# NAME : EDE RENUKA MADHAV

REG NO: 21BCT0244

VIT VELLORE

SLOT: 6:00 PM TO 8:00 PM

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

Load the Dataset

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

   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
```

```
df.shape
```

```
(1599, 12)
```

```
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
```

Data preprocessing

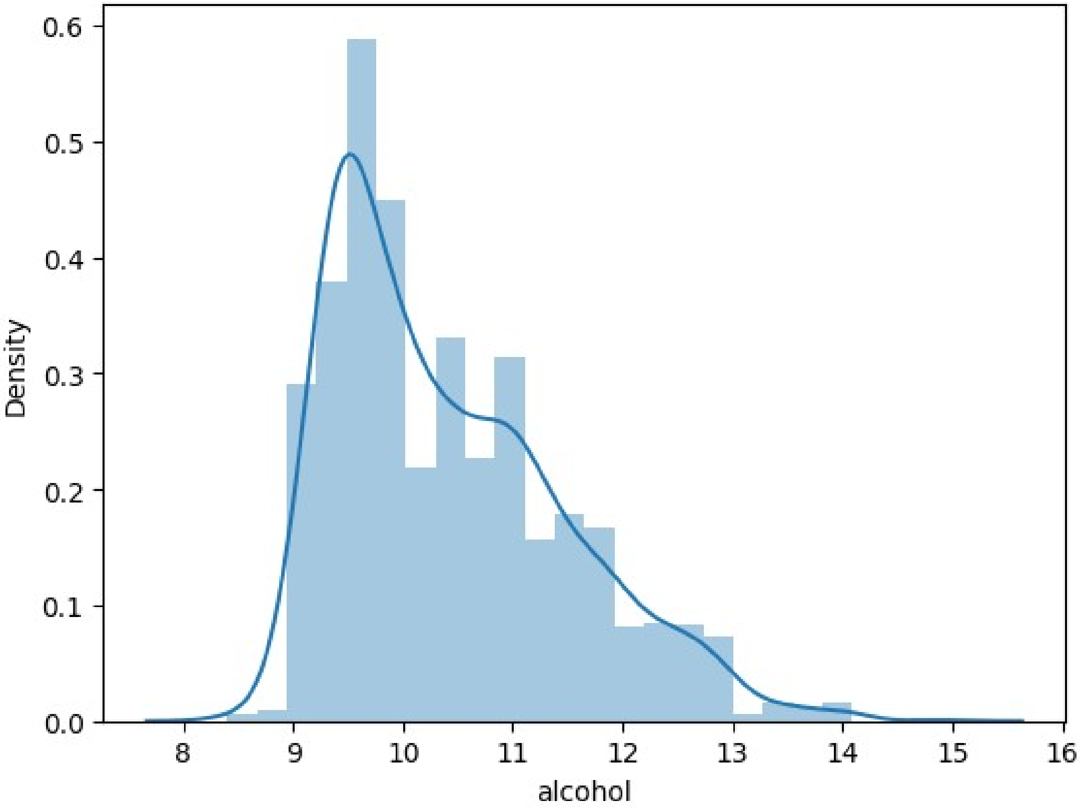
VISUALIZATIONS

1. UNIVARIATE ANALYSIS

```
sns.distplot(df.alcohol)
```

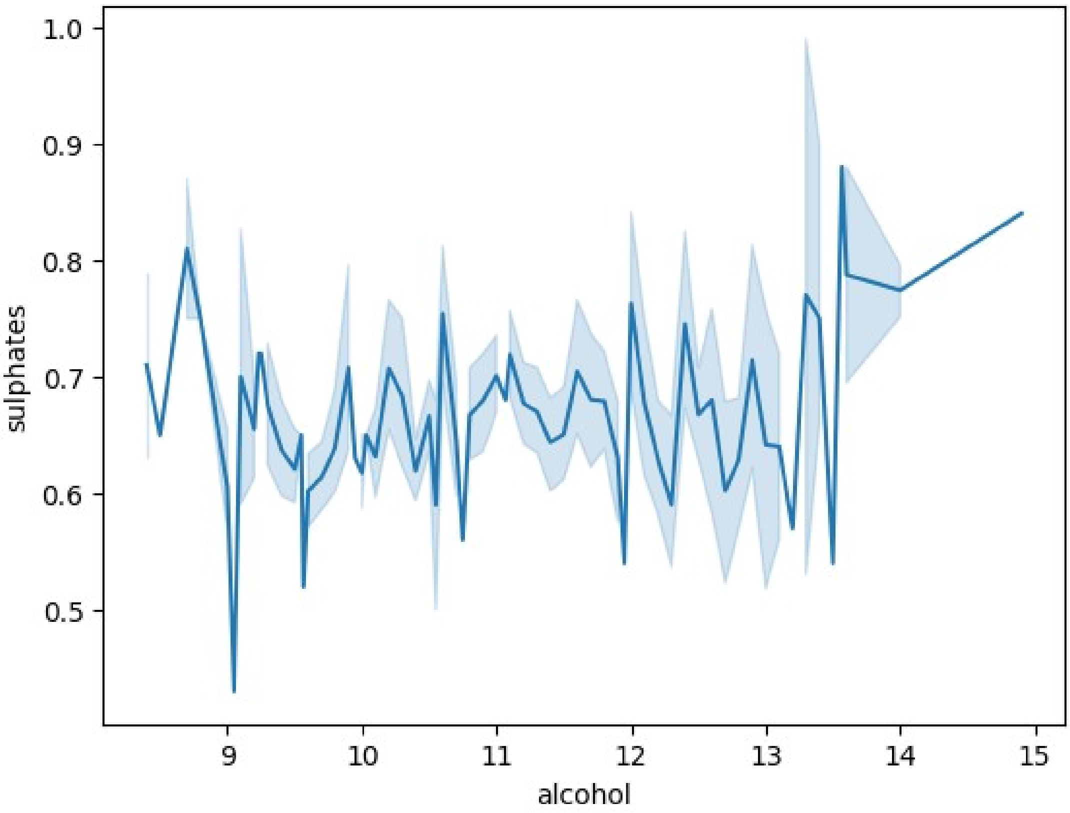
```
<ipython-input-9-cc0e16fd78a8>:1: UserWarning: 

`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.alcohol)

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



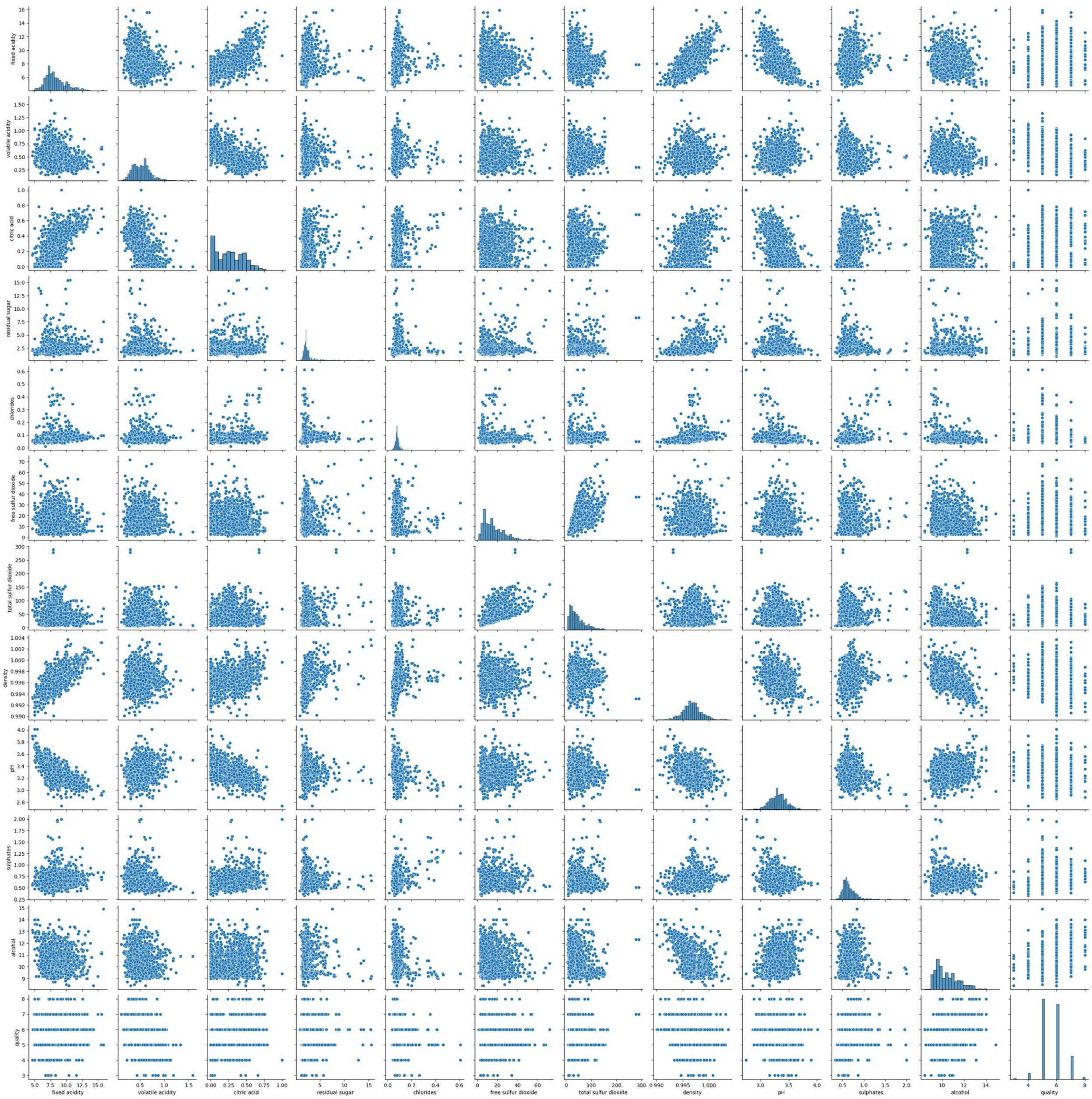
2. BIVARIATE ANALYSIS

```
sns.lineplot(x = df.alcohol ,y = df.sulphates)

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

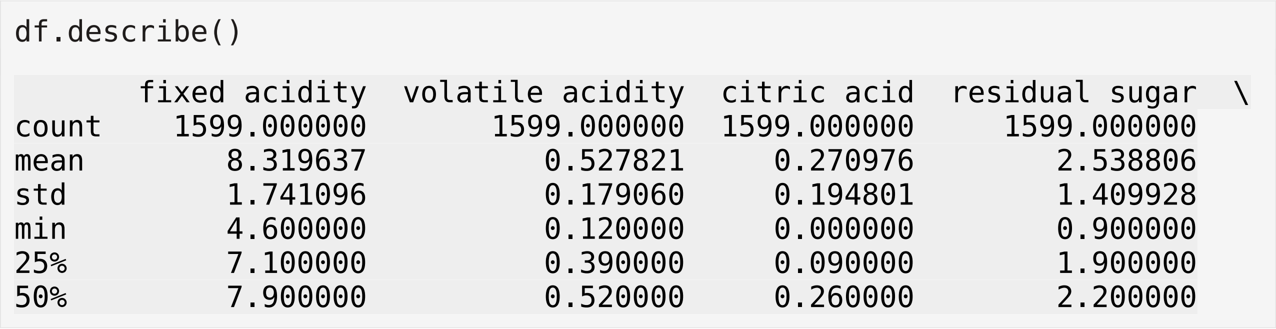


3.MULTIVARIATE ANALYSIS

```
sns.pairplot(df)

<seaborn.axisgrid.PairGrid at 0x7f5c1ddd95d0>
```



STASTISTICAL DATA



```
df.describe()

       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
```

FINDING NULL VALUES

```
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
```

# No Null values(so there are no missing to be handled)

```
df['quality'].value_counts()
```

```
5    611
6    575
7    186
4     46
8     17
3      6
Name: quality, dtype: int64
```

Corelation

```
df.corr()
```

```
                      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
```

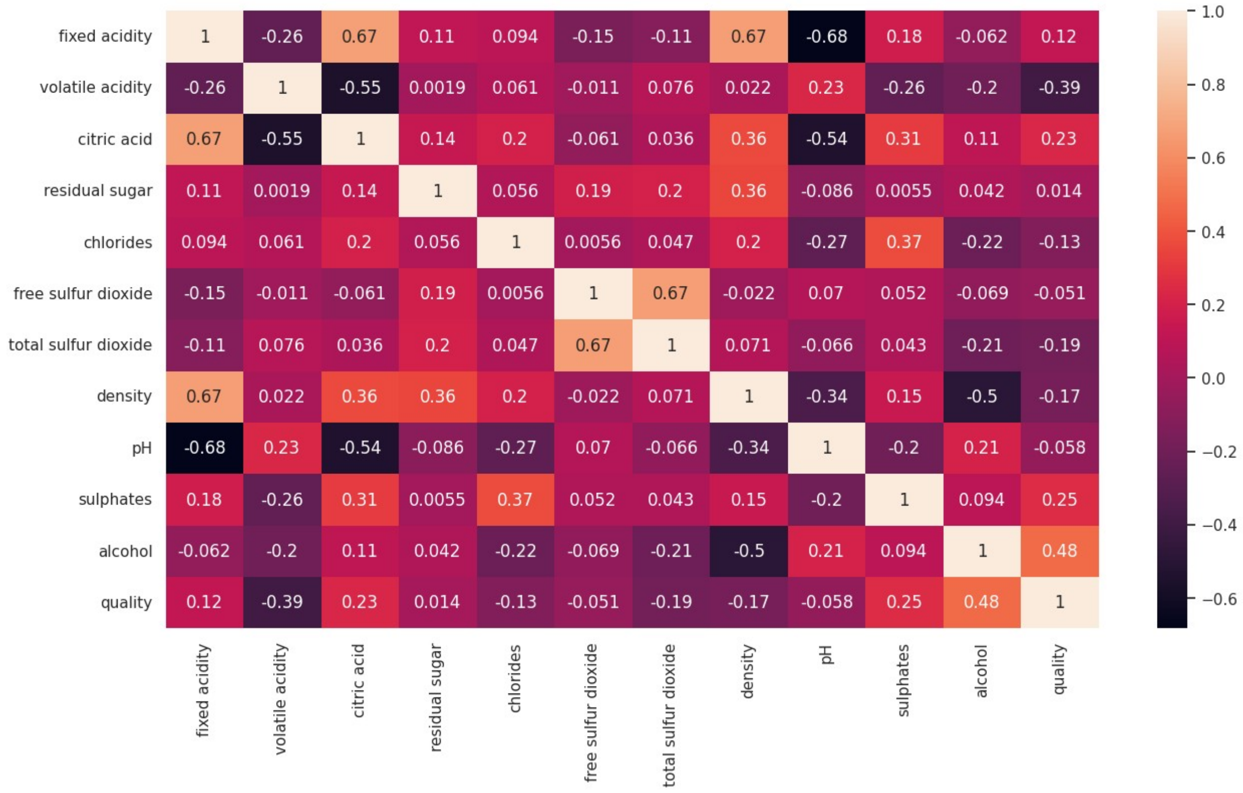
```
alcohol                 1.000000  0.476166
quality                 0.476166  1.000000
```

```
df.corr()['quality'].sort_values(ascending=False)
```

```
quality                   1.000000
alcohol                   0.482143
sulphates                 0.343689
citric acid               0.262596
fixed acidity             0.135510
residual sugar            0.010447
free sulfur dioxide      -0.026502
pH                       -0.084257
density                  -0.175625
total sulfur dioxide     -0.178415
chlorides                -0.194465
volatile acidity         -0.395515
Name: quality, dtype: float64
```

```
sns.heatmap(df.corr(),annot=True)
```

```
<Axes: >
```

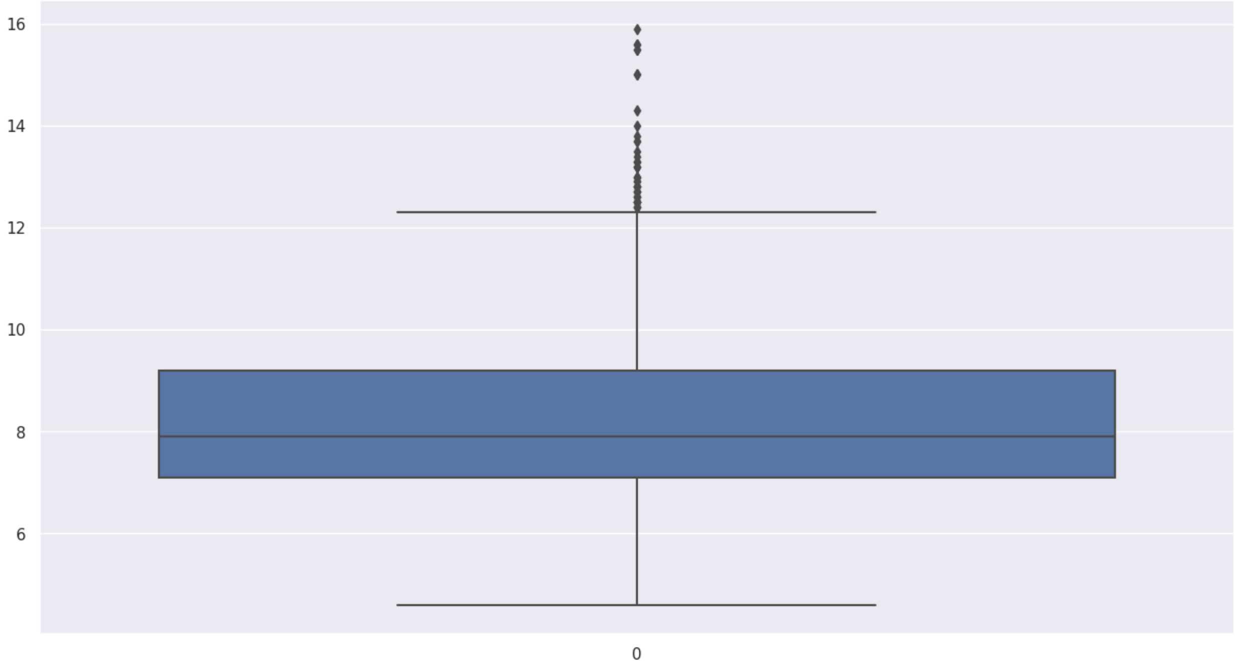


```
df['quality'] = df.quality.apply(lambda x : 1 if x > 6.5 else 0)
```

OUTLIER DETECTION

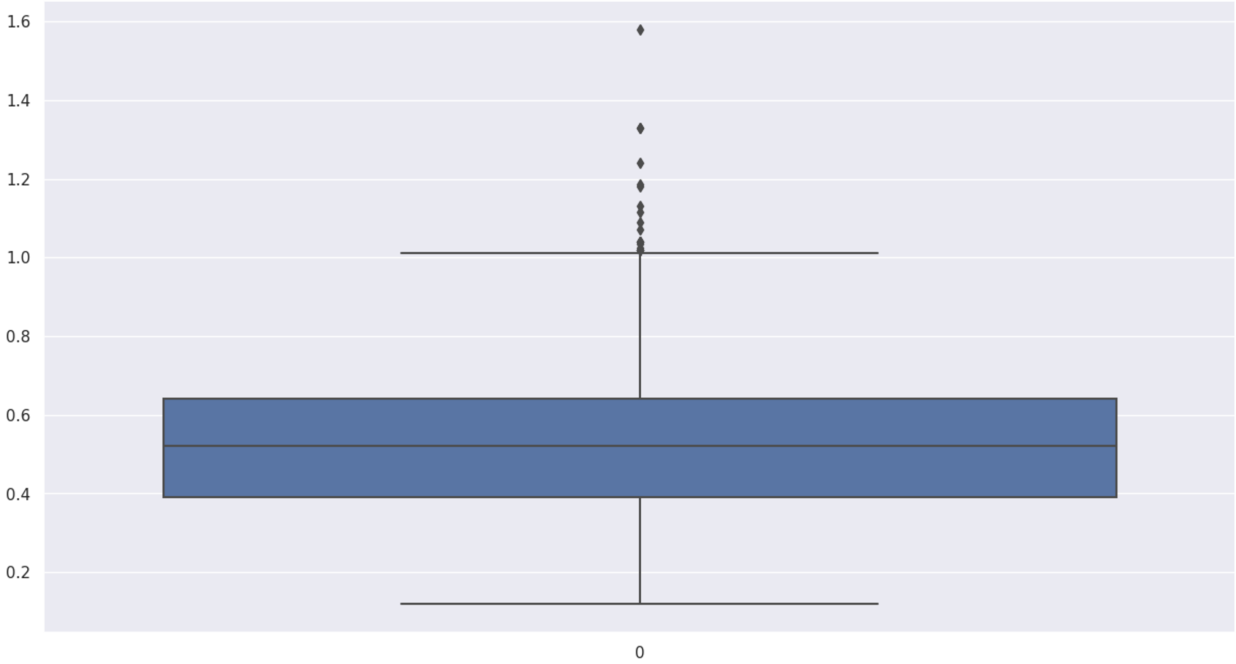
```
sns.boxplot(df["fixed acidity"])
```

```
<Axes: >
```



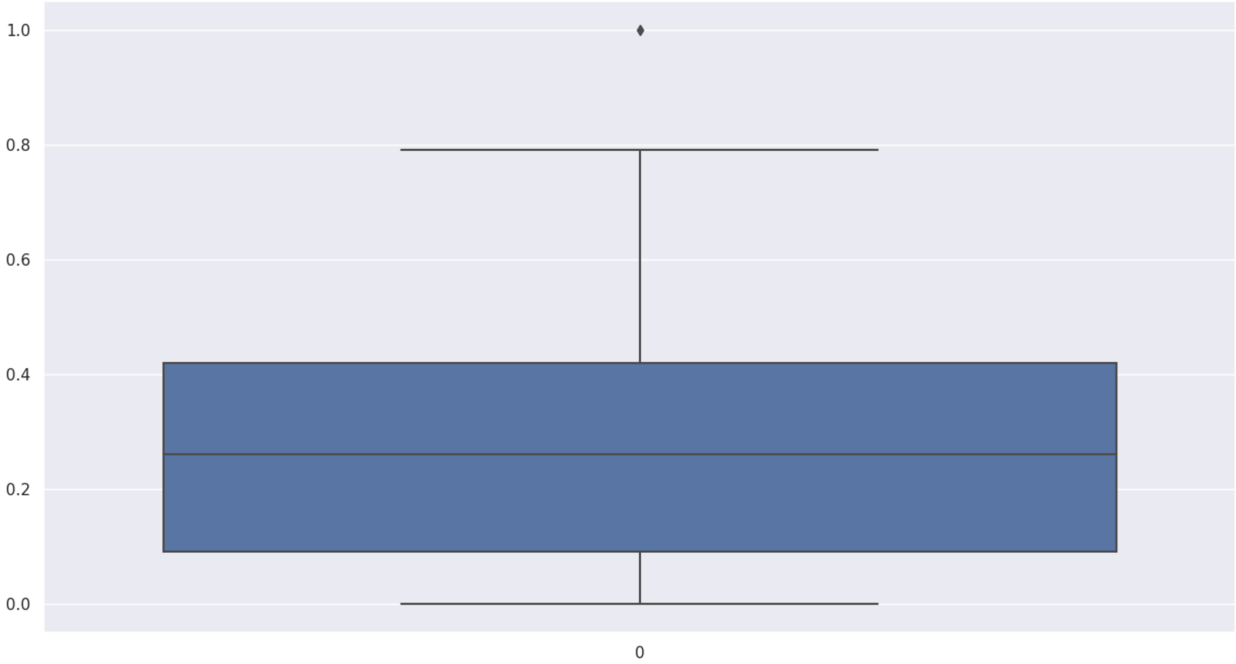
```
sns.boxplot(df["volatile acidity"])
```

```
<Axes: >
```



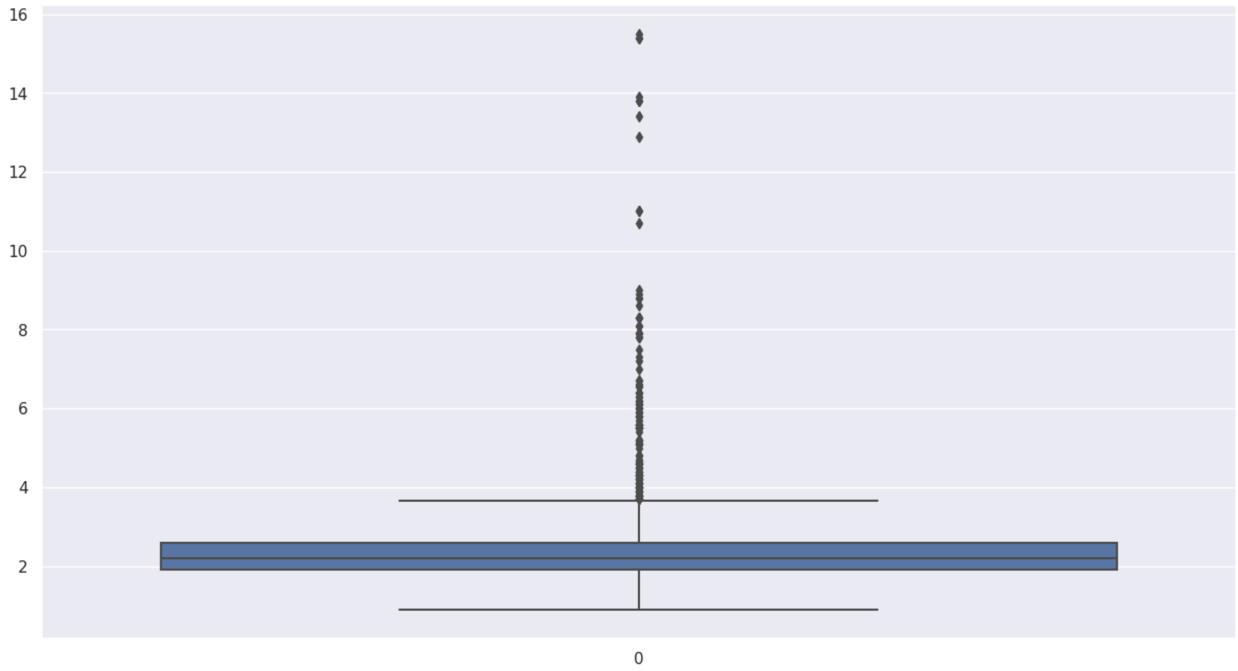
```
sns.boxplot(df["citric acid"])
```

```
<Axes: >
```



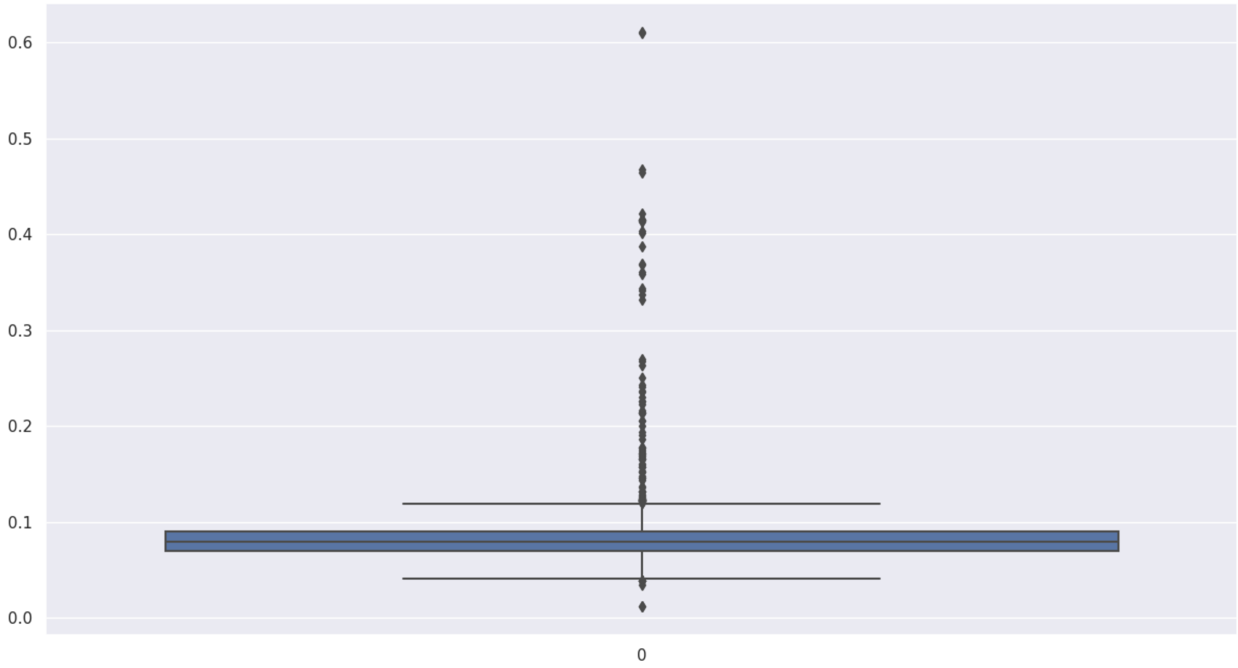
```
sns.boxplot(df["residual sugar"])
```

```
<Axes: >
```



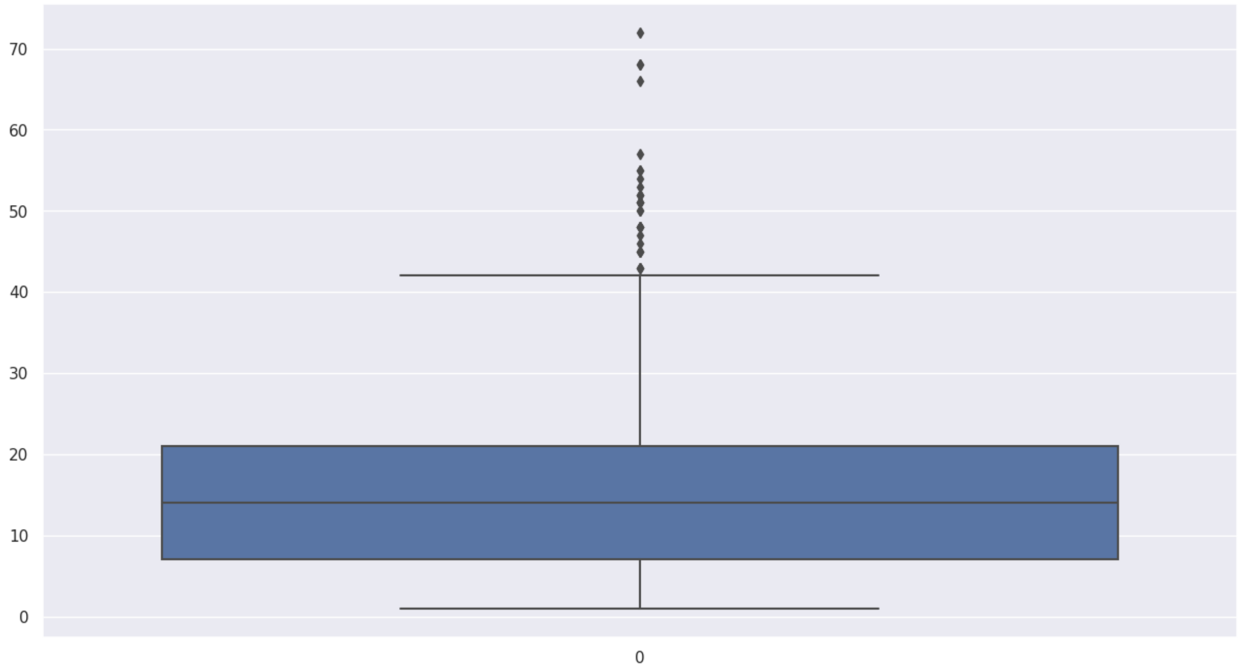
```
sns.boxplot(df["chlorides"])
```

```
<Axes: >
```



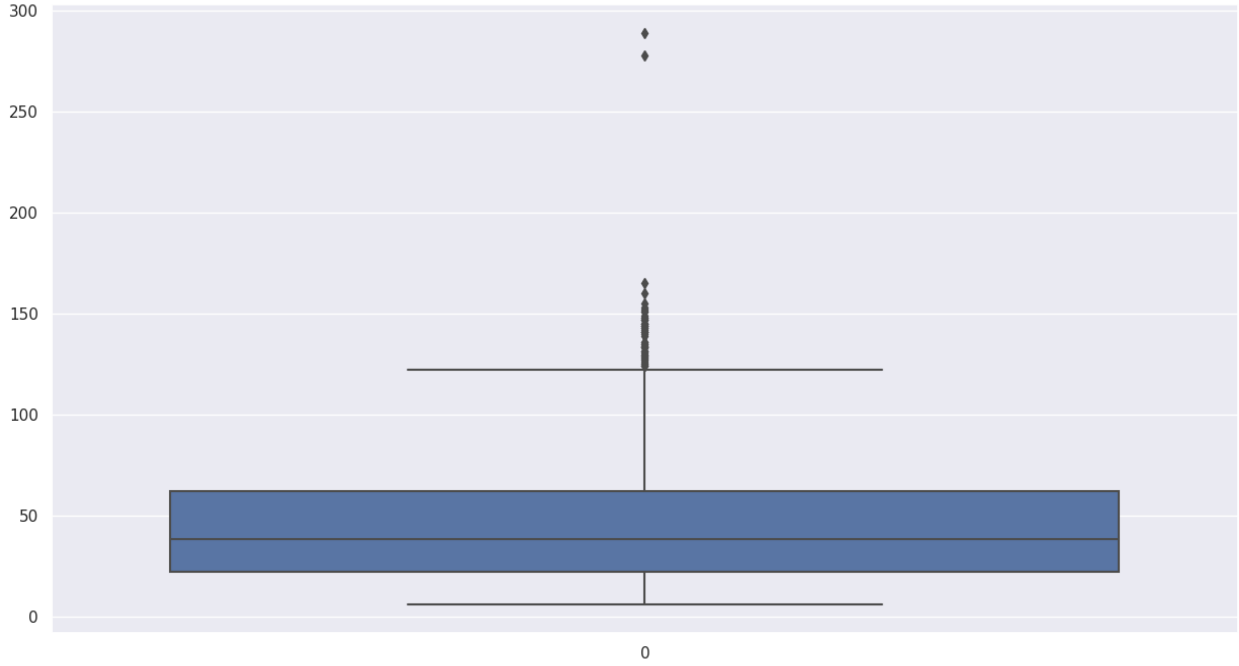
```
sns.boxplot(df['free sulfur dioxide'])
```

```
<Axes: >
```



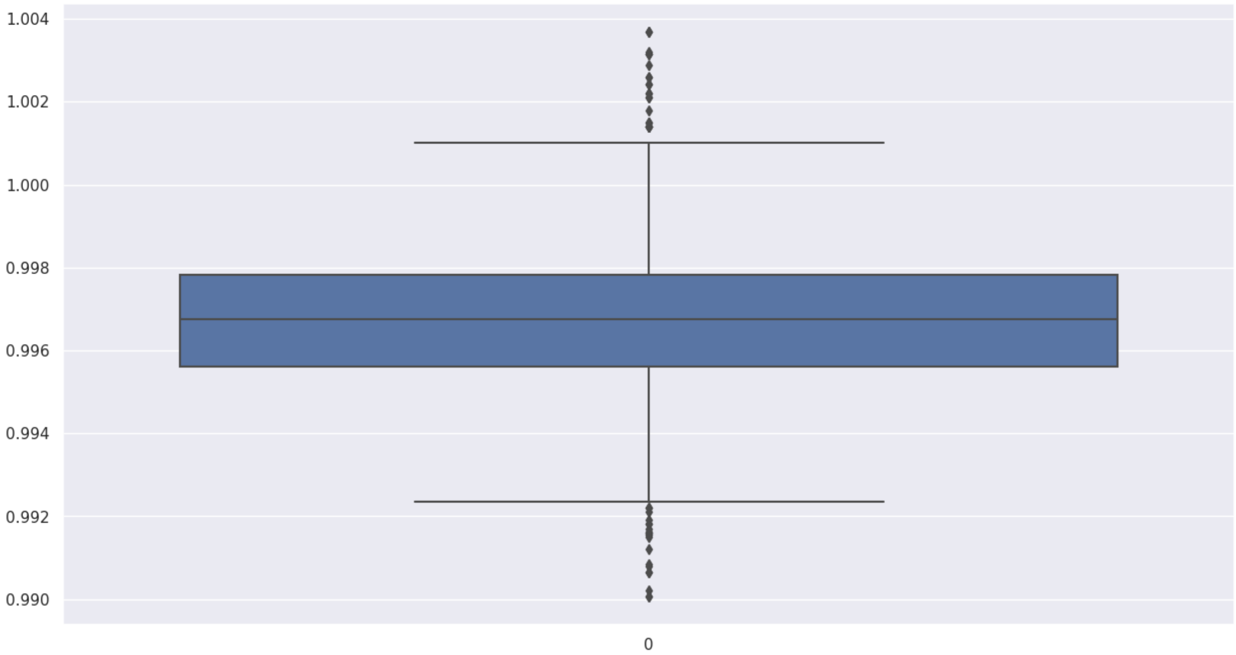
```
sns.boxplot(df["total sulfur dioxide"])
```

```
<Axes: >
```



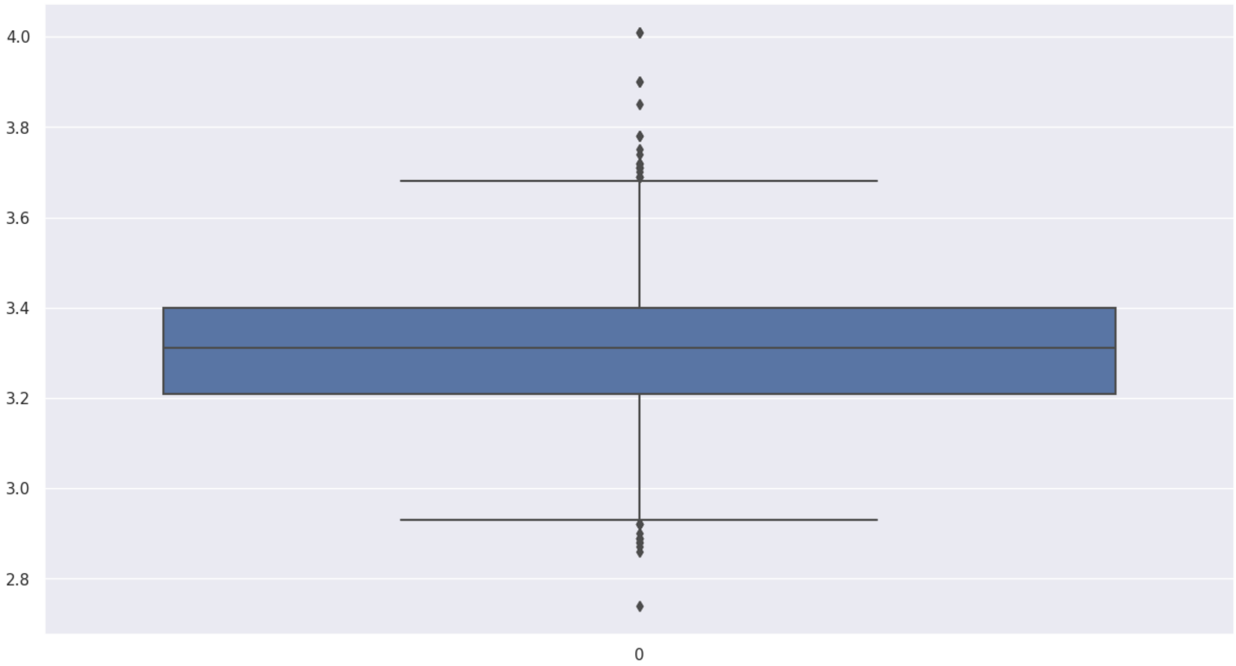
```
sns.boxplot(df["density"])
```

```
<Axes: >
```



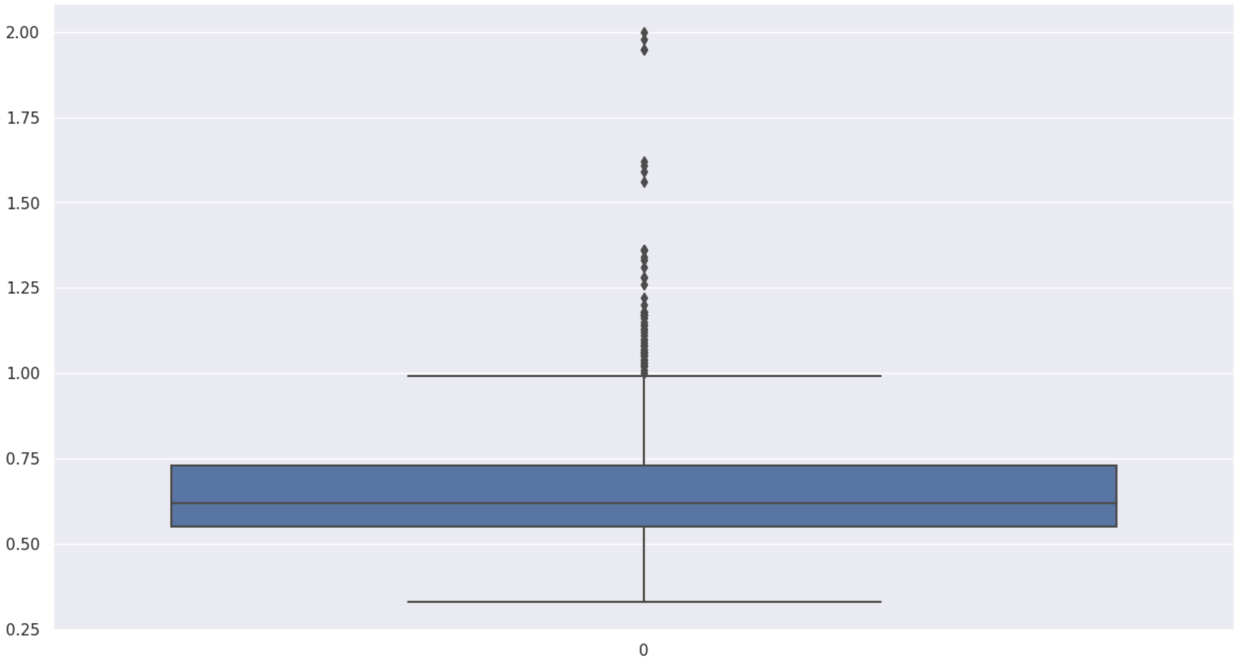
```
sns.boxplot(df["pH"])
```

```
<Axes: >
```



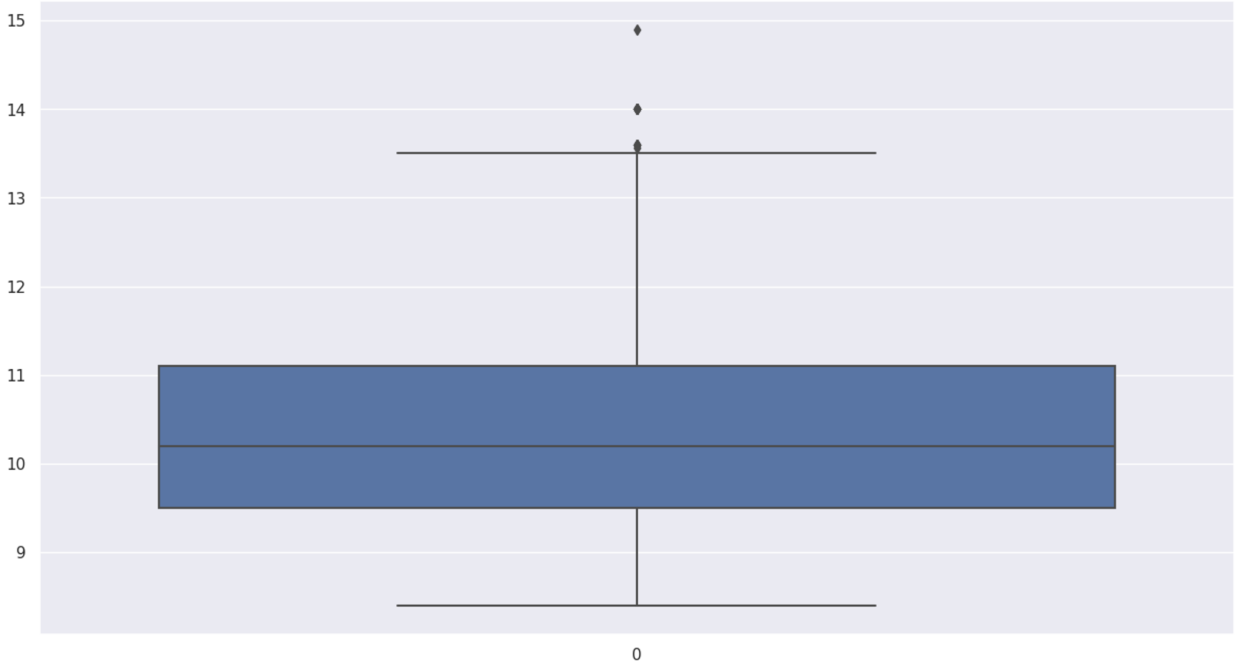
```
sns.boxplot(df["sulphates"])
```

```
<Axes: >
```



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

```
<Axes: >
```



#Replacing the Outliers by IQR

```
# removal of outliers of "fixed acidity" by IQR
q1 = df['fixed acidity'].quantile(0.25)
```

```
q3 = df['fixed acidity'].quantile(0.75)
IQR = q3-q1
IQR
```

```
1.8000000000000007
```

```
upper_limit = q3+1.5*IQR
upper_limit
```

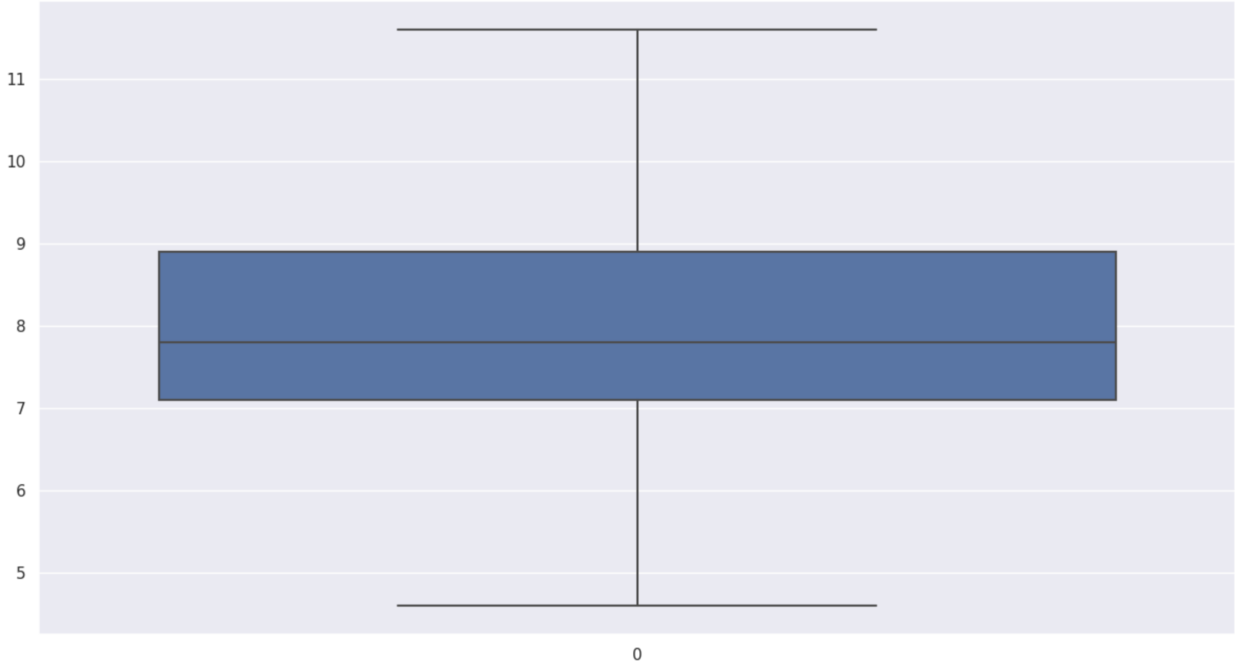
```
11.600000000000001
```

```
lower_limit =q1-1.5*IQR
lower_limit
```

```
4.399999999999999
```

```
df = df[df['fixed acidity']<upper_limit]
sns.boxplot(df["fixed acidity"] )
```

```
<Axes: >
```



```
# removal of outliers of "volatile acidity" by IQR
q1 = df['volatile acidity'].quantile(0.25)
q3 = df['volatile acidity'].quantile(0.75)
IQR = q3-q1
IQR
```

```
0.24375000000000002
```

```
upper_limit = q3+1.5*IQR
upper_limit
```

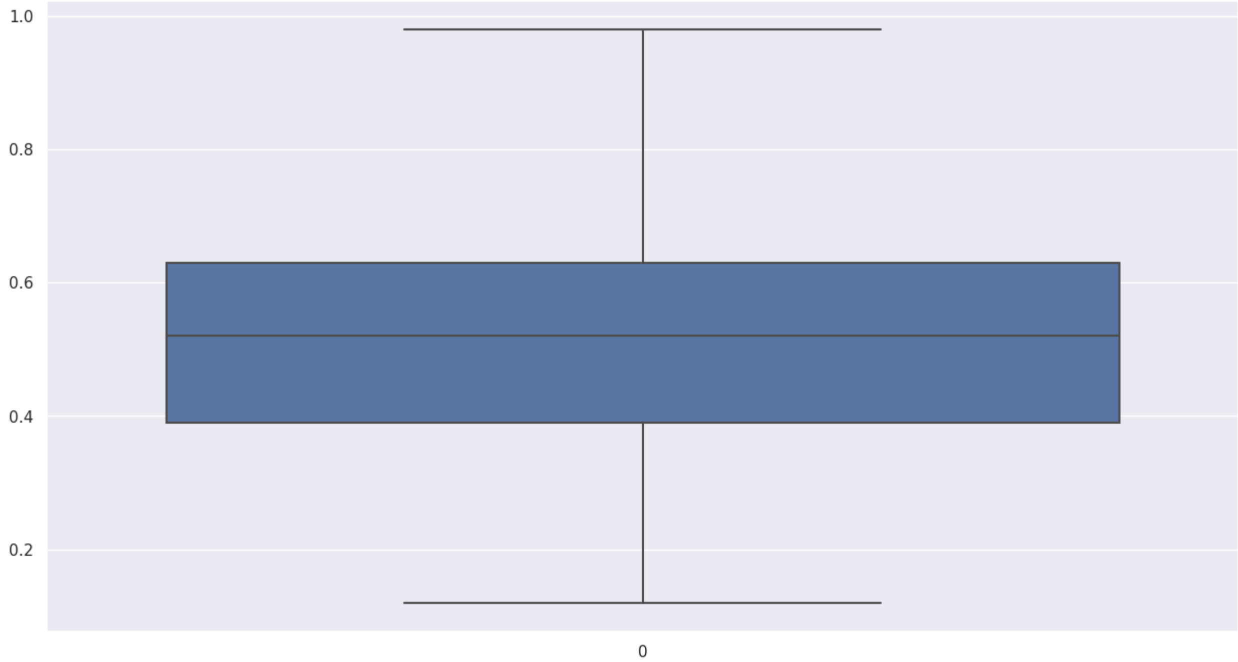
```
0.9993750000000001
```

```
lower_limit =q1-1.5*IQR
lower_limit
```

```
0.024374999999999998
```

```
df = df[df['volatile acidity']<upper_limit]
sns.boxplot(df["volatile acidity"])
```

```
<Axes: >
```



```
# removal of outliers of "citric acid" by IQR
q1 = df['citric acid'].quantile(0.25)
q3 = df['citric acid'].quantile(0.75)
IQR = q3-q1
IQR
```

```
0.31999999999999995
```

```
upper_limit = q3+1.5*IQR
upper_limit
```

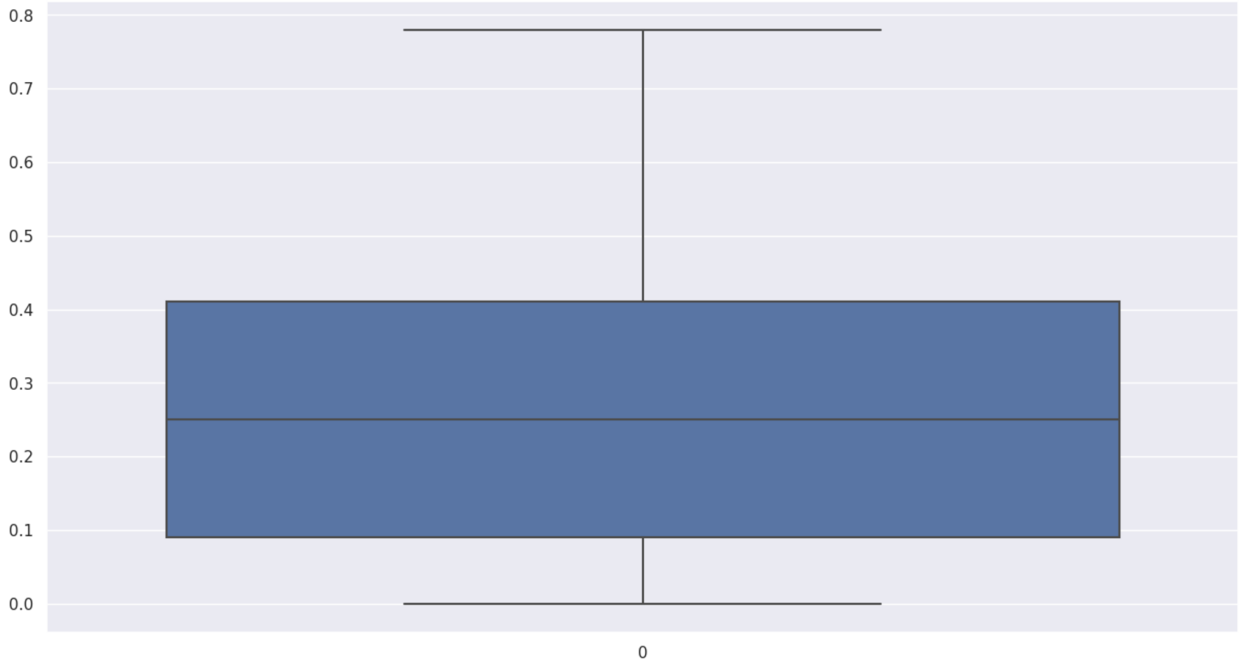
```
0.8899999999999999
```

```
lower_limit =q1-1.5*IQR
lower_limit
```

```
-0.38999999999999999
```

```
df = df[df['citric acid']<upper_limit]
sns.boxplot(df["citric acid"])
```

```
<Axes: >
```



```
# removal of outliers of "residual sugar" by IQR
q1 = df['residual sugar'].quantile(0.25)
q3 = df['residual sugar'].quantile(0.75)
IQR = q3-q1
IQR
```

```
0.5
```

```
upper_limit = q3+1.5*IQR
upper_limit
```

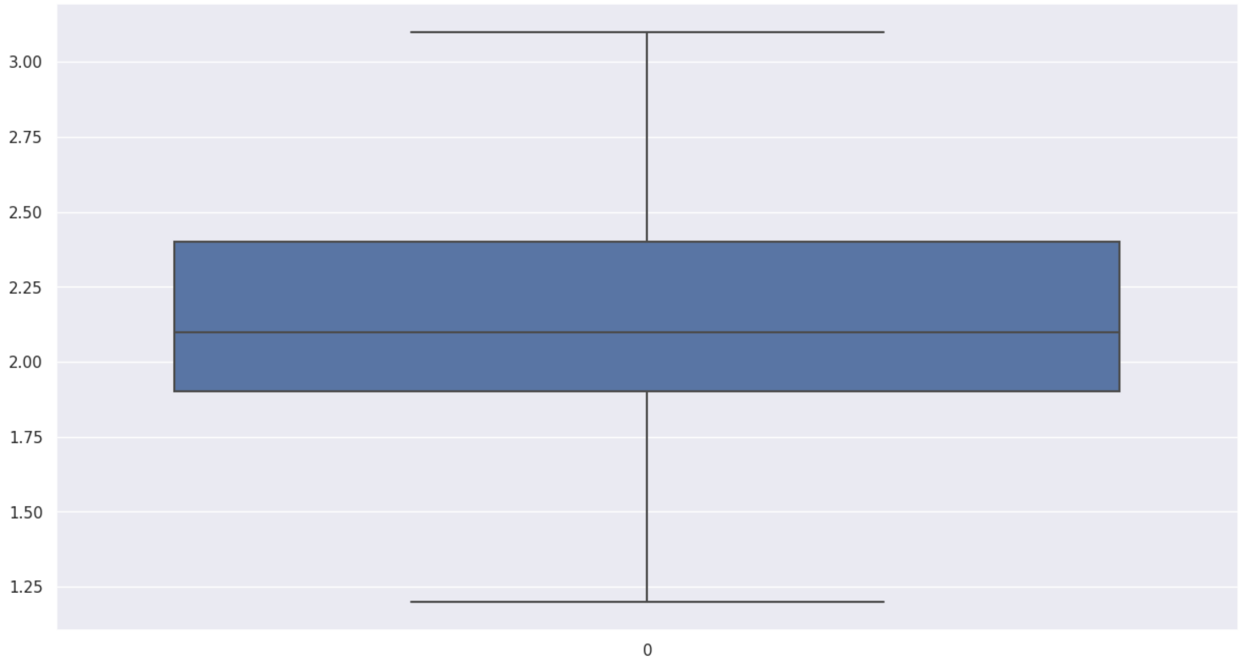
```
3.15
```

```
lower_limit =q1-1.5*IQR
lower_limit
```

```
1.15
```

```
df = df[df['residual sugar']<upper_limit]
sns.boxplot(df["residual sugar"])
```

```
<Axes: >
```



```
# removal of outliers of "chlorides" by IQR
q1 = df['chlorides'].quantile(0.25)
q3 = df['chlorides'].quantile(0.75)
IQR = q3-q1
IQR
```

```
0.016999999999999987
```

```
upper_limit = q3+1.5*IQR
upper_limit
```

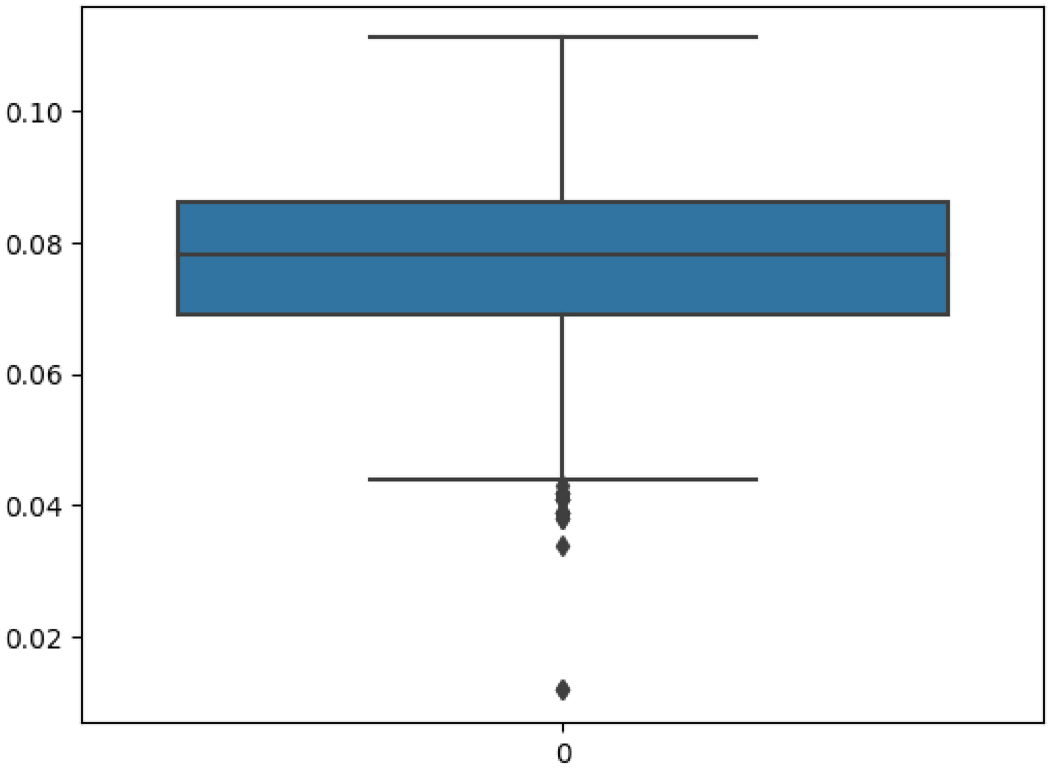
```
0.11149999999999997
```

```
lower_limit =q1-1.5*IQR
lower_limit
```

```
0.043500000000000025
```

```
df = df[df['chlorides']<upper_limit]
sns.boxplot(df["chlorides"])
```

```
<Axes: >
```



```
# removal of outliers of "free sulfur dioxide" by IQR
q1 = df['free sulfur dioxide'].quantile(0.25)
q3 = df['free sulfur dioxide'].quantile(0.75)
IQR = q3-q1
IQR
```

```
13.0
```

```
upper_limit = q3+1.5*IQR
upper_limit
```

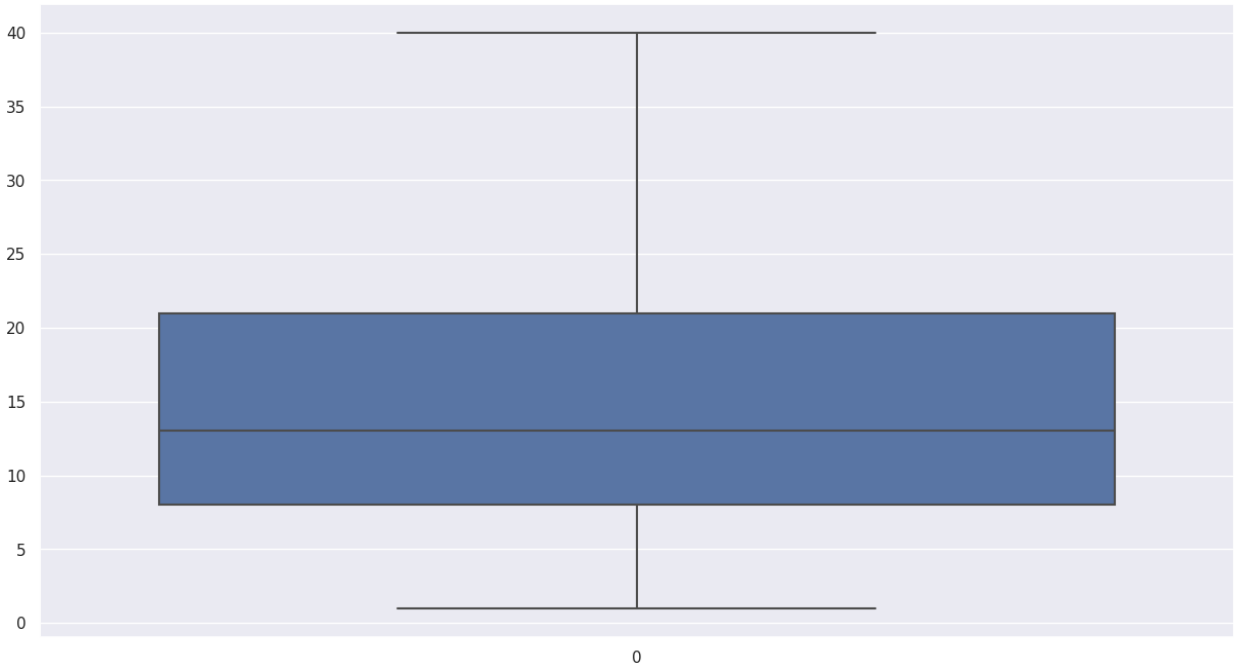
```
40.5
```

```
lower_limit =q1-1.5*IQR
lower_limit
```

```
-11.5
```

```
df = df[df['free sulfur dioxide']<upper_limit]
sns.boxplot(df["free sulfur dioxide"])
```

```
<Axes: >
```



```
# removal of outliers of "total sulfur dioxide" by IQR
q1 = df['total sulfur dioxide'].quantile(0.25)
q3 = df['total sulfur dioxide'].quantile(0.75)
IQR = q3-q1
IQR
```

```
32.0
```

```
upper_limit = q3+1.5*IQR
upper_limit
```

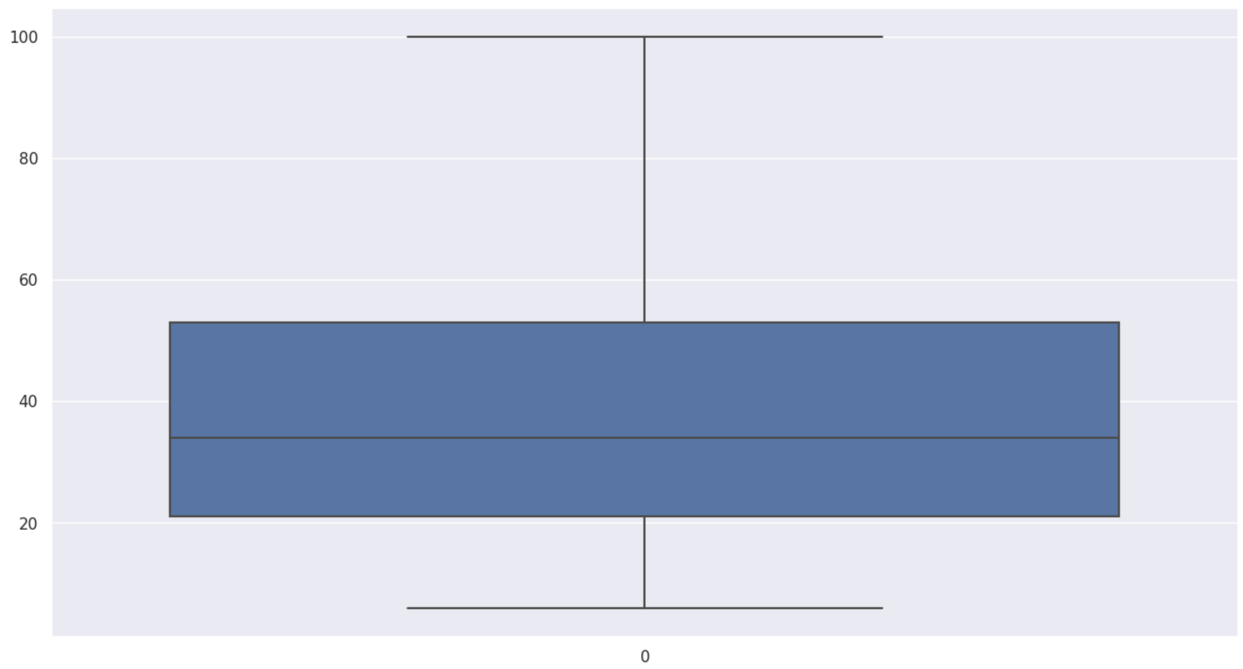
```
101.0
```

```
lower_limit =q1-1.5*IQR
lower_limit
```

```
-27.0
```

```
df = df[df['total sulfur dioxide']<upper_limit]
sns.boxplot(df["total sulfur dioxide"])
```

```
<Axes: >
```



```
# removal of outliers of "density" by IQR
q1 = df['density'].quantile(0.25)
q3 = df['density'].quantile(0.75)
IQR = q3-q1
IQR
```

```
0.0021999999999999797
```

```
upper_limit = q3+1.5*IQR
upper_limit
```

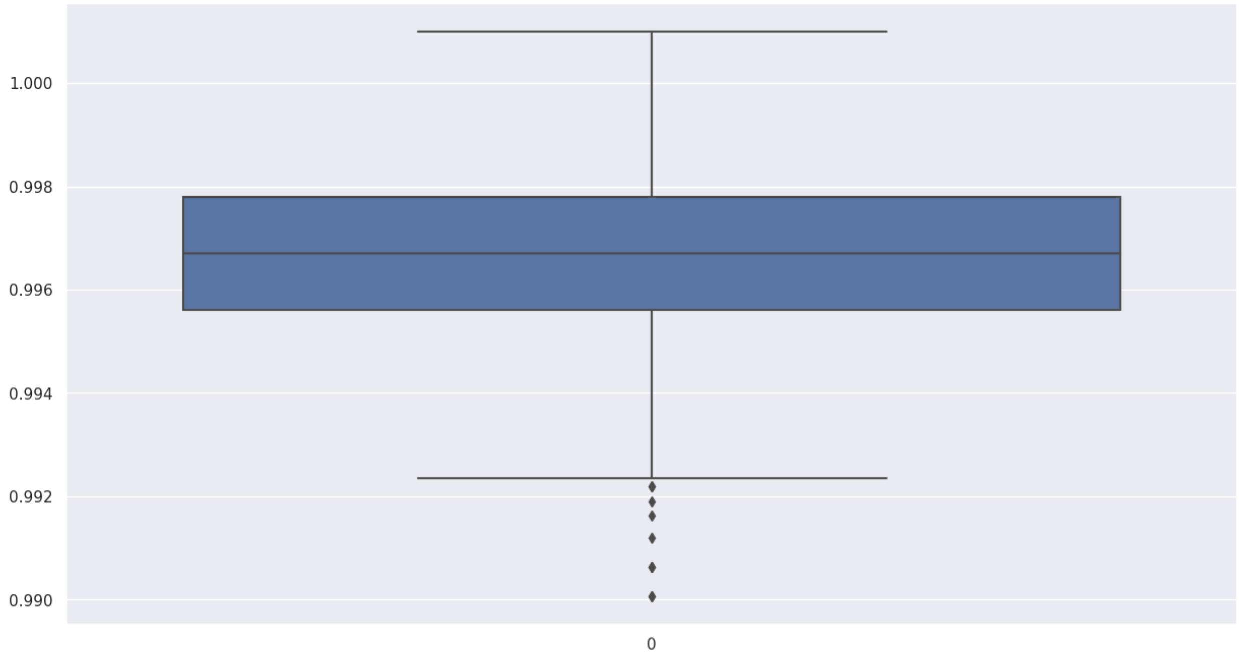
```
1.0011
```

```
lower_limit =q1-1.5*IQR
lower_limit
```

```
0.9923000000000001
```

```
df = df[df['density']<upper_limit]
sns.boxplot(df.density)
```

```
<Axes: >
```



```
# removal of outliers of "pH" by IQR
q1 = df['pH'].quantile(0.25)
q3 = df['pH'].quantile(0.75)
IQR = q3-q1
IQR
```

```
0.18999999999999995
```

```
upper_limit = q3+1.5*IQR
upper_limit
```

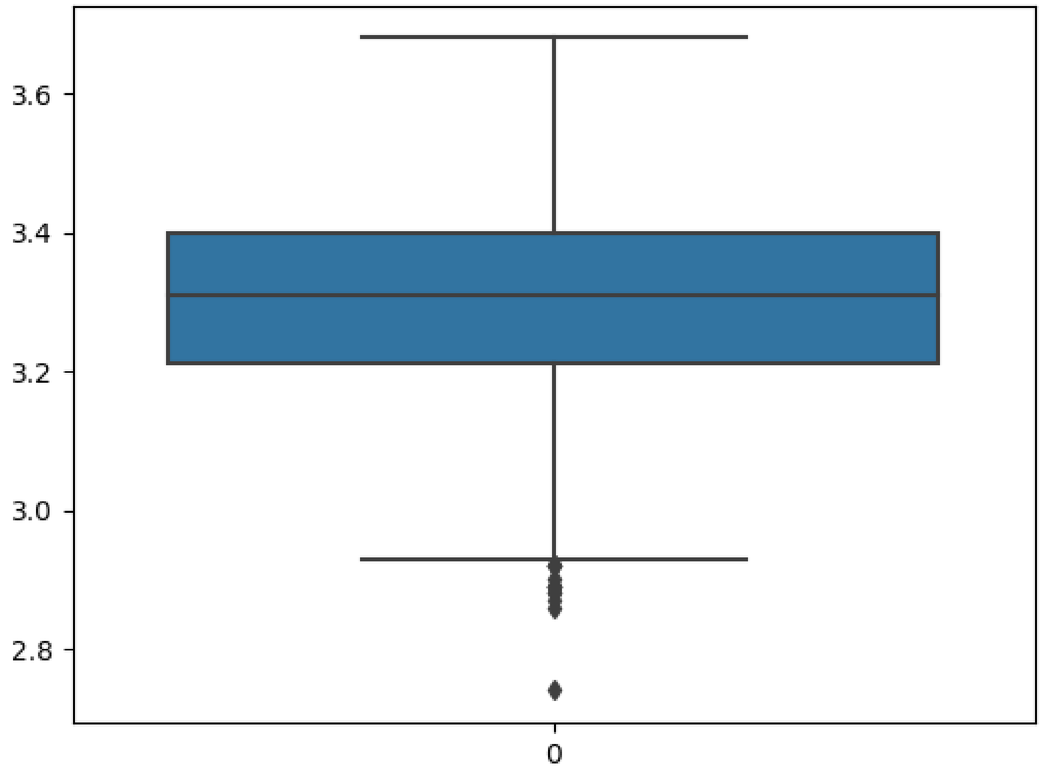
```
3.6849999999999996
```

```
lower_limit =q1-1.5*IQR
lower_limit
```

```
2.925
```

```
df = df[df['pH']<upper_limit]
sns.boxplot(df["pH"])
```

```
<Axes: >
```



```
# removal of outliers of "sulphates" by IQR
q1 = df['sulphates'].quantile(0.25)
q3 = df['sulphates'].quantile(0.75)
IQR = q3-q1
IQR
```

```
0.15249999999999986
```

```
upper_limit = q3+1.5*IQR
upper_limit
```

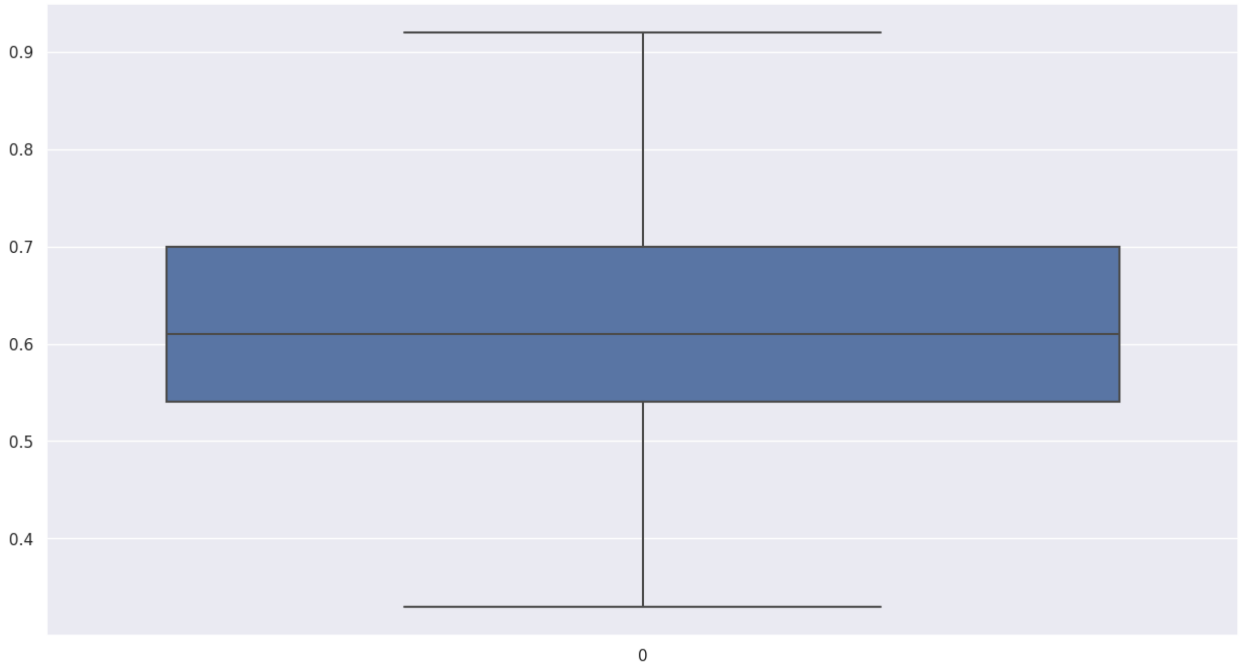
```
0.9287499999999997
```

```
lower_limit =q1-1.5*IQR
lower_limit
```

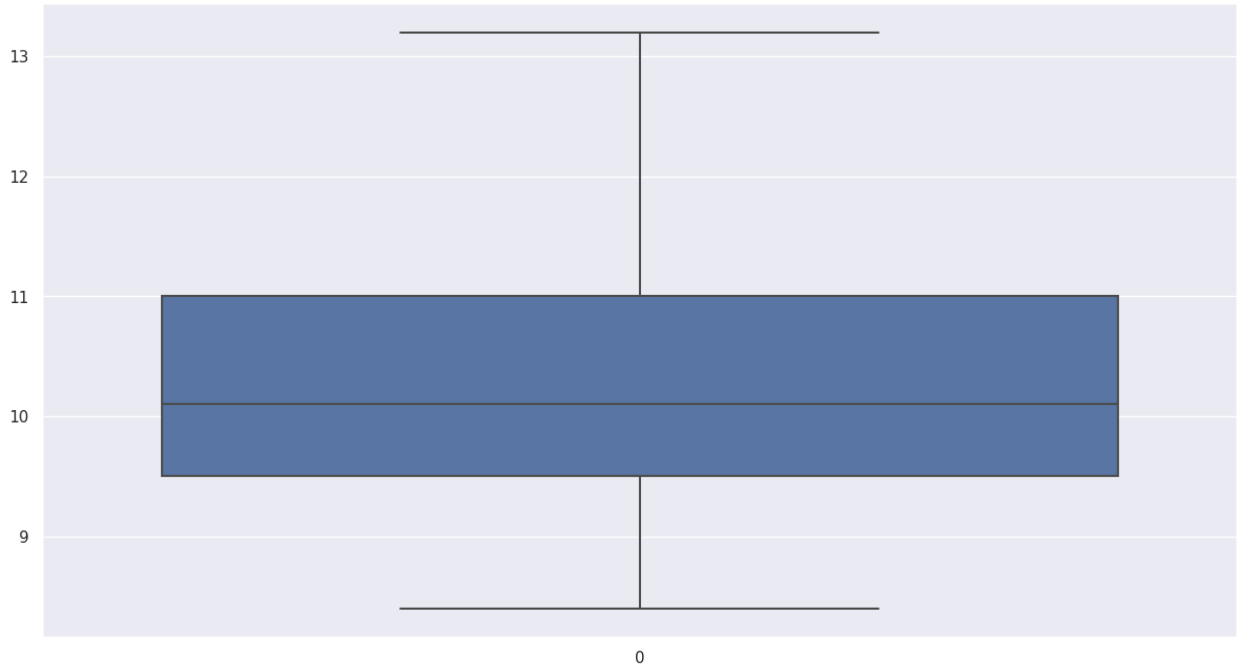
```
0.32125000000000026
```

```
df = df[df['sulphates']<upper_limit]
sns.boxplot(df["sulphates"])
```

```
<Axes: >
```



```
# removal of outliers of "sulphates" by IQR
q1 = df['alcohol'].quantile(0.25)
q3 = df['alcohol'].quantile(0.75)
IQR = q3-q1
IQR

1.5

upper_limit = q3+1.5*IQR
upper_limit

13.25

lower_limit =q1-1.5*IQR
lower_limit

7.25

df = df[df['alcohol']<upper_limit]
sns.boxplot(df["alcohol"])

<Axes: >
```



SPLITING THE VALUES TO X AND Y

```
y=df['quality']
y

0       0
1       0
2       0
3       0
4       0
       ..
1594    0
1595    0
1596    0
1597    0
1598    0
Name: quality, Length: 1599, dtype: int64

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

   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
0      9.4
1      9.8
2      9.8
3      9.8
4      9.4
```

SCALING

```
from sklearn.preprocessing import MinMaxScaler
scale =MinMaxScaler()

x_scaled= pd.DataFrame(scale.fit_transform(x),columns =x.columns)
x_scaled.head()

   fixed acidity  volatile acidity  citric acid  residual sugar
chlorides  \
0       0.247788          0.397260         0.00        0.068493
0.106845
1       0.283186          0.520548         0.00        0.116438
0.143573
2       0.283186          0.438356         0.04        0.095890
0.133556
3       0.584071          0.109589         0.56        0.068493
0.105175
4       0.247788          0.397260         0.00        0.068493
0.106845

   free sulfur dioxide  total sulfur dioxide   density        pH
sulphates  \
0              0.140845              0.098940  0.567548  0.606299
0.137725
1              0.338028              0.215548  0.494126  0.362205
0.209581
2              0.197183              0.169611  0.508811  0.409449
0.191617
3              0.225352              0.190813  0.582232  0.330709
0.149701
4              0.140845              0.098940  0.567548  0.606299
0.137725

    alcohol
0  0.153846
1  0.215385
2  0.215385
3  0.215385
4  0.153846
```

TRAIN TEST SPLIT

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test =
train_test_split(x_scaled,y,test_size=0.3,random_state=10)

x_train.shape

(1119, 11)

x_train.head()

     fixed acidity  volatile acidity  citric acid  residual sugar
chlorides  \
305       0.504425          0.280822         0.48         0.109589
0.085142
984       0.672566          0.226027         0.49         0.034247
0.105175
47        0.362832          0.116438         0.52         0.047945
0.168614
812       0.548673          0.226027         0.33         0.109589
0.145242
3         0.584071          0.109589         0.56         0.068493
0.105175

     free sulfur dioxide  total sulfur dioxide   density        pH
sulphates  \
305             0.070423              0.067138  0.714391  0.299213
0.155689
984             0.028169              0.000000  0.501468  0.307087
0.179641
```

```
47              0.154930              0.109541  0.501468  0.401575
0.149701
812             0.267606              0.113074  0.595448  0.393701
0.227545
3               0.225352              0.190813  0.582232  0.330709
0.149701

      alcohol
305  0.138462
984  0.307692
47   0.169231
812  0.369231
3    0.215385
```

```
y_train.shape
```

```
(1119,)
```

```
y_train.head()
```

```
305    0
984    0
47     0
812    0
3      0
Name: quality, dtype: int64
```

```
x_test.shape
```

```
(480, 11)
```

```
x_test.head()
```

```
      fixed acidity  volatile acidity  citric acid  residual sugar
chlorides  \
1518       0.247788          0.239726         0.46        0.089041
0.170284
1246       0.247788          0.424658         0.07        0.054795
0.123539
544        0.858407          0.130137         0.74        0.061644
0.105175
1343       0.256637          0.267123         0.02        0.054795
0.120200
428        0.398230          0.273973         0.33        0.027397
0.096828

      free sulfur dioxide  total sulfur dioxide   density        pH  \
1518             0.084507              0.049470  0.469897  0.456693
1246             0.197183              0.148410  0.363436  0.299213
544              0.070423              0.031802  0.787812  0.094488
1343             0.169014              0.088339  0.389868  0.488189
```

```
428              0.112676              0.084806  0.567548  0.393701

      sulphates   alcohol
1518   0.179641  0.323077
1246   0.089820  0.246154
544    0.275449  0.000000
1343   0.125749  0.323077
428    0.161677  0.138462
```

```
y_test.shape
```

```
(480,)
```

```
y_test.head()
```

```
1518    0
1246    0
544     0
1343    0
428     0
Name: quality, dtype: int64
```

MODEL BUILDING

```
from sklearn.linear_model import LogisticRegression
```

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

```
LogisticRegression()
```

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

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```python
y_pred =model.predict(x_train)
y_pred
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

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

import sklearn.metrics as metrics

y_pred = model.predict(x_test)
print("Classification Report",metrics.classification_report(y_test,
y_pred))
print("Accuracy score:",metrics.accuracy_score(y_test, y_pred))
print("Precision score:",metrics.precision_score(y_test, y_pred,
average = 'macro'))
print("Recall score:",metrics.recall_score(y_test, y_pred, average =
'macro'))
```

```
Classification Report               precision    recall  f1-score
support
```

```
           0       0.88      0.99      0.93       411
           1       0.72      0.19      0.30        69

    accuracy                           0.87       480
   macro avg       0.80      0.59      0.61       480
weighted avg       0.86      0.87      0.84       480

Accuracy score: 0.8729166666666667
Precision score: 0.8005050505050505
Recall score: 0.5881201734898974
```

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

```
array([[406,   5],
       [ 56,  13]])
```

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

```
col_0      0   1
quality
0        406   5
1         56  13
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.99      0.93       411
           1       0.72      0.19      0.30        69

    accuracy                           0.87       480
   macro avg       0.80      0.59      0.61       480
weighted avg       0.86      0.87      0.84       480
```

MAX ACCURACY SCORE IS 87

DECISION TREE CLASSIFIER

```
from sklearn.tree import DecisionTreeClassifier
```

```
model1 =
DecisionTreeClassifier(max_depth=4,splitter='best',criterion='entropy'
)
```

```
model1.fit(x_train,y_train)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

```
d_y_predict = model1.predict(x_test)
d_y_predict
```

```
array([0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
d_y_predict_train = model1.predict(x_train)
d_y_predict_train
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

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

```
print('Testing Accuracy = ', accuracy_score(y_test,d_y_predict))
print('Training Accuracy = ',
accuracy_score(y_train,d_y_predict_train))
```

```
Testing Accuracy =  0.8541666666666666
Training Accuracy =  0.8972296693476318
```

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

```
col_0      0   1
quality
0        390  21
1         49  20
```

```
print(classification_report(y_test,d_y_predict))
```

```
              precision    recall  f1-score   support

           0       0.89      0.95      0.92       411
           1       0.49      0.29      0.36        69

    accuracy                           0.85       480
   macro avg       0.69      0.62      0.64       480
weighted avg       0.83      0.85      0.84       480
```

MAX ACCURACY SCORE IS 85

RANDOM FOREST CLASSIFIER

```
from sklearn.ensemble import RandomForestClassifier
model2 =RandomForestClassifier(n_estimators=200,criterion='entropy')
```

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

```
RandomForestClassifier(criterion='entropy', n_estimators=200)
```

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

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
r_y_predict_train
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
print('Testing Accuracy = ', accuracy_score(y_test,r_y_predict))
print('Training Accuracy = ',
accuracy_score(y_train,r_y_predict_train))
```

```
Testing Accuracy =  0.9083333333333333
Training Accuracy =  1.0
```

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

```
col_0      0   1
quality
0        404   7
1         37  32
```

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

```
              precision    recall  f1-score   support

           0       0.92      0.98      0.95       411
           1       0.82      0.46      0.59        69

    accuracy                           0.91       480
   macro avg       0.87      0.72      0.77       480
weighted avg       0.90      0.91      0.90       480
```

MAX ACCURACY SCORE IS 91

GOT THE HIGHEST ACCURACY SCORE IN RANDOM FOREST CLASSIFIER

Test with random observation

```
x.head()
```

```
   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  
0      9.4  
1      9.8  
2      9.8  
3      9.8  
4      9.4  
```

```
print("Prediction:",model.predict([[0.283186,0.520548,0.56,0.068493,0.
143573,0.098940,0.567548,0.409449,0.137725,0.191617,0.215385]]))
```

```
Prediction: [0]

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

```
print("Prediction:",model.predict([[0.380531,0.109589,0.45,0.054795,0.
091820,0.084507,0.021201,0.254772,0.401575,0.131737,0.600000]]))
```

```
Prediction: [0]

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

CONCLUSION : all the three models gave the "alchohol quality is BAD"