Arnav

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

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

```
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
```

```
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.corr()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH
fixed acidity	1.000000	-0.256131	0.671703	0.114777	0.093705	-0.153794	-0.113181	0.668047	-0.682978
volatile acidity	-0.256131	1.000000	-0.552496	0.001918	0.061298	-0.010504	0.076470	0.022026	0.234937
citric acid	0.671703	-0.552496	1.000000	0.143577	0.203823	-0.060978	0.035533	0.364947	-0.541904

```
df.corr().quality.sort_values(ascending=False)
```

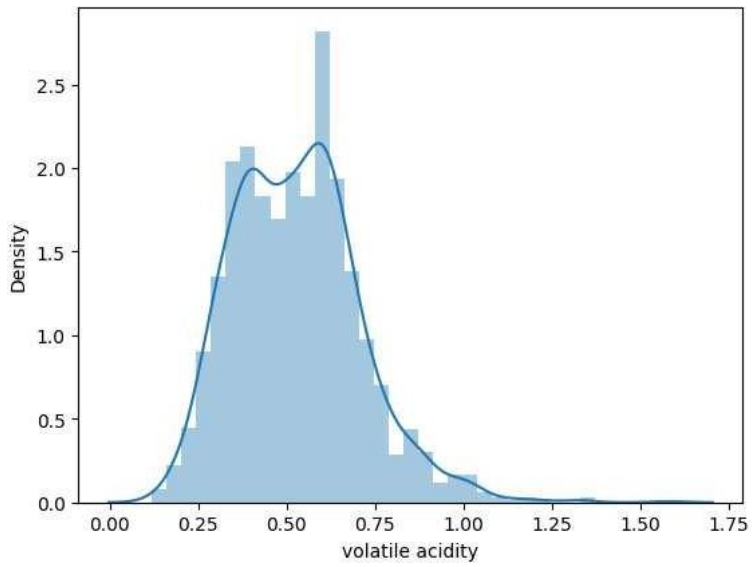
```
quality               1.000000
alcohol               0.476166
sulphates             0.251397
citric acid           0.226373
fixed acidity         0.124052
residual sugar        0.013732
free sulfur dioxide  -0.050656
pH                   -0.057731
chlorides            -0.128907
density              -0.174919
total sulfur dioxide -0.185100
volatile acidity     -0.390558
Name: quality, dtype: float64
```

Visualization

```
import seaborn as sns
from matplotlib import rcParams
```

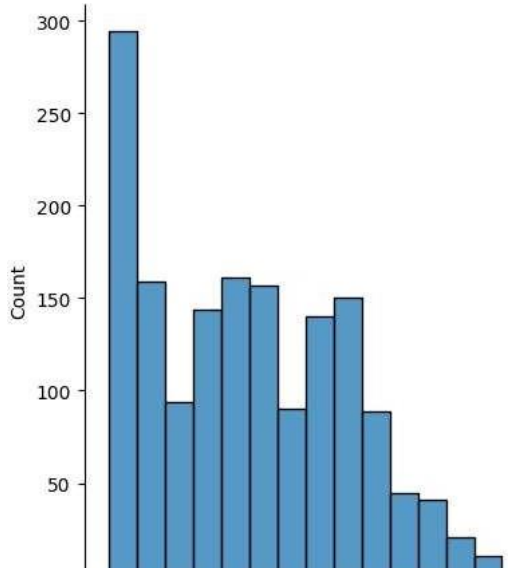
```
sns.distplot(df['volatile acidity'])
```

```
<ipython-input-126-6077730c287e>: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['volatile acidity'])
<Axes: xlabel='volatile acidity', ylabel='Density'>
```



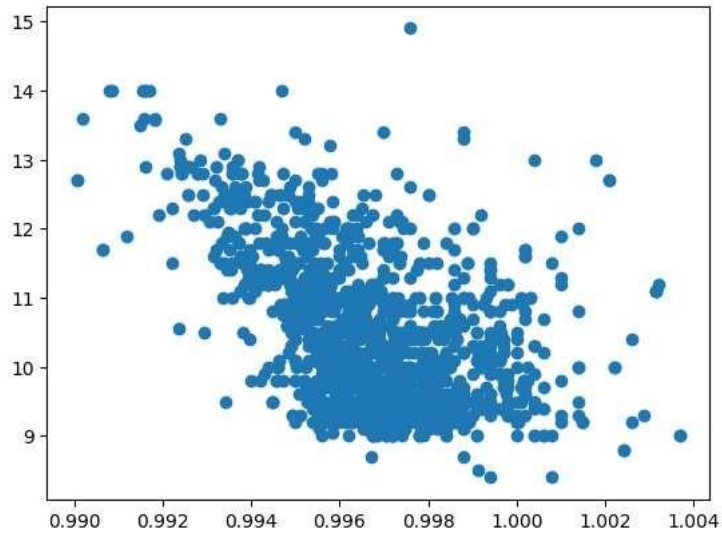

```
sns.displot(df['citric acid'])
```

```
<seaborn.axisgrid.FacetGrid at 0x7cbb7c34e4a0>
```



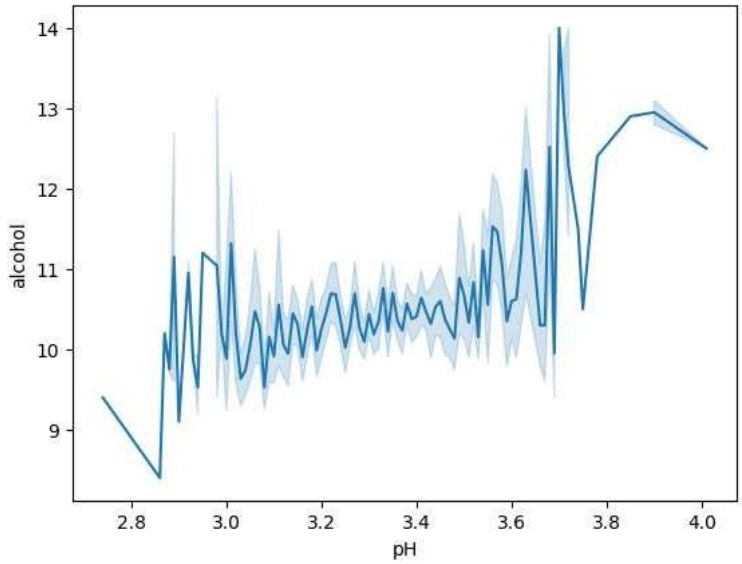

```
plt.scatter(df['density'],df['alcohol'])
```

```
<matplotlib.collections.PathCollection at 0x7cbb7bf5e3e0>
```



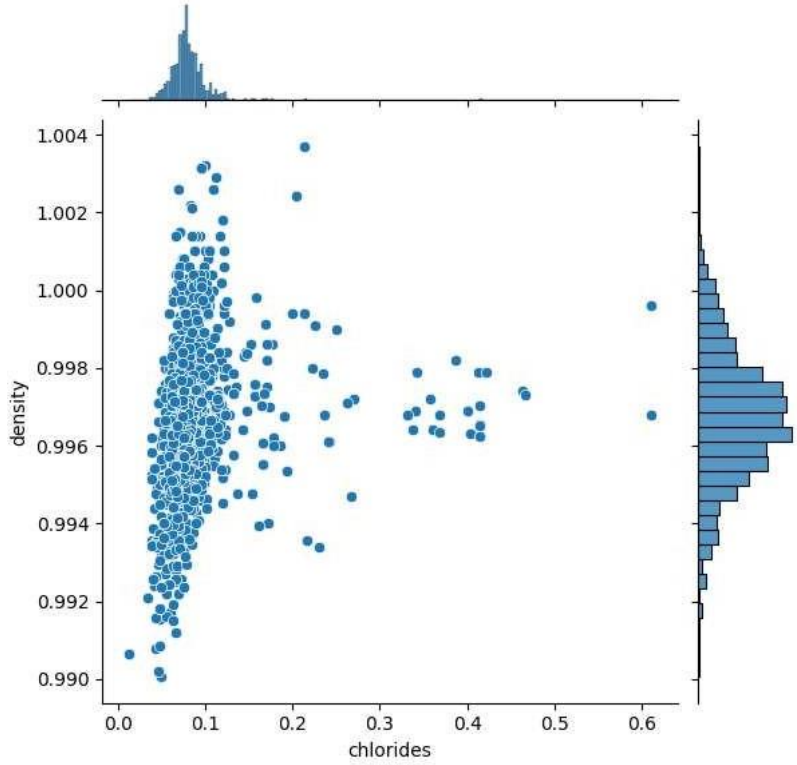

```
sns.lineplot(x=df['pH'],y=df['alcohol'])
```

```
<Axes: xlabel='pH', ylabel='alcohol'>
```



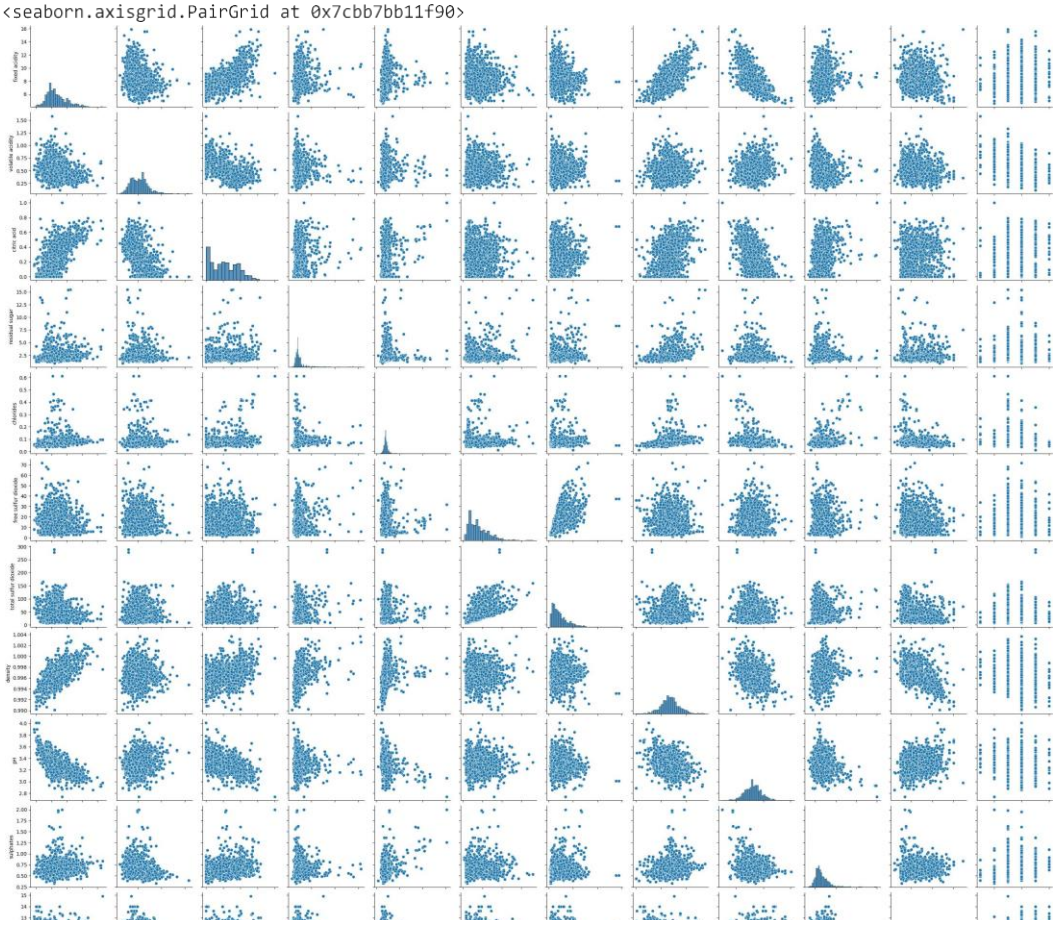
```
sns.jointplot(x=df['chlorides'],y=df['density'],data=df)
```

```
<seaborn.axisgrid.JointGrid at 0x7cbb7d2780a0>
```




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

```
<seaborn.axisgrid.PairGrid at 0x7cbb7bb11f90>
```



```
plt.figure(figsize = (16,5))
sns.heatmap(df.corr(),annot=True,linewidths=.5)
```

```
<Axes: >
```

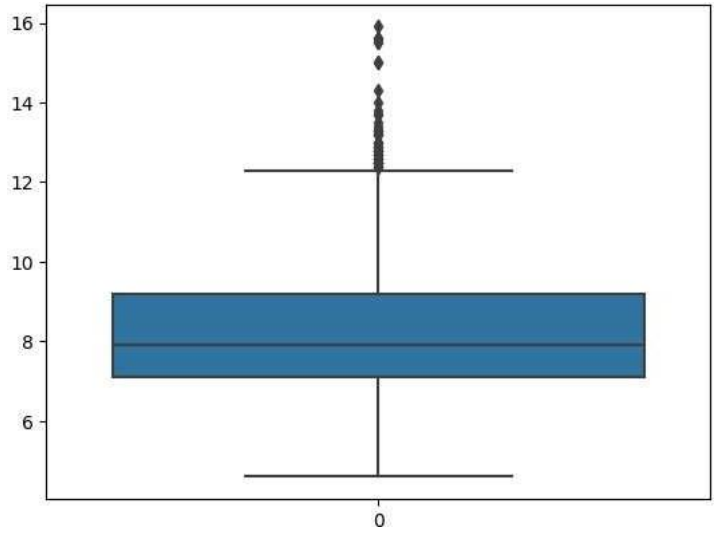

```
df.info()
```

```
<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
```

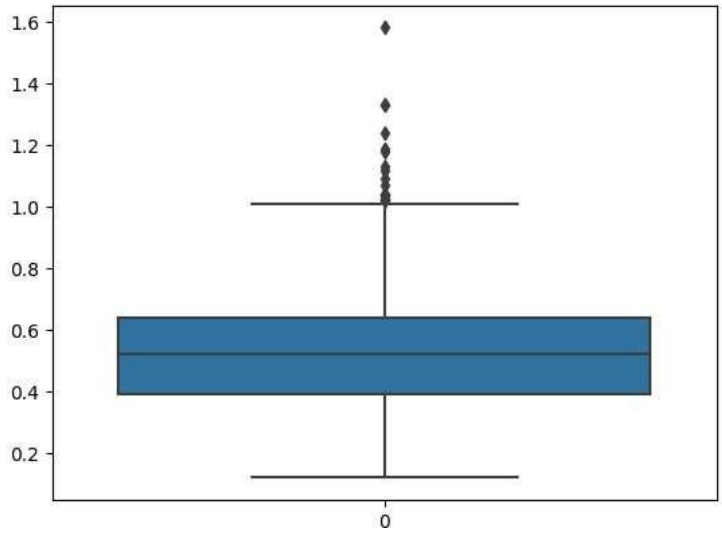
```
sns.boxplot(df['fixed acidity'])
```

```
<Axes: >
```



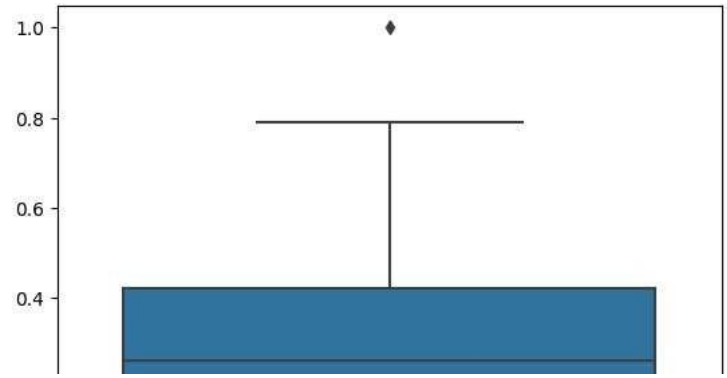

```
sns.boxplot(df['volatile acidity'])
```

```
<Axes: >
```



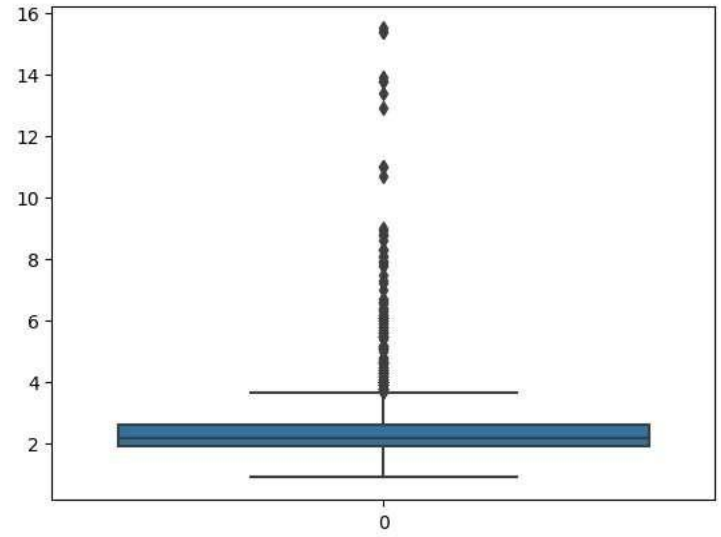

```
sns.boxplot(df['citric acid'])
```

```
<Axes: >
```



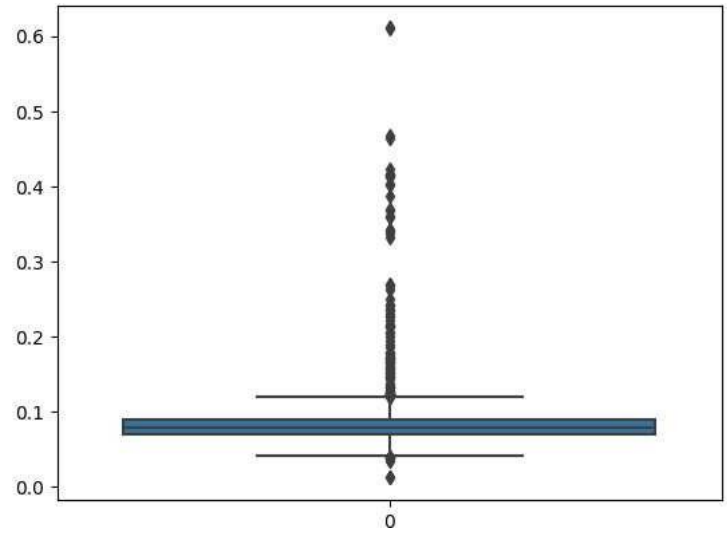

```
sns.boxplot(df['residual sugar'])
```

```
<Axes: >
```



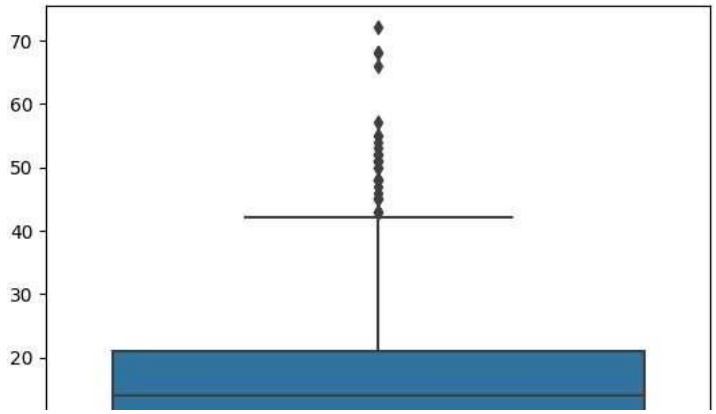

```
sns.boxplot(df['chlorides'])
```

```
<Axes: >
```



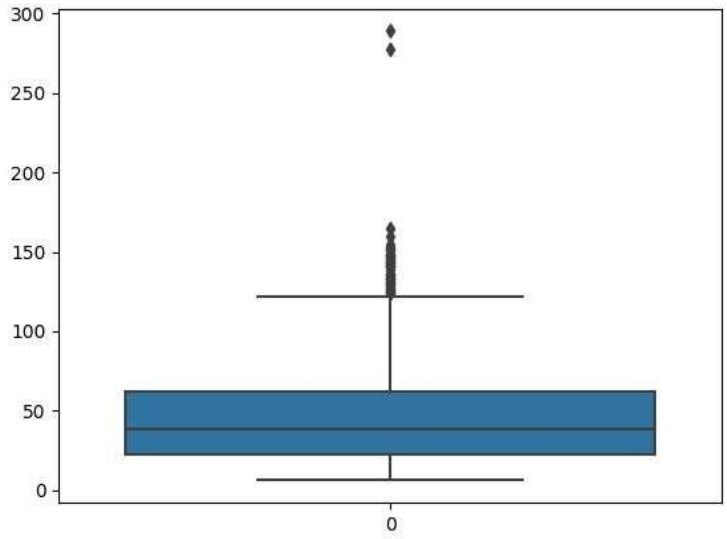

```
sns.boxplot(df['free sulfur dioxide'])
```

```
<Axes: >
```



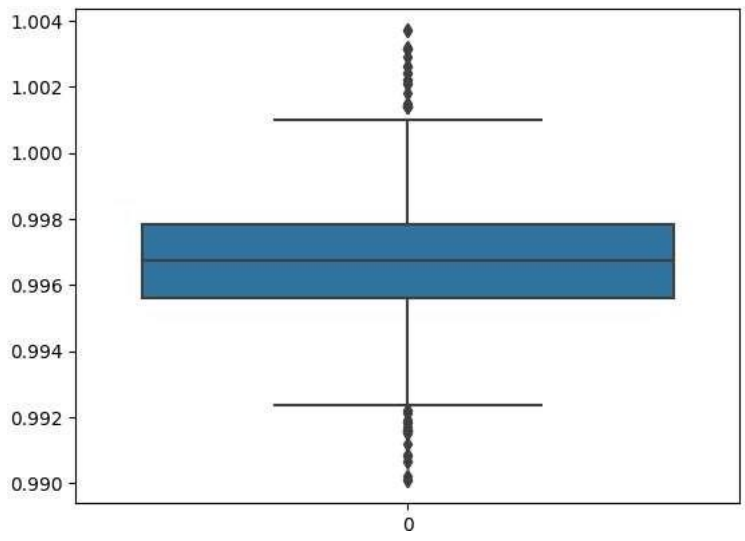

```
sns.boxplot(df['total sulfur dioxide'])
```

```
<Axes: >
```



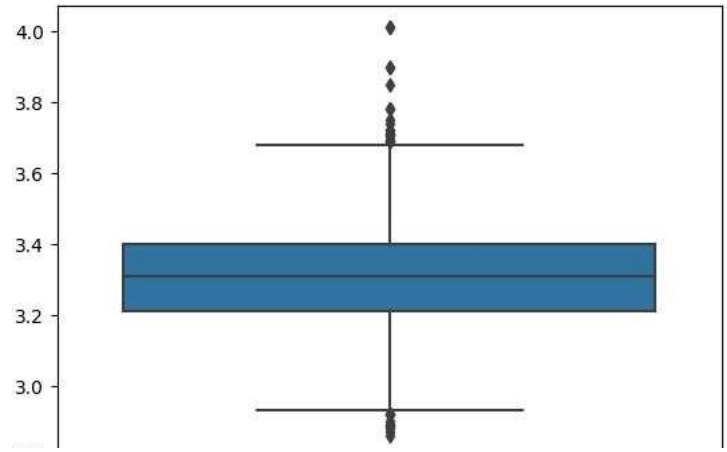

```
sns.boxplot(df['density'])
```

```
<Axes: >
```



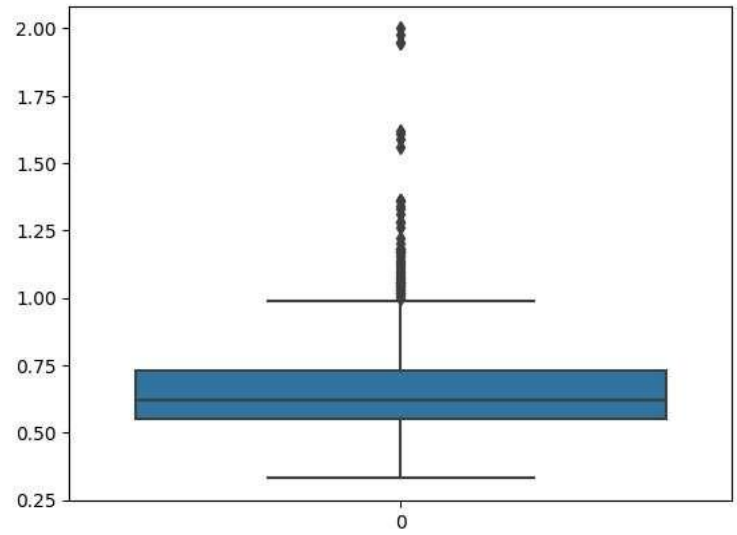

```
sns.boxplot(df['pH'])
```

```
<Axes: >
```



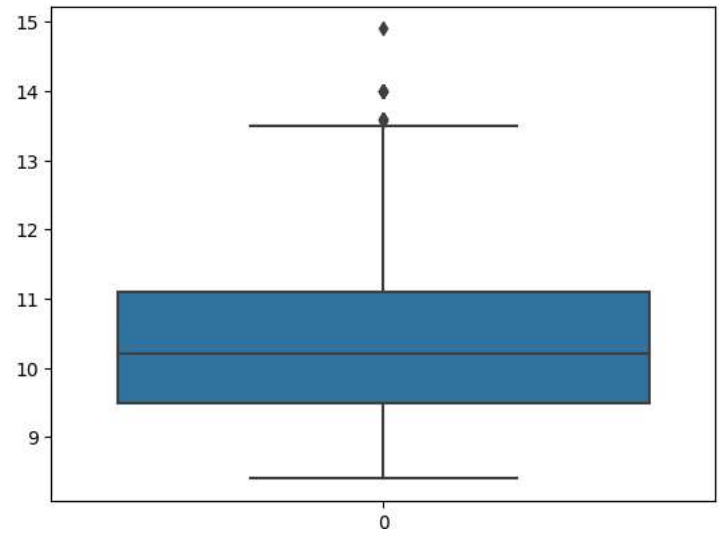

```
sns.boxplot(df['sulphates'])
```

```
<Axes: >
```



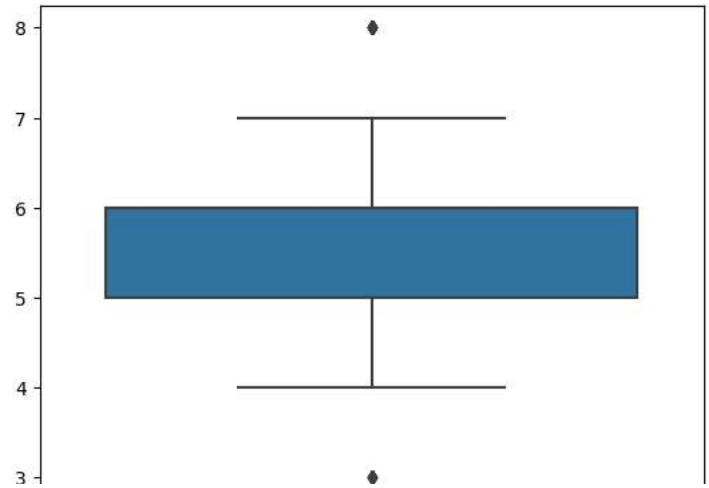

```
sns.boxplot(df['alcohol'])
```

```
<Axes: >
```




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

```
<Axes: >
```

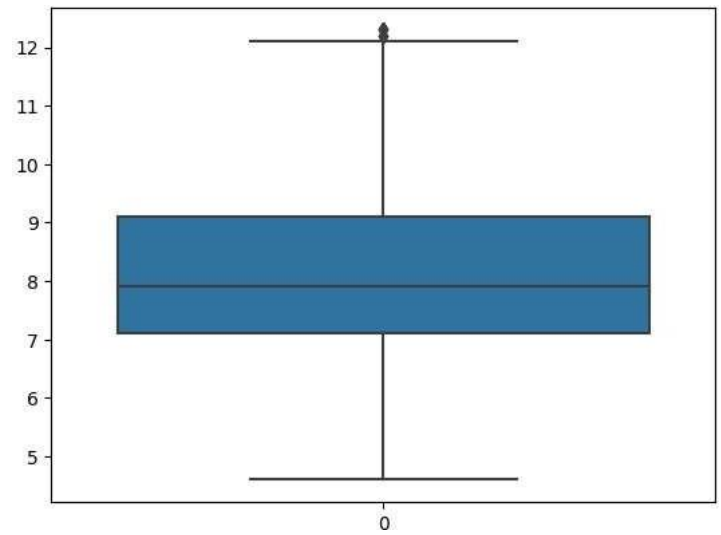



```
df.shape
```

```
(1599, 12)
```

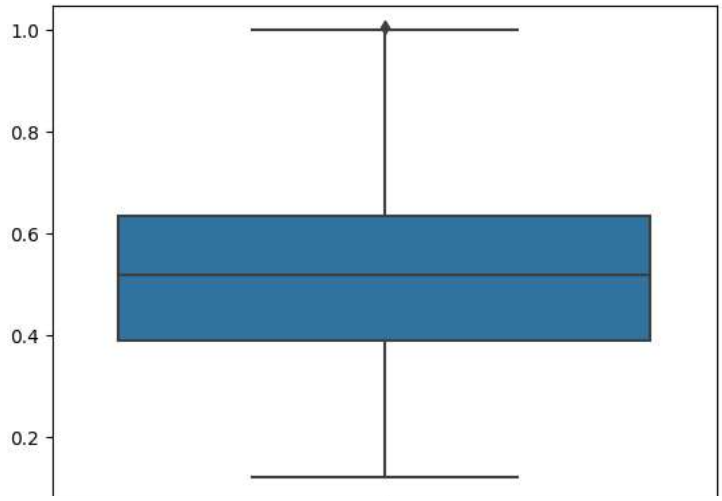
```
# outlier removal for fixed acidity
q1=df['fixed acidity'].quantile(0.25)
q3=df['fixed acidity'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR


df=df[df['fixed acidity']<upper_limit]


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

```
<Axes: >
```



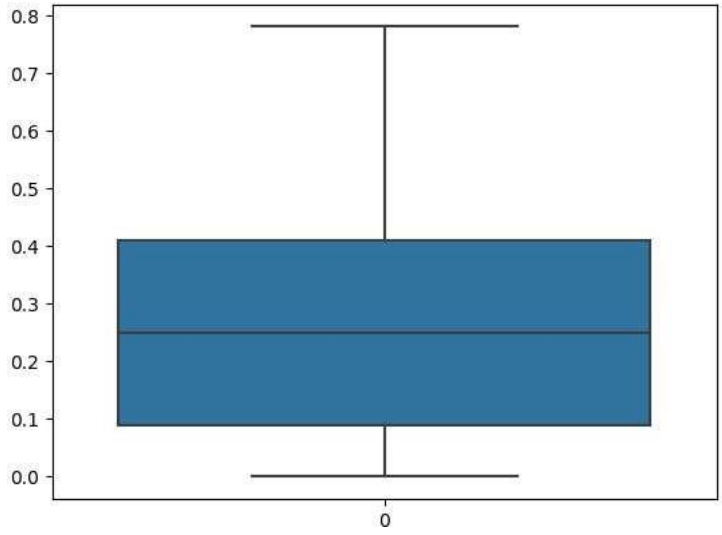

```
# outlier removal for volatile acidity
q1=df['volatile acidity'].quantile(0.25)
q3=df['volatile acidity'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR


df=df[df['volatile acidity']<upper_limit]


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

```
<Axes: >
```



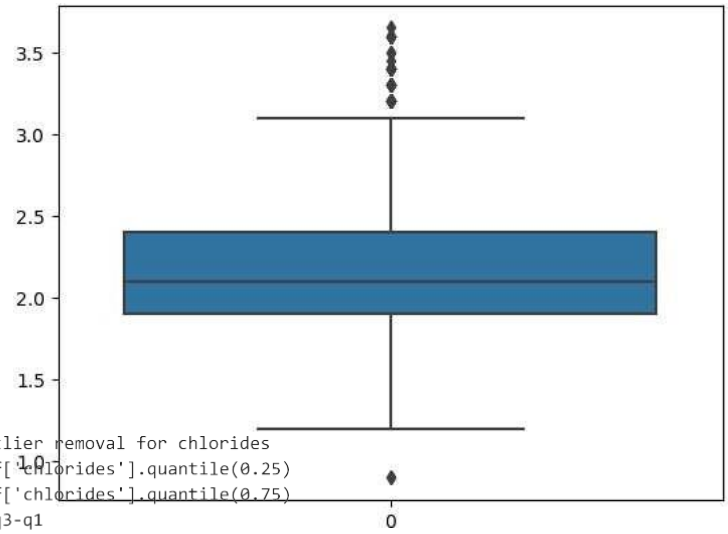

```
# outlier removal for citric acid
q1=df['citric acid'].quantile(0.25)
q3=df['citric acid'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR


df=df[df['citric acid']<upper_limit]


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

```
<Axes: >
```



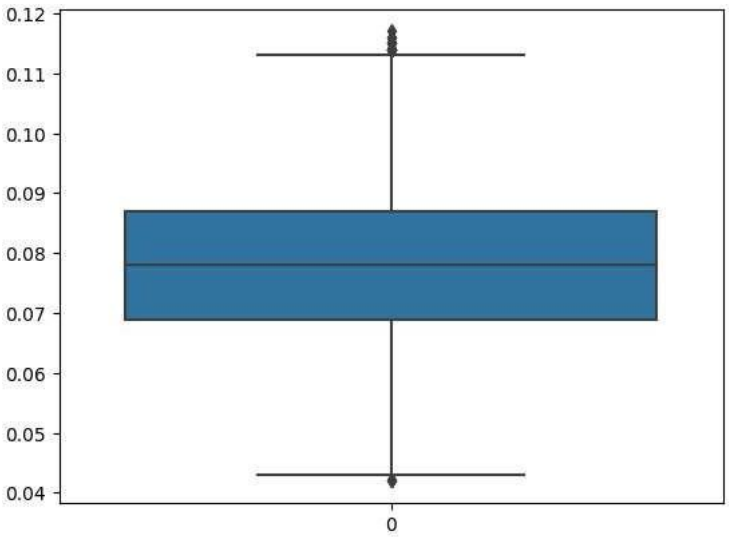

```
# outlier removal for residual sugar
q1=df['residual sugar'].quantile(0.25)
q3=df['residual sugar'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR


df=df[df['residual sugar']<upper_limit]
df=df[df['residual sugar']>lower_limit]


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

```
<Axes: >
```



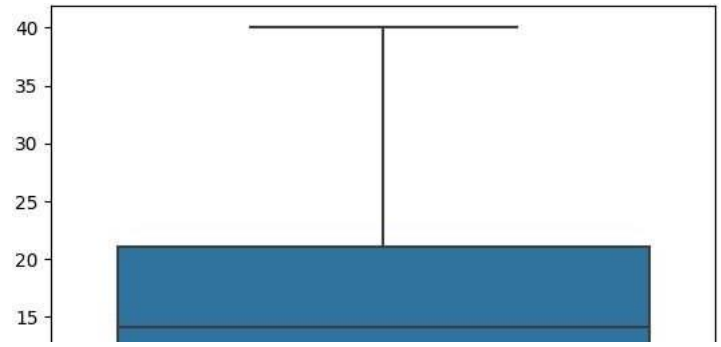

```
# outlier removal for chlorides
q1=df['chlorides'].quantile(0.25)
q3=df['chlorides'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR


df=df[df['chlorides']<upper_limit]
df=df[df['chlorides']>lower_limit]


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

```
<Axes: >
```




```
# outlier removal for free sulfur dioxide
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=df[df['free sulfur dioxide']<upper_limit]


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

```
<Axes: >
```

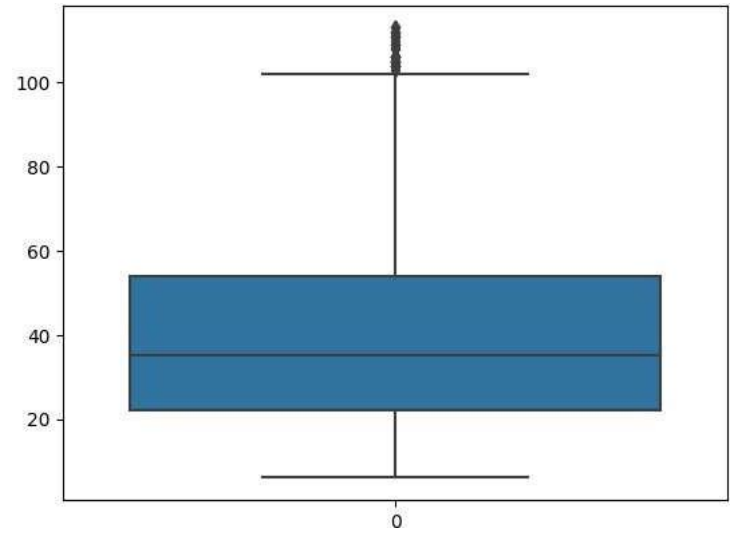



```
# outlier removal for total sulfur dioxide
q1=df['total sulfur dioxide'].quantile(0.25)
q3=df['total sulfur dioxide'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

```
df=df[df['total sulfur dioxide']<upper_limit]
```

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

```
<Axes: >
```

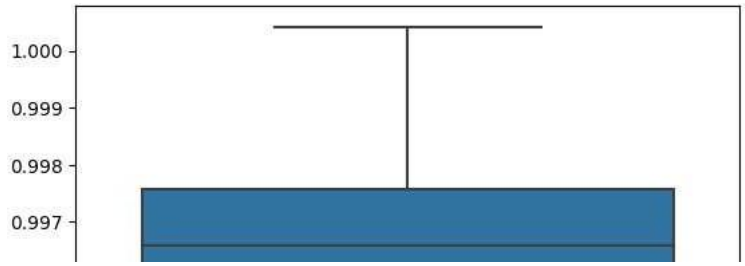



```
# outlier removal for density
q1=df['density'].quantile(0.25)
q3=df['density'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

```
df=df[df['density']<upper_limit]
df=df[df['density']>lower_limit]
```

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

```
<Axes: >
```

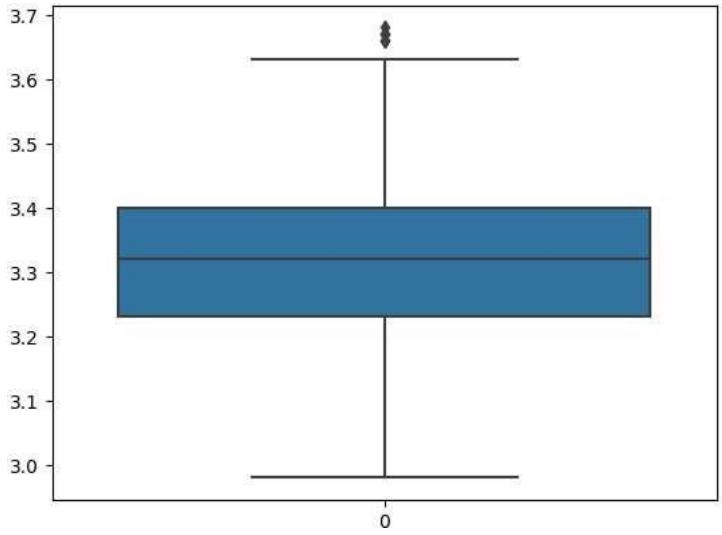



```
# outlier removal for pH
q1=df['pH'].quantile(0.25)
q3=df['pH'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```


```
df=df[df['pH']<upper_limit]
df=df[df['pH']>lower_limit]
```

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

```
<Axes: >
```

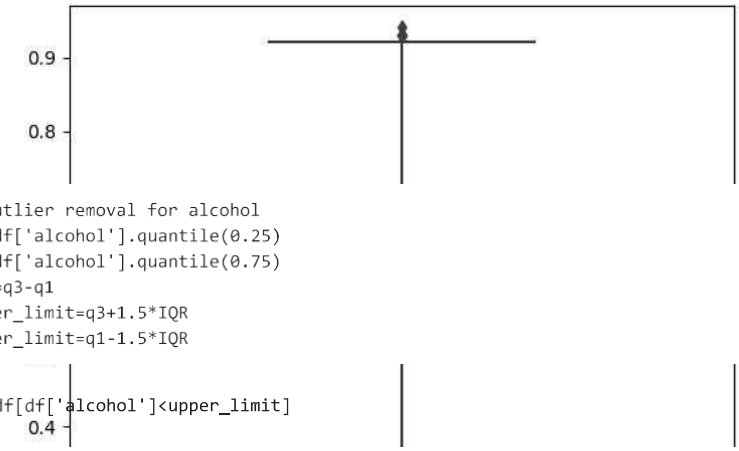



```
# outlier removal for sulphates
q1=df['sulphates'].quantile(0.25)
q3=df['sulphates'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
```

```
df=df[df['sulphates']<upper_limit]
```

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

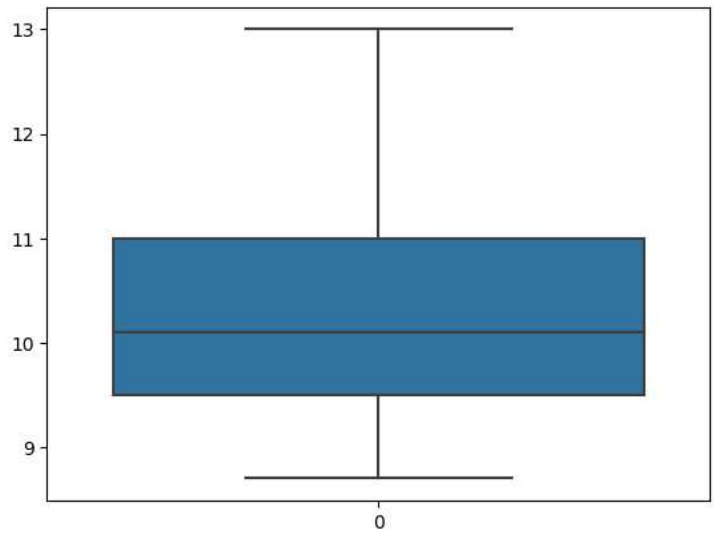
<Axes: >




```
# outlier removal for alcohol
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=df[df['alcohol']<upper_limit]
```

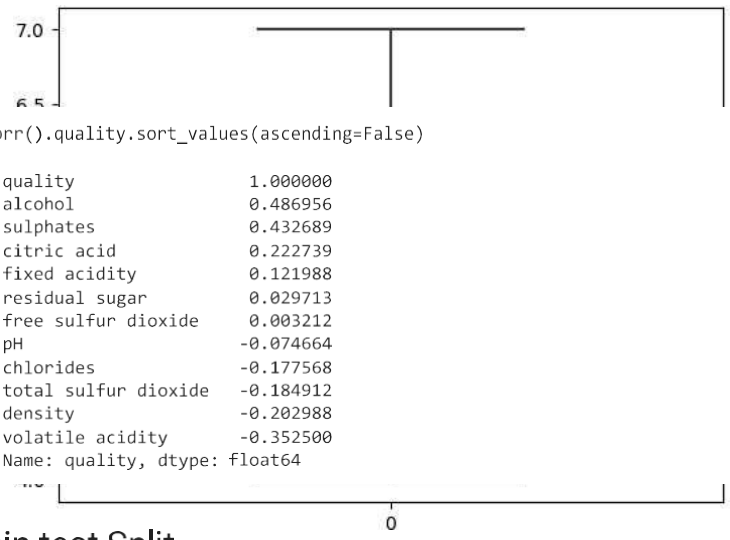
```
sns.boxplot(df['alcohol'])
```

<Axes: >




```
# outlier removal for quality
q1=df['quality'].quantile(0.25)
q3=df['quality'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR


df=df[df['quality']<upper_limit]
df=df[df['quality']>lower_limit]


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

```
<Axes: >
```




```
df.corr().quality.sort_values(ascending=False)
```

```
quality                1.000000
alcohol                0.486956
sulphates              0.432689
citric acid            0.222739
fixed acidity          0.121988
residual sugar         0.029713
free sulfur dioxide    0.003212
pH                    -0.074664
chlorides             -0.177568
total sulfur dioxide  -0.184912
density               -0.202988
volatile acidity      -0.352500
Name: quality, dtype: float64
```

Train test Split

```
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

```
y=df['quality']
y.head()
```

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

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.25,random_state=0)
```

```
x_train.shape
```

```
(843, 11)
```

```
x_test.shape
```

```
(281, 11)
```

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
model2= RandomForestClassifier()
```

```
model2.fit(x_train,y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
r_y_predict=model2.predict(x_test)
r_y_predict_train= model2.predict(x_train)
```

```
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
```

```
print("Testing accuracy", accuracy_score(y_test,r_y_predict))
print("Training accuracy", accuracy_score(y_train,r_y_predict_train))
```

```
Testing accuracy 0.7366548042704626
Training accuracy 1.0
```

```
pd.crosstab(y_test,r_y_predict)
```

col_0	5	6	7
quality			
4	8	3	0
5	104	18	0
6	32	83	2
7	1	10	20

```
print(classification_report(y_test,r_y_predict))
```

```
              precision    recall  f1-score   support

           4       0.00      0.00      0.00        11
           5       0.72      0.85      0.78       122
           6       0.73      0.71      0.72       117
           7       0.91      0.65      0.75        31

    accuracy                           0.74       281
   macro avg       0.59      0.55      0.56       281
weighted avg       0.71      0.74      0.72       281
```

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

```
x_test.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcoho
1095	9.4	0.40	0.47	2.5	0.087	6.0	20.0	0.99772	3.15	0.50	10.
1301	6.7	0.86	0.07	2.0	0.100	20.0	57.0	0.99598	3.60	0.74	11.
1507	7.5	0.38	0.57	2.3	0.106	5.0	12.0	0.99605	3.36	0.55	11.
1532	7.2	0.53	0.13	2.0	0.058	18.0	22.0	0.99573	3.21	0.68	9.

```
random_pred=model2.predict([[6.7,0.38,0.47,2.0,0.079,5.0,20.0,0.99598,3.21,0.68,9.8]])
random_pred
```

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

```
random_pred1=model2.predict([[7.7,0.48,0.40,2.4,0.099,17.0,30.0,0.99698,3.11,0.78,9.8]])
random_pred1
```

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

```
array([5])
```