In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv('D:\\dataset\\kidney_disease.csv')
```

In [3]:
```
df.head(5)
```

Out[3]:

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc
0	0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	...	44.0	7800.0	5.2
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	...	38.0	6000.0	NaN
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	...	31.0	7500.0	NaN
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	...	32.0	6700.0	3.9
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	...	35.0	7300.0	4.6

5 rows × 26 columns

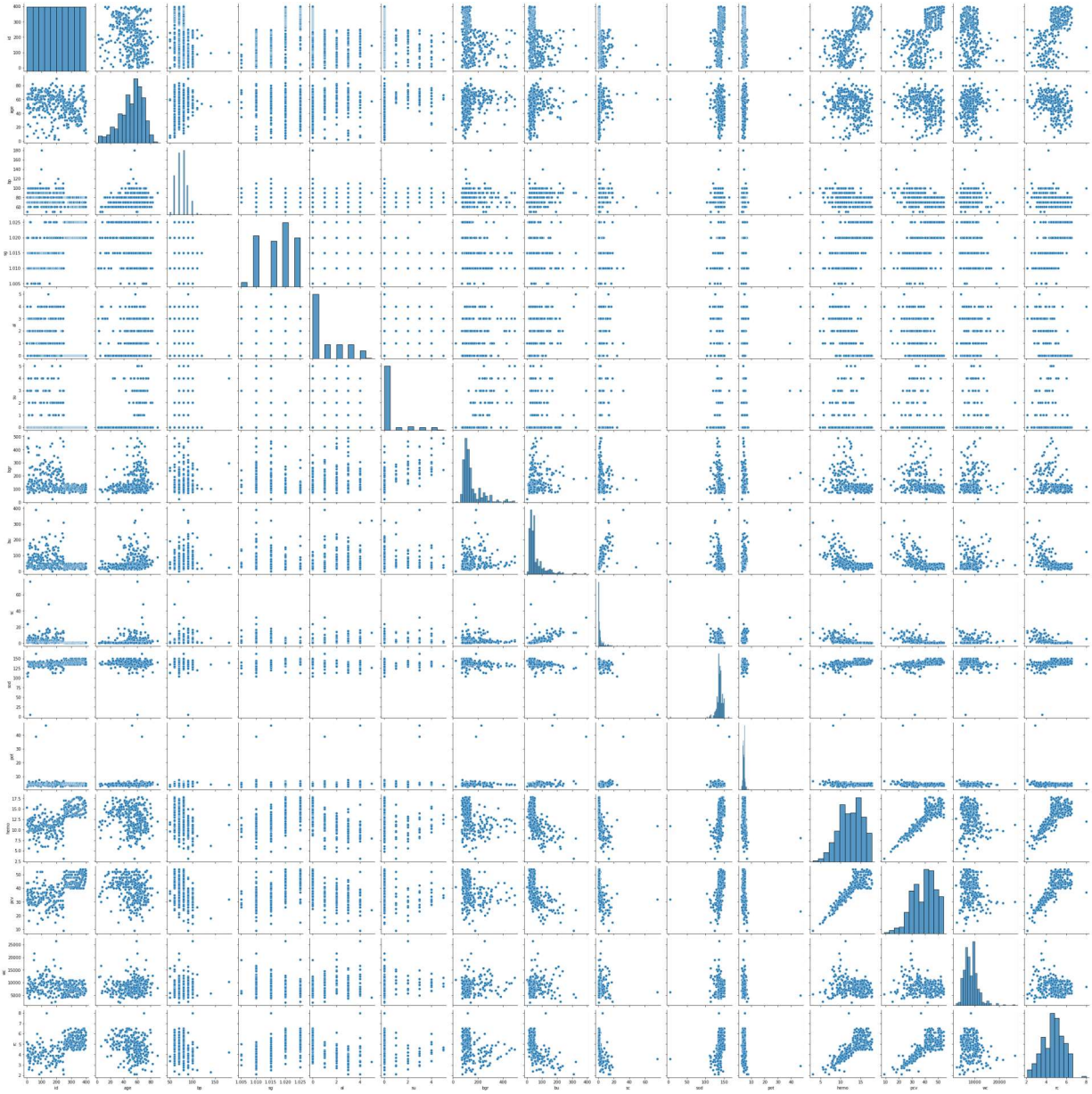
In [4]:
```
df.shape
```

Out[4]:
```
(400, 26)
```

In [5]:
```
import seaborn as sns
```

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

Out[6]:
```
<seaborn.axisgrid.PairGrid at 0x80a8748>
```



In [7]:
```
df.isnull().any()
```

Out[7]:
```
id              False
age              True
bp               True
sg               True
al               True
su               True
rbc              True
pc               True
pcc              True
ba               True
bgr              True
bu               True
sc               True
sod              True
pot              True
hemo             True
pcv              True
wc               True
rc               True
htn              True
dm               True
cad              True
appet            True
pe               True
ane              True
```

```
classification    False
dtype: bool
```

In [8]:
```
df['age']=df['age'].fillna(df['age'].median())
df['bp'] = df['bp'].fillna(df['bp'].median())
df['sg'] = df['sg'].fillna(df['sg'].median())
df['al'] = df['al'].fillna(df['al'].median())
df['su'] = df['su'].fillna(df['su'].median())
df['rbc']=df['rbc'].fillna(df['rbc'].mode())
```

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
id                0
age               0
bp                0
sg                0
al                0
su                0
rbc             151
pc               65
pcc               4
ba                4
bgr              44
bu               19
sc               17
sod              87
pot              88
hemo             52
pcv              71
wc              106
rc              131
htn               2
dm                2
cad               2
appet             1
pe                1
ane               1
classification    0
dtype: int64
```

In [12]:
```
df['rbc'].mode()
```

Out[12]:
```
0    normal
dtype: object
```

In [13]:
```
df['rbc']=df['rbc'].fillna('normal')
```

In [14]:
```
df.isnull().sum()
```

Out[14]:
```
id                0
age               0
bp                0
sg                0
al                0
su                0
rbc               0
pc               65
pcc               4
ba                4
bgr              44
bu               19
sc               17
sod              87
pot              88
hemo             52
pcv              71
wc              106
```

```
rc                131
htn                 2
dm                  2
cad                 2
appet               1
pe                  1
ane                 1
classification      0
dtype: int64
```

In [15]:
```
df['pc'].mode()
```

Out[15]:
```
0    normal
dtype: object
```

In [16]:
```
df['pc']=df['pc'].fillna('normal')
```

In [17]:
```
df['pcc'].mode()
```

Out[17]:
```
0    notpresent
dtype: object
```

In [18]:
```
df['pcc']=df['pcc'].fillna('notpresent')
```

In [19]:
```
df['ba'].mode()
```

Out[19]:
```
0    notpresent
dtype: object
```

In [20]:
```
df['ba']=df['ba'].fillna('notpresent')
```

In [21]:
```
df['bgr'].median()
```

Out[21]:
```
121.0
```

In [22]:
```
df['bgr']=df['bgr'].fillna(df['bgr'].median())
```

In [23]:
```
df['bu']=df['bu'].fillna(df['bu'].median())
```

In [24]:
```
df['sc']=df['sc'].fillna(df['sc'].median())
```

In [25]:
```
df['sod']=df['sod'].fillna(df['sod'].median())
```

In [26]:
```
df['pot']=df['pot'].fillna(df['pot'].median())
```

In [27]:
```
df['hemo']=df['hemo'].fillna(df['hemo'].median())
```

In [28]:
```
df['pcv']=df['pcv'].fillna(df['pcv'].median())
```

In [29]:
```
df['wc']=df['wc'].fillna(df['wc'].median())
```

In [30]:
```
df['rc']=df['rc'].fillna(df['rc'].median())
```

In [31]:
```
df.isnull().sum()
```

Out[31]:
```
id                0
age               0
bp                0
sg                0
al                0
```

```
su                0
rbc               0
pc                0
pcc               0
ba                0
bgr               0
bu                0
sc                0
sod               0
pot               0
hemo              0
pcv               0
wc                0
rc                0
htn               2
dm                2
cad               2
appet             1
pe                1
ane               1
classification    0
dtype: int64
```

In [32]:
```
df['htn'].mode()
```

Out[32]:
```
0    no
dtype: object
```

In [33]:
```
df['htn']=df['htn'].fillna('no')
```

In [34]:
```
df['dm'].mode()
```

Out[34]:
```
0    no
dtype: object
```

In [35]:
```
df['dm']=df['dm'].fillna('no')
```

In [36]:
```
df['cad'].mode()
```

Out[36]:
```
0    no
dtype: object
```

In [37]:
```
df['cad']=df['cad'].fillna('no')
```

In [38]:
```
df['appet'].mode()
```

Out[38]:
```
0    good
dtype: object
```

In [39]:
```
df['appet']=df['appet'].fillna('good')
```

In [40]:
```
df['pe'].mode()
```

Out[40]:
```
0    no
dtype: object
```

In [41]:
```
df['pe']=df['pe'].fillna('no')
```

In [42]:
```
df['ane'].mode()
```

Out[42]:
```
0    no
dtype: object
```

In [43]:
```
df['ane']=df['ane'].fillna('no')
```

In [44]:
```
df.isnull().sum()
```

Out[44]:
```
id                0
age               0
bp                0
sg                0
al                0
su                0
rbc               0
pc                0
pcc               0
ba                0
bgr               0
bu                0
sc                0
sod               0
pot               0
hemo              0
pcv               0
wc                0
rc                0
htn               0
dm                0
cad               0
appet             0
pe                0
ane               0
classification    0
dtype: int64
```

In [45]:
```
from sklearn.preprocessing import LabelEncoder
```

In [46]:
```
df.columns
```

Out[46]:
```
Index(['id', 'age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr',
       'bu', 'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad',
       'appet', 'pe', 'ane', 'classification'],
      dtype='object')
```

In [47]:
```
df['rbc']=LabelEncoder().fit_transform(df['rbc'])
```

In [48]:
```
df['pc']=LabelEncoder().fit_transform(df['pc'])
```

In [49]:
```
df['pcc']=LabelEncoder().fit_transform(df['pcc'])
```

In [50]:
```
df['ba']=LabelEncoder().fit_transform(df['ba'])
```

In [51]:
```
df['htn']=LabelEncoder().fit_transform(df['htn'])
```

In [52]:
```
df['dm']=LabelEncoder().fit_transform(df['dm'])
```

In [53]:
```
df['cad']=LabelEncoder().fit_transform(df['cad'])
```

In [54]:
```
df['appet']=LabelEncoder().fit_transform(df['appet'])
```

In [55]:
```
df['pe']=LabelEncoder().fit_transform(df['pe'])
```

In [56]:
```
df['ane']=LabelEncoder().fit_transform(df['ane'])
```

In [57]:
```
df['classification']=LabelEncoder().fit_transform(df['classification'])
```

In [58]:
```
df.head()
```

Out[58]:

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe
0	0	48.0	80.0	1.020	1.0	0.0	1	1	0	0	...	44.0	7800.0	5.2	1	4	1	0	(
1	1	7.0	50.0	1.020	4.0	0.0	1	1	0	0	...	38.0	6000.0	4.8	0	3	1	0	(
2	2	62.0	80.0	1.010	2.0	3.0	1	1	0	0	...	31.0	7500.0	4.8	0	4	1	1	(
3	3	48.0	70.0	1.005	4.0	0.0	1	0	1	0	...	32.0	6700.0	3.9	1	3	1	1	1
4	4	51.0	80.0	1.010	2.0	0.0	1	1	0	0	...	35.0	7300.0	4.6	0	3	1	0	(

5 rows × 26 columns

In [59]:
```
Q1_df =df.quantile(0.25)
Q3_df = df.quantile(0.75)
IQR = Q3_df - Q1_df
print(IQR)
```

```
id                199.500
age                22.000
bp                 10.000
sg                  0.005
al                  2.000
su                  0.000
rbc                 0.000
pc                  0.000
pcc                 0.000
ba                  0.000
bgr                49.000
bu                 34.750
sc                  1.825
sod                 6.000
pot                 0.800
hemo                3.750
pcv                10.000
wc               2425.000
rc                  0.600
htn                 1.000
dm                  1.000
cad                 0.000
appet               0.000
pe                  0.000
ane                 0.000
classification      2.000
dtype: float64
```

In [60]:
```
df_out= df[~((df<(Q1_df-1.5* IQR))|(df>(Q3_df + 1.5 * IQR))).any(axis=1)]
df_out.shape
```

Out[60]:
```
(148, 26)
```

In [61]:
```
df_out.columns
```

Out[61]:
```
Index(['id', 'age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr',
       'bu', 'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad',
       'appet', 'pe', 'ane', 'classification'],
      dtype='object')
```

In [62]:
```
x =df_out[['id', 'age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr',
       'bu', 'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad',
       'appet', 'pe', 'ane']]
```

In [63]:
```
y = df_out[['classification']]
```

In [64]:
```
from sklearn.model_selection import train_test_split
```

In [65]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [66]:
```
from sklearn import tree
```

In [67]:
```
model = tree.DecisionTreeRegressor()
```

In [68]:
```
model.fit(x_train,y_train)
```

Out[68]:
```
DecisionTreeRegressor()
```

In [69]:
```
y_pred = model.predict(x_test)
```

In [70]:
```
y_pred
```

Out[70]:
```
array([2., 2., 2., 2., 0., 2., 2., 2., 2., 0., 2., 2., 2., 2., 2., 2., 0.,
       2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 0., 2., 2., 2., 2., 2.,
       2., 2., 2., 2., 2., 2., 2., 2., 0., 2., 2.])
```

In [71]:
```
import numpy  as np
from sklearn import metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test,y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
Mean Absolute Error: 0.0
Mean Squared Error: 0.0
Root Mean Squared Error: 0.0
```

In [72]:
```
from sklearn.linear_model import LogisticRegression
```

In [73]:
```
model = LogisticRegression()
```

In [74]:
```
model.fit(x_train,y_train)
```

```
D:\data science\lib\site-packages\sklearn\utils\validation.py:73: DataConversionWarn
ing: A column-vector y was passed when a 1d array was expected. Please change the sh
ape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
D:\data science\lib\site-packages\sklearn\linear_model\_logistic.py:762: Convergence
Warning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[74]:
```
LogisticRegression()
```

In [86]:
```
y_pred=model.predict(x_test)
```

In [87]:
```
y_pred
```

Out[87]:
```
array([2, 2, 2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2,
       2])
```

In [88]:
```
from sklearn import metrics
```

In [89]:
```
cm = metrics.confusion_matrix(y_test,y_pred)
```

In [90]:
```
cm
```

Out[90]:
```
array([[ 5,  0],
       [ 0, 40]], dtype=int64)
```

In [91]:
```
metrics.accuracy_score(y_test,y_pred)
```

Out[91]:
```
1.0
```

In [96]:
```
y_test = y_true
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-96-687f3cc81c8d> in <module>
----> 1 y_test = y_true

NameError: name 'y_true' is not defined
```

In [ ]: