21BIT0389 Pavithra S Assingment 4

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

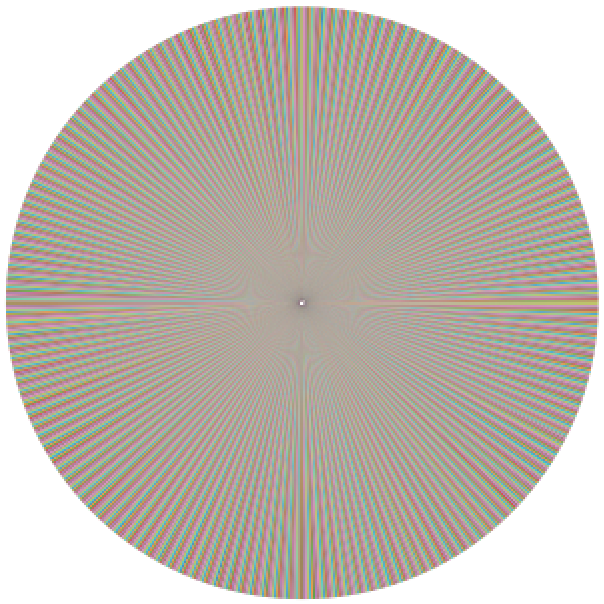
```
df=pd.read_csv('/content/winequality-red.csv')
df
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

1599 rows × 12 columns

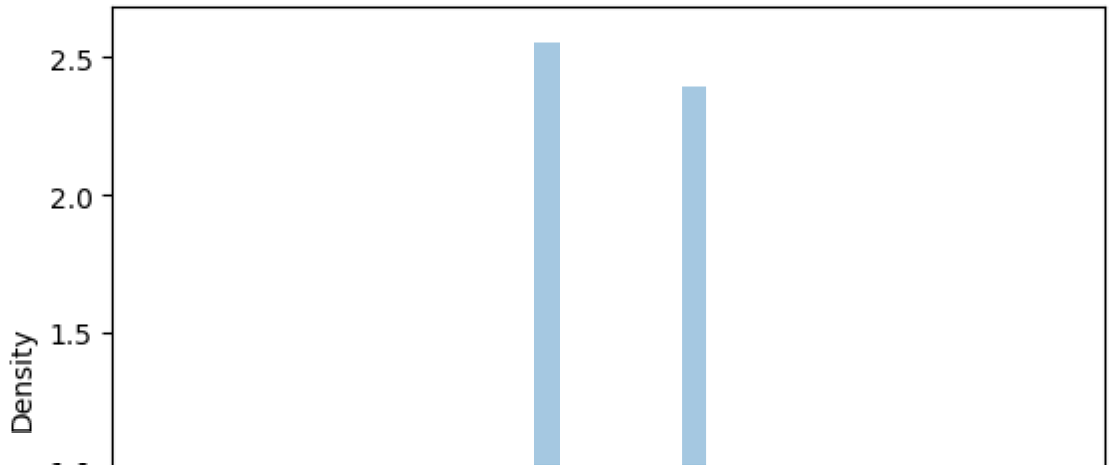
```
plt.pie(df.quality)
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



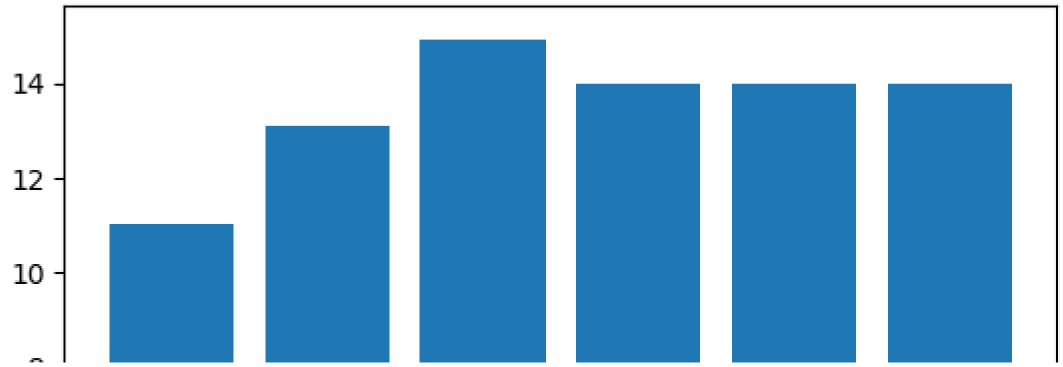
```
sns.distplot(df.quality)
```

```
<ipython-input-4-e8684199aa87>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df.quality)
<Axes: xlabel='quality', ylabel='Density'>
```



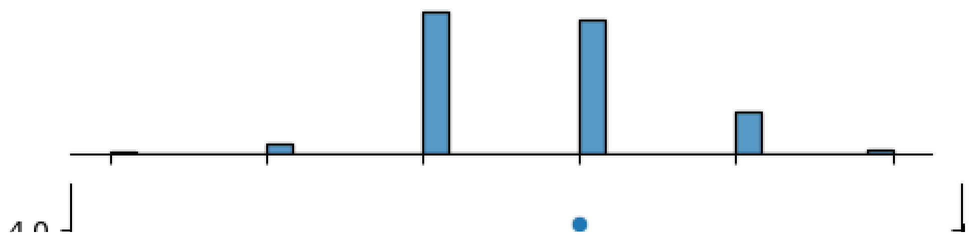
```
plt.bar(df.quality,df.alcohol)
```

```
<BarContainer object of 1599 artists>
```



```
sns.jointplot(x='quality',y='pH',data=df)
```

```
<seaborn.axisgrid.JointGrid at 0x7c9175c7bd60>
```



```
sns.heatmap(df.corr())
```

```
    <Axes: >
```

```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7c916a514a30>
```

```
df.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	dens
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003

```
df.isnull().any()
```

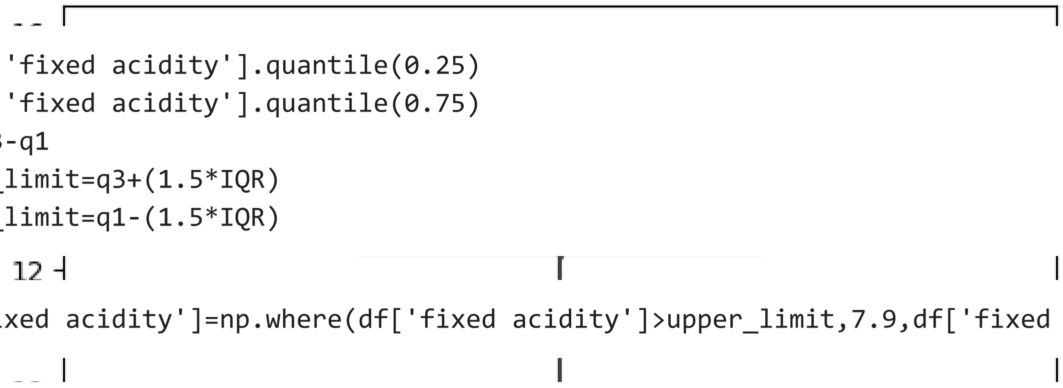
```
fixed acidity           False
volatile acidity        False
citric acid             False
residual sugar          False
chlorides               False
free sulfur dioxide     False
total sulfur dioxide    False
density                 False
pH                      False
sulphates               False
alcohol                 False
quality                 False
dtype: bool
```

```
df.median()
```

```
fixed acidity            7.90000
volatile acidity         0.52000
citric acid              0.26000
residual sugar           2.20000
chlorides                0.07900
free sulfur dioxide     14.00000
total sulfur dioxide    38.00000
density                  0.99675
pH                       3.31000
sulphates                0.62000
alcohol                 10.20000
quality                  6.00000
dtype: float64
```

```
sns.boxplot(df['fixed acidity'])
```

```
<Axes: >
```

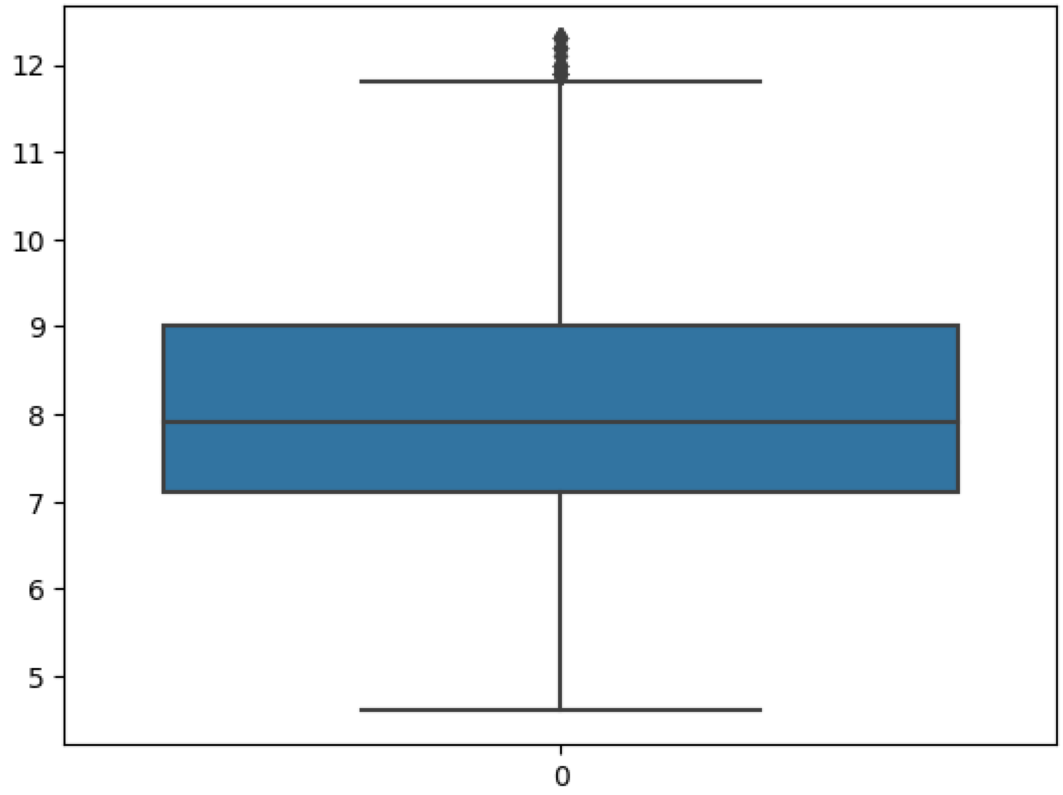


```
q1=df['fixed acidity'].quantile(0.25)
q3=df['fixed acidity'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+(1.5*IQR)
lower_limit=q1-(1.5*IQR)
```

```
df['fixed acidity']=np.where(df['fixed acidity']>upper_limit,7.9,df['fixed acidity'
```

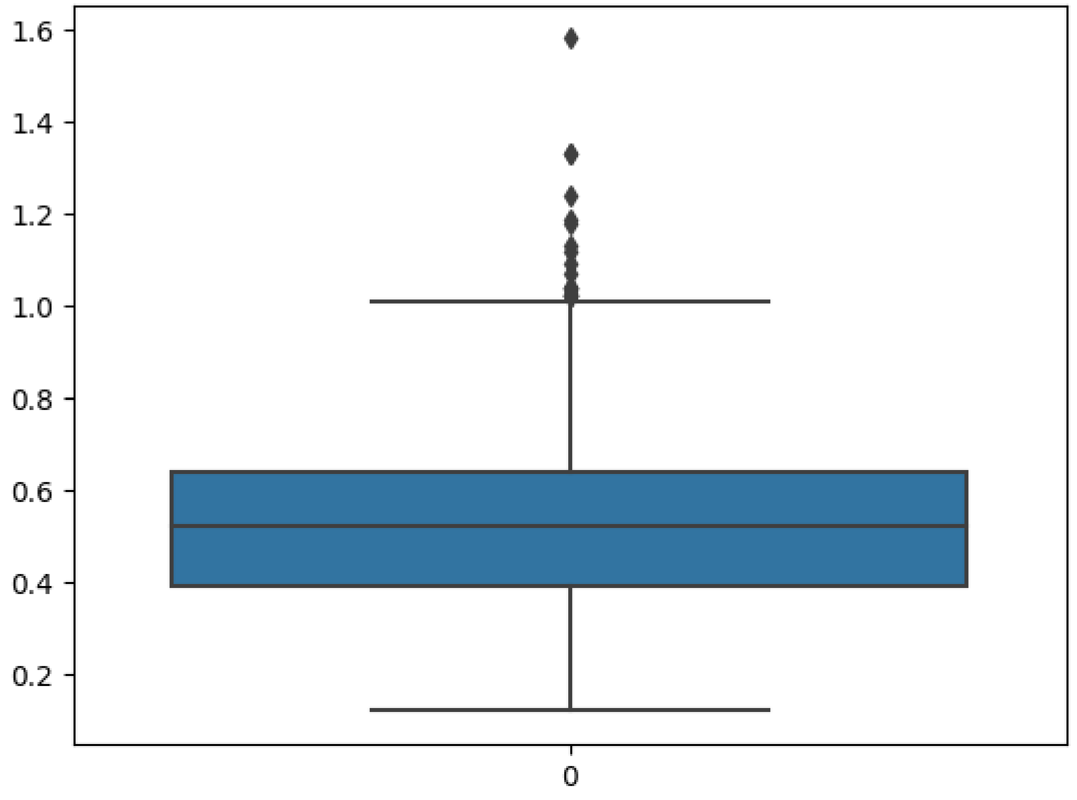
```
sns.boxplot(df['fixed acidity'])
```

```
<Axes: >
```



```
sns.boxplot(df['volatile acidity'])
```

```
<Axes: >
```

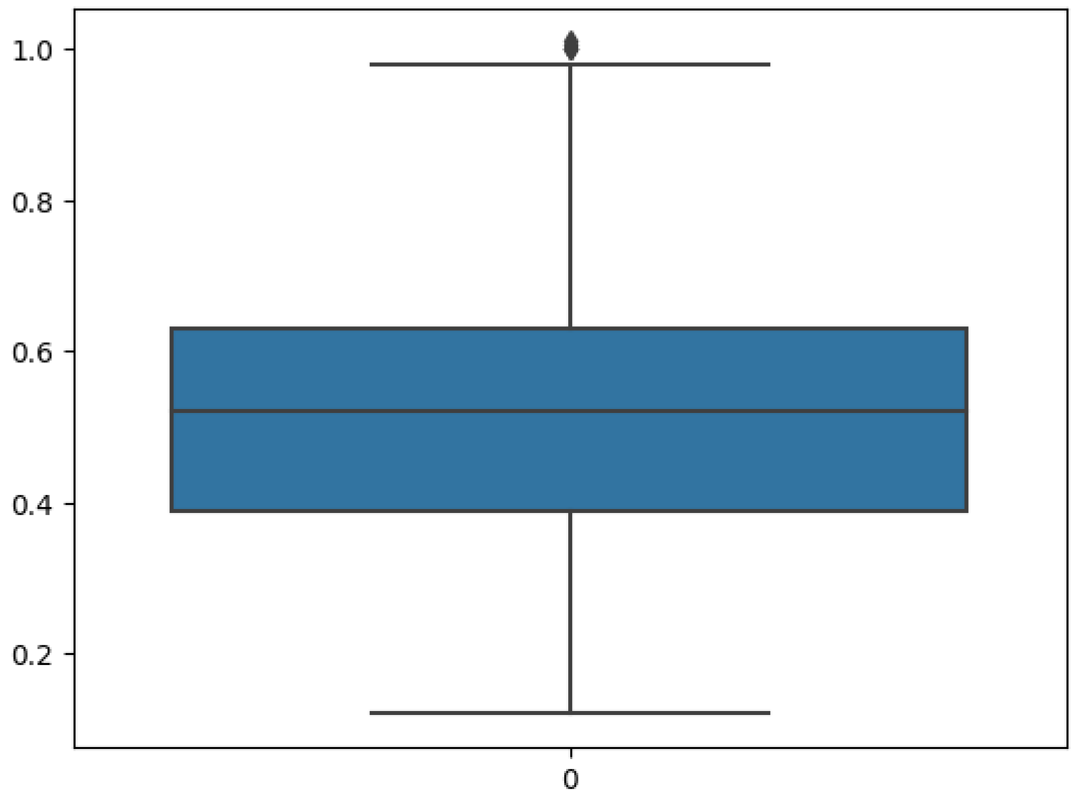


```
q1=df['volatile acidity'].quantile(0.25)
q3=df['volatile acidity'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+(1.5*IQR)
lower_limit=q1-(1.5*IQR)
```

```
df['volatile acidity']=np.where(df['volatile acidity']>upper_limit,0.52,df['volatil
```

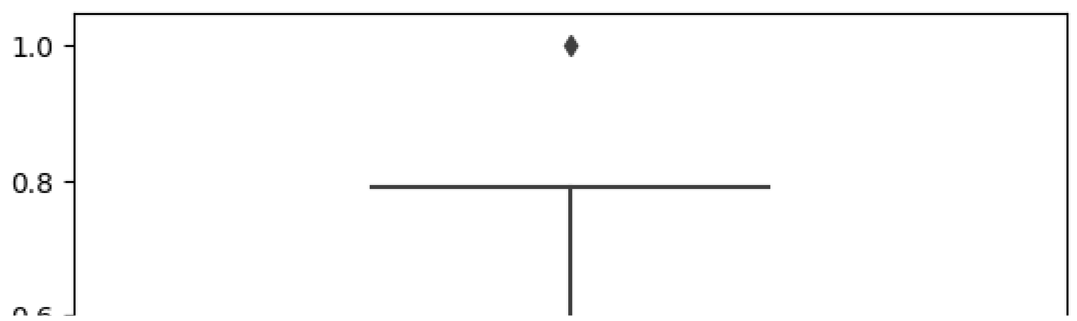
```
sns.boxplot(df['volatile acidity'])
```

```
<Axes: >
```



```
sns.boxplot(df['citric acid'])
```

```
<Axes: >
```

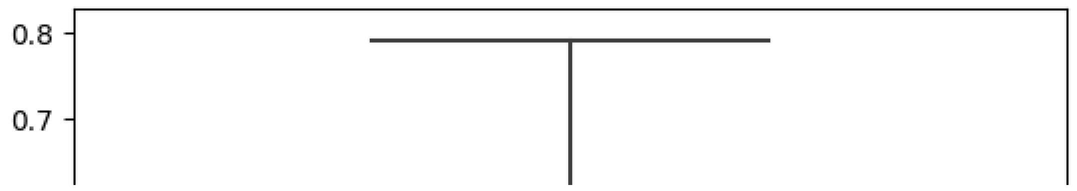


```
q1=df['citric acid'].quantile(0.25)
q3=df['citric acid'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

```
df['citric acid'] = np.where(df['citric acid']>upper_limit,0.26,df['citric acid'])
```

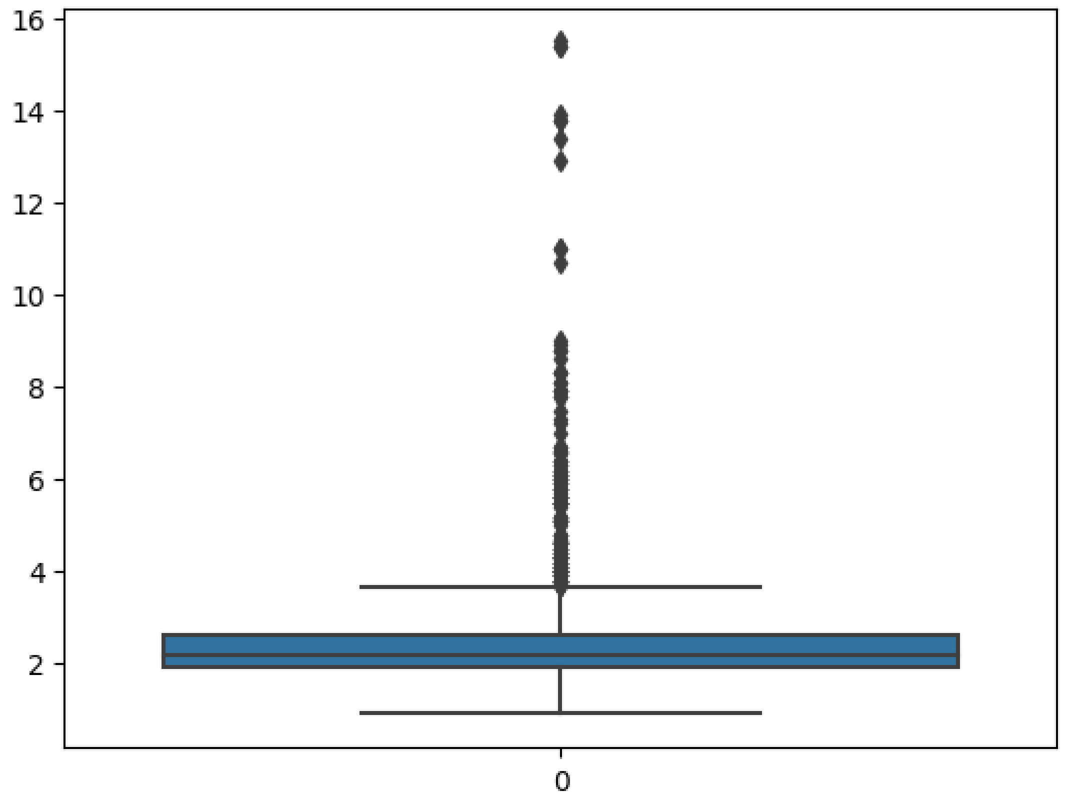
```
sns.boxplot(df['citric acid'])
```

```
<Axes: >
```



```
sns.boxplot(df['residual sugar'])
```

```
<Axes: >
```

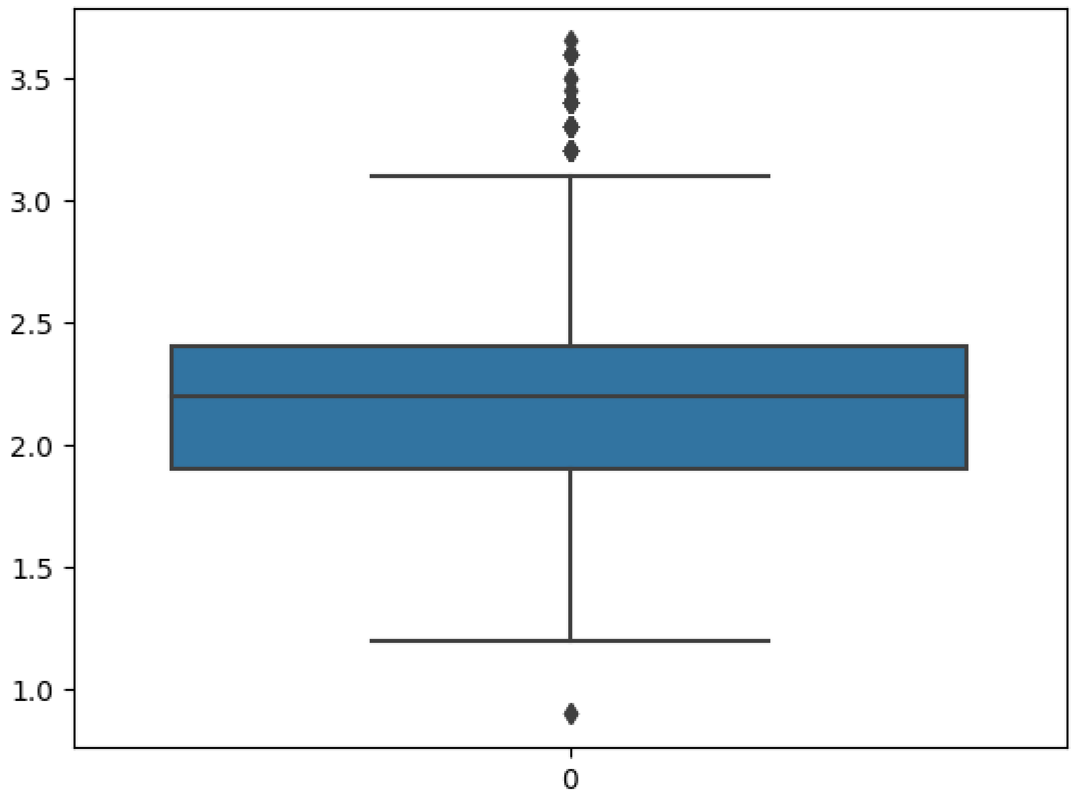


```
q1=df['residual sugar'].quantile(0.25)
q3=df['residual sugar'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

```
df['residual sugar'] = np.where(df['residual sugar']>upper_limit,2.2,df['residual sugar'])
```

```
sns.boxplot(df['residual sugar'])
```

```
<Axes: >
```

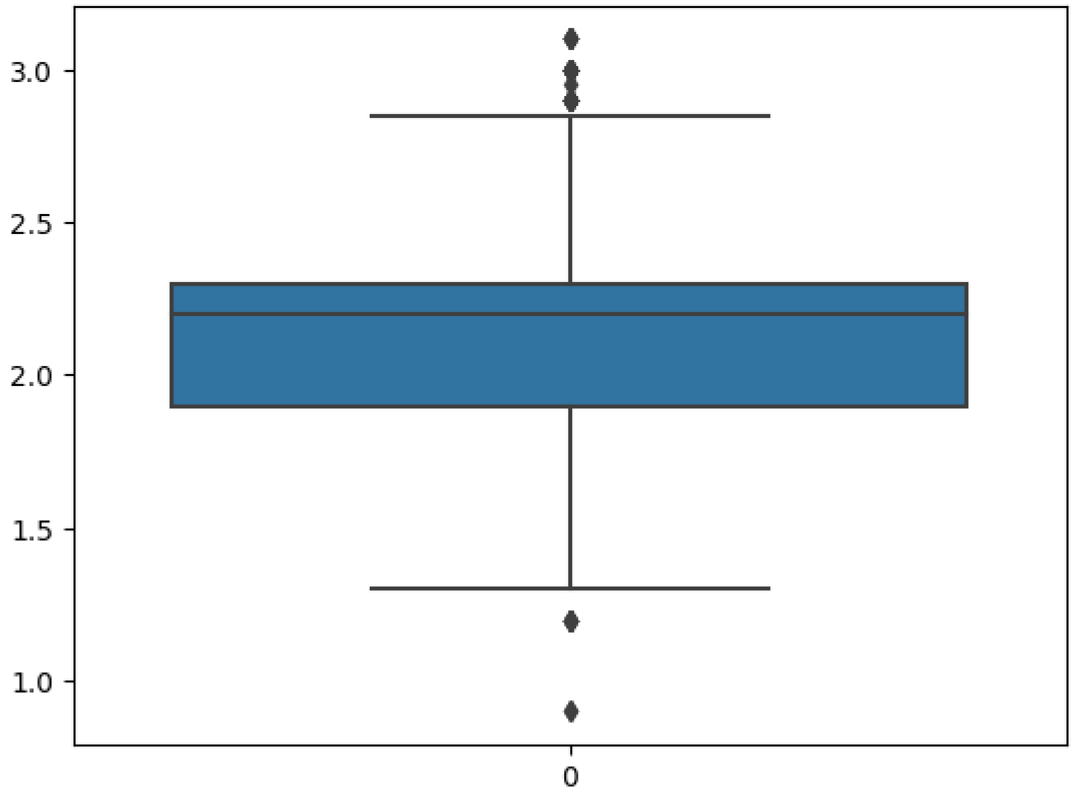


```
q1=df['residual sugar'].quantile(0.25)
q3=df['residual sugar'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

```
df['residual sugar'] = np.where(df['residual sugar']>upper_limit,2.2,df['residual sugar'])
```

```
sns.boxplot(df['residual sugar'])
```

```
<Axes: >
```

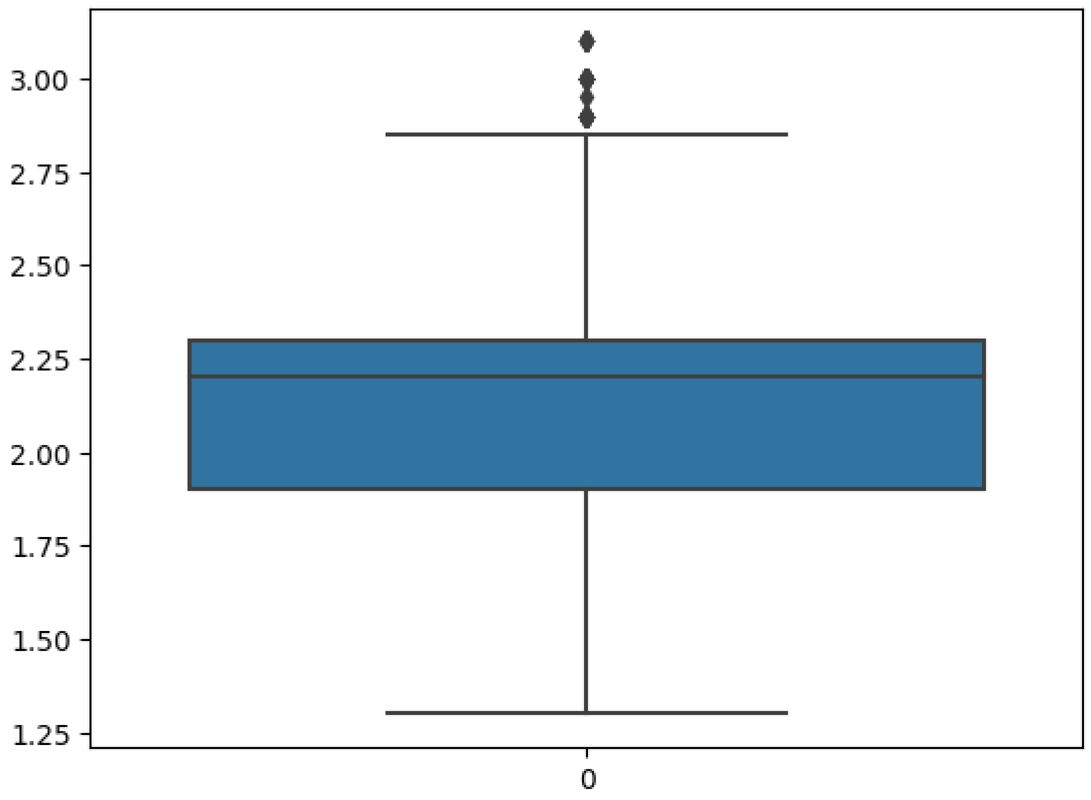


```
q1=df['residual sugar'].quantile(0.25)
q3=df['residual sugar'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

```
df['residual sugar'] = np.where(df['residual sugar']<lower_limit,2.2,df['residual sugar'])
```

```
sns.boxplot(df['residual sugar'])
```

```
<Axes: >
```

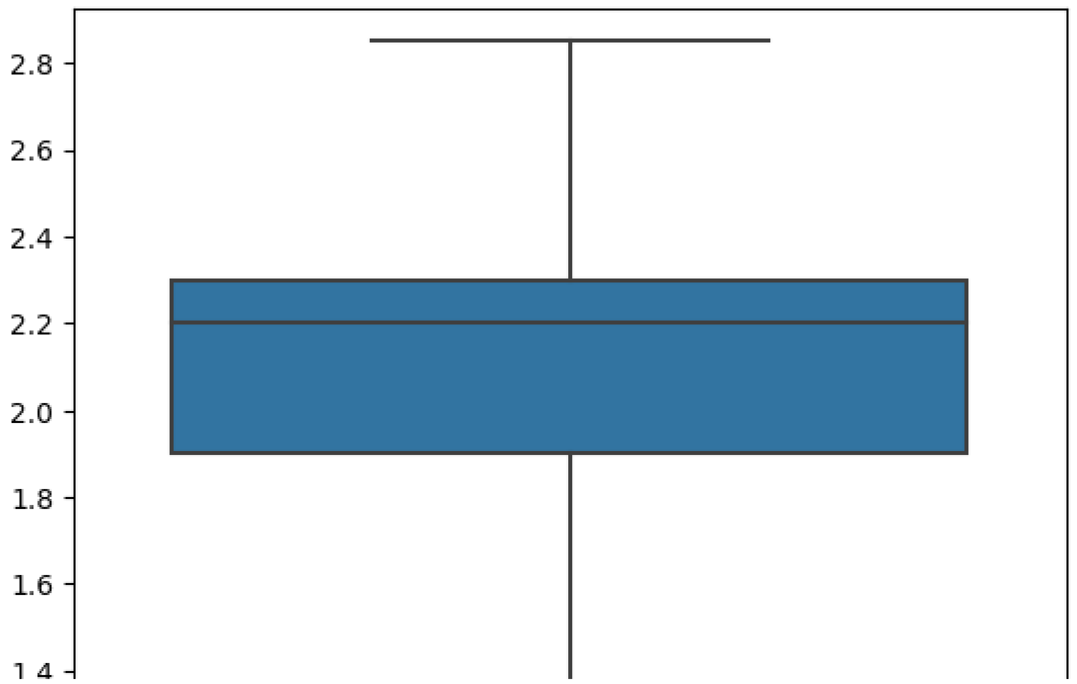


```
q1=df['residual sugar'].quantile(0.25)
q3=df['residual sugar'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

```
df['residual sugar'] = np.where(df['residual sugar']>upper_limit,2.2,df['residual sugar'])
```

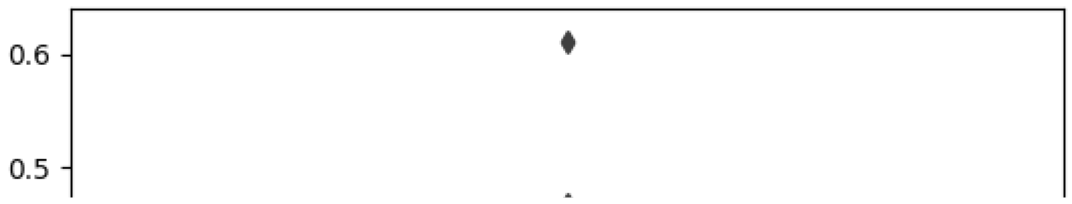
```
sns.boxplot(df['residual sugar'])
```

```
<Axes: >
```



```
sns.boxplot(df['chlorides'])
```

```
<Axes: >
```

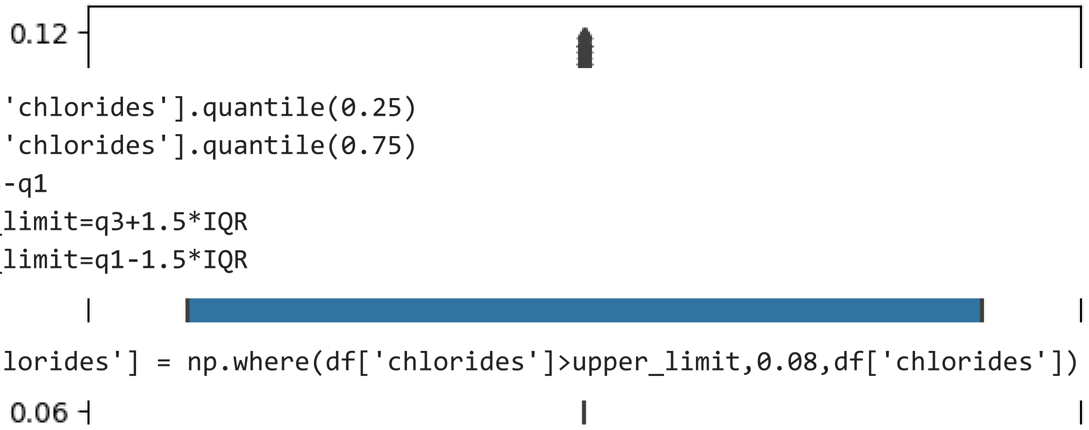


```
q1=df['chlorides'].quantile(0.25)
q3=df['chlorides'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

```
df['chlorides'] = np.where(df['chlorides']>upper_limit,0.08,df['chlorides'])
```

```
sns.boxplot(df['chlorides'])
```

```
<Axes: >
```

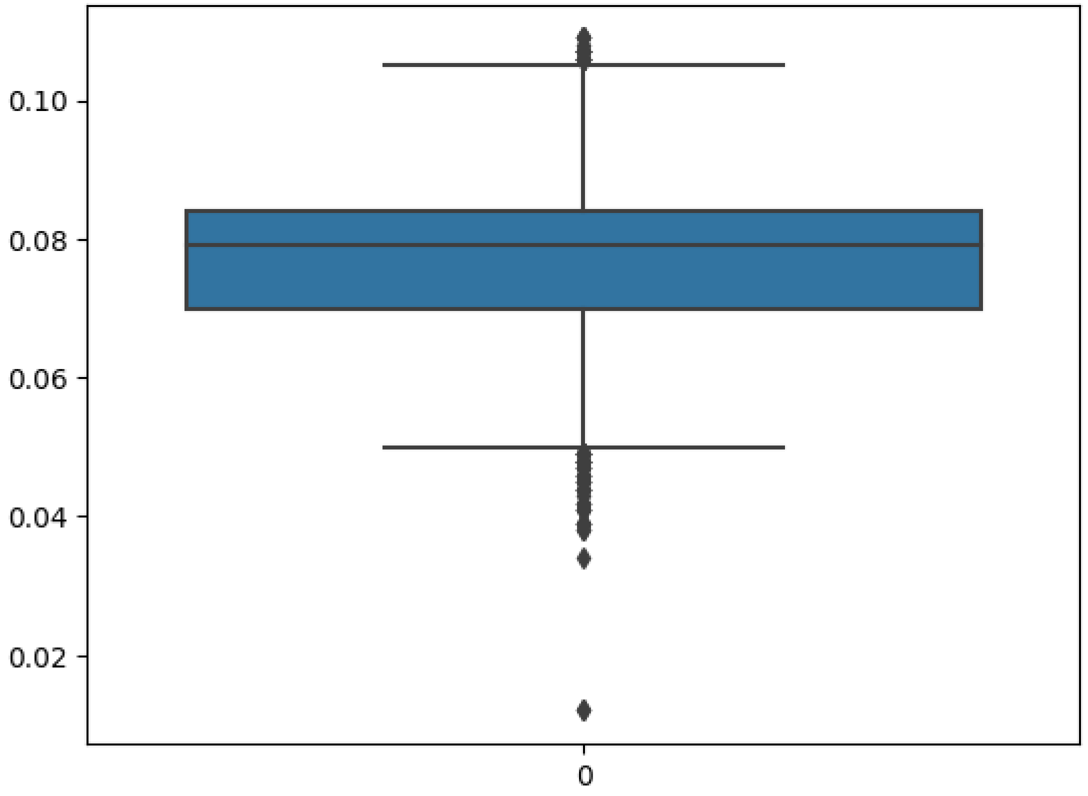


```
q1=df['chlorides'].quantile(0.25)
q3=df['chlorides'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

```
df['chlorides'] = np.where(df['chlorides']>upper_limit,0.08,df['chlorides'])
```

```
sns.boxplot(df['chlorides'])
```

```
<Axes: >
```

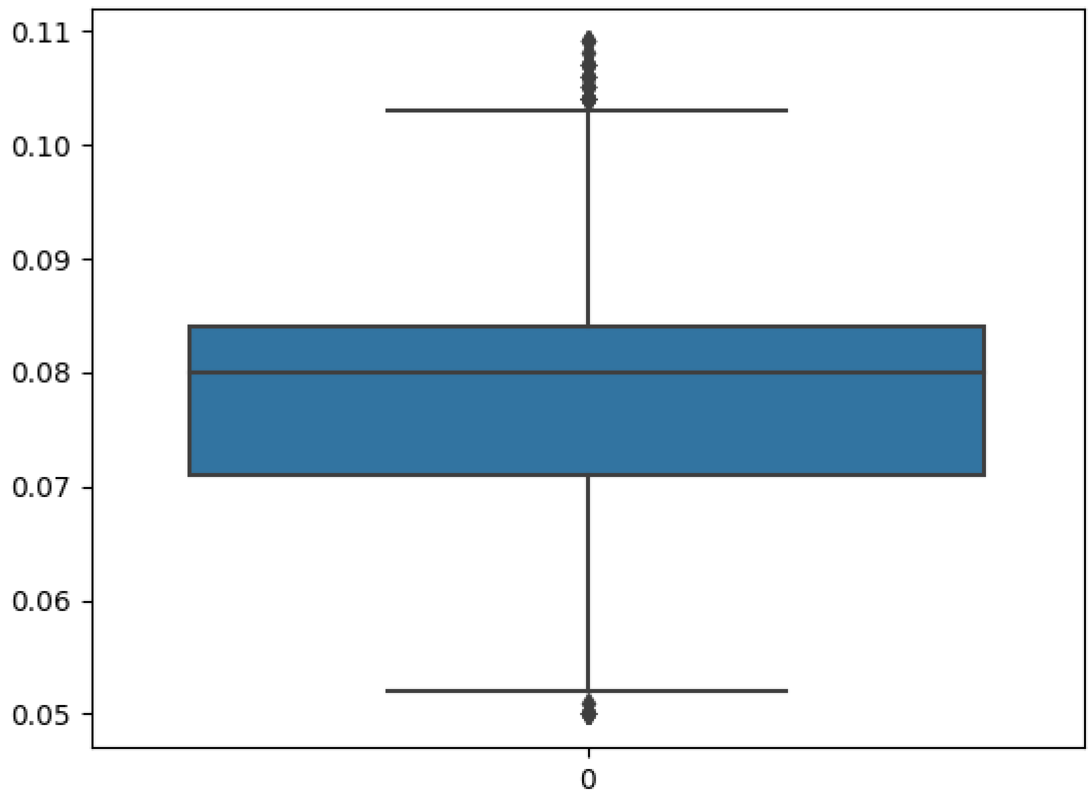


```
q1=df['chlorides'].quantile(0.25)
q3=df['chlorides'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

```
df['chlorides'] = np.where(df['chlorides']<lower_limit,0.08,df['chlorides'])
```

```
sns.boxplot(df['chlorides'])
```

```
<Axes: >
```



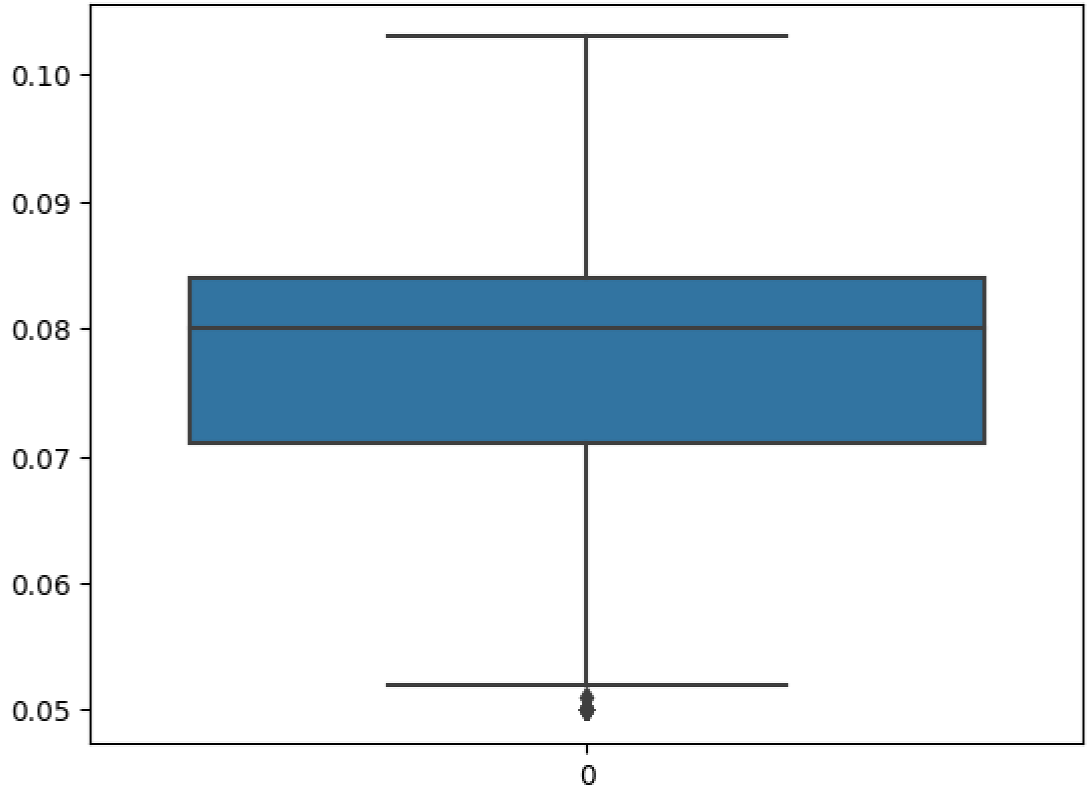
```
q1=df['chlorides'].quantile(0.25)
q3=df['chlorides'].quantile(0.75)
IQR=q3-q1
```

```
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

```
df['chlorides'] = np.where(df['chlorides']>upper_limit,0.08,df['chlorides'])
```

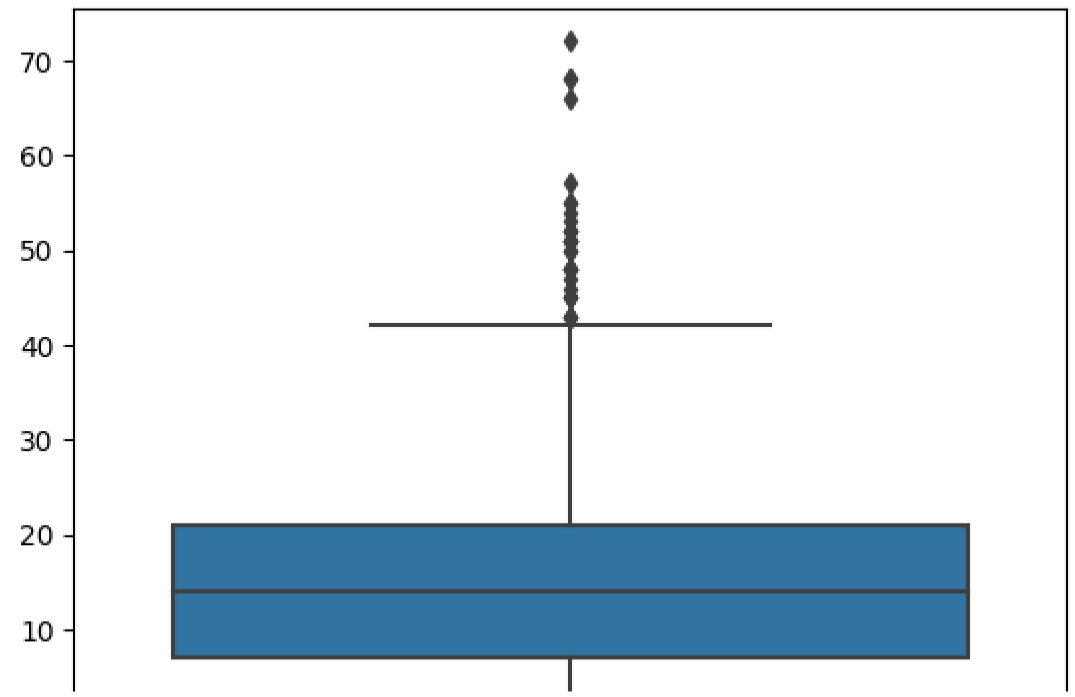
```
sns.boxplot(df['chlorides'])
```

```
<Axes: >
```



```
sns.boxplot(df['free sulfur dioxide'])
```

```
<Axes: >
```

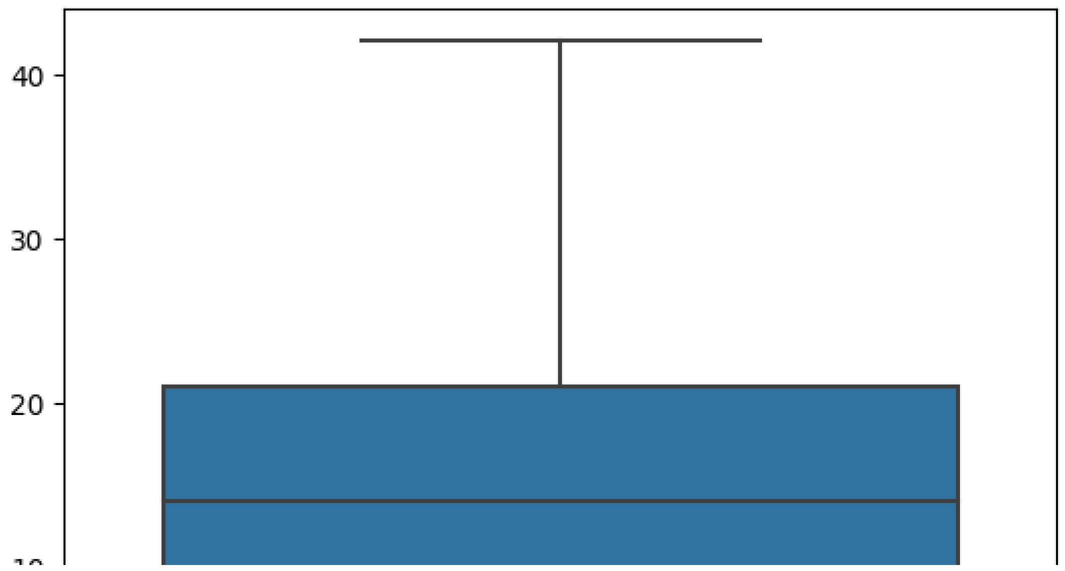


```
q1=df['free sulfur dioxide'].quantile(0.25)
q3=df['free sulfur dioxide'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

```
df['free sulfur dioxide'] = np.where(df['free sulfur dioxide']>upper_limit,14,df['free sulfur dioxide'])
```

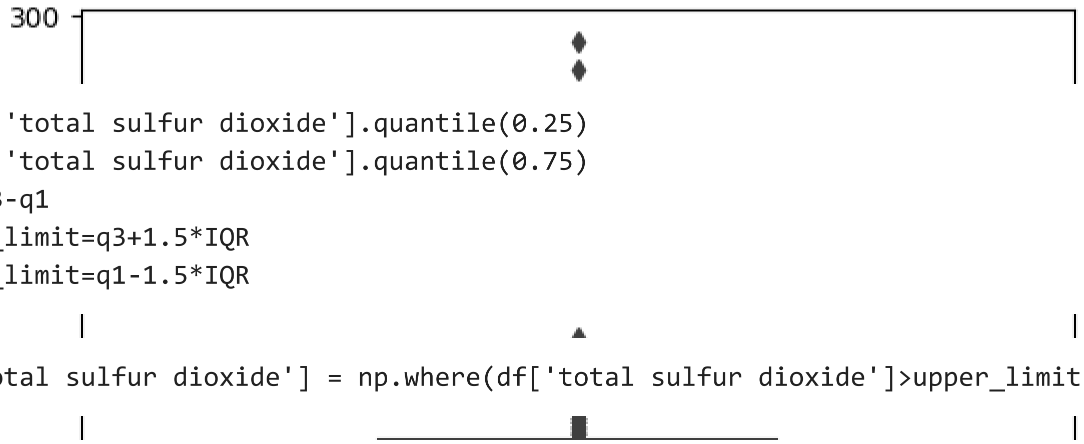
```
sns.boxplot(df['free sulfur dioxide'])
```

```
<Axes: >
```



```
sns.boxplot(df['total sulfur dioxide'])
```

```
<Axes: >
```

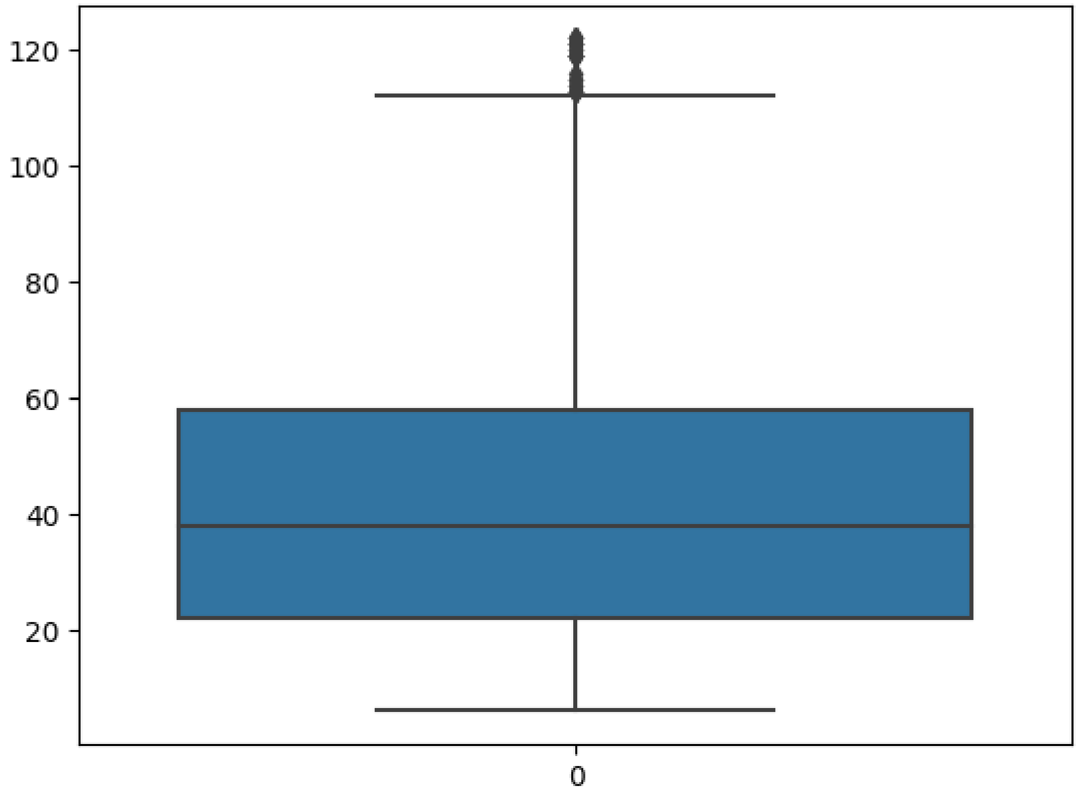


```
q1=df['total sulfur dioxide'].quantile(0.25)
q3=df['total sulfur dioxide'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

```
df['total sulfur dioxide'] = np.where(df['total sulfur dioxide']>upper_limit,38,df['total sulfur dioxide'])
```

```
sns.boxplot(df['total sulfur dioxide'])
```

```
<Axes: >
```

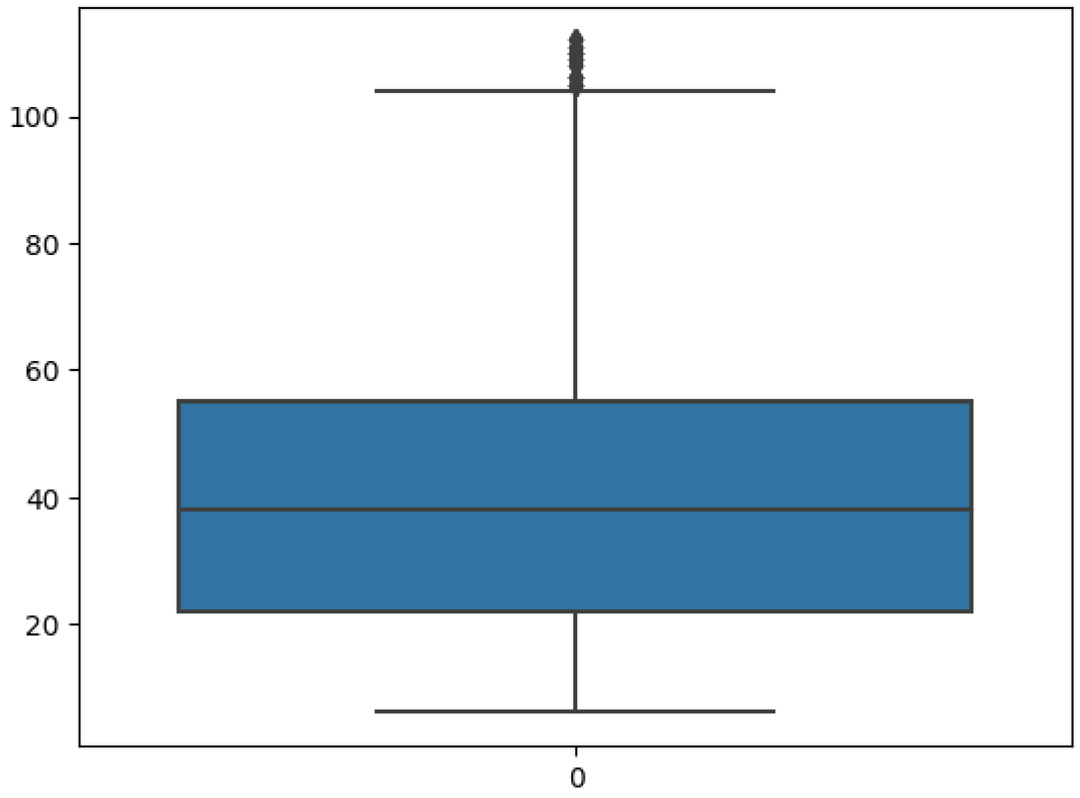


```
q1=df['total sulfur dioxide'].quantile(0.25)
q3=df['total sulfur dioxide'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

```
df['total sulfur dioxide'] = np.where(df['total sulfur dioxide']>upper_limit,38,df['total sulfur dioxide'])
```

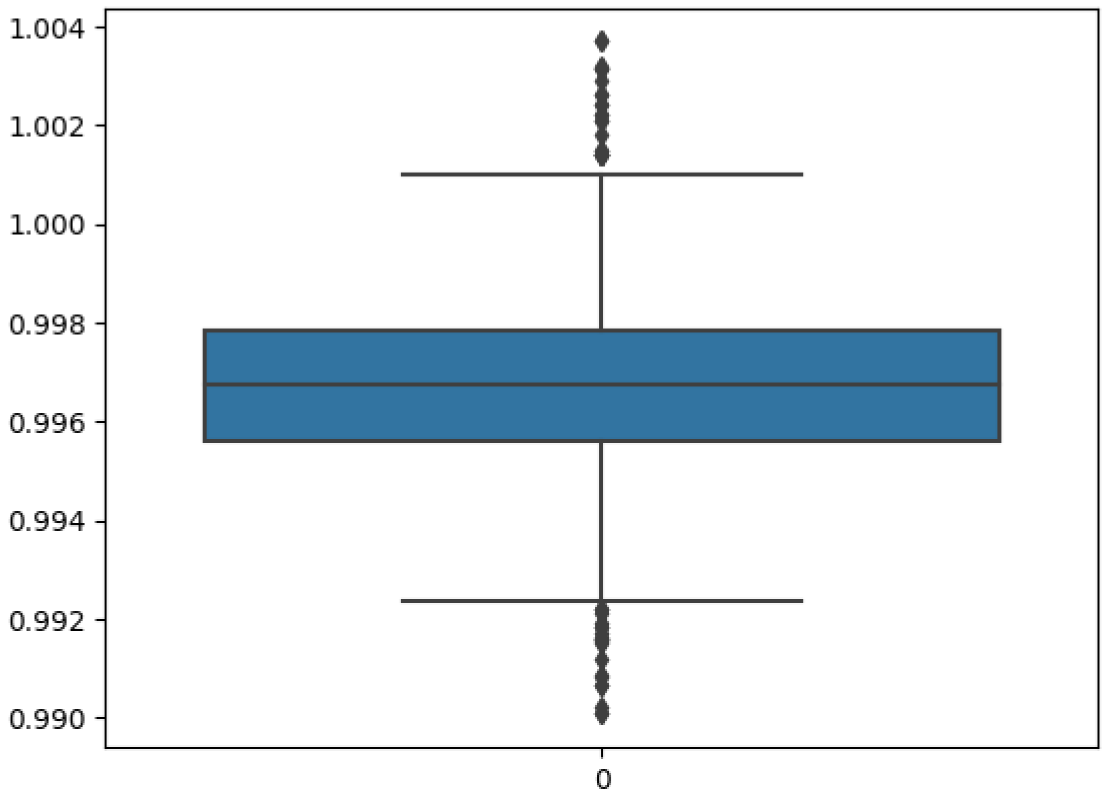
```
sns.boxplot(df['total sulfur dioxide'])
```

```
<Axes: >
```



```
sns.boxplot(df.density)
```

```
<Axes: >
```

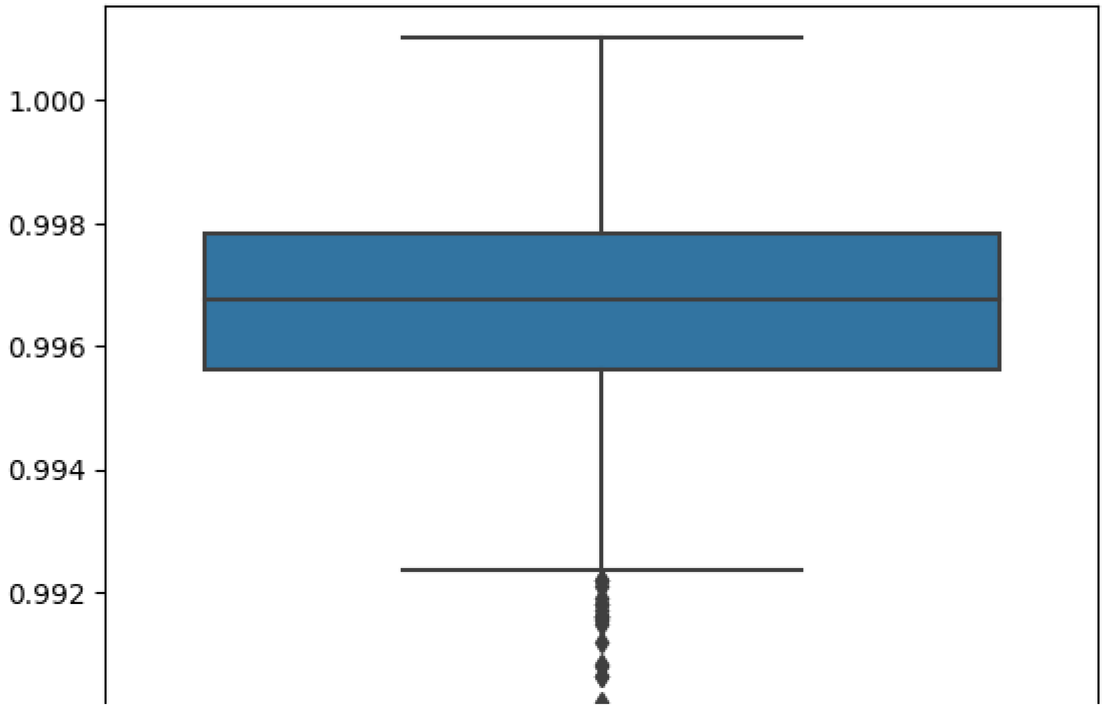


```
q1=df.density.quantile(0.25)
q3=df.density.quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

```
df.density = np.where(df.density>upper_limit,1,df.density)
```

```
sns.boxplot(df.density)
```

```
<Axes: >
```

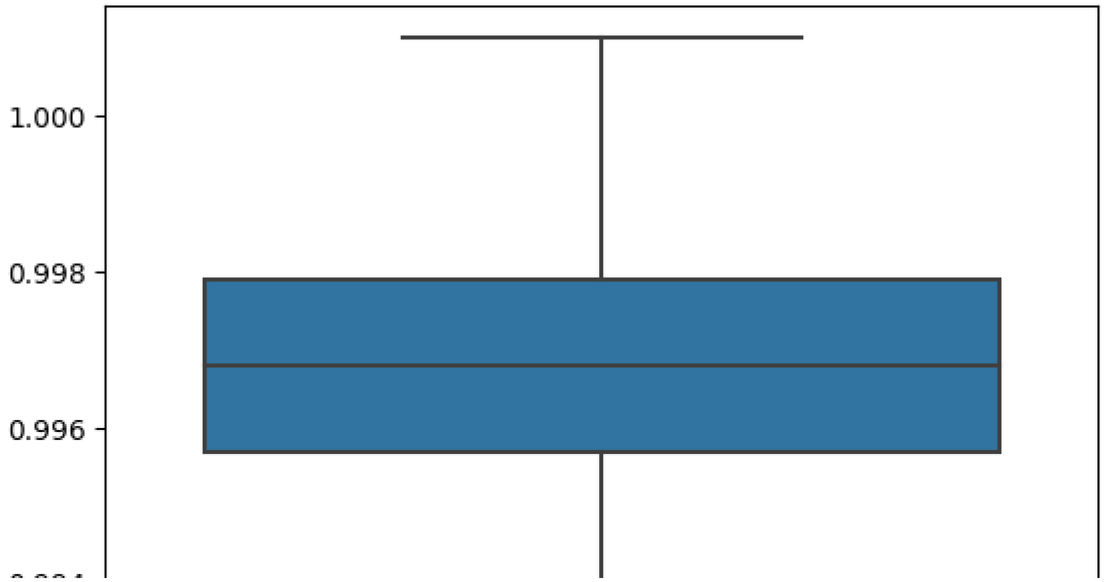


```
q1=df.density.quantile(0.25)
q3=df.density.quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

```
df.density = np.where(df.density<lower_limit,1,df.density)
```

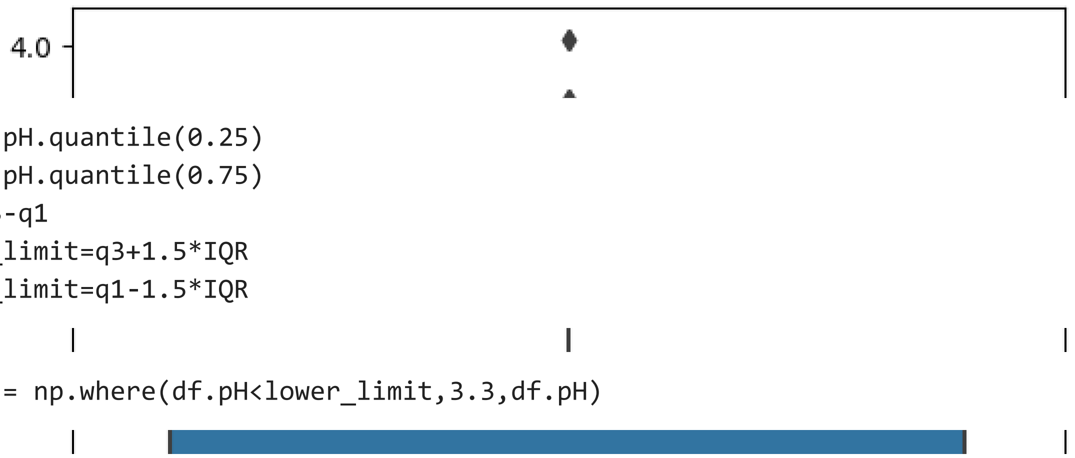
```
sns.boxplot(df.density)
```

```
<Axes: >
```



```
sns.boxplot(df.pH)
```

```
<Axes: >
```



```
q1=df.pH.quantile(0.25)
q3=df.pH.quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

```
df.pH = np.where(df.pH<lower_limit,3.3,df.pH)
```

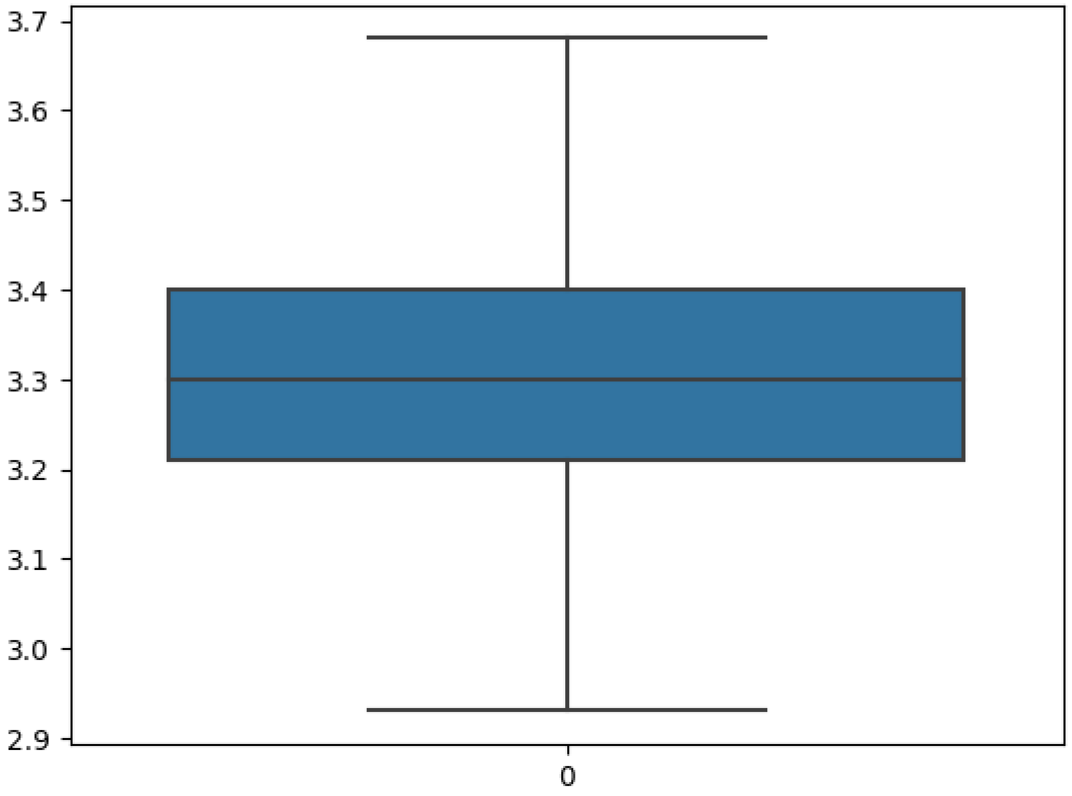
```
sns.boxplot(df.pH)
```

```
q1=df.pH.quantile(0.25)
q3=df.pH.quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

```
df.pH = np.where(df.pH>upper_limit,3.3,df.pH)
```

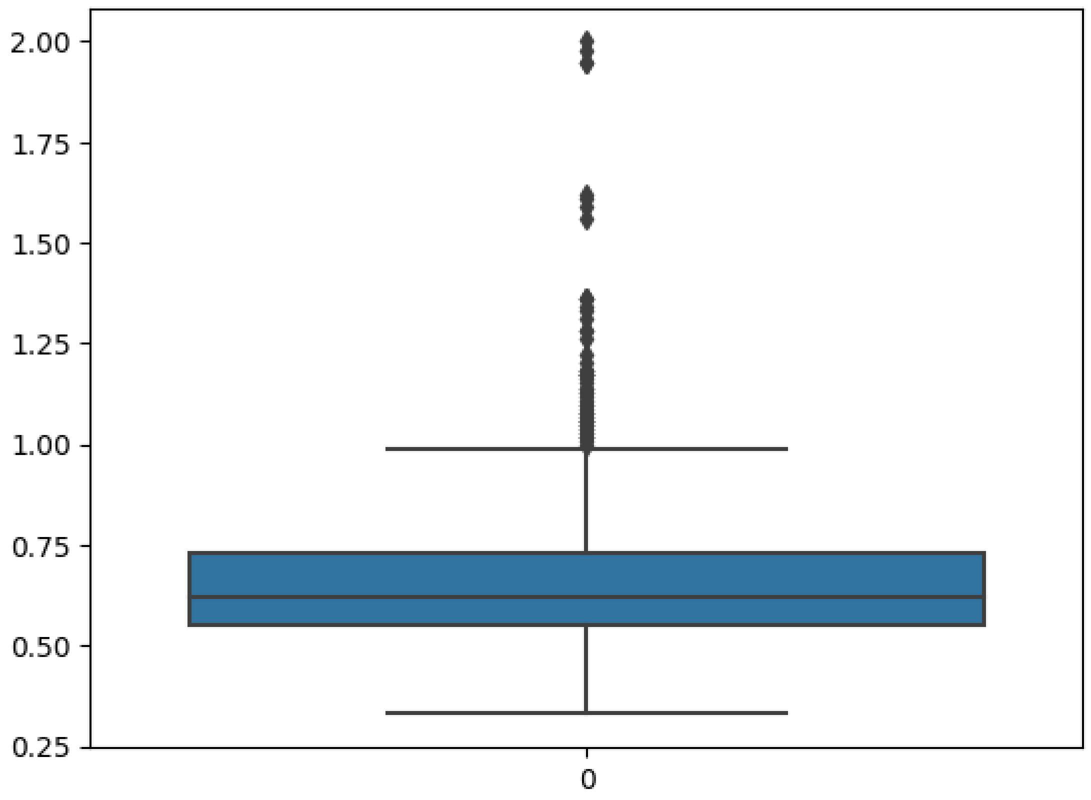
```
sns.boxplot(df.pH)
```

```
<Axes: >
```



```
sns.boxplot(df.sulphates)
```

```
<Axes: >
```

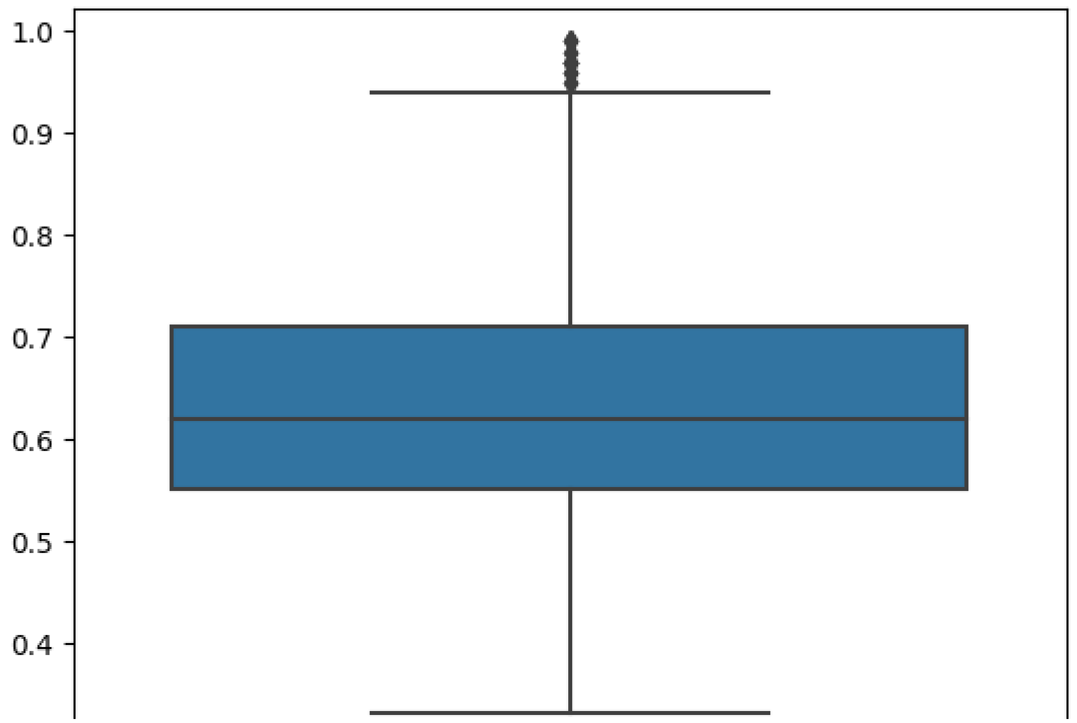


```
q1=df.sulphates.quantile(0.25)
q3=df.sulphates.quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

```
df.sulphates = np.where(df.sulphates>upper_limit,0.62,df.sulphates)
```

```
sns.boxplot(df.sulphates)
```

```
<Axes: >
```

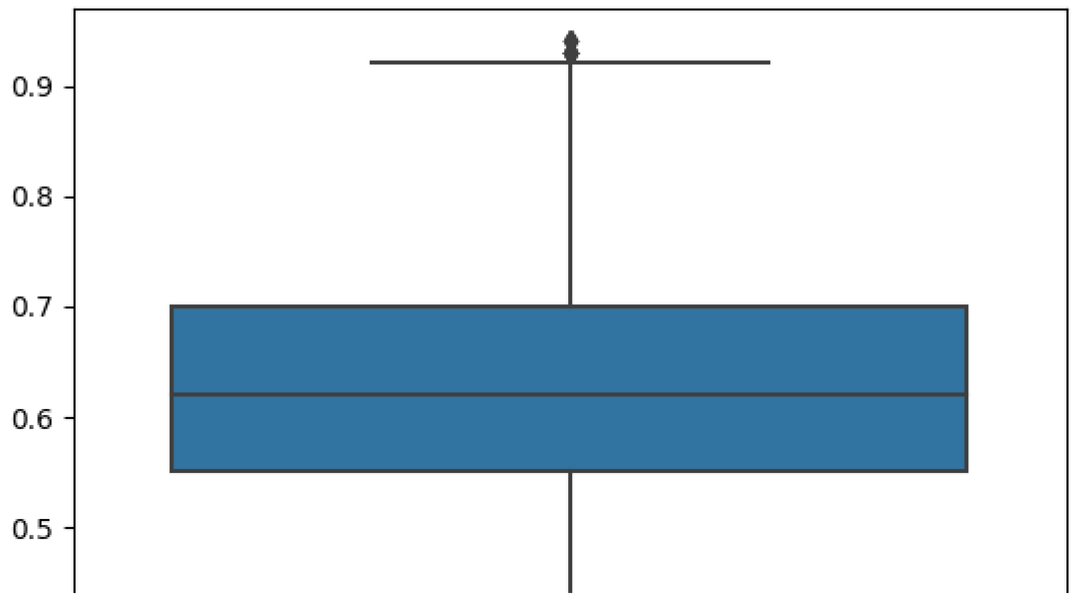


```
q1=df.sulphates.quantile(0.25)
q3=df.sulphates.quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

```
df.sulphates = np.where(df.sulphates>upper_limit,0.62,df.sulphates)
```

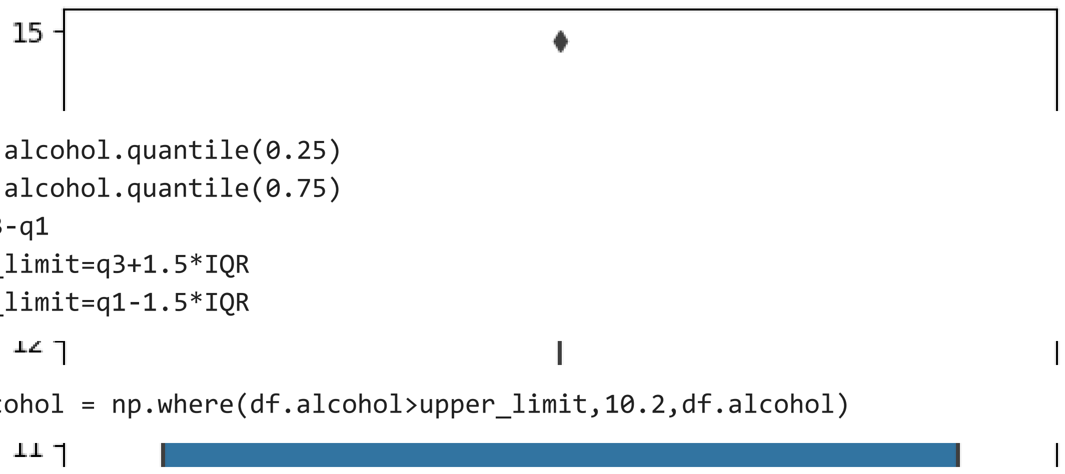
```
sns.boxplot(df.sulphates)
```

```
<Axes: >
```



```
sns.boxplot(df.alcohol)
```

```
<Axes: >
```



```
q1=df.alcohol.quantile(0.25)
q3=df.alcohol.quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

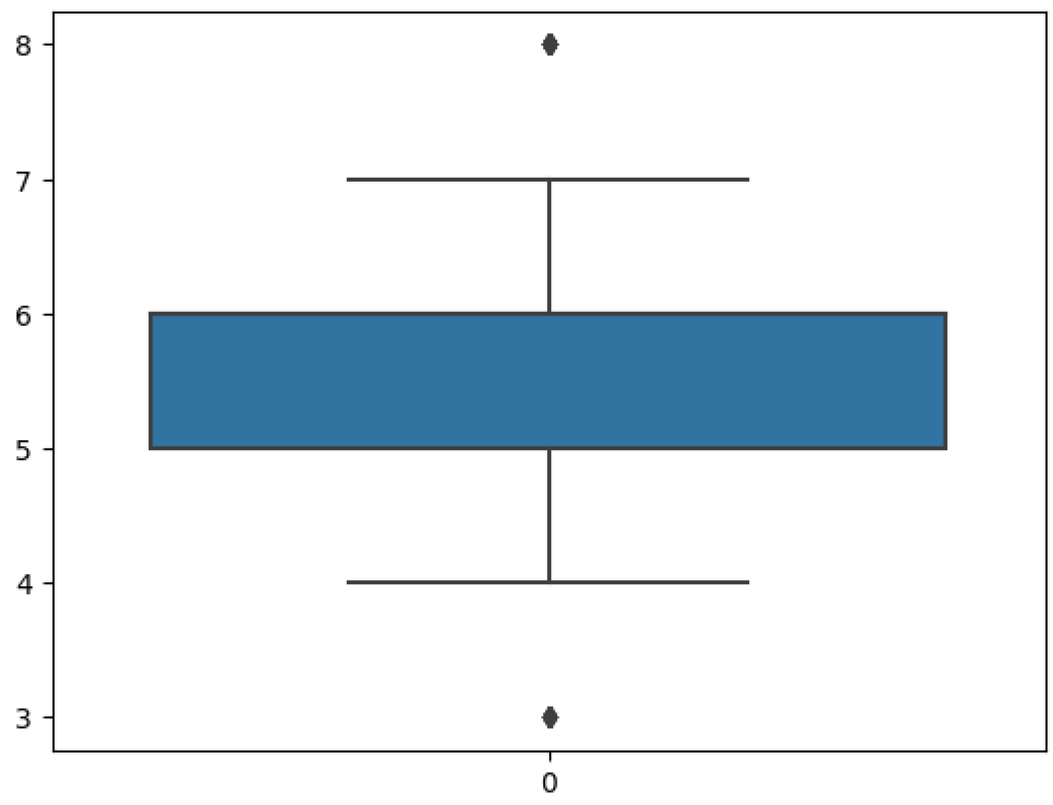
```
df.alcohol = np.where(df.alcohol>upper_limit,10.2,df.alcohol)
```

```
sns.boxplot(df.alcohol)
```

```
<Axes: >
```

```
sns.boxplot(df.quality)
```

```
<Axes: >
```



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

```
x=df.drop(columns=['quality'],axis=1)
```

```
x.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4

```
y=df.quality
y.head()
```

```
0    5
1    5
2    5
3    6
4    5
Name: quality, dtype: int64
```

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
```

```
scale=MinMaxScaler()
```

```
x_s=pd.DataFrame(scale.fit_transform(x),columns=x.columns)
```

```
x_train,x_test,y_train,y_test=train_test_split(x_s,y,test_size=0.2,random_state=0)
```

```
x_train.shape,y_train.shape
```

```
((1279, 11), (1279,))
```

```
x_test.shape,y_test.shape
```

```
((320, 11), (320,))
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
```

```
model=RandomForestClassifier(n_estimators=400)
```

```
model.fit(x_train,y_train)
```

```
▾        RandomForestClassifier
RandomForestClassifier(n_estimators=400)
```

```
y_pred=model.predict(x_test)
```

```
y_pred_train=model.predict(x_train)
```

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_train,y_pred_train))
```

```
    1.0
```

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

```
    0.728125
```

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

```
                  precision    recall  f1-score   support

               3       0.00      0.00      0.00         2
               4       0.00      0.00      0.00        11
               5       0.78      0.80      0.79       135
               6       0.73      0.77      0.75       142
               7       0.53      0.59      0.56        27
               8       0.00      0.00      0.00         3

        accuracy                           0.73       320
       macro avg       0.34      0.36      0.35       320
    weighted avg       0.70      0.73      0.71       320

    /usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and
      _warn_prf(average, modifier, msg_start, len(result))
    /usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and
      _warn_prf(average, modifier, msg_start, len(result))
    /usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and
      _warn_prf(average, modifier, msg_start, len(result))
```

```
confusion_matrix(y_test, y_pred)
```

```
array([[  0,   0,   1,   1,   0,   0],
       [  0,   0,   6,   5,   0,   0],
       [  0,   0, 108,  25,   2,   0],
       [  0,   0,  23, 109,  10,   0],
       [  0,   0,   1,   8,  16,   2],
       [  0,   0,   0,   1,   2,   0]])
```

```
model.predict([[7.5, 0.8, 0.0, 2, 0.75, 15, 38, 0.96, 3.5, 0.5, 9.8]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but Ranc
  warnings.warn(
array([5])
```

```
model.predict([[7.4, 0.7, 0.1, 2, 0.75, 14, 35, 0.96, 3.5, 0.5, 9.9]])
```

```
base.py:439: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
```

```
model.predict([[7.9, 0.52, 0.26, 2.2, 0.08, 14, 38, 0.99, 3.3, 0.62, 10.2]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but Ranc
  warnings.warn(
array([7])
```