assignment-4-21bit0376

September 22, 2023

NYSA SINGH

21BIT0376

[1]:
```
#Importing essential libraries

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

[2]:
```
#Loading the dataset

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

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

[4]:
```
(1599, 12)
```

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

[5]:
```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
4            7.4              0.70         0.00             1.9      0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56

   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
```

```
3      9.8        6
4      9.4        5
```

[18]:
```
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
```

Checking for null values

[9]:
```
df.isnull().any()
```

[9]:
```
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
```

We observe there are no null values, hence we move to data analysis step.

[10]:
```
#Getting description of data

df.describe()
```

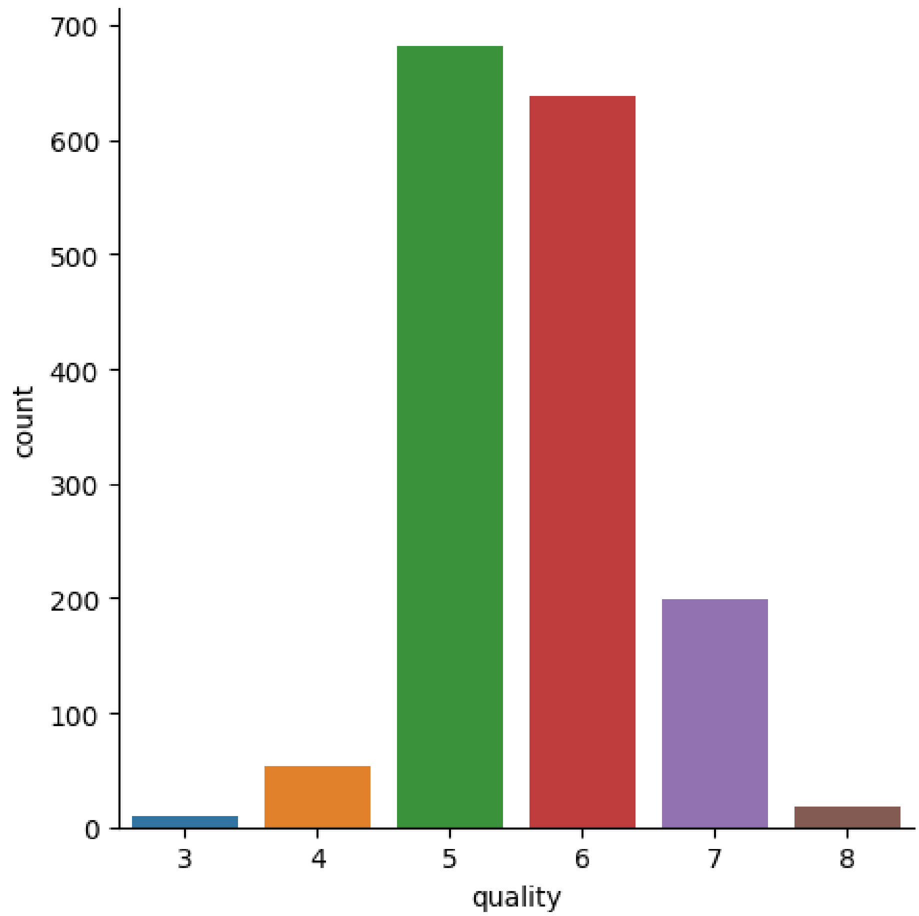
[10]:
```
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    1599.000000       1599.000000  1599.000000     1599.000000
mean        8.319637          0.527821     0.270976        2.538806
std         1.741096          0.179060     0.194801        1.409928
min         4.600000          0.120000     0.000000        0.900000
25%         7.100000          0.390000     0.090000        1.900000
50%         7.900000          0.520000     0.260000        2.200000
75%         9.200000          0.640000     0.420000        2.600000
max        15.900000          1.580000     1.000000       15.500000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  1599.000000          1599.000000           1599.000000  1599.000000
mean      0.087467            15.874922             46.467792     0.996747
std       0.047065            10.460157             32.895324     0.001887
min       0.012000             1.000000              6.000000     0.990070
25%       0.070000             7.000000             22.000000     0.995600
50%       0.079000            14.000000             38.000000     0.996750
75%       0.090000            21.000000             62.000000     0.997835
max       0.611000            72.000000            289.000000     1.003690

                pH    sulphates      alcohol      quality
count  1599.000000  1599.000000  1599.000000  1599.000000
mean      3.311113     0.658149    10.422983     5.636023
std       0.154386     0.169507     1.065668     0.807569
min       2.740000     0.330000     8.400000     3.000000
25%       3.210000     0.550000     9.500000     5.000000
50%       3.310000     0.620000    10.200000     6.000000
75%       3.400000     0.730000    11.100000     6.000000
max       4.010000     2.000000    14.900000     8.000000
```

[11]:
```python
#Checking values in each quality

df.quality.value_counts()
```

[11]:
```
5    681
6    638
7    199
4     53
8     18
3     10
Name: quality, dtype: int64
```

[12]:
```python
#Using catplot

sns.catplot(x='quality', data=df, kind='count')
```

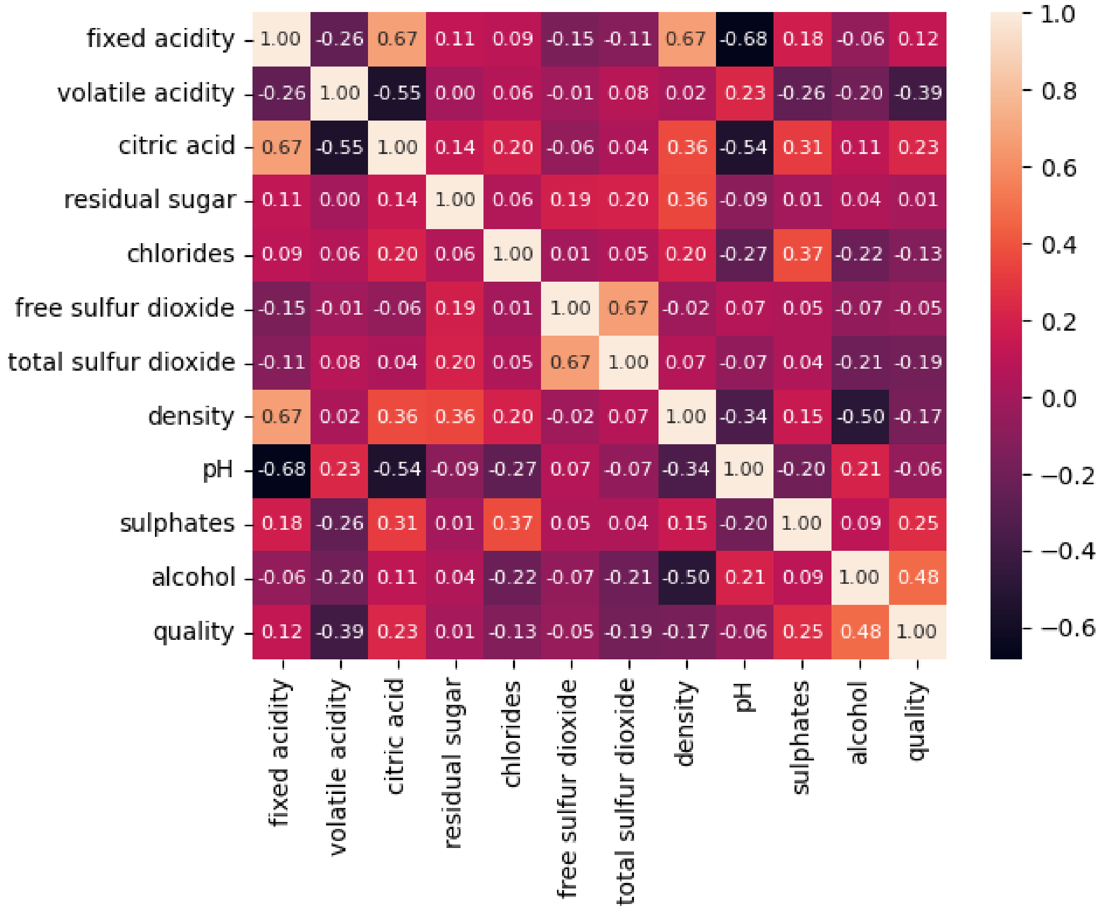
[12]:
```
<seaborn.axisgrid.FacetGrid at 0x7ea18a9b0880>
```



Checking for coloumn related to quality

[17]:
```
sns.heatmap(df.corr(),annot=True, annot_kws={'size': 8}, fmt='.2f')
```

[17]:
```
<Axes: >
```



[14]:
```
df.corr()
```

[14]:
```
                      fixed acidity  volatile acidity  citric acid  \
fixed acidity              1.000000         -0.256131     0.671703
volatile acidity          -0.256131          1.000000    -0.552496
citric acid                0.671703         -0.552496     1.000000
residual sugar             0.114777          0.001918     0.143577
chlorides                  0.093705          0.061298     0.203823
free sulfur dioxide       -0.153794         -0.010504    -0.060978
total sulfur dioxide      -0.113181          0.076470     0.035533
density                    0.668047          0.022026     0.364947
pH                        -0.682978          0.234937    -0.541904
sulphates                  0.183006         -0.260987     0.312770
alcohol                   -0.061668         -0.202288     0.109903
quality                    0.124052         -0.390558     0.226373

                      residual sugar  chlorides  free sulfur dioxide  \
fixed acidity               0.114777   0.093705            -0.153794
```

volatile acidity	0.001918	0.061298	-0.010504
citric acid	0.143577	0.203823	-0.060978
residual sugar	1.000000	0.055610	0.187049
chlorides	0.055610	1.000000	0.005562
free sulfur dioxide	0.187049	0.005562	1.000000
total sulfur dioxide	0.203028	0.047400	0.667666
density	0.355283	0.200632	-0.021946
pH	-0.085652	-0.265026	0.070377
sulphates	0.005527	0.371260	0.051658
alcohol	0.042075	-0.221141	-0.069408
quality	0.013732	-0.128907	-0.050656

	total sulfur dioxide	density	pH	sulphates \
fixed acidity	-0.113181	0.668047	-0.682978	0.183006
volatile acidity	0.076470	0.022026	0.234937	-0.260987
citric acid	0.035533	0.364947	-0.541904	0.312770
residual sugar	0.203028	0.355283	-0.085652	0.005527
chlorides	0.047400	0.200632	-0.265026	0.371260
free sulfur dioxide	0.667666	-0.021946	0.070377	0.051658
total sulfur dioxide	1.000000	0.071269	-0.066495	0.042947
density	0.071269	1.000000	-0.341699	0.148506
pH	-0.066495	-0.341699	1.000000	-0.196648
sulphates	0.042947	0.148506	-0.196648	1.000000
alcohol	-0.205654	-0.496180	0.205633	0.093595
quality	-0.185100	-0.174919	-0.057731	0.251397

	alcohol	quality
fixed acidity	-0.061668	0.124052
volatile acidity	-0.202288	-0.390558
citric acid	0.109903	0.226373
residual sugar	0.042075	0.013732
chlorides	-0.221141	-0.128907
free sulfur dioxide	-0.069408	-0.050656
total sulfur dioxide	-0.205654	-0.185100
density	-0.496180	-0.174919
pH	0.205633	-0.057731
sulphates	0.093595	0.251397
alcohol	1.000000	0.476166
quality	0.476166	1.000000

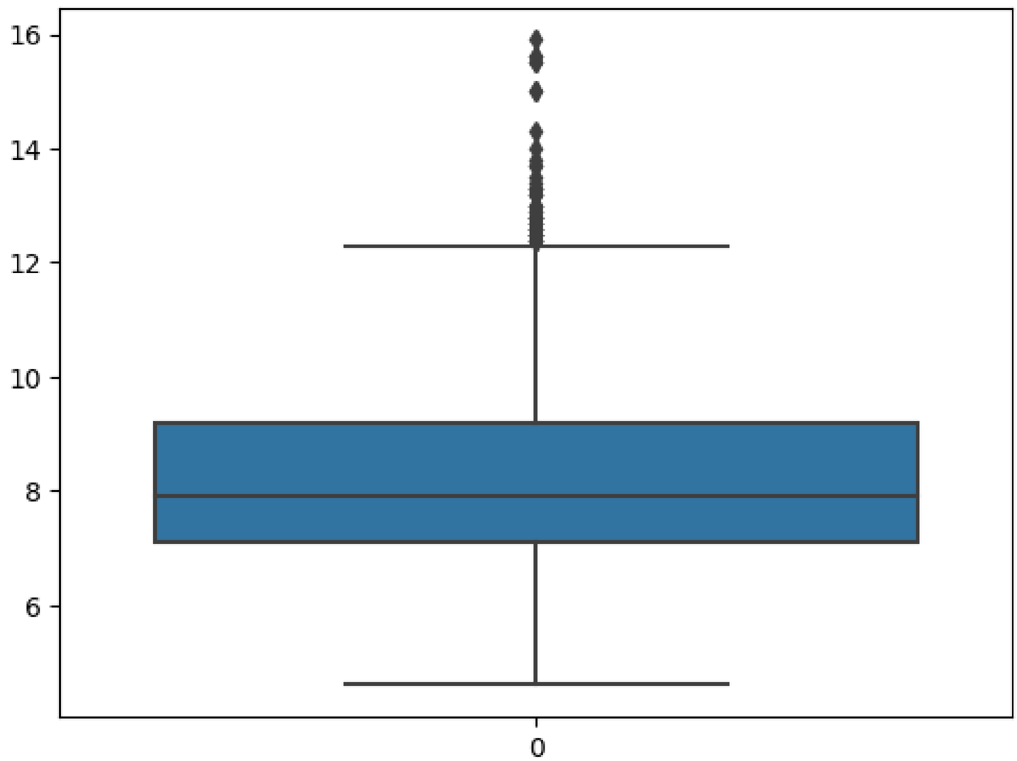
We observe that wine quality is highlt influenced by alcohol and sulphates positively and presence of volatile acidity affects the quality of wine negatively.

Since we are performing classification and we know that Classifiaction Algorithm like decision tree and Random forest are not affected by the UnScaled data. Hence we are Avoiding sacling of data.

Classification Algorithm are also insesnitive to outliers, but as a part of data preprocessing we will try to detect and remove outliers.

[19]:
```
#Checking for fixed Acidity

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

[19]: <Axes: >

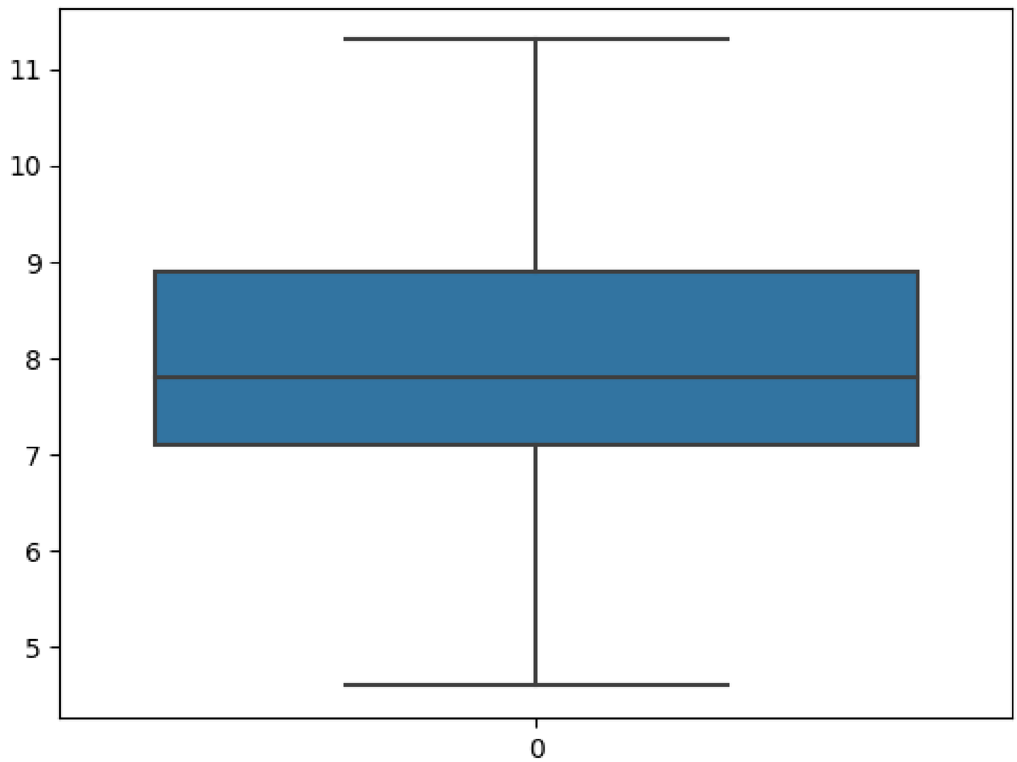


We Observe there are outliers in the fixed Acidity columns. Hence, we will remove them.

[27]:
```
#Using IQR method

f1= df['fixed acidity'].quantile(0.25) #First Quartile
f3= df['fixed acidity'].quantile(0.75) #Third Quartile
IQR=f3-f1 #Inter Quertile range

Upper_limit = f3+(1.5)*IQR
Lower_limit = f1-(1.5)*IQR

df=df[(df['fixed acidity']<Upper_limit) & (df['fixed acidity']>Lower_limit)]
```

[26]:
```
sns.boxplot(df['fixed acidity'])
```

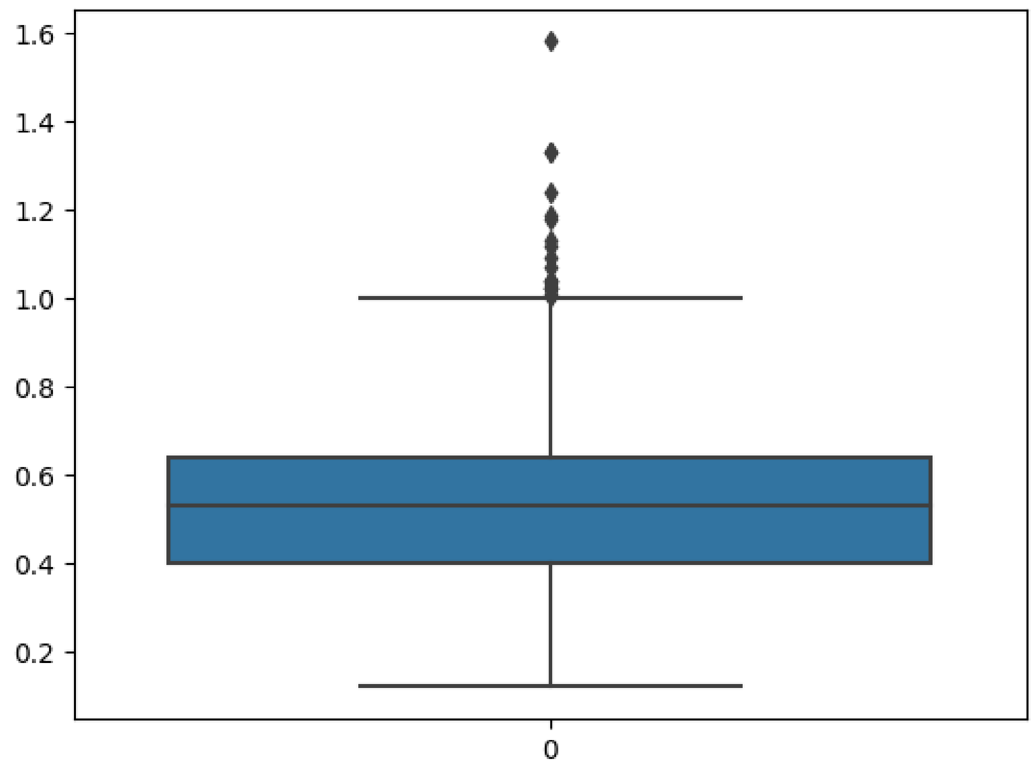
[26]: <Axes: >



checking for outliers in volatile acidity column

[28]:
```
sns.boxplot(df['volatile acidity'])
```

[28]: <Axes: >

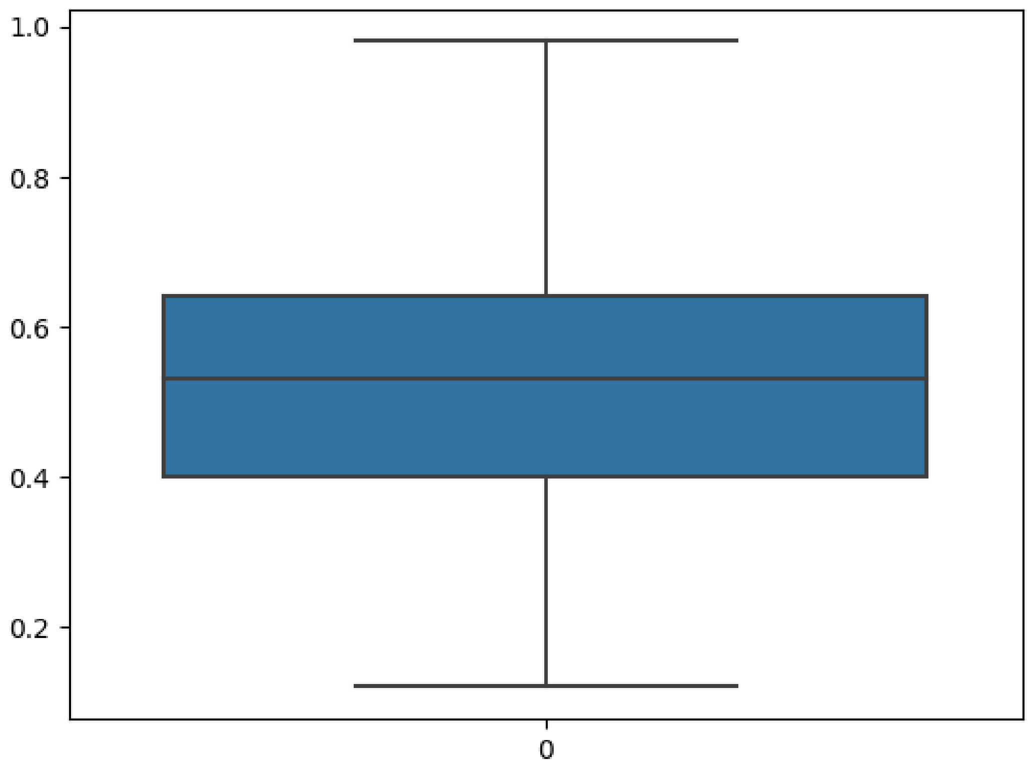


We see outliers are present in Volatile Acidity column. Hence we will remove the outliers

[29]:
```
#Using IQR method

f1= df['volatile acidity'].quantile(0.25) #First Quartile
f3= df['volatile acidity'].quantile(0.75) #Third Quartile
IQR=f3-f1 #Inter Quertile range

Upper_limit = f3+(1.5)*IQR
Lower_limit = f1-(1.5)*IQR

df=df[(df['volatile acidity']<Upper_limit) & (df['volatile␣
 ↪acidity']>Lower_limit)]
```

[30]:
```
sns.boxplot(df['volatile acidity'])
```

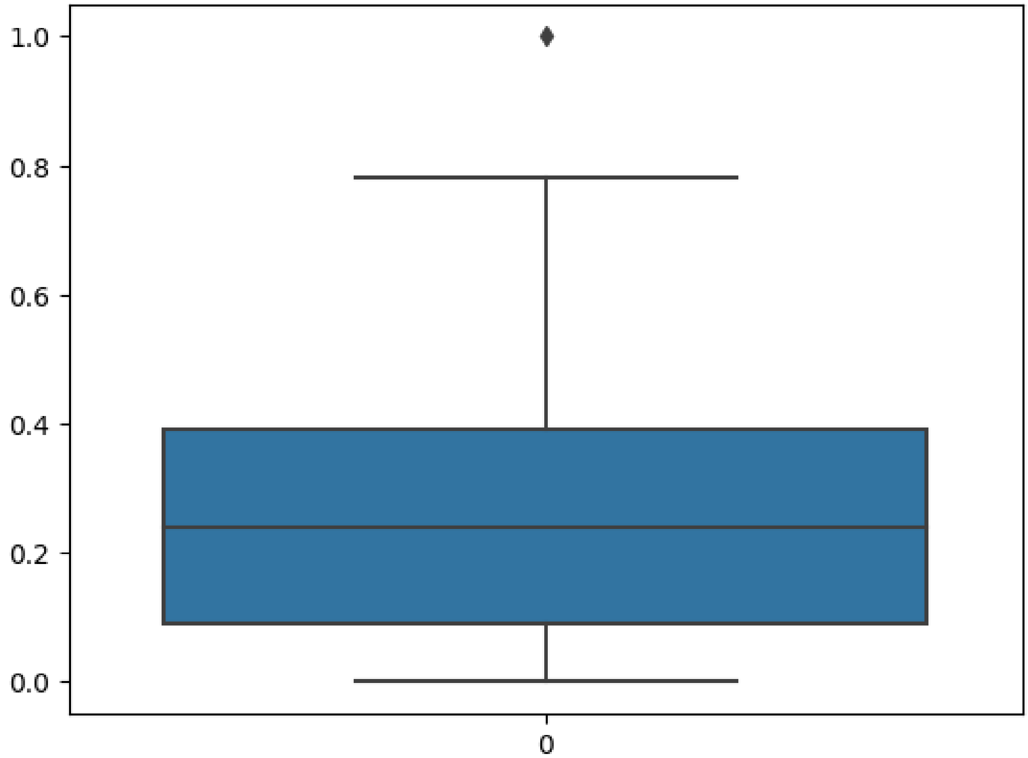
[30]: <Axes: >



Checking for outliers in citric Acid column

[31]:
```
sns.boxplot(df['citric acid'])
```

[31]: <Axes: >

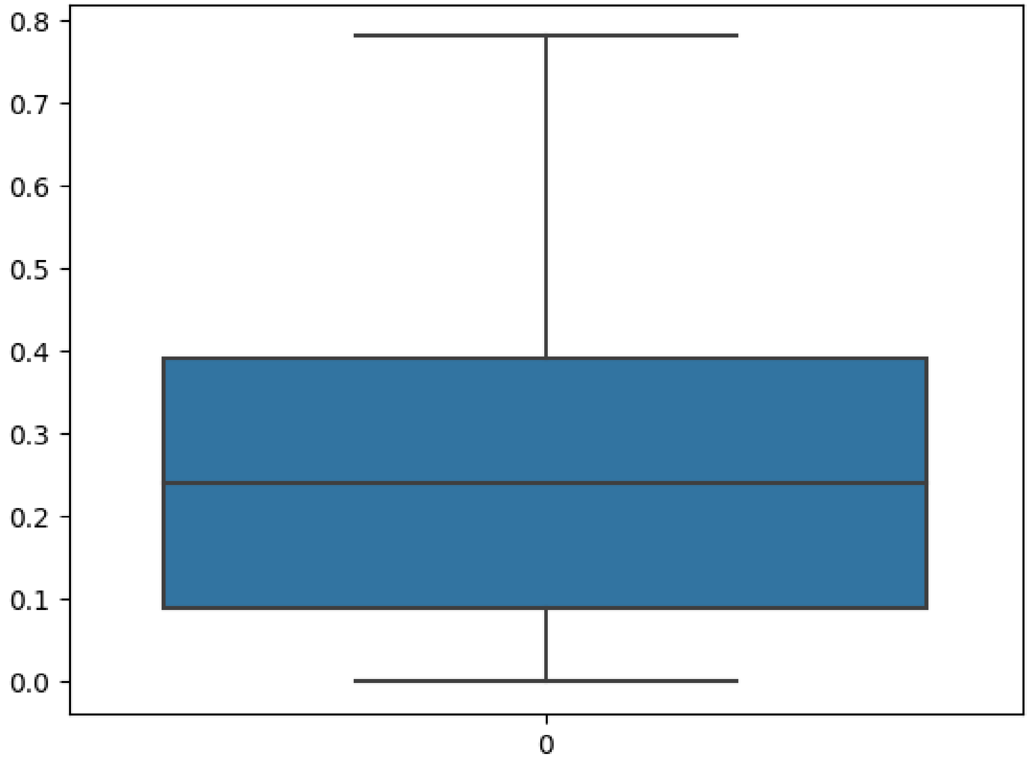


We observe there is a outlier in citric Acid column. Hence we will remove it.

[32]:
```
#Using IQR method

f1= df['citric acid'].quantile(0.25) #First Quartile
f3= df['citric acid'].quantile(0.75) #Third Quartile
IQR=f3-f1 #Inter Quertile range

Upper_limit = f3+(1.5)*IQR
Lower_limit = f1-(1.5)*IQR

df=df[(df['citric acid']<Upper_limit) & (df['citric acid']>Lower_limit)]
```

[33]:
```
sns.boxplot(df['citric acid'])
```

[33]: <Axes: >

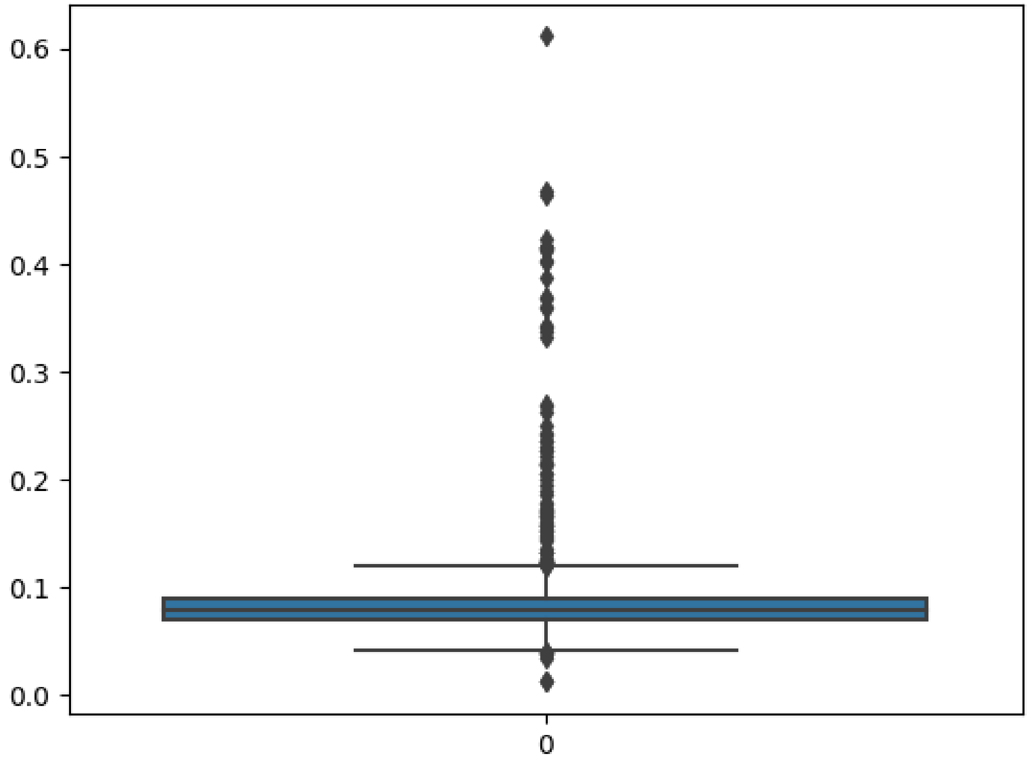


We see that Residual sugar does not affect Quality much. Hence we skip the Residual Sugar Column.

Checking for Outliers in chlorides

[34]:
```
sns.boxplot(df['chlorides'])
```

[34]: <Axes: >

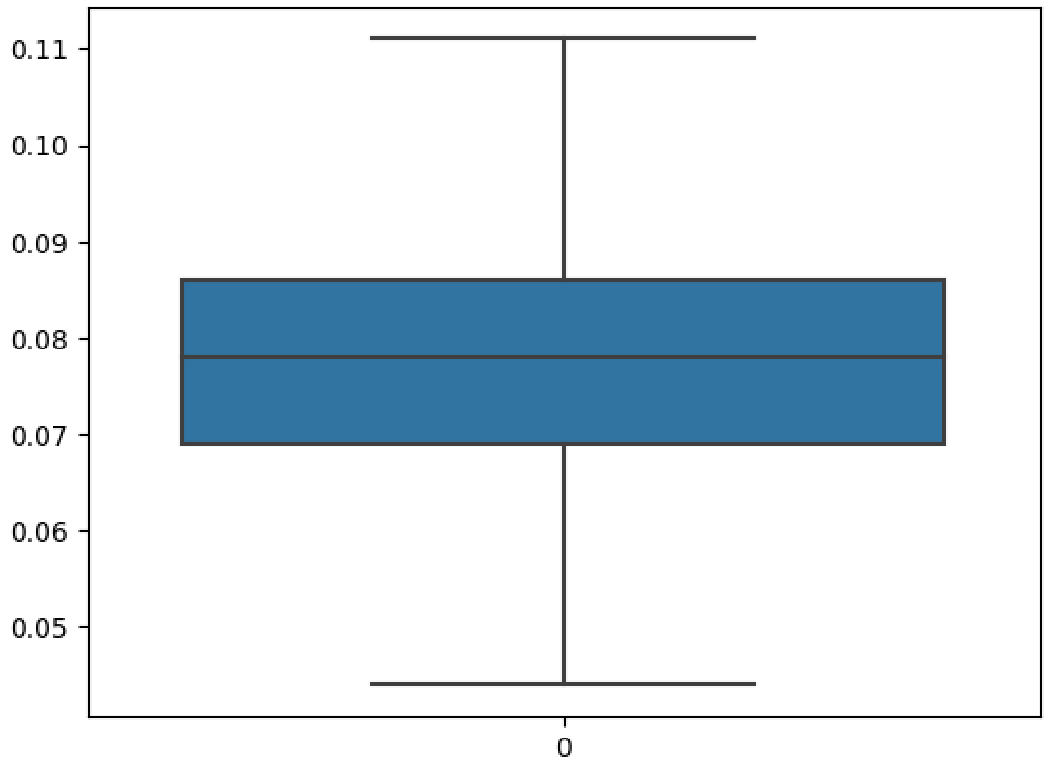


We observe there are a large number of iutliers in chlorides columns. Hence removing them.

[39]:
```
#Using IQR method

f1= df['chlorides'].quantile(0.25) #First Quartile
f3= df['chlorides'].quantile(0.75) #Third Quartile
IQR=f3-f1 #Inter Quertile range

Upper_limit = f3+(1.5)*IQR
Lower_limit = f1-(1.5)*IQR

df=df[(df['chlorides']<Upper_limit) & (df['chlorides']>Lower_limit)]
```

[40]:
```
sns.boxplot(df['chlorides'])
```

[40]: <Axes: >

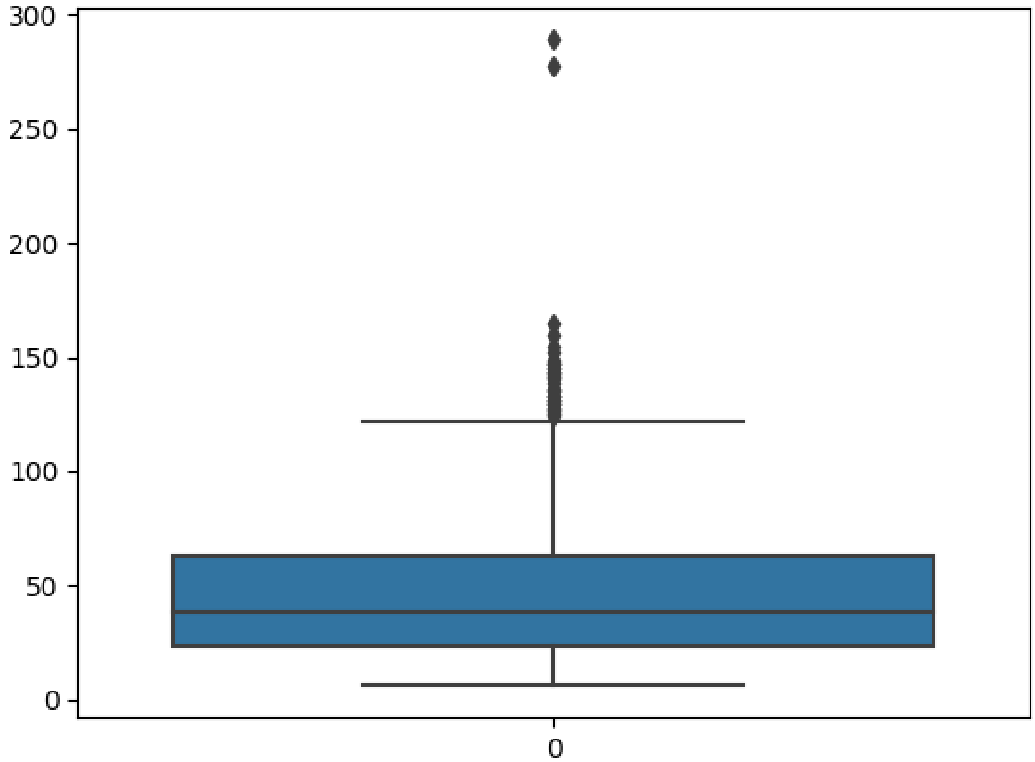


We see that Free Sulphur Dioxide does not affect Quality much. Hence we skip the Free sulphur dioxide Column.

Checking for Outliers in Total sulphur dioxide

[41]:
```
sns.boxplot(df['total sulfur dioxide'])
```

[41]: <Axes: >

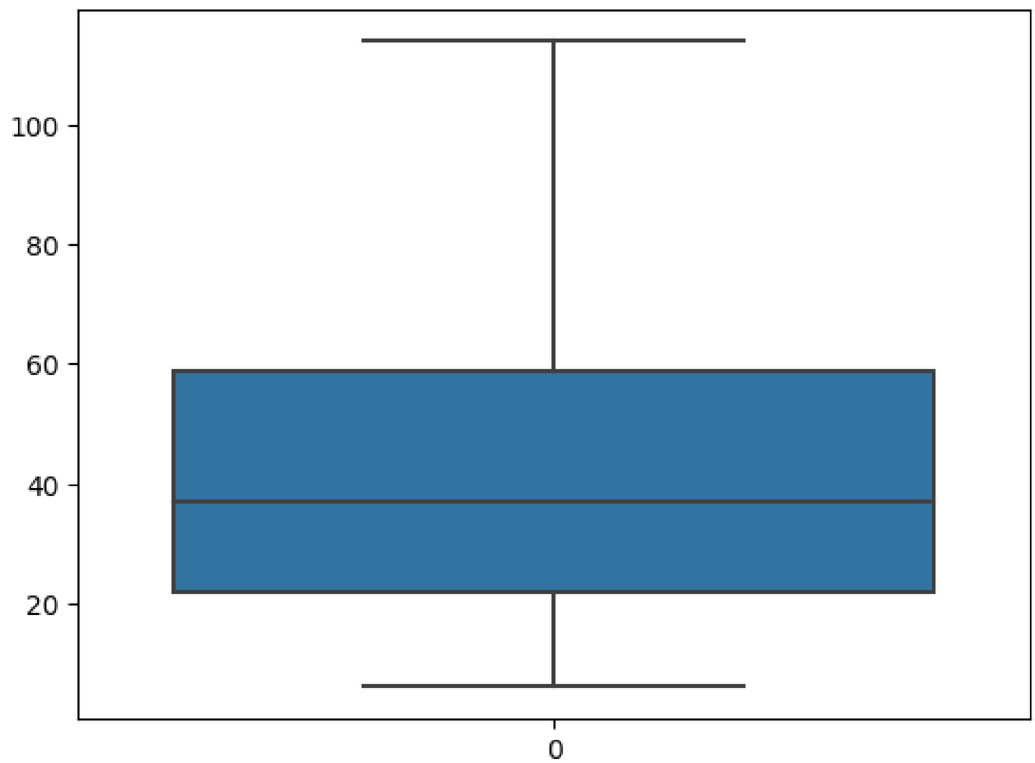


We observe there are a large number of iutliers in Total sulphur Dioxide columns. Hence removing them.

[46]:
```
#Using IQR method

f1= df['total sulfur dioxide'].quantile(0.25) #First Quartile
f3= df['total sulfur dioxide'].quantile(0.75) #Third Quartile
IQR=f3-f1 #Inter Quertile range

Upper_limit = f3+(1.5)*IQR
Lower_limit = f1-(1.5)*IQR

df=df[(df['total sulfur dioxide']<Upper_limit) & (df['total sulfur␣
 ↪dioxide']>Lower_limit)]
```

[47]:
```
sns.boxplot(df['total sulfur dioxide'])
```

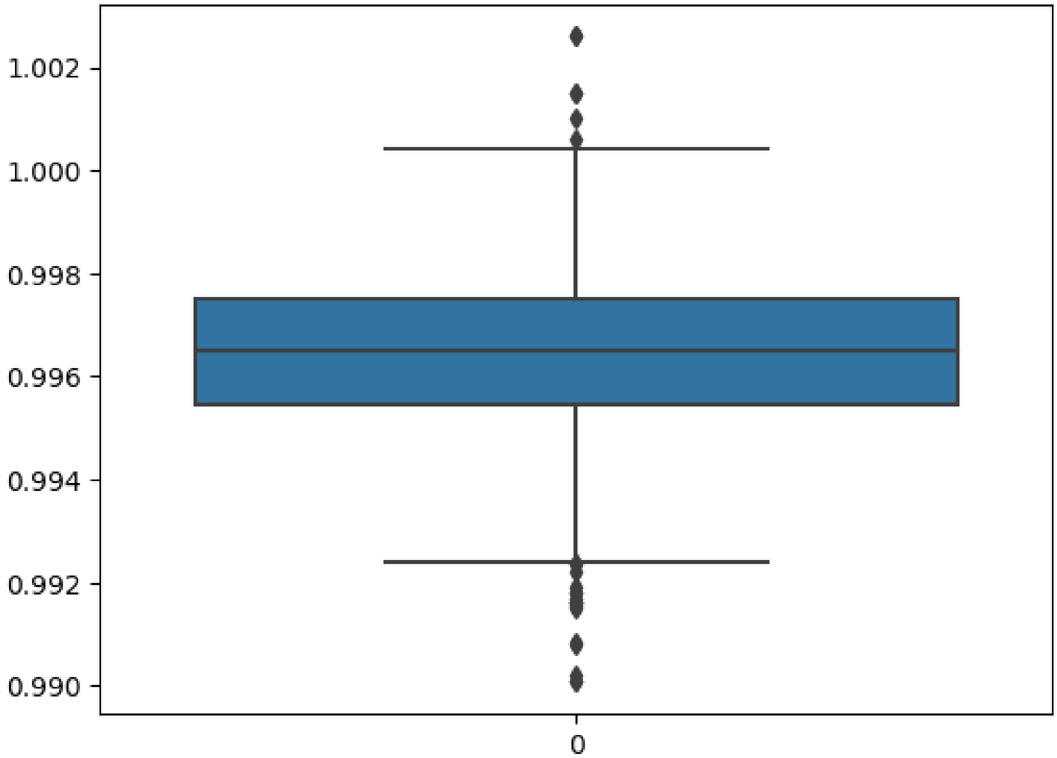
[47]: <Axes: >



Checking for Outliers in Density

[48]:
```
sns.boxplot(df.density)
```

[48]:
```
<Axes: >
```

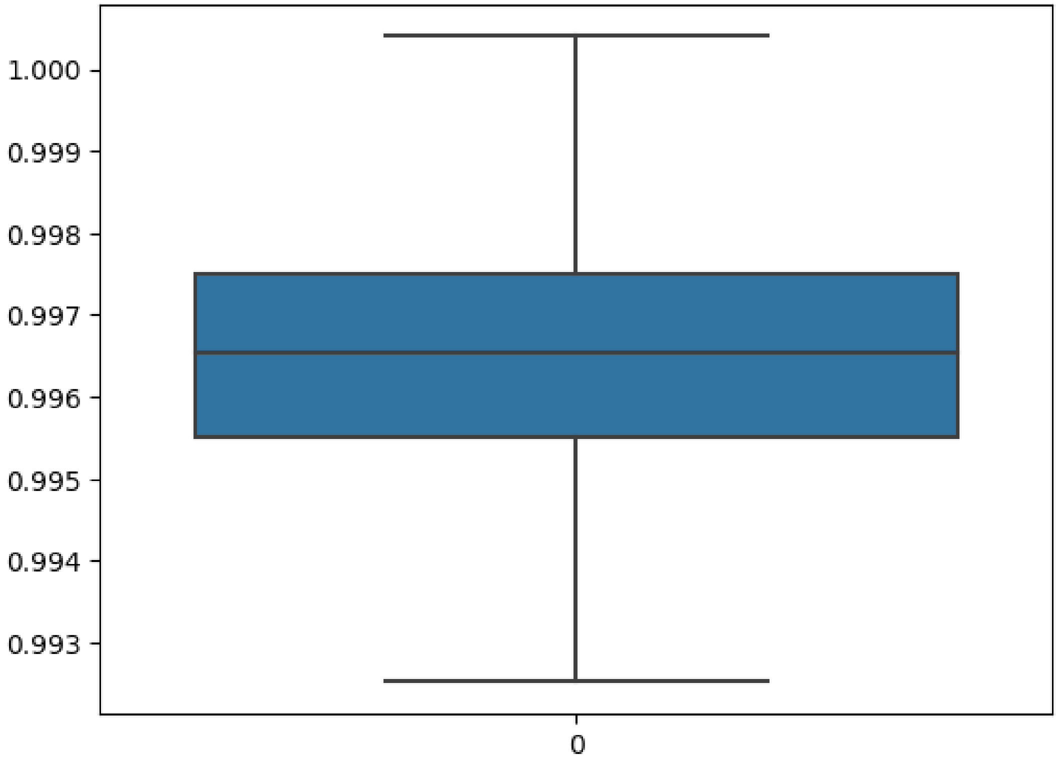


We observe there are outliers in Density columns. Hence removing them.

[51]:
```
#Using IQR method

f1= df['density'].quantile(0.25) #First Quartile
f3= df['density'].quantile(0.75) #Third Quartile
IQR=f3-f1 #Inter Quertile range

Upper_limit = f3+(1.5)*IQR
Lower_limit = f1-(1.5)*IQR

df=df[(df['density']<Upper_limit) & (df['density']>Lower_limit)]
```

[52]:
```
sns.boxplot(df.density)
```

[52]: <Axes: >

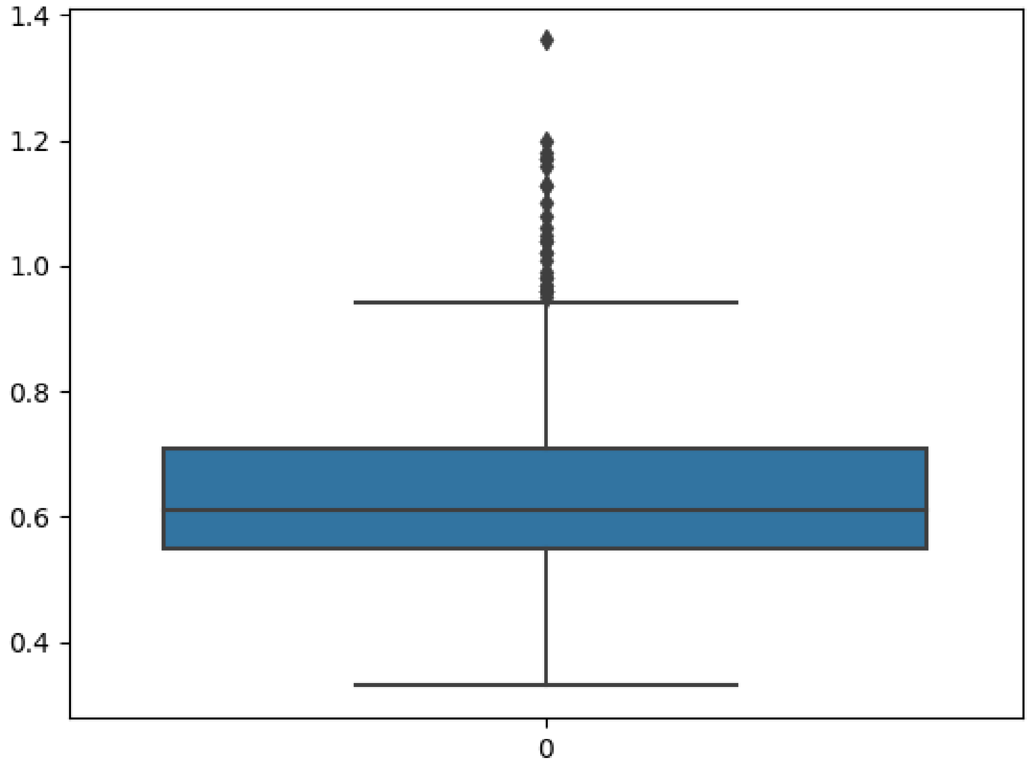


We see that PH does not affect Quality much. Hence we skip the PH Column.

Checking for outliers in Sulpahtes column

[53]:
```
sns.boxplot(df.sulphates)
```

[53]: <Axes: >

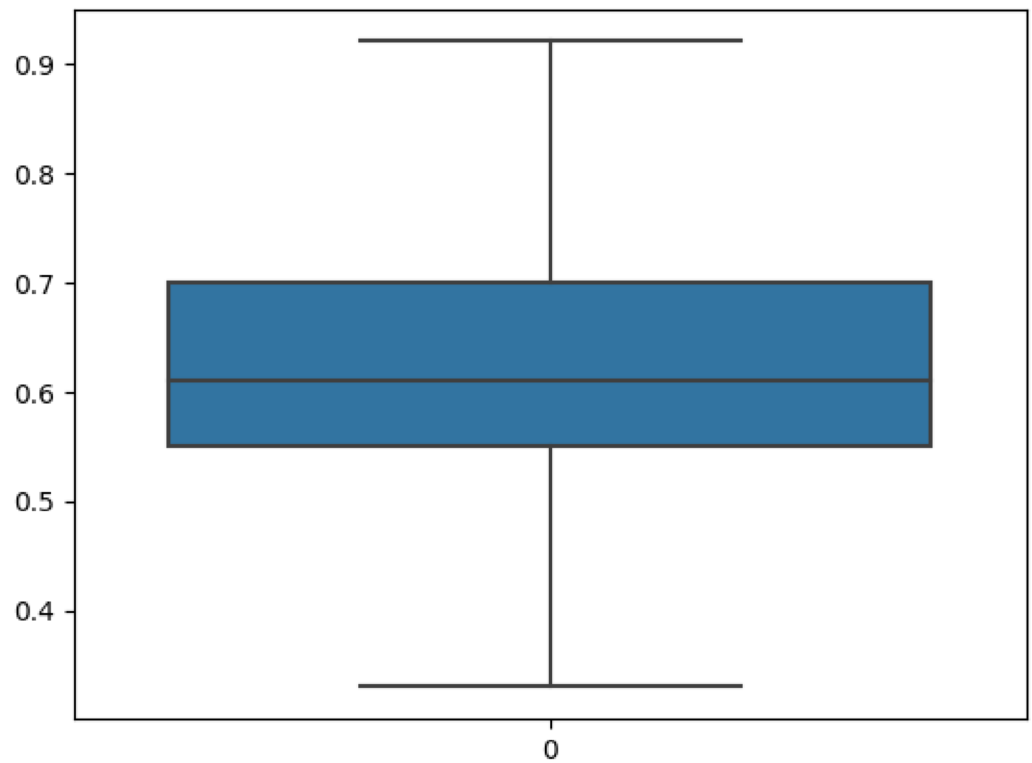


We observe there are a large number of outliers in Sulpahtes columns. Hence removing them.

[56]:
```
#Using IQR method

f1= df['sulphates'].quantile(0.25) #First Quartile
f3= df['sulphates'].quantile(0.75) #Third Quartile
IQR=f3-f1 #Inter Quertile range

Upper_limit = f3+(1.5)*IQR
Lower_limit = f1-(1.5)*IQR

df=df[(df['sulphates']<Upper_limit) & (df['sulphates']>Lower_limit)]
```

[57]:
```
sns.boxplot(df.sulphates)
```

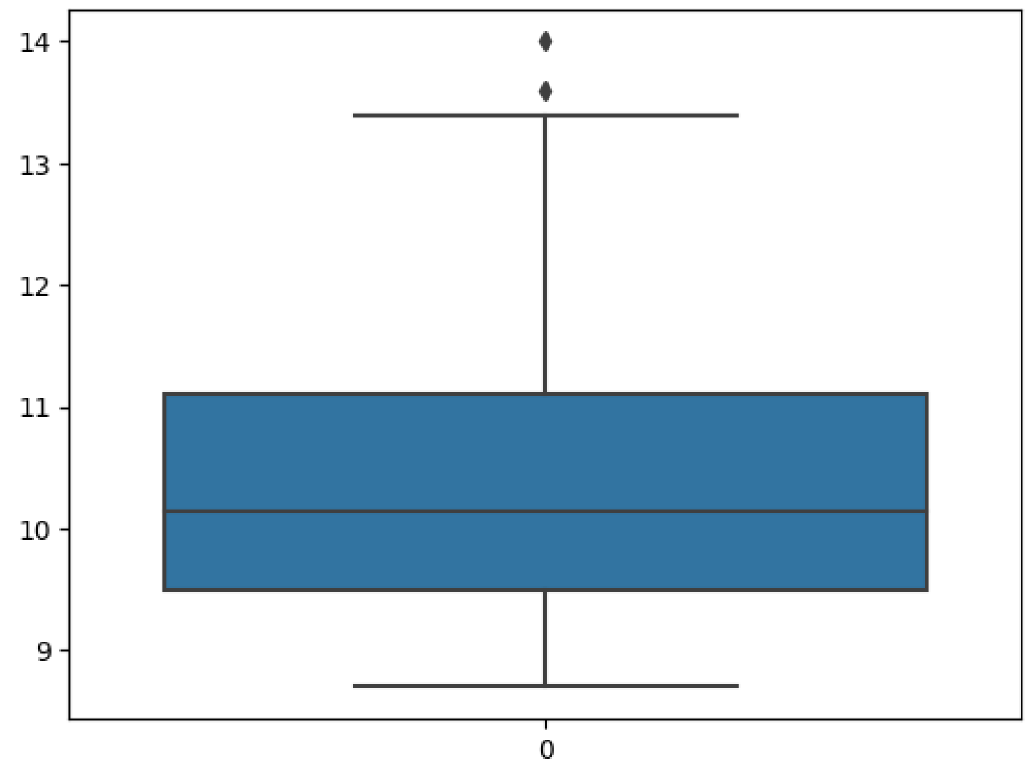
[57]: <Axes: >



Checking for outliers in Alcohol Column

[58]:
```
sns.boxplot(df.alcohol)
```

[58]: <Axes: >

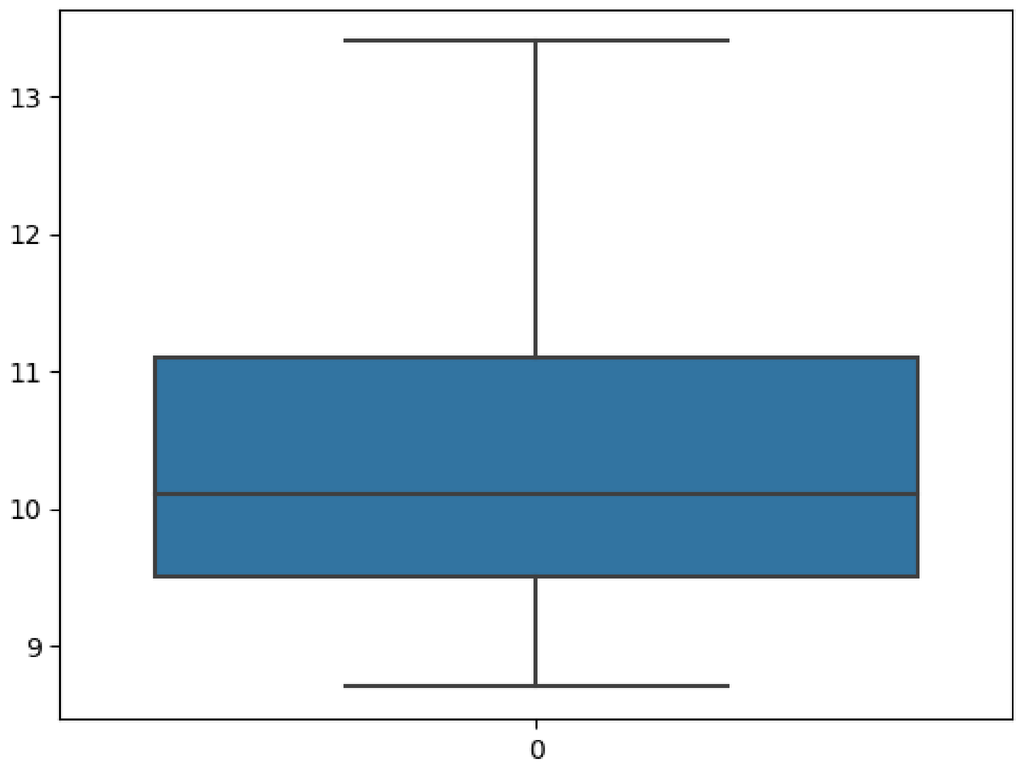


There are outliers in Alcohol. Hence removing them.

[59]:
```
#Using IQR method

f1= df['alcohol'].quantile(0.25) #First Quartile
f3= df['alcohol'].quantile(0.75) #Third Quartile
IQR=f3-f1 #Inter Quertile range

Upper_limit = f3+(1.5)*IQR
Lower_limit = f1-(1.5)*IQR

df=df[(df['alcohol']<Upper_limit) & (df['alcohol']>Lower_limit)]
```

[61]:
```
sns.boxplot(df.alcohol)
```

[61]: <Axes: >



We Observe that outliers in all the Columns are removed.

Seperating data into dependent and independent columns

[63]:
```
X= df.drop('quality',axis=1);
```

[64]:
```
X
```

[64]:
```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0               7.4             0.700         0.00             1.9      0.076
1               7.8             0.880         0.00             2.6      0.098
2               7.8             0.760         0.04             2.3      0.092
3              11.2             0.280         0.56             1.9      0.075
4               7.4             0.700         0.00             1.9      0.076
…                …                 …           …               …        …
1594            6.2             0.600         0.08             2.0      0.090
1595            5.9             0.550         0.10             2.2      0.062
1596            6.3             0.510         0.13             2.3      0.076
1597            5.9             0.645         0.12             2.0      0.075
1598            6.0             0.310         0.47             3.6      0.067

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                    11.0                  34.0  0.99780  3.51       0.56
```

```
1                    25.0                     67.0  0.99680  3.20        0.68
2                    15.0                     54.0  0.99700  3.26        0.65
3                    17.0                     60.0  0.99800  3.16        0.58
4                    11.0                     34.0  0.99780  3.51        0.56
…                     …                    …      …     …          …
1594                 32.0                     44.0  0.99490  3.45        0.58
1595                 39.0                     51.0  0.99512  3.52        0.76
1596                 29.0                     40.0  0.99574  3.42        0.75
1597                 32.0                     44.0  0.99547  3.57        0.71
1598                 18.0                     42.0  0.99549  3.39        0.66

      alcohol
0         9.4
1         9.8
2         9.8
3         9.8
4         9.4
…       …
1594     10.5
1595     11.2
1596     11.0
1597     10.2
1598     11.0

[1184 rows x 11 columns]
```

[65]:
```
#Getting Dependent variable
Y=df.quality
```

[66]:
```
Y
```

[66]:
```
0       5
1       5
2       5
3       6
4       5
       ..
1594    5
1595    6
1596    6
1597    5
1598    6
Name: quality, Length: 1184, dtype: int64
```

Performing Binarization

We are considering quality above 6 means wine is good denoted as 1 else its bad denoted as 0

[67]:
```
Y=Y.apply(lambda y_value :1 if y_value>=7 else 0)
```

[68]:
```
Y
```

[68]:
```
0       0
1       0
2       0
3       0
4       0
       ..
1594    0
1595    0
1596    0
1597    0
1598    0
Name: quality, Length: 1184, dtype: int64
```

Performing train Test split

[62]:
```
from sklearn.model_selection import train_test_split
```

[92]:
```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.
  ↪3,random_state=4)
```

[93]:
```
print(df.shape, X_train.shape, X_test.shape, Y_train.shape, Y_test.shape)
```

```
(1184, 12) (828, 11) (356, 11) (828,) (356,)
```

Using Random Forest Classifier

[81]:
```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()
```

[94]:
```
#Fitting Data

model.fit(X_train,Y_train)
```

[94]:
```
RandomForestClassifier()
```

[95]:
```
y_predict = model.predict(X_test)

y_predict_train = model.predict(X_train)
```

Model Evaluation

[98]:
```
from sklearn.metrics import accuracy_score
```

[96]:
```
print("Accuracy: ", accuracy_score(Y_test,y_predict))
```

Accuracy: 0.9297752808988764

[97]:
```
print("Accuracy: ", accuracy_score(Y_train,y_predict_train))
```

Accuracy: 1.0

Testing with Random Values

[104]:
```
data = (8.3,        0.84,        0.07,        1.9,        0.1,        18.
 ↪0,        43,        0.884,        4.56,        0.87,        8.9)
model.predict((np.asarray(data)).reshape(1,-1))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but RandomForestClassifier was fitted with feature
names
  warnings.warn(
```

[104]:
```
array([0])
```

We got label as 0, Means the wine is of Bad Quality