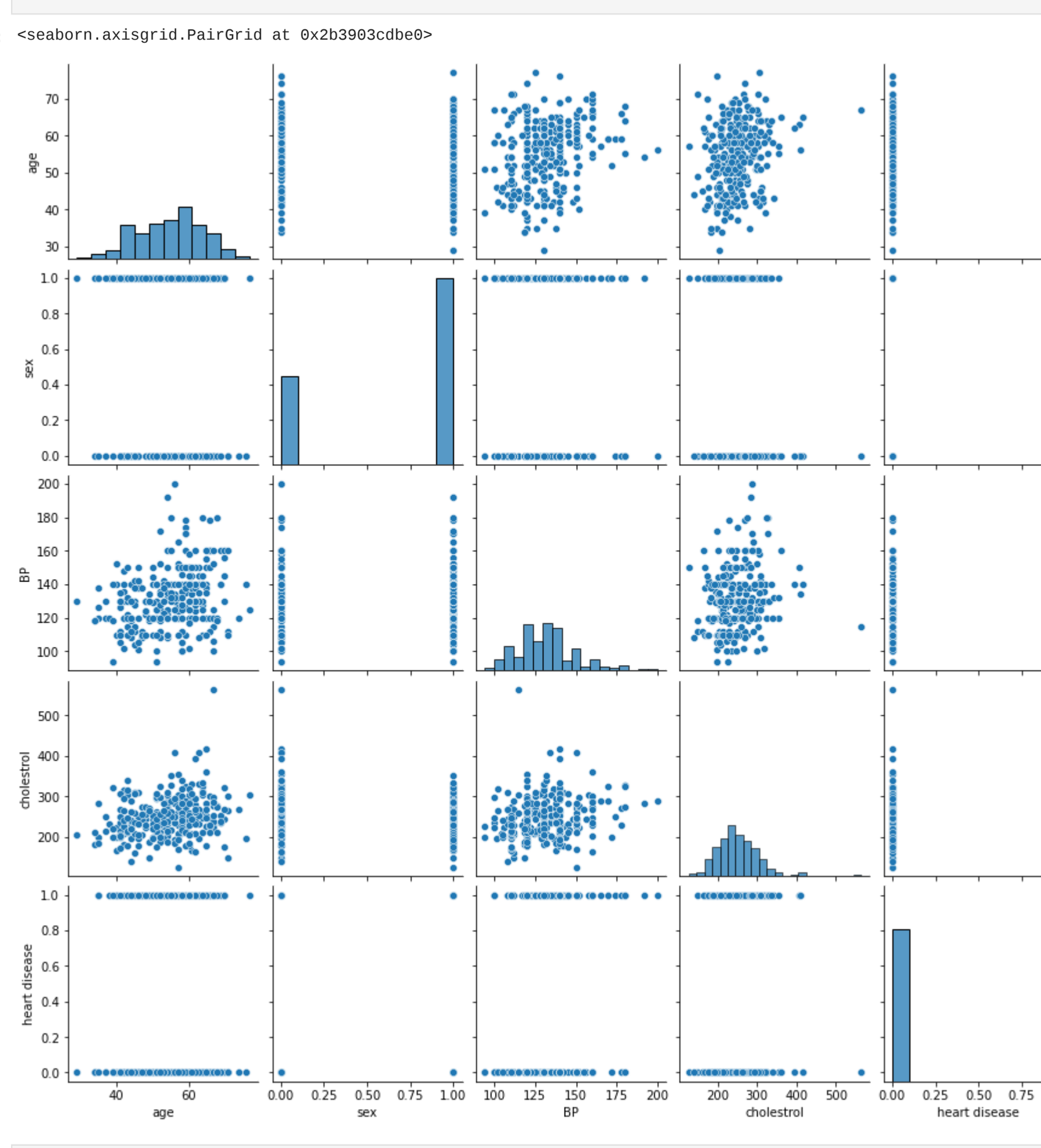
In [5]:
```python
df.describe()
```

Out[5]:

	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

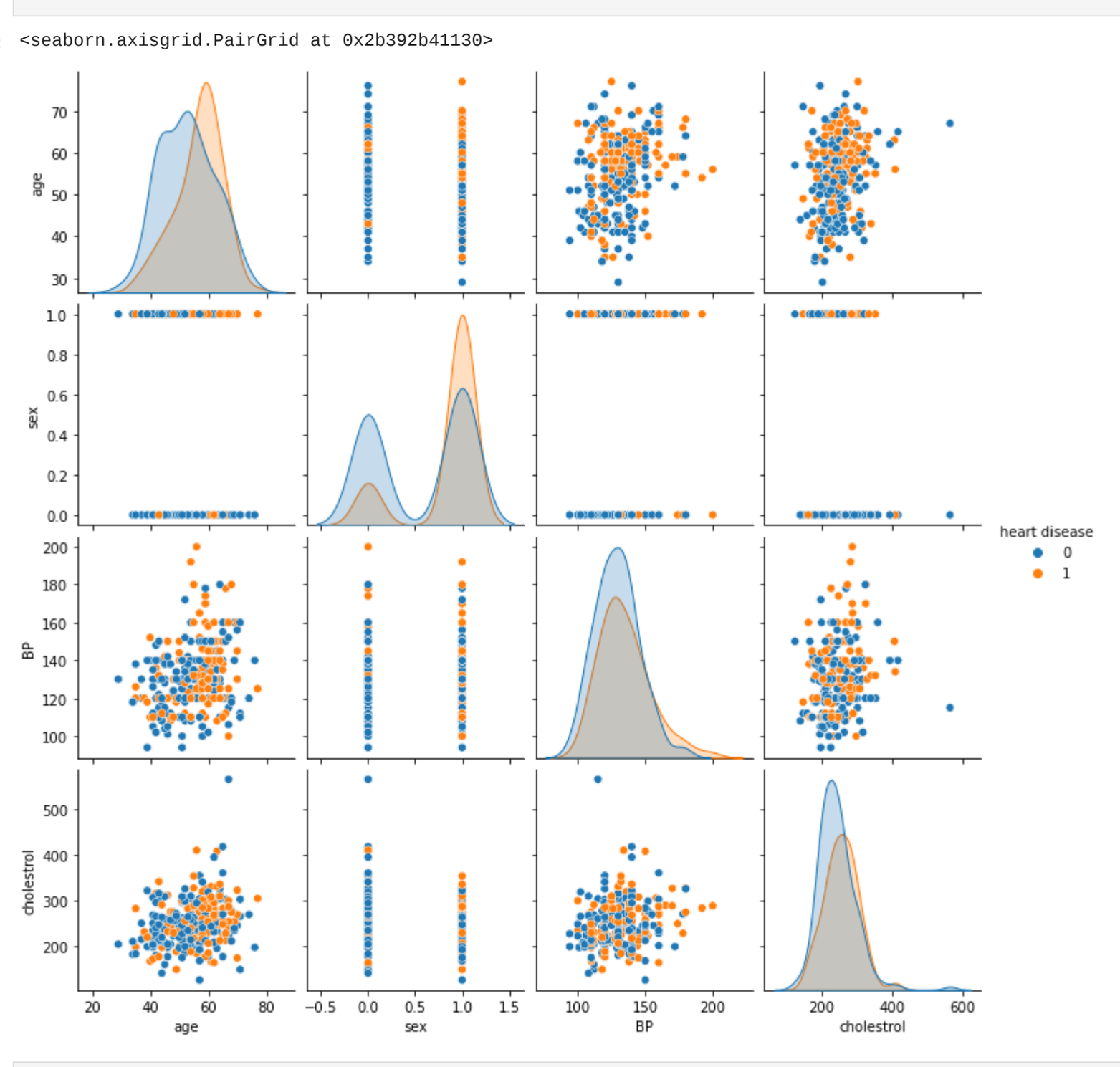
In [6]:
```python
sns.pairplot(df)
```

Out[6]: <seaborn.axisgrid.PairGrid at 0x2b3903cdbe0>

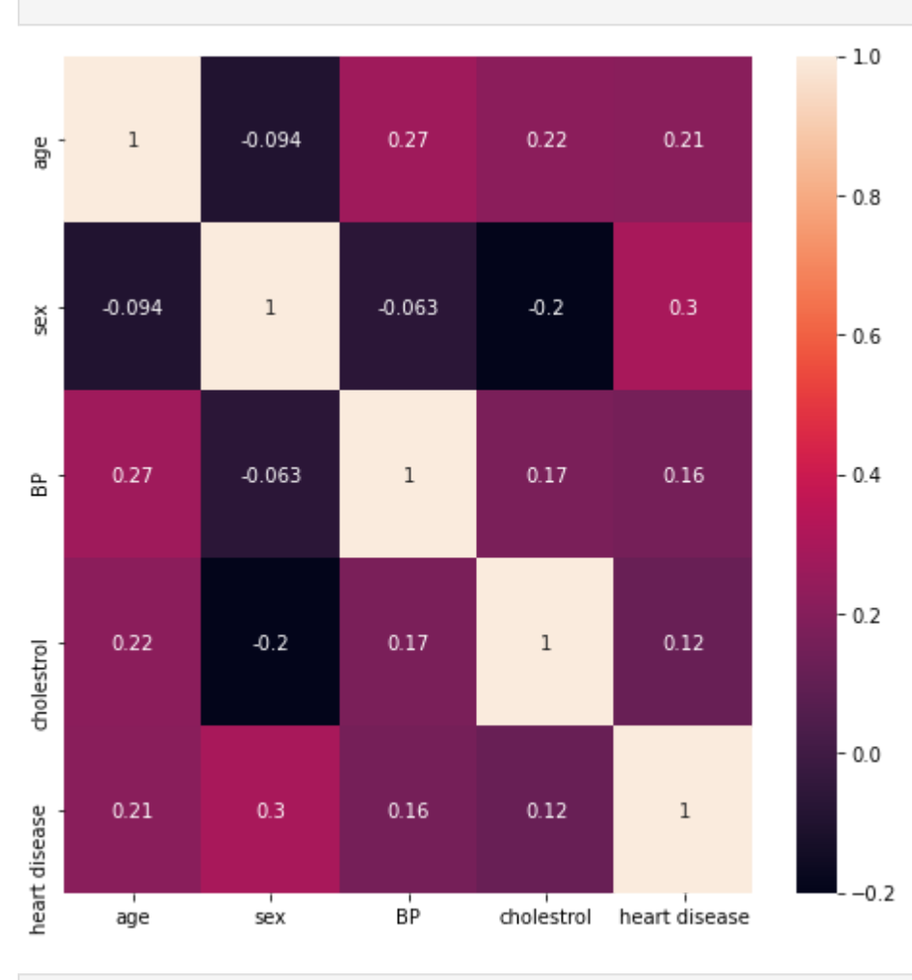


In [7]:
```python
sns.pairplot(df,hue='heart disease')
```

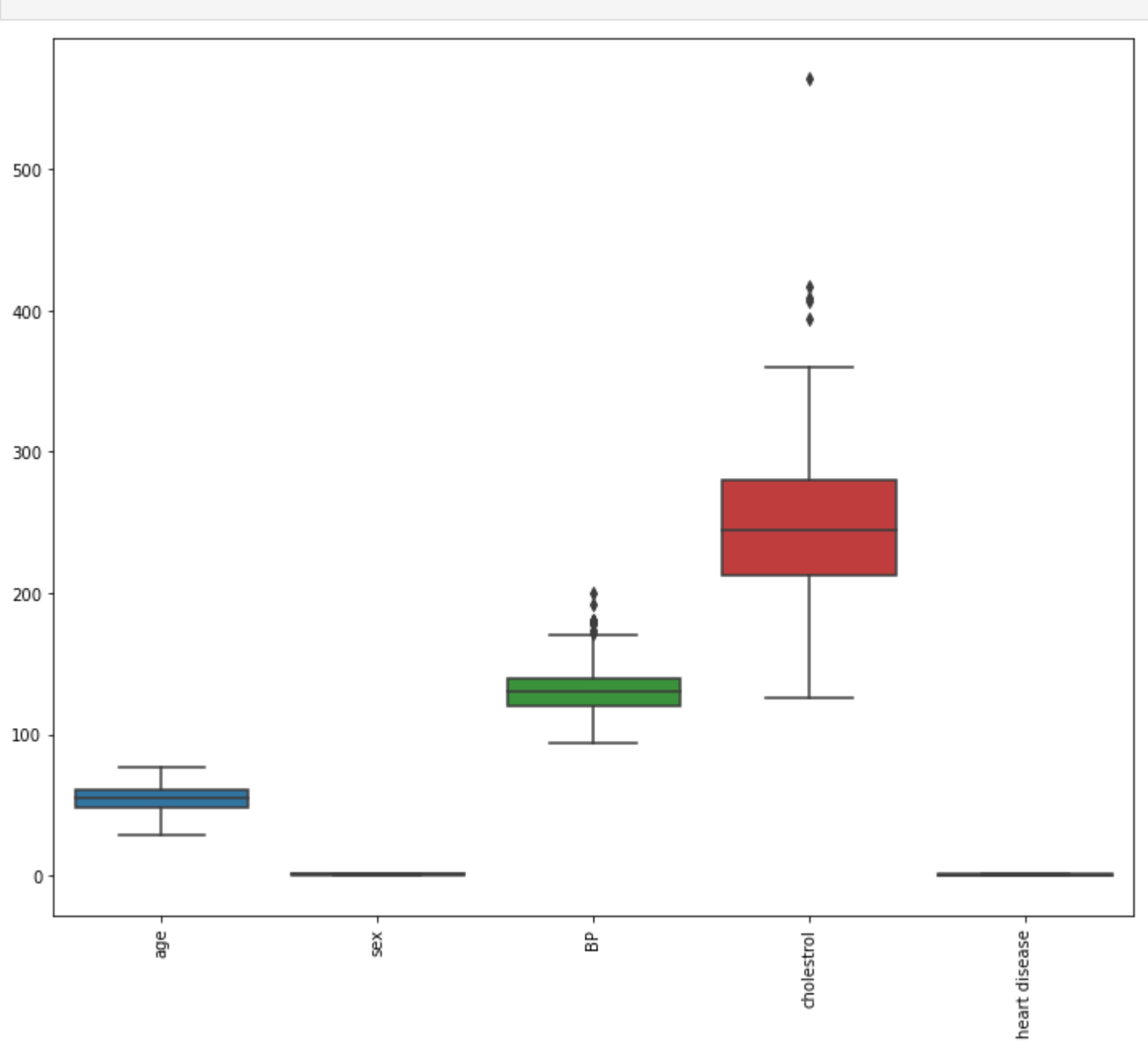
Out[7]: <seaborn.axisgrid.PairGrid at 0x2b392b41130>



In [8]:
```python
plt.figure(figsize=(8,8))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



In [9]:
```python
plt.figure(figsize=(12,10))
sns.boxplot(data=df)
plt.xticks(rotation=90)
plt.show()
```



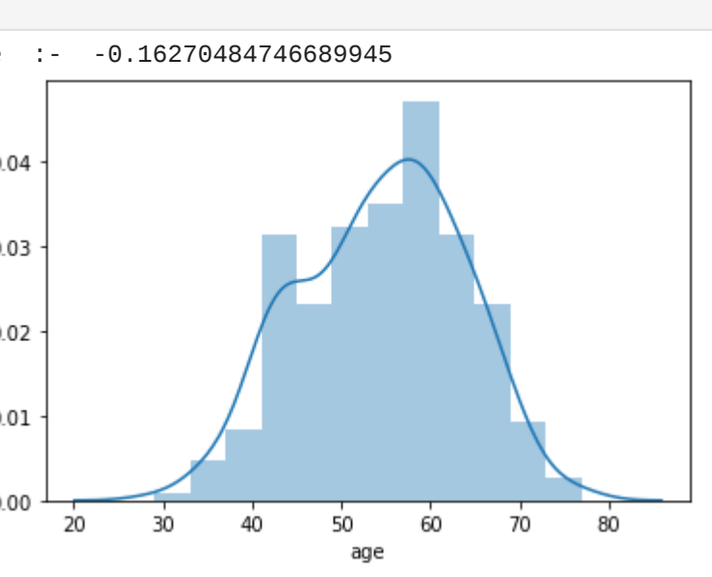
In [10]:
```python
df.skew()
```

Out[10]:
```
age             -0.163615
sex             -0.765084
BP               0.722618
cholestrol       1.183721
heart disease    0.224858
dtype: float64
```

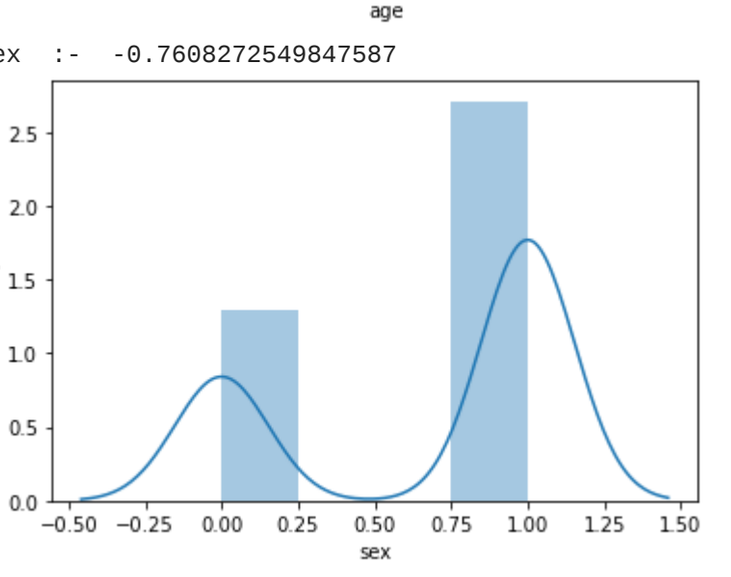
In [11]:
```python
from scipy.stats import skew
```

In [12]:
```python
for col in df:

    print(col,  ' :-  ',end='')
    print(skew(df[col]))


    sns.distplot(df[col])
    plt.show()
```

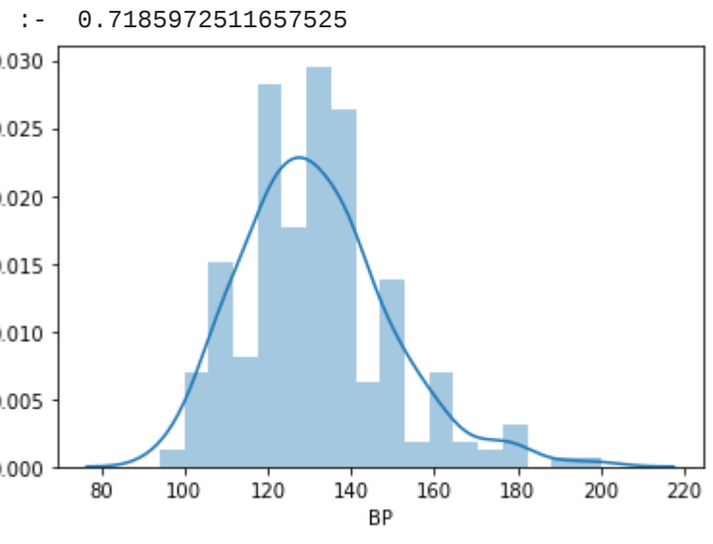
age :- -0.16270484746689945



sex :- -0.7608272549847587

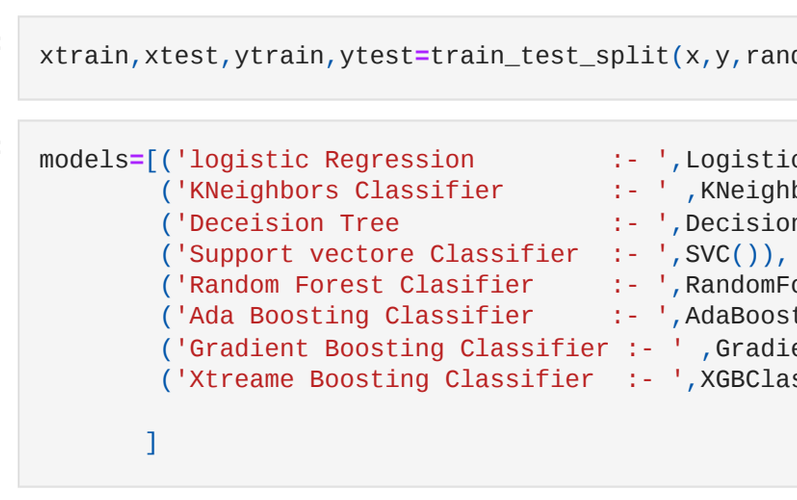


BP :- 0.7185972511657525



cholestrol :- 1.1771344865188054

heart disease :- 0.22360679774997907



In [13]:
```python
x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
```

In [14]:
```python
from sklearn.metrics import accuracy_score, confusion_matrix,classification_report
from sklearn.model_selection import train_test_split ,cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier , AdaBoostClassifier   ,GradientBoostingClassifier
from xgboost import XGBClassifier
```

In [15]:
```python
xtrain,xtest,ytrain,ytest=train_test_split(x,y,random_state=0,test_size=0.25,stratify=y)
```

In [16]:
```python
models=[('logistic Regression         :- ',LogisticRegression()),
        ('KNeighbors Classifier       :- ' ,KNeighborsClassifier()),
        ('Deceision Tree              :- ',DecisionTreeClassifier()),
        ('Support vectore Classifier  :- ',SVC()),
        ('Random Forest Clasifier     :- ',RandomForestClassifier()),
        ('Ada Boosting Classifier     :- ',AdaBoostClassifier()),
        ('Gradient Boosting Classifier :- ' ,GradientBoostingClassifier()),
        ('Xtreame Boosting Classifier  :- ',XGBClassifier())

       ]
```

In [17]:
```python
accuracy=[]
for name,model in models:
    print(name)
    model.fit(xtrain,ytrain)
    ypred=model.predict(xtest)
    print('Confusion matrix  :-  ')
    print(confusion_matrix(ytest,ypred))
    print(classification_report(ytest,ypred))
    accuracy.append(accuracy_score(ytest,ypred))
```

```
logistic Regression         :-
Confusion matrix  :-
[[25 13]
 [10 20]]
              precision    recall  f1-score   support

           0       0.71      0.66      0.68        38
           1       0.61      0.67      0.63        30

    accuracy                           0.66        68
   macro avg       0.66      0.66      0.66        68
weighted avg       0.67      0.66      0.66        68

KNeighbors Classifier       :-
Confusion matrix  :-
[[27 11]
 [13 17]]
              precision    recall  f1-score   support

           0       0.68      0.71      0.69        38
           1       0.61      0.57      0.59        30

    accuracy                           0.65        68
   macro avg       0.64      0.64      0.64        68
weighted avg       0.65      0.65      0.65        68

Deceision Tree              :-
Confusion matrix  :-
[[26 12]
 [12 18]]
              precision    recall  f1-score   support

           0       0.68      0.68      0.68        38
           1       0.60      0.60      0.60        30

    accuracy                           0.65        68
   macro avg       0.64      0.64      0.64        68
weighted avg       0.65      0.65      0.65        68

Support vectore Classifier  :-
Confusion matrix  :-
[[33  5]
 [20 10]]
              precision    recall  f1-score   support

           0       0.62      0.87      0.73        38
           1       0.67      0.33      0.44        30

    accuracy                           0.63        68
   macro avg       0.64      0.60      0.58        68
weighted avg       0.64      0.63      0.60        68

Random Forest Clasifier     :-
Confusion matrix  :-
[[26 12]
 [10 20]]
              precision    recall  f1-score   support

           0       0.72      0.68      0.70        38
           1       0.62      0.67      0.65        30

    accuracy                           0.68        68
   macro avg       0.67      0.68      0.67        68
weighted avg       0.68      0.68      0.68        68

Ada Boosting Classifier     :-
Confusion matrix  :-
[[22 16]
 [ 9 21]]
              precision    recall  f1-score   support

           0       0.71      0.58      0.64        38
           1       0.57      0.70      0.63        30

    accuracy                           0.63        68
   macro avg       0.64      0.64      0.63        68
weighted avg       0.65      0.63      0.63        68

Gradient Boosting Classifier :-
Confusion matrix  :-
[[24 14]
 [11 19]]
              precision    recall  f1-score   support

           0       0.69      0.63      0.66        38
           1       0.58      0.63      0.60        30

    accuracy                           0.63        68
   macro avg       0.63      0.63      0.63        68
weighted avg       0.64      0.63      0.63        68

Xtreame Boosting Classifier  :-
[20:57:31] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'er
ror' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
Confusion matrix  :-
[[25 13]
 [10 20]]
              precision    recall  f1-score   support

           0       0.71      0.66      0.68        38
           1       0.61      0.67      0.63        30

    accuracy                           0.66        68
   macro avg       0.66      0.66      0.66        68
weighted avg       0.67      0.66      0.66        68
```

In [18]:
```python
dtree=DecisionTreeClassifier()
dtree.fit(xtrain,ytrain)
ypred=dtree.predict(xtest)
```

In [19]:
```python
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

           0       0.68      0.66      0.67        38
           1       0.58      0.60      0.59        30

    accuracy                           0.63        68
   macro avg       0.63      0.63      0.63        68
weighted avg       0.63      0.63      0.63        68
```

In [20]:
```python
print('trainig score  :-  ',dtree.score(xtrain,ytrain)*100)
print('testing score   :-  ',dtree.score(xtest,ytest)*100)
```

```
trainig score  :-   100.0
testing score   :-   63.23529411764706
```

In [21]:
```python
path=dtree.cost_complexity_pruning_path(xtrain,ytrain)
path
```

Out[21]:
```
{'ccp_alphas': array([0.        , 0.00412541, 0.00440044, 0.00458903, 0.0049505 ,
        0.00550055, 0.00577558, 0.00594059, 0.00594059, 0.00618812,
        0.00621466, 0.00660066, 0.00660066, 0.00660066, 0.00796454,
        0.00825083, 0.00858086, 0.00873617, 0.00923348, 0.00960096,
        0.01019977, 0.01155116, 0.01211304, 0.01369137, 0.01699524,
        0.02097879, 0.03423991, 0.03944089]),
 'impurities': array([0.        , 0.00825083, 0.01705171, 0.03540783, 0.05520981,
        0.07721201, 0.09453874, 0.10641993, 0.12424171, 0.14280607,
        0.16145004, 0.1680507 , 0.17465136, 0.18125202, 0.20514563,
        0.21339646, 0.23055817, 0.24803051, 0.25726399, 0.26686495,
        0.32806357, 0.33961473, 0.35172776, 0.36541913, 0.39940962,
        0.42038841, 0.45462832, 0.49406921])}
```

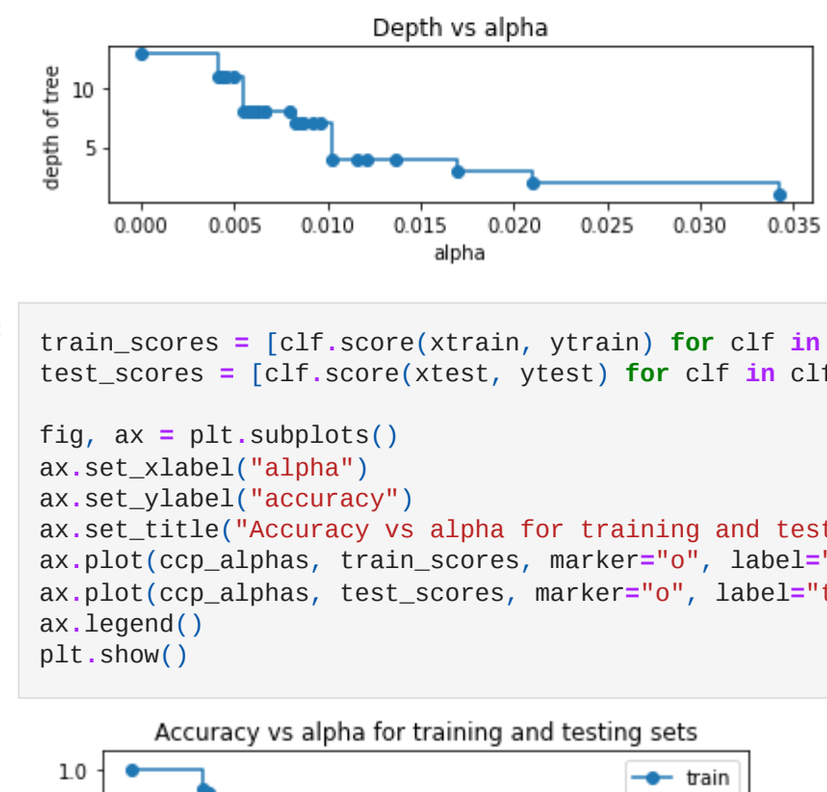
In [22]:
```python
ccp_alphas, impurities = path.ccp_alphas, path.impurities
```

In [ ]:

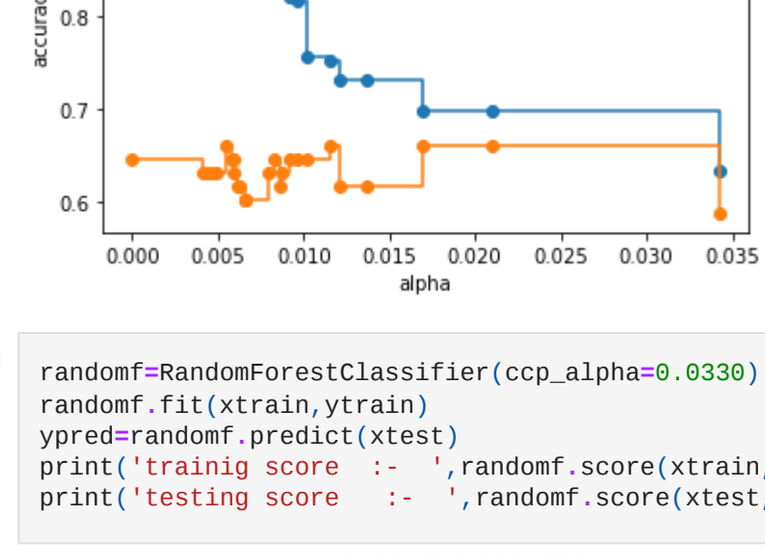
In [23]:
```python
clfs = []
for ccp_alpha in ccp_alphas:
    clf = DecisionTreeClassifier(random_state=0, ccp_alpha=ccp_alpha)
    clf.fit(xtrain, ytrain)
    clfs.append(clf)
print(
    "Number of nodes in the last tree is: {} with ccp_alpha: {}".format(
        clfs[-1].tree_.node_count, ccp_alphas[-1]
    )
)
```

Number of nodes in the last tree is: 1 with ccp_alpha: 0.03944088892623343

In [24]:
```python
clfs = clfs[:-1]
ccp_alphas = ccp_alphas[:-1]

node_counts = [clf.tree_.node_count for clf in clfs]
depth = [clf.tree_.max_depth for clf in clfs]
fig, ax = plt.subplots(2, 1)
ax[0].plot(ccp_alphas, node_counts, marker="o", drawstyle="steps-post")
ax[0].set_xlabel("alpha")
ax[0].set_ylabel("number of nodes")
ax[0].set_title("Number of nodes vs alpha")
ax[1].plot(ccp_alphas, depth, marker="o", drawstyle="steps-post")
ax[1].set_xlabel("alpha")
ax[1].set_ylabel("depth of tree")
ax[1].set_title("Depth vs alpha")
fig.tight_layout()
```



In [25]:
```python
train_scores = [clf.score(xtrain, ytrain) for clf in clfs]
test_scores = [clf.score(xtest, ytest) for clf in clfs]

fig, ax = plt.subplots()
ax.set_xlabel("alpha")
ax.set_ylabel("accuracy")
ax.set_title("Accuracy vs alpha for training and testing sets")
ax.plot(ccp_alphas, train_scores, marker="o", label="train", drawstyle="steps-post")
ax.plot(ccp_alphas, test_scores, marker="o", label="test", drawstyle="steps-post")
ax.legend()
plt.show()
```



In [26]:
```python
randomf=RandomForestClassifier(ccp_alpha=0.0330)
randomf.fit(xtrain,ytrain)
ypred=randomf.predict(xtest)
print('trainig score  :-  ',randomf.score(xtrain,ytrain)*100)
print('testing score   :-  ',randomf.score(xtest,ytest)*100)
```

```
trainig score  :-   70.29702970297029
testing score   :-   64.70588235294117
```

In [27]:
```python
print(confusion_matrix(ytest,ypred))

print(classification_report(ytest,ypred))
```

```
[[27 11]
 [13 17]]
              precision    recall  f1-score   support

           0       0.68      0.71      0.69        38
           1       0.61      0.57      0.59        30

    accuracy                           0.65        68
   macro avg       0.64      0.64      0.64        68
weighted avg       0.65      0.65      0.65        68
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: