

kidney_disease Dataset Project

```
In [3]: import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
In [4]: df=pd.read_csv("kidney_disease12.csv")
df.head()
```

Out[4]:

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	cla
0	0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	...	44	7800	5.2	yes	yes	no	good	no	no	
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	...	38	6000	NaN	no	no	no	good	no	no	
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	...	31	7500	NaN	no	yes	no	poor	no	yes	
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	...	32	6700	3.9	yes	no	no	poor	yes	yes	
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	...	35	7300	4.6	no	no	no	good	no	no	

5 rows × 26 columns



Getting features type

```
In [335... df.dtypes
```

Out[335...

id	int64
age	float64
bp	float64
sg	float64
al	float64
su	float64
rbc	object
pc	object
pcc	object
ba	object
bgr	float64
bu	float64
sc	float64
sod	float64
pot	float64
hemo	float64
pcv	object
wc	object
rc	object
htn	object
dm	object
cad	object
appet	object
pe	object
ane	object
classification	object
dtype:	object

print columns

```
In [336... df.columns
```

Out[336...

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')
--

missing values

```
In [337... df.isnull().sum()
```

Out[337...
id 0
age 9
bp 12
sg 47
al 46
su 49
rbc 152
pc 65
pcc 4
ba 4
bgr 44
bu 19
sc 17
sod 87
pot 88
hemo 52
pcv 70
wc 105
rc 130
htn 2
dm 2
cad 2
appet 1
pe 1
ane 1
classification 0
dtype: int64

Handle missing values.

In [338...
missing_percentage = df.isnull().mean() * 100
Filter columns with more than 30% (or 50%) missing values
columns_with_missing_30 = missing_percentage[missing_percentage > 30]
Display the columns with more than 30% missing values
print("Columns with more than 30% missing values:")
print(columns_with_missing_30)

Columns with more than 30% missing values:
rbc 38.0
rc 32.5
dtype: float64

Deleting columns that have 30% missing values

In [339...
missing_percentage = df.isnull().mean() * 100
threshold = 30
Filter columns with more than the threshold percentage of missing values
columns_to_drop = missing_percentage[missing_percentage > threshold].index
Drop the columns
df = df.drop(columns=columns_to_drop)
Display the resulting dataframe after removing columns with too many missing values
print(f"Columns dropped: {list(columns_to_drop)}")
print(f"Remaining columns: {df.columns}")

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

In [340...
df = pd.read_csv("kidney_disease12.csv")
df.head()
df.to_csv("clean_dataset.csv")

In [341...
df = pd.read_csv("clean_dataset.csv")

In [342...
len(df.columns)

Out[342...
27

Pre-processing on dataset

In [343...
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
for column in df.select_dtypes(include=['object']).columns:
 df[column] = label_encoder.fit_transform(df[column])
df.head()

Out[343...

	Unnamed: 0	id	age	bp	sg	al	su	rbc	pc	pcc	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	0	0	48.0	80.0	1.020	1.0	0.0	2	1	0	...	32	72	34	1	4	1	0	0	0	0
1	1	1	7.0	50.0	1.020	4.0	0.0	2	1	0	...	26	56	49	0	3	1	0	0	0	0
2	2	2	62.0	80.0	1.010	2.0	3.0	1	1	0	...	19	70	49	0	4	1	1	0	1	0
3	3	3	48.0	70.0	1.005	4.0	0.0	1	0	1	...	20	62	19	1	3	1	1	1	1	0
4	4	4	51.0	80.0	1.010	2.0	0.0	1	1	0	...	23	68	27	0	3	1	0	0	0	0

5 rows × 27 columns

In [360...

df.shape

Out[360...

(400, 26)

Filling missing values

In [344...

```
# Fill missing values in columns with mode
for column in df.select_dtypes(include=['float64']).columns:
    mean_value = df[column].mean() # Get the mode (most frequent value)
    df[column] = df[column].fillna(mean_value)
# Fill missing values in categorical columns with mode
for column in df.select_dtypes(include=['object']).columns:
    mode_value = df[column].mode()[0] # Get the mode (most frequent value)
    df[column] = df[column].fillna(mode_value)
df.isnull().sum()
```

Out[344...

```
Unnamed: 0      0
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
```

2. Handle outliers.

In [346...

```
num_cols = df.select_dtypes(include=[np.number]).columns.tolist()
display(df[num_cols].describe().T)

def iqr_bounds(s, k=1.5):# IQR bounds per numeric column
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

def outlier_report(df, cols=None, k=1.5): # Build an outlier
    if cols is None:
        cols = df.select_dtypes(include=[np.number]).columns
    rows = []
    caps = {}
    for c in cols:
        s = df[c].dropna()
        if s.empty:
            continue
        lo, hi = iqr_bounds(s, k=k)
```

```

        n_lo = (df[c] < lo).sum()
        n_hi = (df[c] > hi).sum()
        frac = (n_lo + n_hi) / len(df)
        rows.append({"column": c, "lower_bound": lo, "upper_bound": hi, "n_below": int(n_lo),
                     "n_above": int(n_hi), "pct_outliers": round(100*frac, 2) })
        caps[c] = (lo, hi)
    rep = pd.DataFrame(rows).sort_values("pct_outliers", ascending=False)
    return rep, caps
    outliers_only
```

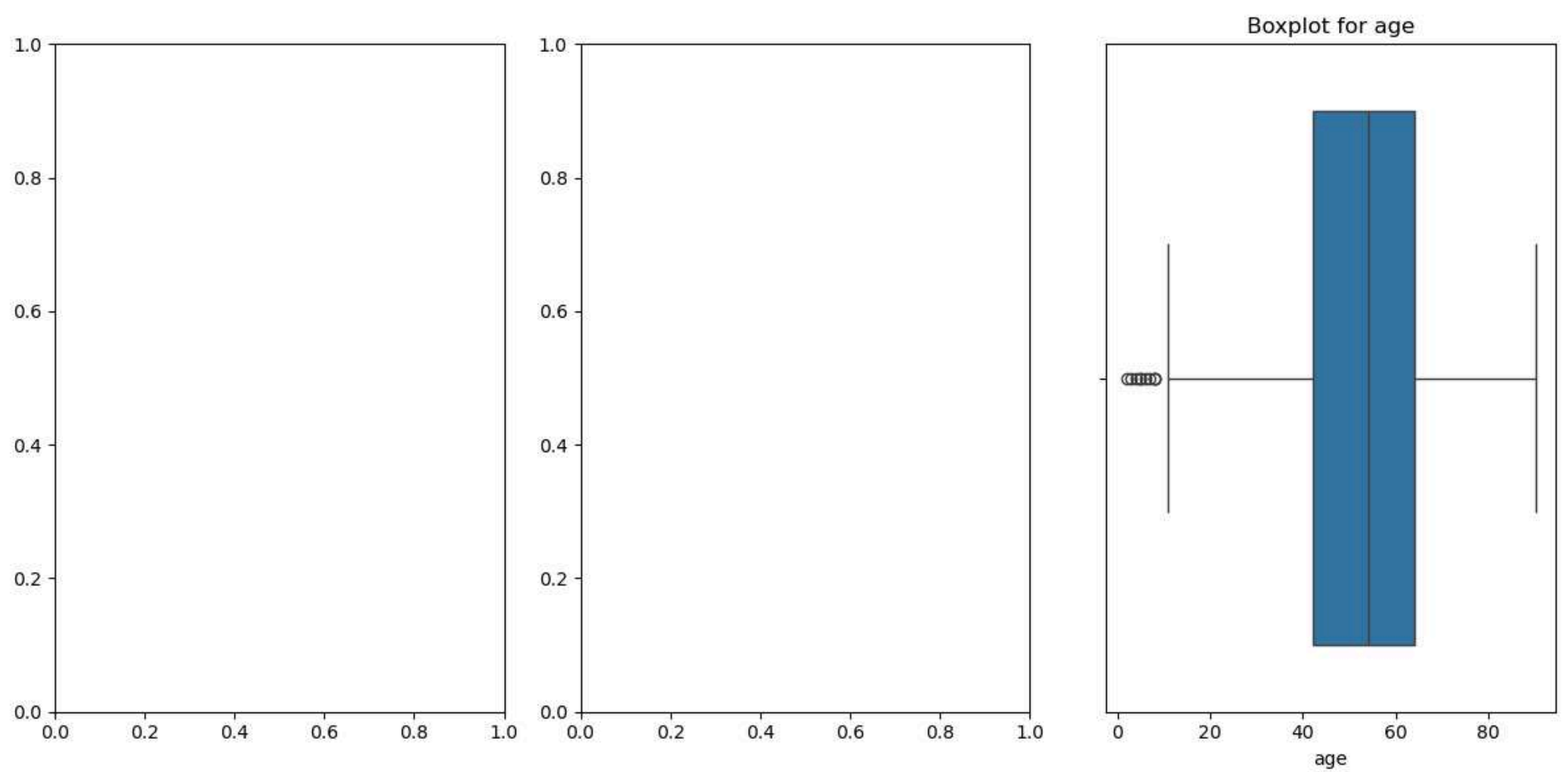
	count	mean	std	min	25%	50%	75%	max
Unnamed: 0	400.0	199.500000	115.614301	0.000	99.750	199.500000	299.250000	399.000
id	400.0	199.500000	115.614301	0.000	99.750	199.500000	299.250000	399.000
age	400.0	51.483376	16.974966	2.000	42.000	54.000000	64.000000	90.000
bp	400.0	76.469072	13.476298	50.000	70.000	78.234536	80.000000	180.000
sg	400.0	1.017408	0.005369	1.005	1.015	1.017408	1.020000	1.025
al	400.0	1.016949	1.272318	0.000	0.000	1.000000	2.000000	5.000
su	400.0	0.450142	1.029487	0.000	0.000	0.000000	0.450142	5.000
rbc	400.0	1.262500	0.655491	0.000	1.000	1.000000	2.000000	2.000
pc	400.0	0.972500	0.593823	0.000	1.000	1.000000	1.000000	2.000
pcc	400.0	0.125000	0.360138	0.000	0.000	0.000000	0.000000	2.000
ba	400.0	0.075000	0.299331	0.000	0.000	0.000000	0.000000	2.000
bgr	400.0	148.036517	74.782634	22.000	101.000	126.000000	150.000000	490.000
bu	400.0	57.425722	49.285887	1.500	27.000	44.000000	61.750000	391.000
sc	400.0	3.072454	5.617490	0.400	0.900	1.400000	3.072454	76.000
sod	400.0	137.528754	9.204273	4.500	135.000	137.528754	141.000000	163.000
pot	400.0	4.627244	2.819783	2.500	4.000	4.627244	4.800000	47.000
hemo	400.0	12.526437	2.716171	3.100	10.875	12.526437	14.625000	17.800
pcv	400.0	29.852500	10.529145	0.000	22.000	30.000000	39.250000	44.000
wc	400.0	64.430000	28.149620	0.000	49.000	71.000000	92.000000	92.000
rc	400.0	34.837500	13.397381	0.000	26.000	36.000000	49.000000	49.000
htn	400.0	0.377500	0.495588	0.000	0.000	0.000000	1.000000	2.000
dm	400.0	3.310000	0.591417	0.000	3.000	3.000000	4.000000	5.000
cad	400.0	1.090000	0.319618	0.000	1.000	1.000000	1.000000	3.000
appet	400.0	0.210000	0.413918	0.000	0.000	0.000000	0.000000	2.000
pe	400.0	0.195000	0.402965	0.000	0.000	0.000000	0.000000	2.000
ane	400.0	0.155000	0.369210	0.000	0.000	0.000000	0.000000	2.000
classification	400.0	0.755000	0.968152	0.000	0.000	0.000000	2.000000	2.000

Data Visulization

In [347...

```

import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(12, 6))
for i, col in enumerate(df.columns[:3]): # Only plotting the first 3 features
    plt.subplot(1, 3, i+1) # Create subplots for each feature
    sns.boxplot(x=df[col])
plt.title(f"Boxplot for {col}")
plt.tight_layout()
plt.show()
```

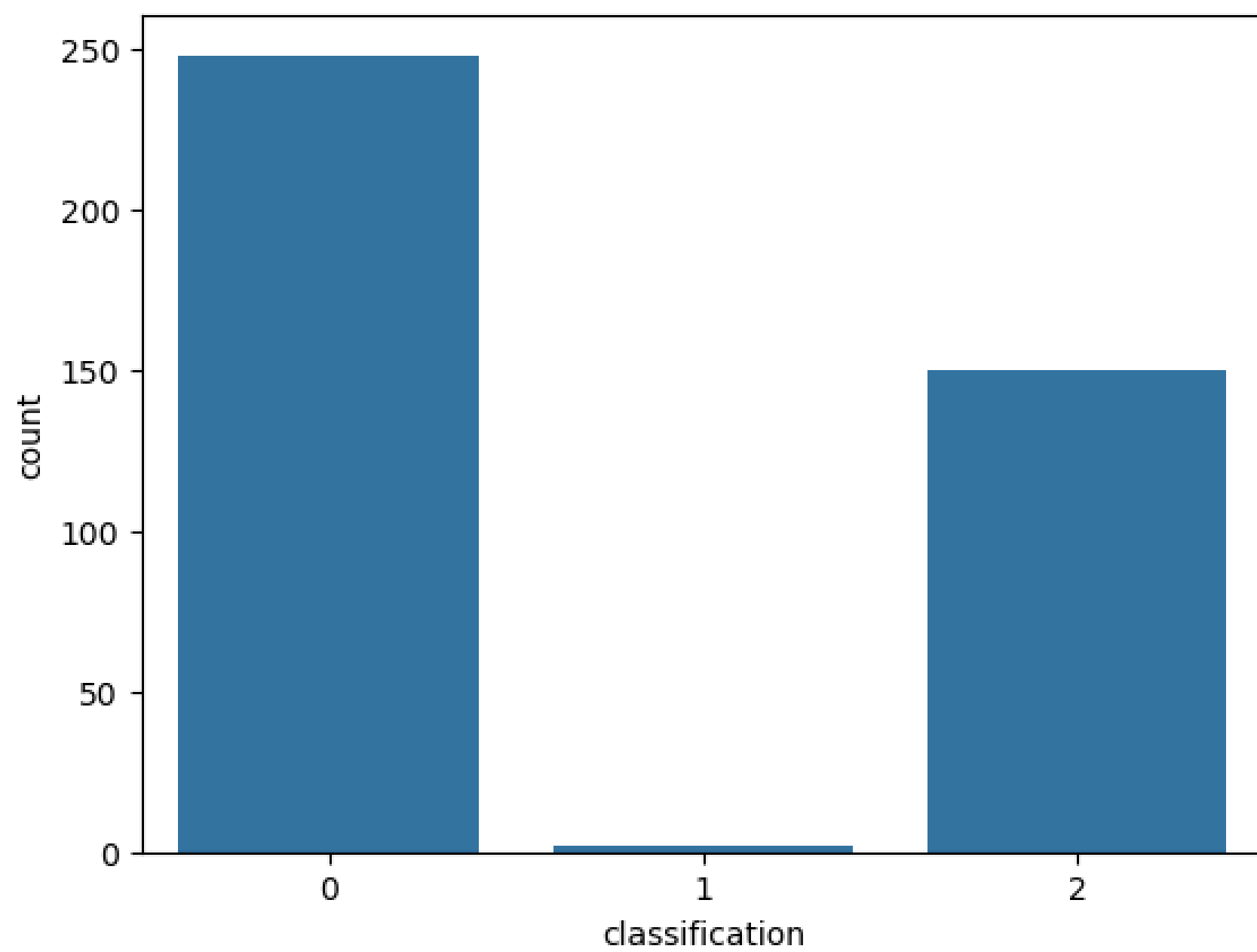


```
In [348...] df['classification'].value_counts()
```

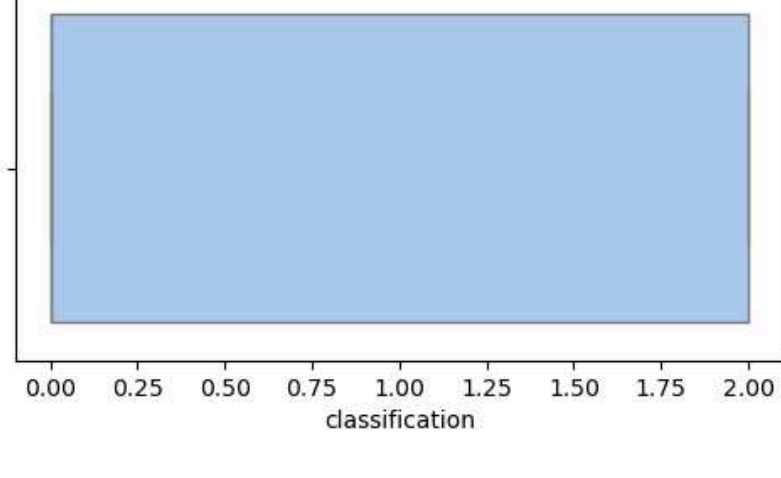
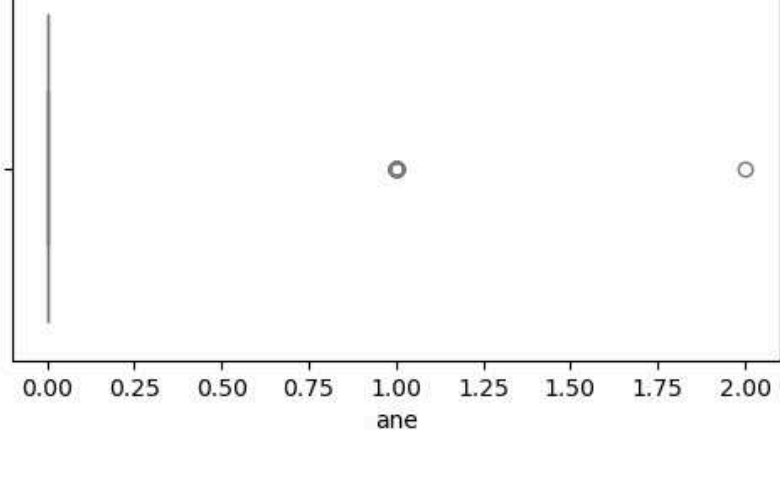
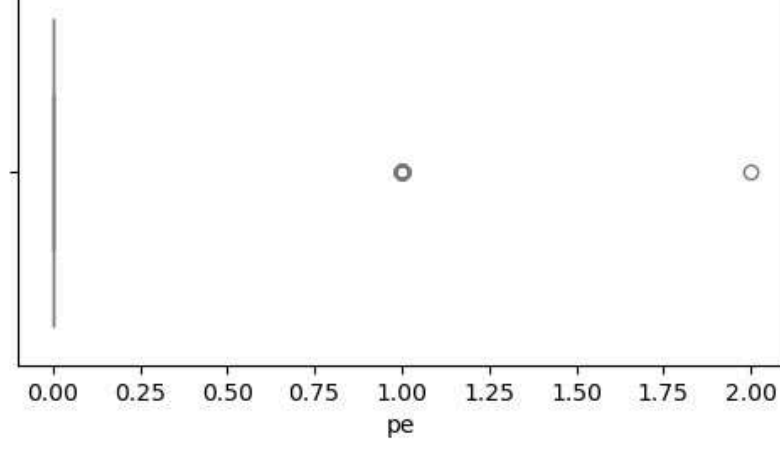
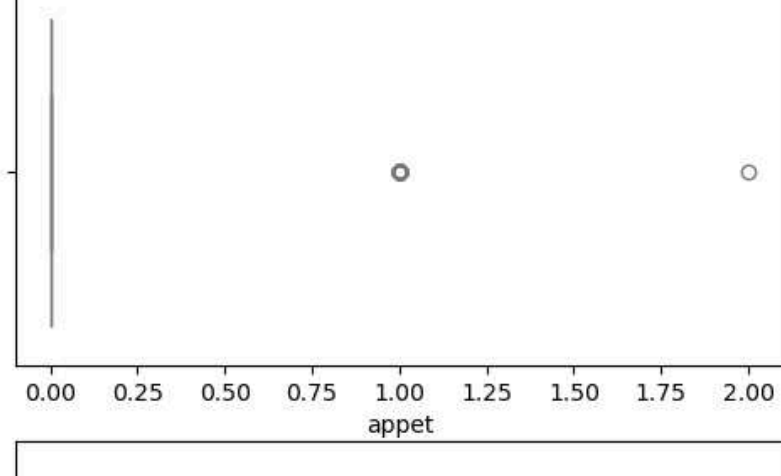
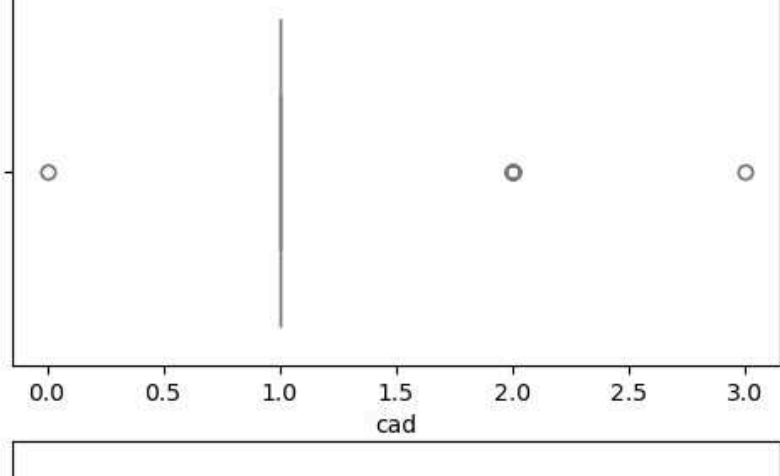
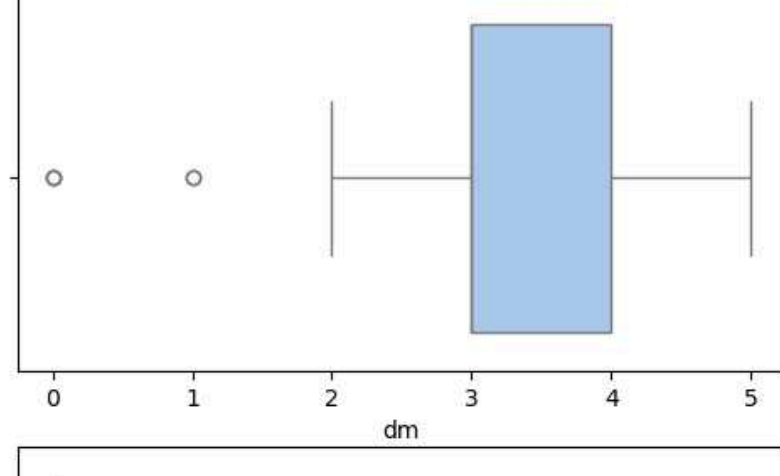
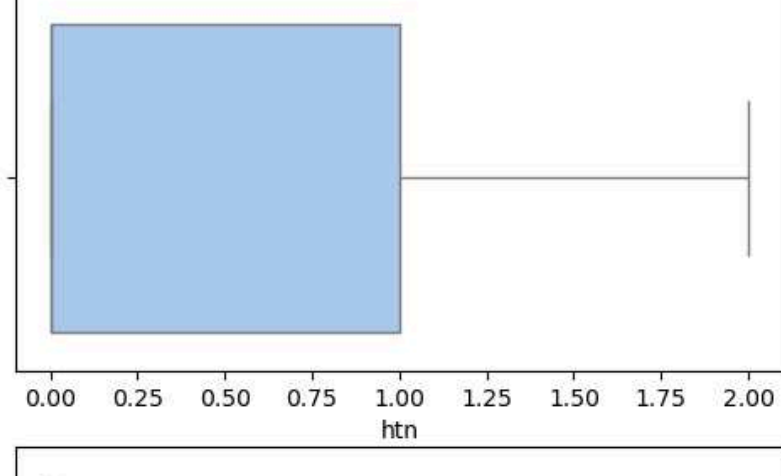
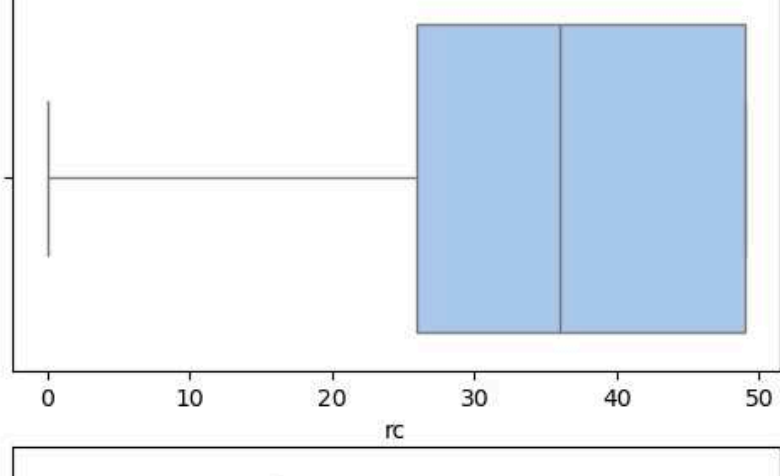
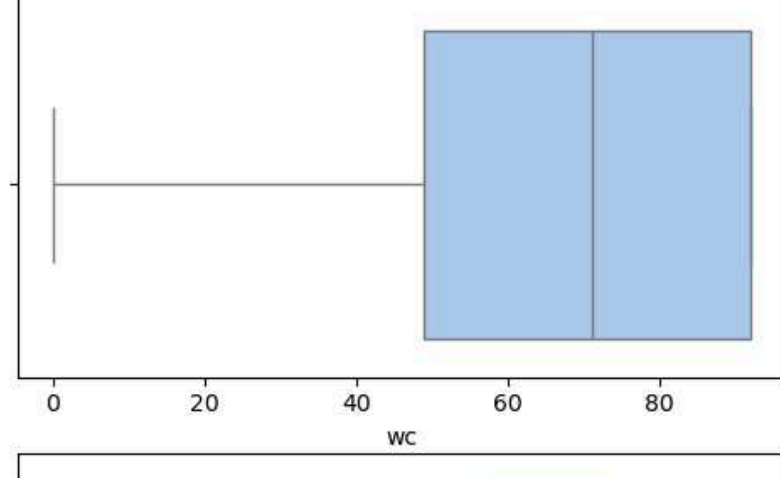
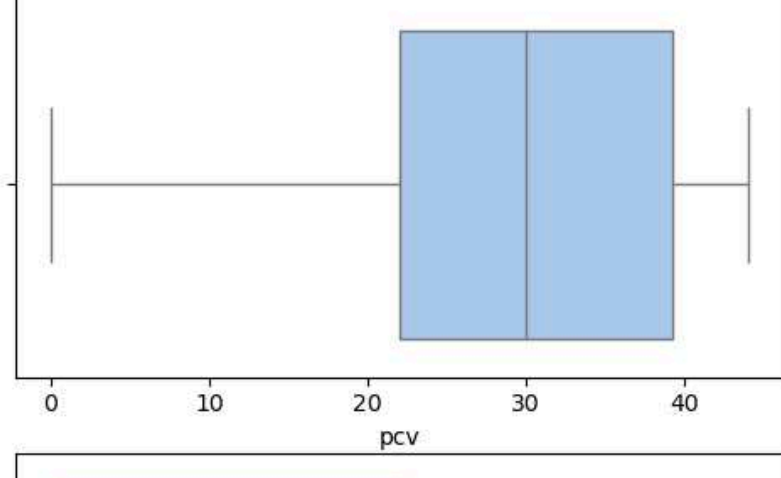
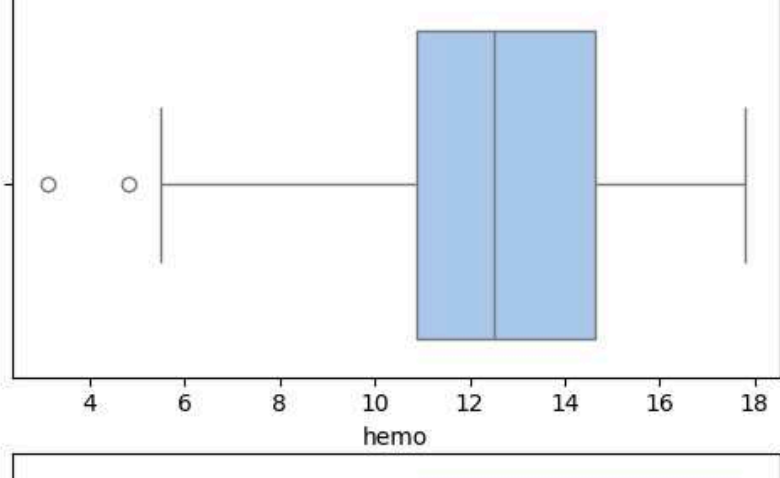
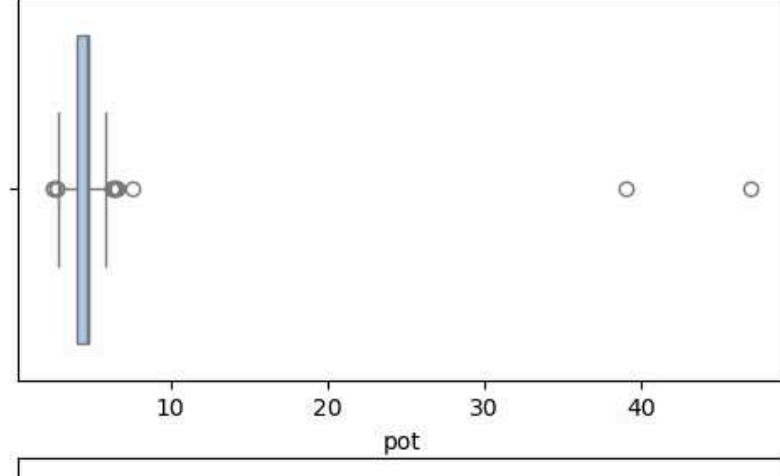
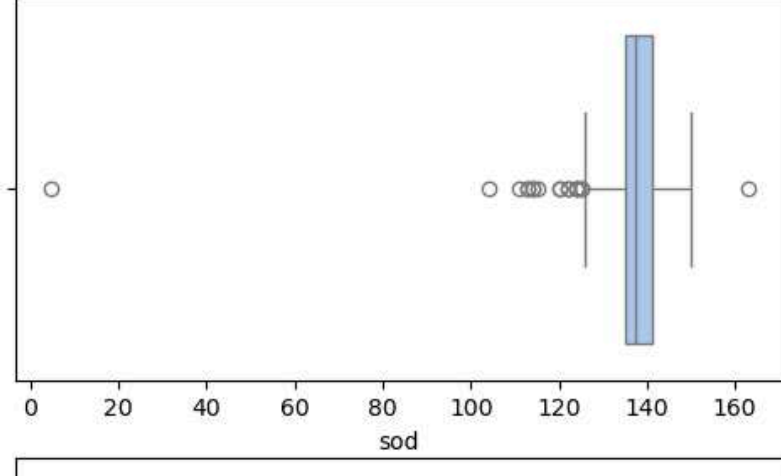
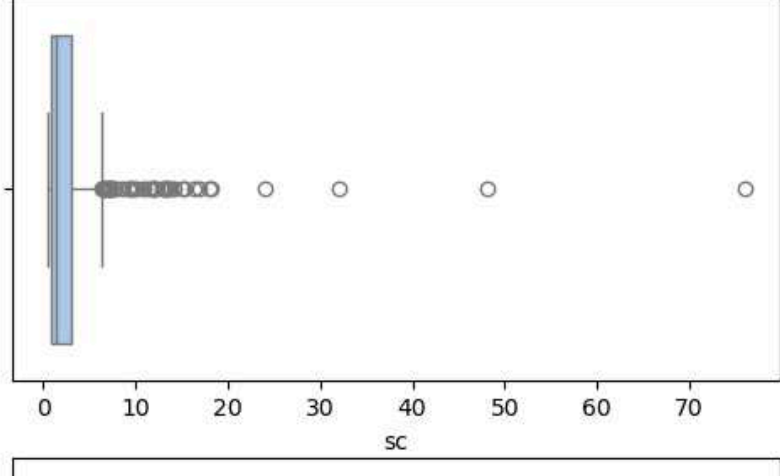
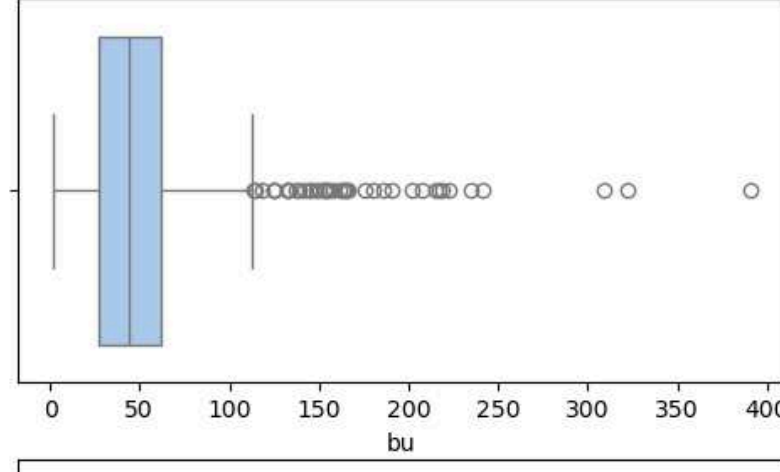
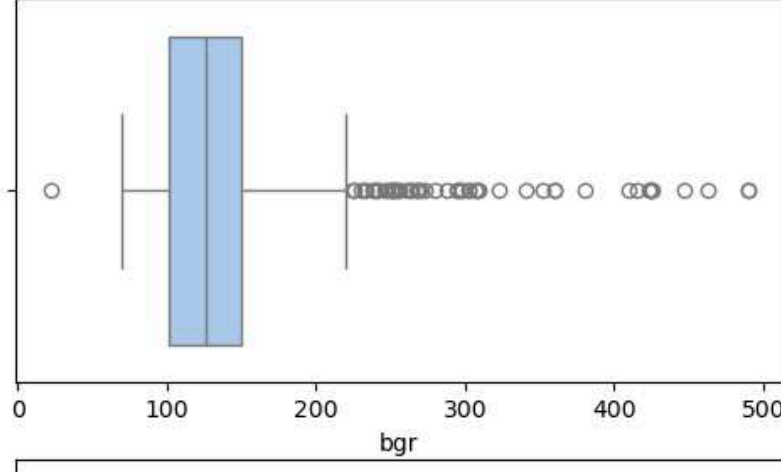
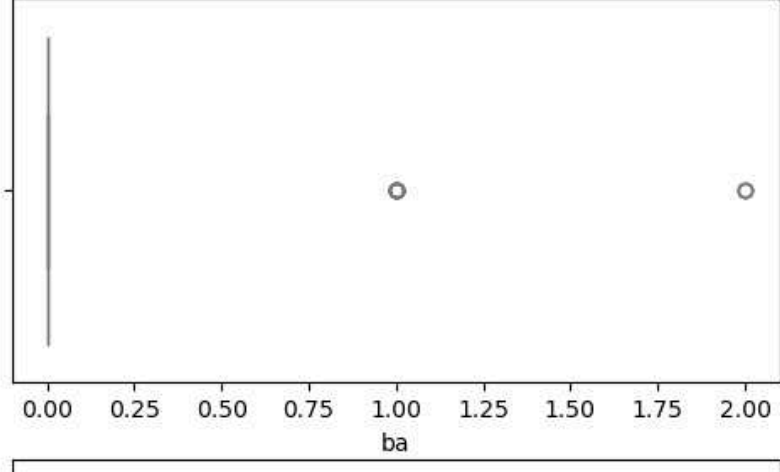
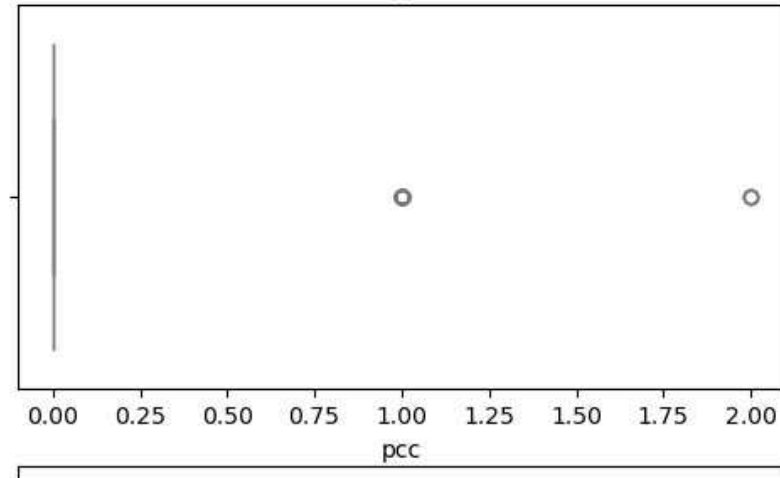
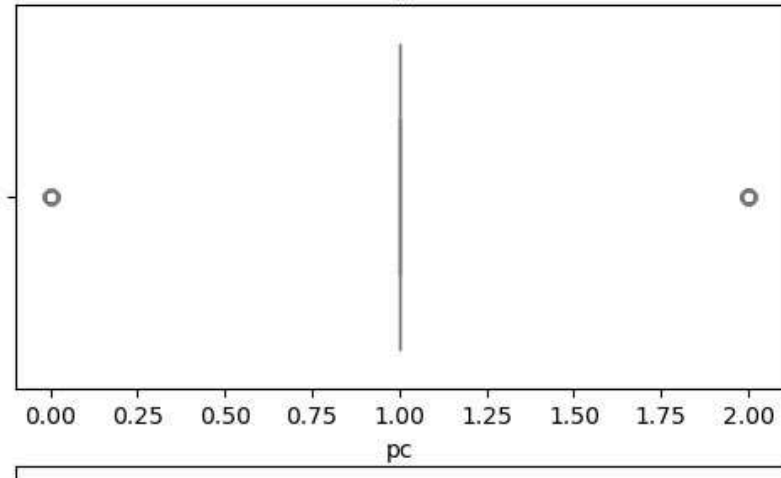
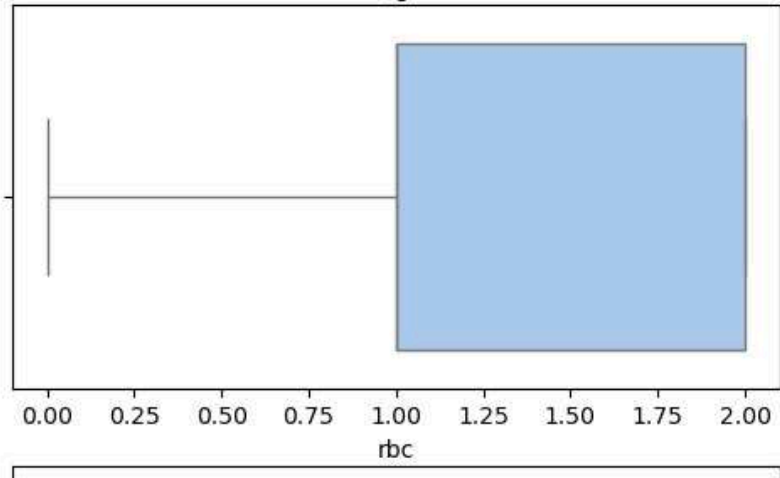
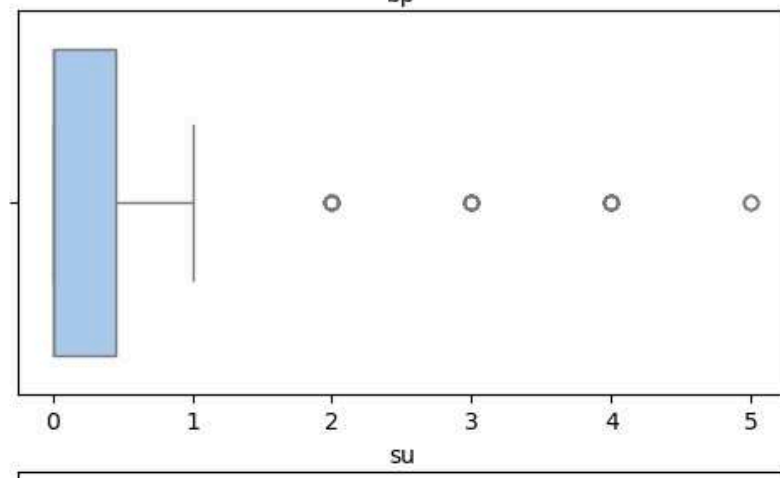
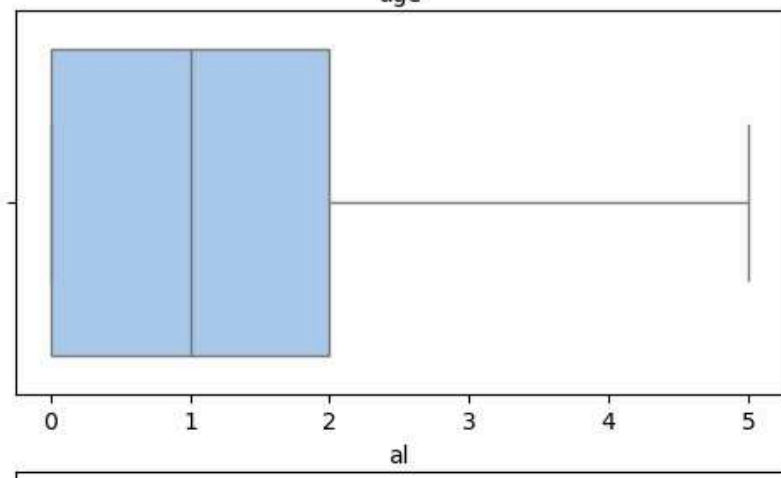
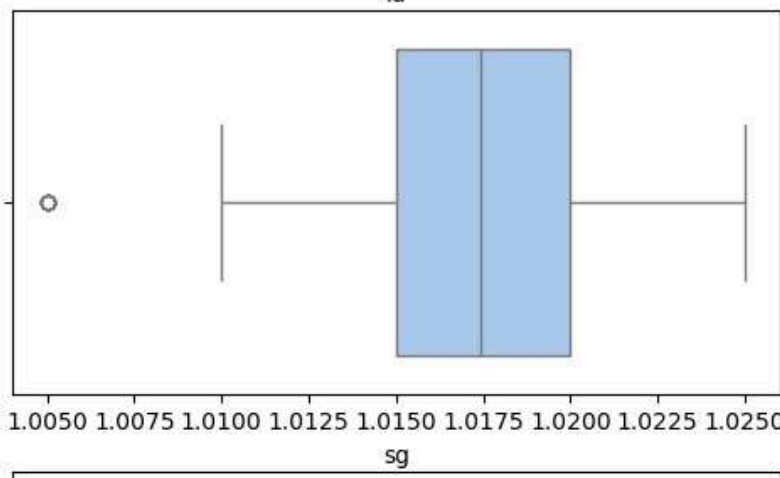
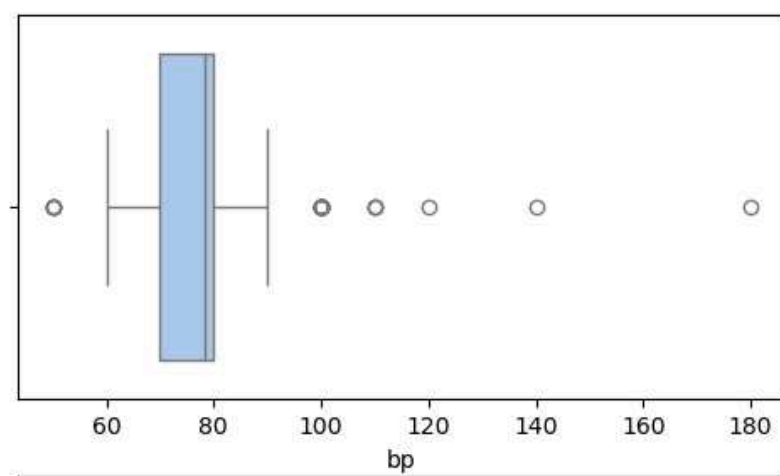
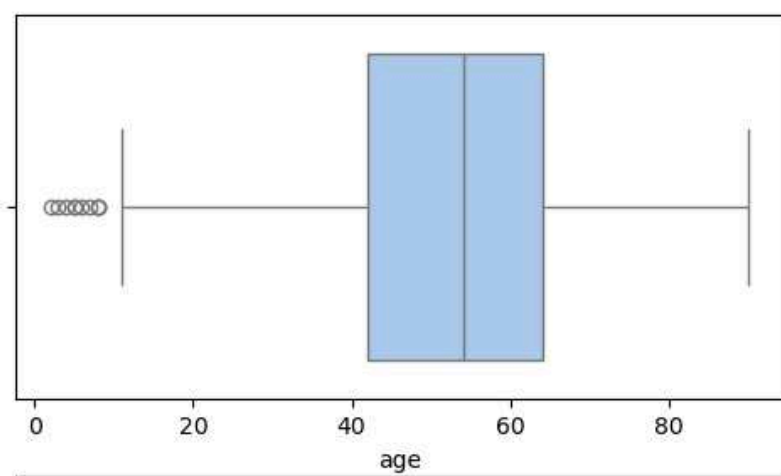
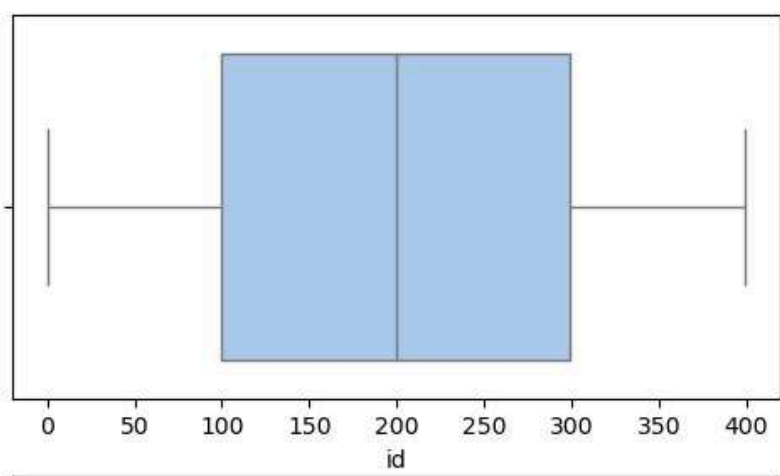
```
Out[348...] classification
0      248
2      150
1         2
Name: count, dtype: int64
```

```
In [349...] sns.countplot(x='classification', data=df)
```

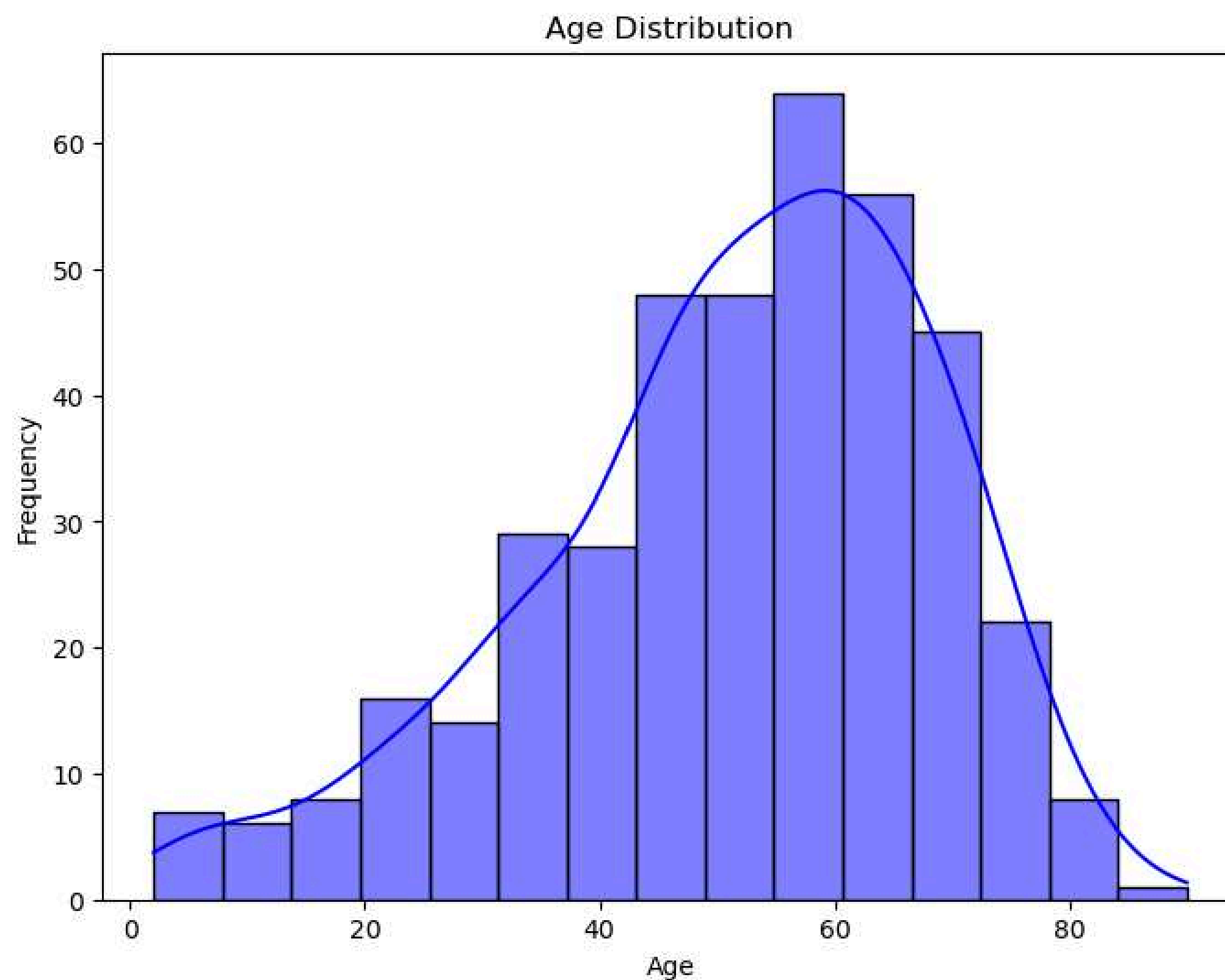
```
Out[349...] <Axes: xlabel='classification', ylabel='count'>
```



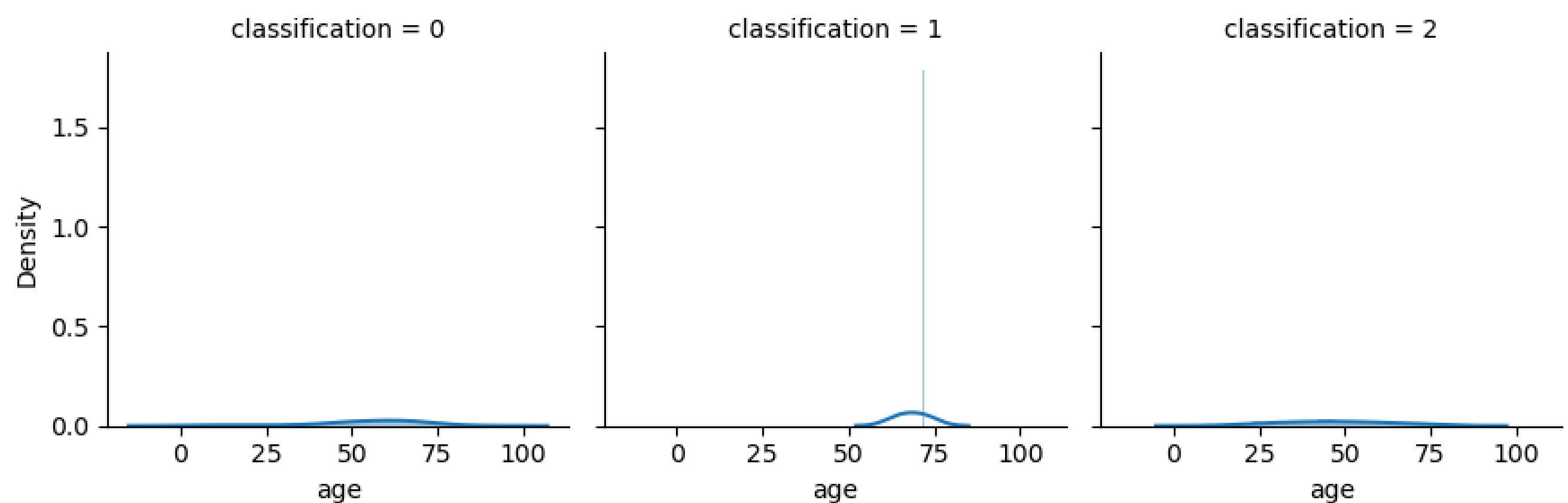
```
In [350...] plt.figure(figsize=(20,35))
plotnumber =1
for column in df.columns[1:]:
    ax = plt.subplot(10,3,plotnumber)
    sns.boxplot(data =df, x = column, palette='pastel')
    plt.xlabel(column)
    plotnumber+=1
plt.show()
```



```
In [351... cat_cols = [col for col in df.columns if df[col].dtype=='object']
num_cols = [col for col in df.columns if df[col].dtype!='object']
for col in cat_cols:
    print(f'{col} has {df[col].unique()} values')
# Distribution of Age
plt.figure(figsize=(8, 6))
sns.histplot(df['age'], bins=15, kde=True, color='blue')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```

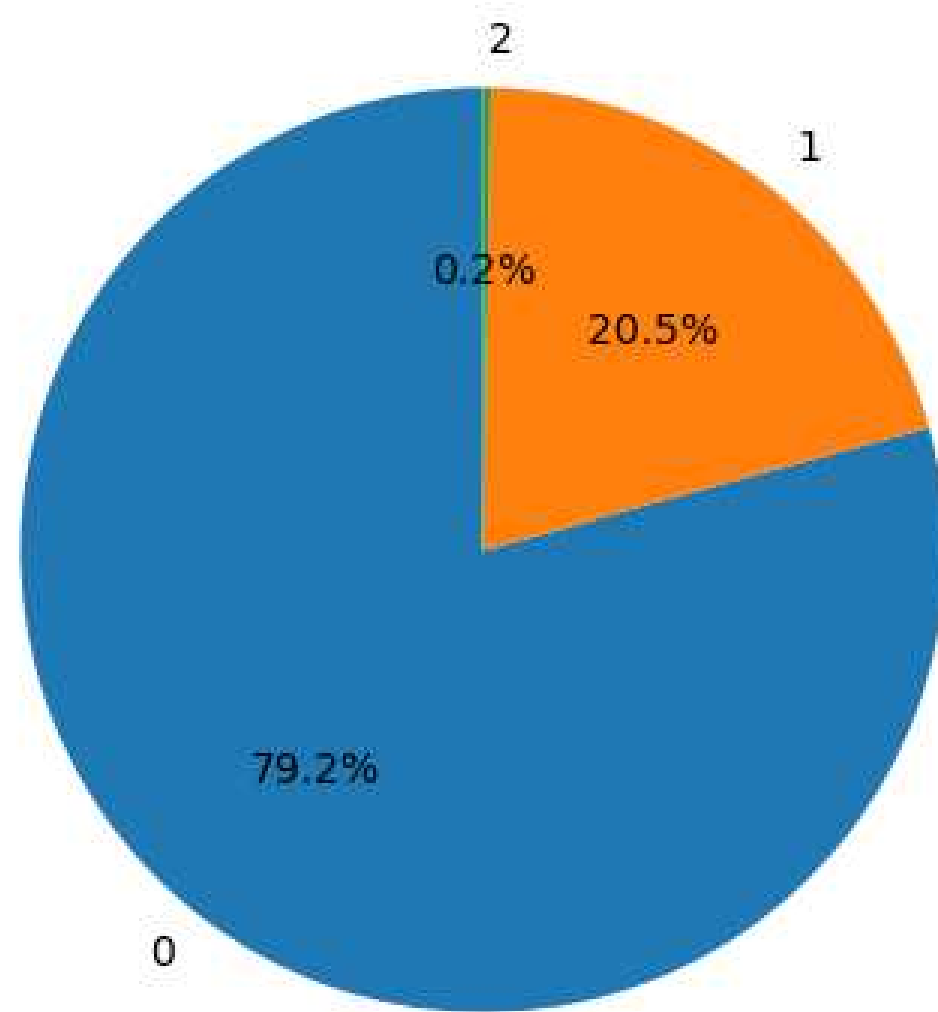


```
In [352... g = sns.FacetGrid(df,col="classification")
g.map(sns.distplot,"age", bins=25)
plt.show()
```

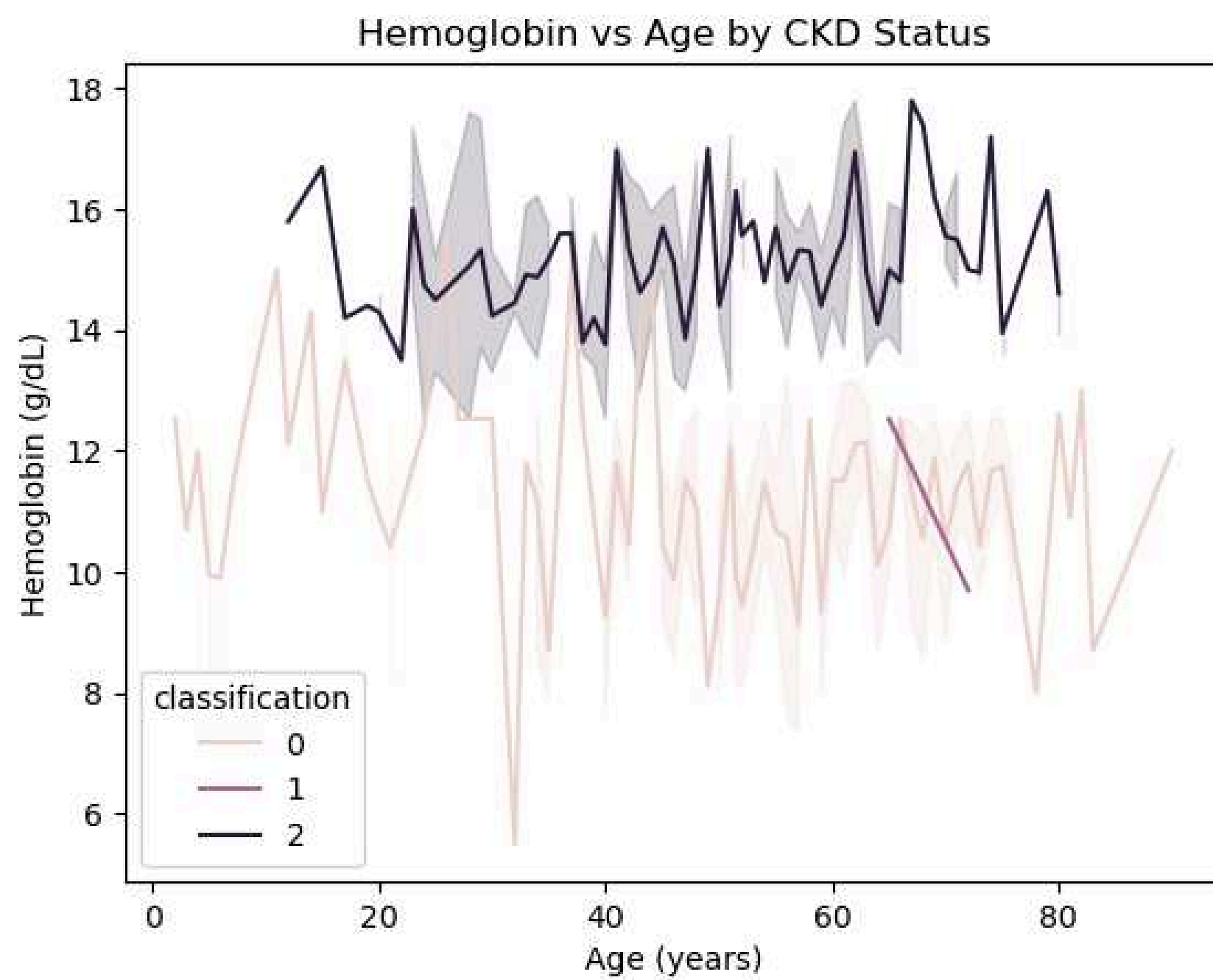


```
In [354... appet_counts = df['appet'].value_counts()
plt.figure(figsize=(5,5))
plt.pie(appet_counts, labels=appet_counts.index, autopct='%1.1f%%', startangle=90)
plt.title("Appetite Status Distribution")
plt.show()
```

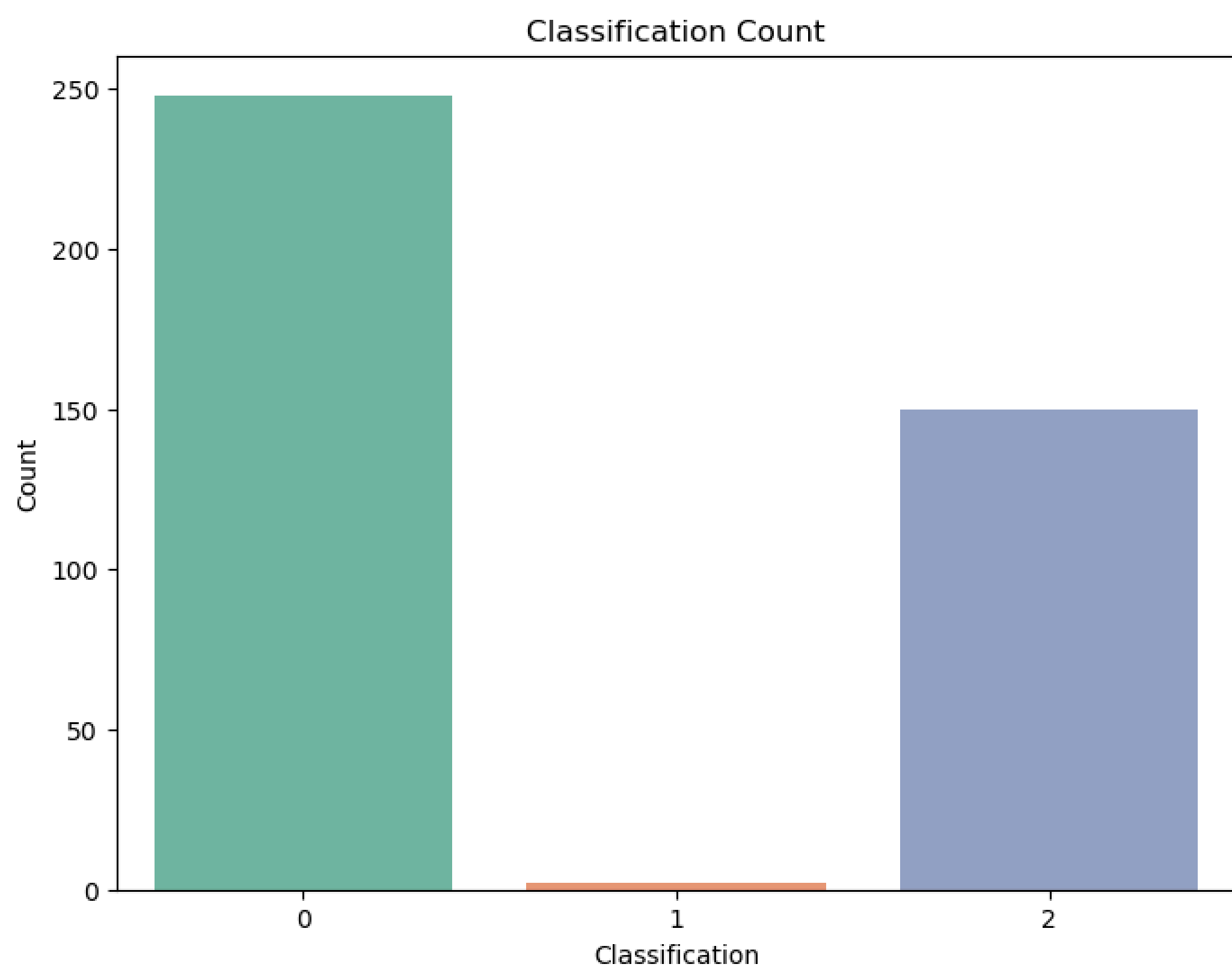
Appetite Status Distribution



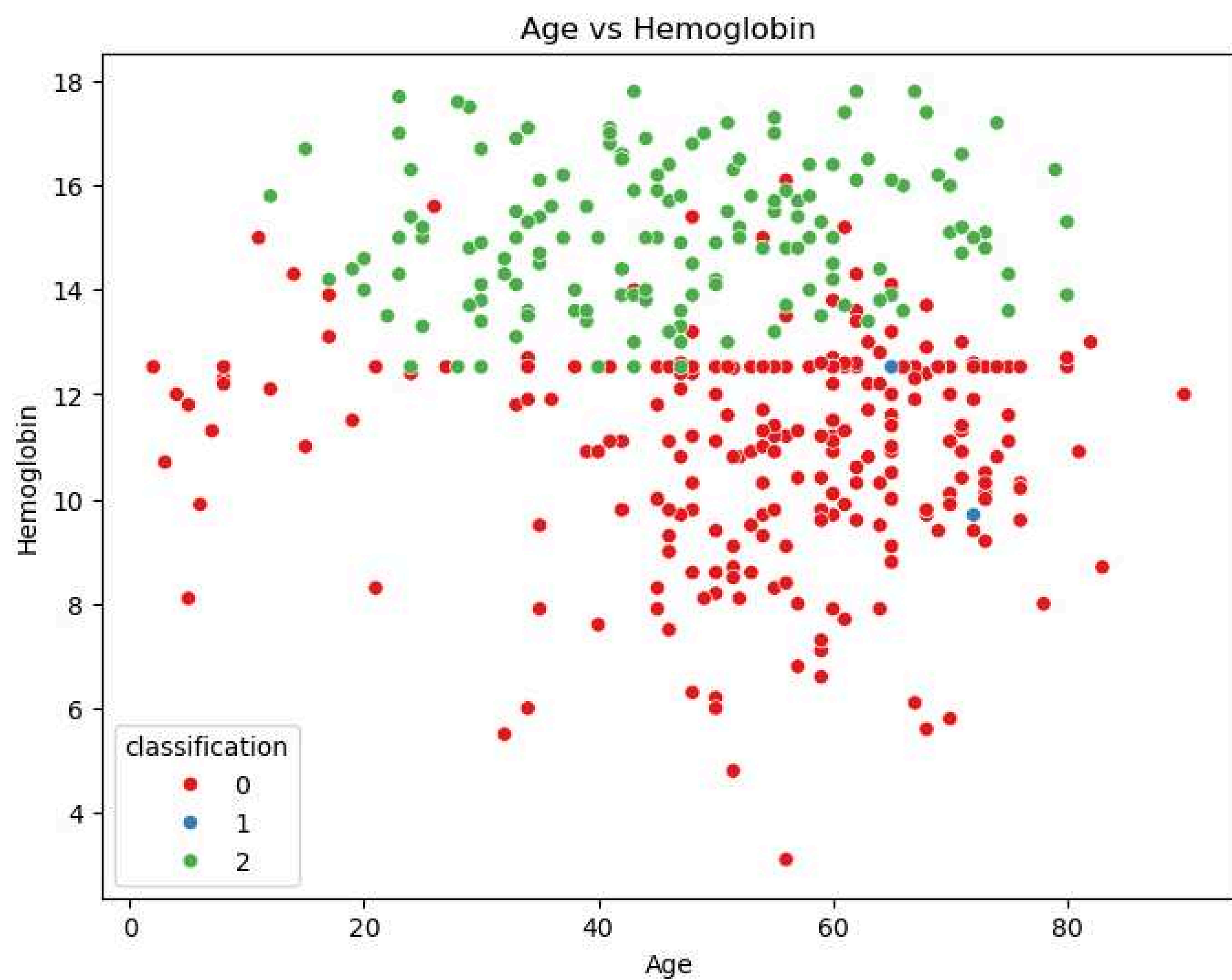
```
In [355... sns.lineplot(x='age', y='hemo', hue='classification', data=df)
plt.title("Hemoglobin vs Age by CKD Status")
plt.xlabel("Age (years)")
plt.ylabel("Hemoglobin (g/dL)")
plt.show()
```



```
In [356... # 3. Countplot for Classification
plt.figure(figsize=(8, 6))
sns.countplot(x='classification', data=df, palette='Set2')
plt.title('Classification Count')
plt.xlabel('Classification')
plt.ylabel('Count')
plt.show()
```



```
In [357... plt.figure(figsize=(8, 6))
sns.scatterplot(x='age', y='hemo', data=df, hue='classification', palette='Set1')
plt.title('Age vs Hemoglobin')
plt.xlabel('Age')
plt.ylabel('Hemoglobin')
plt.show()
```



```
In [358... # Boxplot for Hemoglobin by Classification
plt.figure(figsize=(8, 6))
sns.boxplot(x='classification', y='hemo', data=df, palette='Set1')
plt.title('Hemoglobin Levels by Classification')
plt.xlabel('Classification')
plt.ylabel('Hemoglobin')
plt.show()
```

